# OTRS Developer Manual

***Kiadás 7.0***

**OTRS AG**

**júl. 08, 2020**

# Tartalom

FEJEZET 1

# Kezdeti lépések

Az OTRS egy többplatformos webes alkalmazás-keretrendszer, amelyet eredetileg hibajegyrendszernek fejlesztettek. Különböző webkiszolgálókat és adatbázisokat támogat.

Ez a kézikönyv azt mutatja be, hogy hogyan fejleszthet saját OTRS modulokat és alkalmazásokat az OTRS stílus útmutatók alapján.

## 1.1 Fejlesztői környezet

Az OTRS bővítőmodulok írásának megkönnyítéséhez egy fejlesztői környezet létrehozása szükséges. Az OTRS és a további nyilvános modulok forráskódja megtalálható a GitHubon.

### 1.1.1 A keretrendszer letöltése

Először is létre kell hozni egy könyvtárat, amelyben eltárolhatók a modulok. Ezután lépjen be az új könyvtárba a parancssor használatával, és töltse le azokat a következő parancs használatával:

```
# for git master
shell> git clone git@github.com:OTRS/otrs.git -b master
# for a specific branch like OTRS 3.3
shell> git clone git@github.com:OTRS/otrs.git -b rel-3_3
```

Töltse le a `module-tools` modult is (a GitHubról) a fejlesztői környezetéhez. Ez számos hasznos eszközt tartalmaz:

```
shell> git clone git@github.com:OTRS/module-tools.git
```

Állítsa be az OTRS rendszert a telepítési utasítások szerint.

### 1.1.2 Hasznos eszközök

Létezik két modul, amely erősen ajánlott az OTRS fejlesztésnél:

- OTRSCodePolicy
- Fred

Az *OTRSCodePolicy* egy kódolási minőség-ellenőrző, amely kikényszeríti a közös kódolási szabványok használatát az OTRS fejlesztőcsapatnál is. Erősen ajánlott ennek használata, ha hozzájárulásokat tervez készíteni. Használhatja önálló tesztelő parancsfájlként, vagy akár regisztrálhatja egy olyan git véglegesítés horogként, amely minden alkalommal lefut, amikor egy véglegesítést készít. A részletekért nézze meg a modul dokumentációját.

A *Fred* egy kicsi fejlesztést segítő modul, amelyet ténylegesen telepíthet vagy hozzákapcsolhat (a lent leírt módon) a fejlesztői rendszeréhez. Számos hasznos modult szerepeltet, amelyet bekapcsolhat, mint például egy SQL naplózóként vagy egy szabványos hibakimenet konzolként. Néhány további részletet találhat a modul dokumentációjában.

Mellesleg ezek az eszközök szintén nyílt forrásúak, és nagyon örülnénk bármilyen továbbfejlesztésnek, amellyel hozzájárul.

### 1.1.3 Bővítőmodulok hozzákapcsolása

Egyértelmű szétválasztás szükséges az OTRS és a modulok között a megfelelő fejlesztéshez. Különösen egy git klón használatakor kritikus az egyértelmű szétválasztás. Annak érdekében, hogy megkönnyítse az OTRS-nek a fájlokhoz történő hozzáférést, linkeket kell létrehozni. Ez megtehető a könyvtármodul eszközök tárolójában lévő parancsfájllal.

Példa: a *Naptár* modul hozzákapcsolása:

```
shell> ~/src/module-tools/link.pl ~/src/Calendar/ ~/src/otrs/
```

Amikor új fájlok kerülnek hozzáadásara, akkor azokat a fent bemutatott módon kell hozzákapcsolni.

Amint a hozzákapcsolás befejeződött, újra kell építeni a rendszerbeállításokat a modul regisztrálásához az OTRS-be. A további SQL vagy Perl-kódot is végre kell hajtani a modulból.

Példa:

```
shell> ~/src/otrs/bin/otrs.Console.pl Maint::Config::Rebuild
shell> ~/src/module-tools/DatabaseInstall.pl -m Calendar.sopm -a install
shell> ~/src/module-tools/CodeInstall.pl -m Calendar.sopm -a install
```

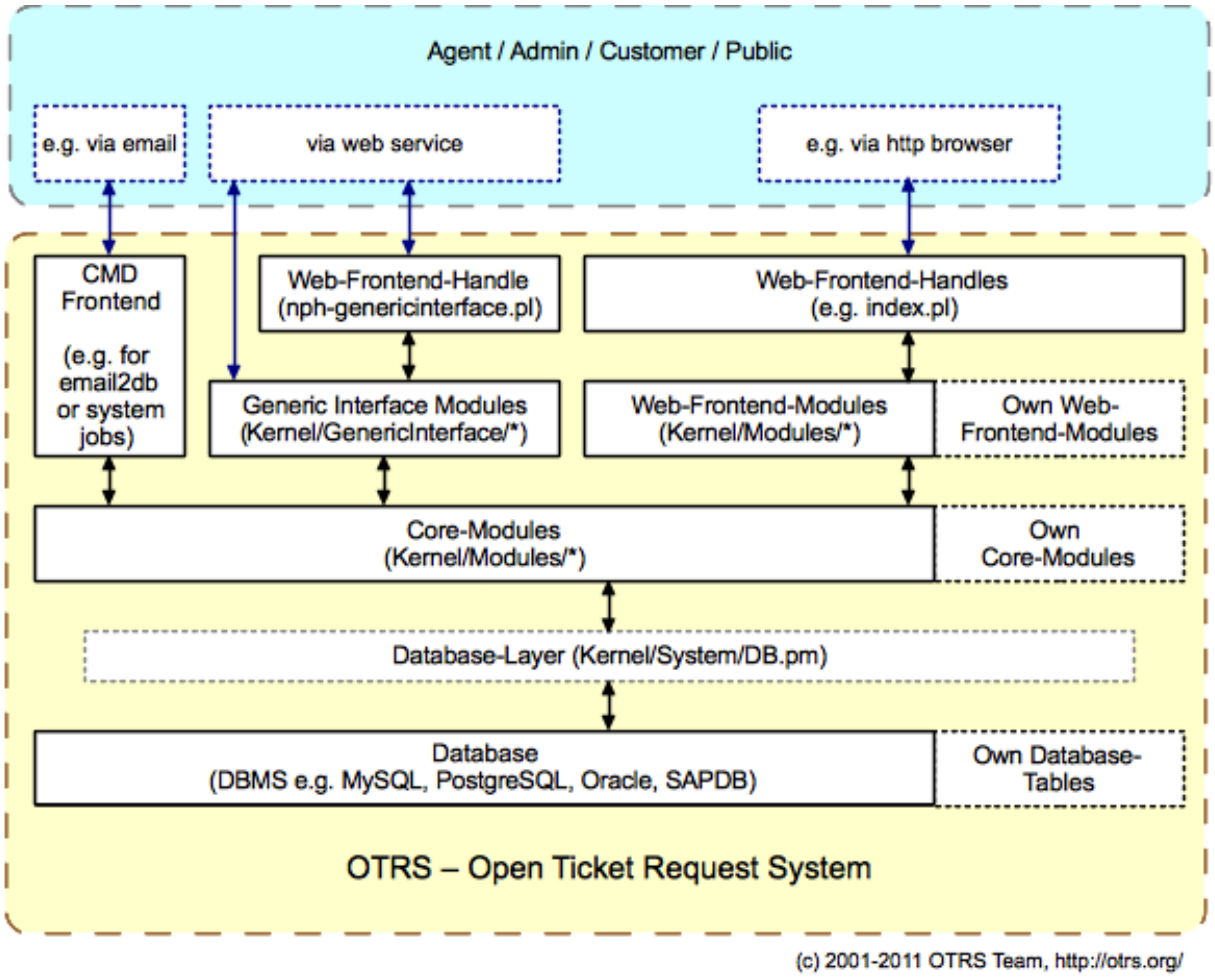
A kapcsolatoknak az OTRS-ből történő eltávolításához adja ki a következő parancsot:

```
shell> ~/src/module-tools/remove_links.pl ~/src/otrs/
```

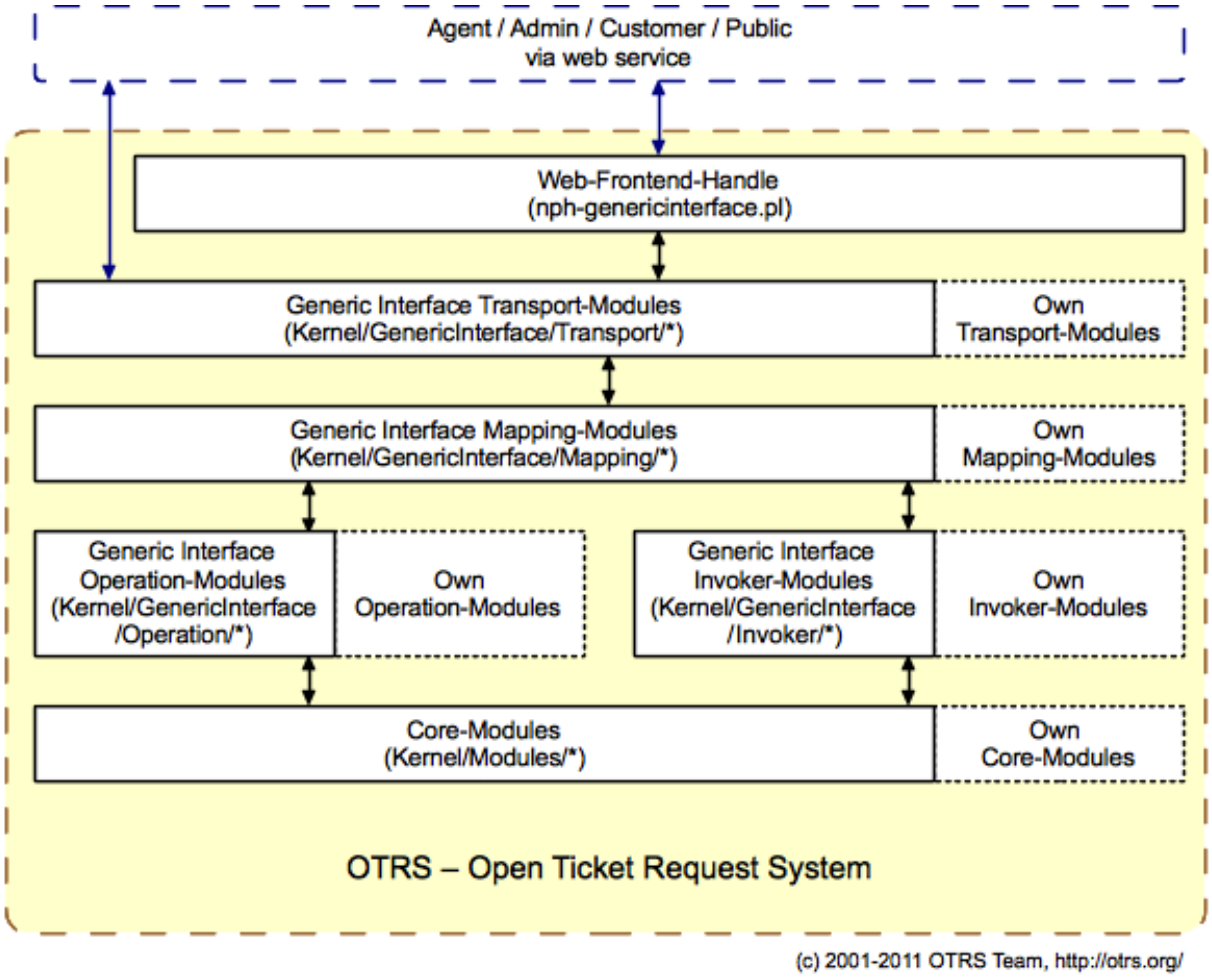
## 1.2 Szerkezeti áttekintés

Az OTRS keretrendszer moduláris. A következő kép az OTRS alapvető rétegszerkezetét jeleníti meg.

Az OTRS általános felület folytatja az OTRS modularitását. A következő kép az általános felület alapvető rétegszerkezetét jeleníti meg.




1. ábra: Az OTRS szerkezete




2. ábra: Az általános felület szerkezete

### 1.2.1 Könyvtárak

| Könyvtár | Leírás |
|---|---|
| bin/ | parancssori eszközök |
| bin/cgi-bin/ | webes kezelő |
| bin/fcgi-bin/ | fast CGI webes kezelő |
| Kernel | alkalmazás kódbázisa |
| Kernel/Config/ | beállítófájlok |
| Kernel/Config/Files | beállítófájlok |
| Kernel/GenericInterface/ | az általános felület API-ja |
| Kernel/GenericInterface/Invoker/ | meghívó modulok az általános felülethez |
| Kernel/GenericInterface/Mapping/ | leképező modulok az általános felülethez, például: Simple |
| Kernel/GenericInterface/Operation/ | műveleti modulok az általános felülethez |
| Kernel/GenericInterface/Transport/ | átviteli modulok az általános felülethez, például: „HTTP SOAP” |
| Kernel/Language | nyelvi fordítási fájlok |
| Kernel/Scheduler/ | az ütemező fájljai |
| Kernel/Scheduler/TaskHandler | kezelőmodulok az ütemező feladatokhoz, például: GenericInterface |
| Kernel/System/ | alapmodulok, például: Log, Ticket |
| Kernel/Modules/ | előtétprogram-modulok, például: QueueView |
| Kernel/Output/HTML/ | HTML sablonok |
| var/ | változó adatok |
| var/log | naplófájlok |
| var/cron/ | cron fájlok |
| var/httpd/htdocs/ | htdocs könyvtár az index.html fájllal |
| var/httpd/htdocs/skins/Agent/ | az ügyintéző felületéhez elérhető felszínek |
| var/httpd/htdocs/skins/Customer/ | az ügyfélfelülethez elérhető felszínek |
| var/httpd/htdocs/js/ | JavaScript fájlok |
| scripts/ | egyéb fájlok |
| scripts/test/ | egységteszt fájlok |
| scripts/test/sample/ | egységteszt példaadat fájlok |

### 1.2.2 Fájlok

- .pl = Perl
- .pm = Perl-modul
- .tt = Template::Toolkit sablonfájlok
- .dist = Fájlok alapértelmezett sablonjai
- .yaml vagy .yml = A webszolgáltatás beállításaihoz használt YAML-fájlok

### 1.2.3 Alapmodulok

Az alapmodulok az `$OTRS_HOME/Kernel/System/*` alatt találhatók. Ez a réteg a logikai munkához van. Az alapmodulok a rendszer rutinjai kezeléséhez használhatók, mint például *jegy zárolása* és *jegy létrehozása*. Néhány fő alapmodul a következő:

- `Kernel::System::Config` a beállítási lehetőségek eléréséhez.

- `Kernel::System::Log` a naplózáshoz az OTRS naplózó háttérprogramjába.
- `Kernel::System::DB` az adatbázis háttérprogram eléréséhez.
- `Kernel::System::Auth` a felhasználói hitelesítés ellenőrzéséhez.
- `Kernel::System::User` a felhasználók kezeléséhez.
- `Kernel::System::Group` a csoportok kezeléséhez.
- `Kernel::System::Email` az e-mailek küldéséhez.

További információkért nézze meg a dokumentációs portált.

### 1.2.4 Előtétprogram kezelés

A böngésző, a webkiszolgáló és az előtétprogram-modulok közti felület. Egy előtétprogram-modul használható HTTP-hivatkozáson keresztül.

```
http://localhost/otrs/index.pl?Action=Module
```

### 1.2.5 Előtétprogram-modulok

Az előtétprogram-modulok az `$OTRS_HOME/Kernel/Modules/*.pm` alatt találhatók. Két nyilvános függvény van ebben - `new()` és `run()` - amelyek az előtétprogram kezelésből érhetők el (például `index.pl`).

A `new()` előtétprogram-modul objektumok létrehozásához használható. Az előtétprogram kezelés biztosítja a használt előtétprogram-modulokat az alapvető keretrendszer objektumokkal. Ezek például a következők:

- `ParamObject` a webes űrlapparaméterek lekéréséhez.
- `DBObject` a meglévő adatbázis-kapcsolatok használatához.
- `LayoutObject` a sablonok és egyéb HTML elrendezés függvények használatához.
- `ConfigObject` a konfigurációs beállítások hozzáféréséhez.
- `LogObject` a keretrendszer naplózó rendszerének használatához.
- `UserObject` a felhasználói függvények lekéréséhez a jelenlegi felhasználótól.
- `GroupObject` a csoportfüggvények lekéréséhez.

További információkért nézze meg a dokumentációs portált.

### 1.2.6 Parancssori előtétprogram

A CMD (parancssori) előtétprogram a webes előtétprogram kezeléshez és a webes előtétprogram-modulokhoz együtt hasonló (csak a `LayoutObject` nélkül), és az alapmodulokat használja néhány műveletnél a rendszeren.

### 1.2.7 Általános felület modulok

Az általános felület modulok az `$OTRS_HOME/Kernel/GenericInterface/*` alatt találhatók. Az általános felület modulok egy webszolgáltatás végrehajtásának egyes részei kezeléséhez használhatók a rendszeren. Az általános felület fő moduljai a következők:

- `Kernel::GenericInterface::Transport` a kölcsönhatásba lépéshez távoli rendszerekkel.

- `Kernel::GenericInterface::Mapping` az adatok átalakításához egy szükséges formátumba.
- `Kernel::GenericInterface::Requester` az OTRS kliensként való használatához a webszolgáltatásnál.
- `Kernel::GenericInterface::Provider` az OTRS kiszolgálóként való használatához a webszolgáltatásnál.
- `Kernel::GenericInterface::Operation` a szolgáltató műveletek végrehajtásához.
- `Kernel::GenericInterface::Invoker` a kérelmező műveletek végrehajtásához.
- `Kernel::GenericInterface::Debugger` a webszolgáltatás kommunikációjának követéséhez naplóbejegyzések használatával.

További információkért nézze meg a dokumentációs portált.

#### Általános felület meghívó modulok

Az általános felület meghívó modulok az `$OTRS_HOME/Kernel/GenericInterface/Invoker/*` alatt találhatók. Minden egyes meghívót egy `Controller` elnevezésű mappa tartalmaz. Ez a megközelítés nem csak a belső osztályoknak és metódusoknak segít egy névteret meghatározni, hanem a fájlneveknek is. Például az `$OTRS_HOME/Kernel/GenericInterface/Invoker/Test/` az összes teszt típusú meghívó vezérlője.

Az általános felület meghívó modulok háttérprogramként használhatók kérések létrehozásához a távoli rendszereknél műveletek végrehajtásához.

További információkért nézze meg a dokumentációs portált.

#### Általános felület leképező modulok

Az általános felület leképező modulok az `$OTRS_HOME/Kernel/GenericInterface/Mapping/*` alatt találhatók. Ezek a modulok adatok (kulcsok és értékek) átalakításához használhatók az egyik formátumról egy másikra.

További információkért nézze meg a dokumentációs portált.

#### Általános felület műveleti modulok

Az általános felület műveleti modulok az `$OTRS_HOME/Kernel/GenericInterface/Operation/*` alatt találhatók. Minden egyes műveletet egy `Controller` elnevezésű mappa tartalmaz. Ez a megközelítés nem csak a belső osztályoknak és metódusoknak segít egy névteret meghatározni, hanem a fájlneveknek is. Például az `$OTRS_HOME/Kernel/GenericInterface/Operation/Ticket/` az összes jegy típusú művelet vezérlője.

Az általános felület műveleti modulok háttérprogramként használhatók egy távoli rendszer által kért műveletek végrehajtásához.

További információkért nézze meg a dokumentációs portált.

#### Általános felület átviteli modulok

Az általános felület átviteli modulok az `$OTRS_HOME/Kernel/GenericInterface/Transport/*` alatt találhatók. Minden egyes átviteli modult egy olyan elnevezésű könyvtárba kell elhelyezni, ahogy a használt hálózati protokollt hívják. Például a „HTTP SOAP” átviteli modul a `$OTRS_HOME/Kernel/GenericInterface/Transport/HTTP/SOAP.pm` fájlban található.

Az általános felület átviteli modulok adatok küldéséhez használhatók egy távoli rendszer számára, valamint adatok fogadásához tőle.

További információkért nézze meg a dokumentációs portált.

### 1.2.8 Ütemező feladatkezelő modulok

Az ütemező feladatkezelő modulok az `$OTRS_HOME/Kernel/Scheduler/TaskHandler/*` alatt találhatók. Ezek a modulok aszinkron feladatok végrehajtásához használhatók. Például a `GenericInterface` feladatkezelő általános felület kéréseket hajt végre a távoli rendszerekhez az Apache folyamaton kívül. Ez abban segíti a rendszert, hogy fogékonyabb legyen, megelőzve a lehetséges teljesítményproblémákat.

További információkért nézze meg a dokumentációs portált.

### 1.2.9 Adatbázis

Az adatbázis-felület különböző adatbázisokat támogat.

Az OTRS adatmodelljéhez nézze meg a `/doc` könyvtárban lévő fájlokat. Alternatívaként megnézheti az adatmodellt a GitHubon is.

FEJEZET 2

# OTRS belsőségek - hogyan működik

## 2.1 Beállítási mechanizmus

Az OTRS dedikált mechanizmussal érkezik a konfigurációs beállítások kezeléséhez egy grafikus felületen (rendszerbeállításokon) keresztül. Ez a szakasz azt írja le, hogy hogyan működik belsőleg, és hogyan adhat meg új konfigurációs beállításokat, vagy hogyan változtathatja meg a meglévő alapértelmezett értékeket.

### 2.1.1 `Defaults.pm`: az OTRS alapértelmezett beállításai

Az OTRS alapértelmezett beállítófájlja a `Kernel/Config/Defaults.pm`. Ez a fájl szükséges az üzembe állított XML-beállítások nélküli, frissen telepített rendszerek működéséhez, és a fájlt érintetlenül kell hagyni, mivel automatikusan frissítésre kerül a keretrendszer frissítéseikor.

### 2.1.2 Automatikusan előállított beállítófájlok

A `Kernel/Config/Files` mappában néhány automatikusan előállított beállítófájl található:

**`ZZZAAuto.pm`** Az XML beállítások aktuális értékeinek (alapértelmezett vagy a felhasználó által módosított) Perl gyorsítótára.

**`ZZZACL.pm`** Az adatbázisból származó ACL beállítások Perl gyorsítótára.

**`ZZZProcessManagement.pm`** Az adatbázisból származó folyamatmenedzsment beállítások Perl gyorsítótára.

Ezek a fájlok az aktuális rendszerbeállítások Perl változatai. Sosem szabad kézzel megváltoztatni, mivel az OTRS felülírja azokat.

### 2.1.3 XML beállítófájlok

Az OTRS-ben azok a beállítási lehetőségek, amelyeket az adminisztrátor a rendszerbeállításokon keresztül be tud állítani, különleges formátumban lévő XML-fájlokon keresztül biztosítottak. A régi XML-ek átalakítá-

sához használhatja a következő parancsot:

```
otrs> /opt/otrs/bin/otrs.Console.pl Dev::Tools::Migrate::ConfigXMLStructure
```

A `Kernel/Config/Files/ZZZAAuto.pm` fájl az XML gyorsítótárazott Perl verziója, amely tartalmazza az összes beállítást azok aktuális értékeivel. Újra előállíthatók a következő paranccsal:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Config::Rebuild
```

Az egyes XML beállítófájloknak a következő elrendezésük van:

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="2.0" init="Changes">

    <!--  settings will be here -->

</otrs_config>
```

**init** A globális `init` attribútum azt írja le, hogy honnan kell a beállítási lehetőségeket betölteni. Különböző szintek érhetők el, és a következő sorrendben lesznek betöltve/felülírva:

- `Framework` (a keretrendszer beállításaihoz, például munkamenet beállítás)
- `Application` (az alkalmazás beállításaihoz, például jegybeállítások)
- `Config` (kiterjesztésekhez a meglévő alkalmazásoknál, például ITSM beállítások)
- `Changes` (egyéni fejlesztésekhez, például keretrendszer vagy jegybeállítások felülírásához).

A beállítási elemek `Setting` elemekként vannak írva egy `Description` leírással, egy `Navigation` csoporttal (a fa alapú navigációhoz a grafikus felhasználói felületen) és a `Value` értékkel, amely azt képviseli. Álljon itt egy példa:

```
<Setting Name="Ticket::Hook" Required="1" Valid="1">
    <Description Translatable="1">The identifier for a ticket, e.g. Ticket#,␣
↪Call#, MyTicket#. The default is Ticket#.</Description>
    <Navigation>Core::Ticket</Navigation>
    <Value>
        <Item ValueType="String" ValueRegex="">Ticket#</Item>
    </Value>
</Setting>
```

**Required** Ha ez 1 értékre van állítva, akkor a konfigurációs beállítást nem lehet letiltani.

**Valid** Meghatározza, hogy a konfigurációs beállítás alapértelmezetten be van kapcsolva (1) vagy ki van kapcsolva (0).

**ConfigLevel** Ha az opcionális `ConfigLevel` attribútum be van állítva, akkor a beállítási változót esetleg nem szerkesztheti az adminisztrátor a saját beállítási szintjétől függően. A `ConfigLevel` beállítási változó állítja be az adminisztrátor szakmai tapasztalatának szintjét. Lehet *100 (Szakértő)*, *200 (Speciális)* vagy *300 (Kezdő)*. Iránymutatásként, hogy mely beállítási szintet kell egy lehetőséghez megadni, az az ajánlott, hogy az összes olyan lehetőségnek, amelyet külső interakció beállításával kell megtenni (mint például Sendmail, LDAP, SOAP és egyebek), legalább *200 (Speciális)* beállítási szintet kell kapnia.

**Invisible** Ha 1 értékre van állítva, akkor a konfigurációs beállítás nem jelenik meg a grafikus felhasználói felületen.

**Readonly** Ha 1 értékre van állítva, akkor a konfigurációs beállítást nem lehet megváltoztatni a grafikus felhasználói felületen.

**UserModificationPossible** Ha a `UserModificationPossible` `1` értékre van állítva, akkor az adminisztrátorok engedélyezhetik a beállítás felhasználói módosításait (a felhasználói beállításokban).

**UserModificationActive** Ha a `UserModificationActive` `1` értékre van állítva, akkor a beállítás felhasználói módosításai engedélyezve vannak (a felhasználói beállításokban). Ezt az attribútumot a `UserModificationPossible` attribútummal együtt kell használnia.

**UserPreferencesGroup** Használja a `UserPreferencesGroup` attribútumot annak meghatározásához, hogy mely csoport alá tartozik a konfigurációs érték (a `UserPreferences` képernyőn). Ezt az attribútumot a `UserModificationPossible` attribútummal együtt kell használnia.

Iránymutatások a beállítások megfelelő navigációs csomópontokba való elhelyezéséhez:

- Csak akkor hozzon létre új csomópontot, ha szükséges. Kerülje a csak nagyon kevés beállítással rendelkező csomópontokat, ha lehetséges.
- A fa első szintjén nem szabad új csomópontokat hozzáadni.
- Ne tegyen új beállításokat a `Core` csomópontba közvetlenül. Ez néhány fontos globális beállításnak van fenntartva.
- A `Core::*` kaphat új csoportokat, amelyek hasonló témát lefedő beállításokat tartalmaznak (például `Core::Email`) vagy ugyanahhoz az entitáshoz kapcsolódnak (például `Core::Queue`).
- Az összes eseménykezelő regisztráció a `Core::Event` csoportba kerüljön.
- Ne adjon hozzá új közvetlen gyermekcsomópontokat a `Frontend` csomóponton belül. A globális előtétprogram beállítások a `Frontend::Base` csoportba kerüljenek, a csak a rendszer egy részét érintő beállítások a megfelelő `Admin`, `Agent`, `Customer` vagy `Public` alcsoportokba kerüljenek.
- Azok az előtétprogram beállítások, amelyek csak egy képernyőt érintenek, a kapcsolódó képernyő (`View`) csomópontjába (hozzon létre egyet, ha szükséges) kerüljenek. Például az `AgentTicketZoom` képernyőhöz kapcsolódó beállítások a `Frontend::Agent::View::TicketZoom` csoportba kerüljenek. Ha kapcsolódó beállítások csoportjaival rendelkező modulrétegek vannak egy képernyőn belül, akkor azok is kerülhetnek ide egy alcsoportba (például `Frontend::Agent::View::TicketZoom::MenuModule` az összes jegynagyítás menü modulregisztrációjához).
- Az összes globális betöltő beállítások a `Frontend::Base::Loader` csoportba, a képernyőre jellemző betöltő beállítások a `Frontend::*::ModuleRegistration::Loader` csoportba kerüljenek.

### A `Value` elemek szerkezete

A `Value` elemek tartalmazzák a tényleges beállítási adatok hasznos terhét. Tartalmazhatnak önálló értékeket, kivonatokat vagy tömböket.

#### `Item`

Egy `Item` elem egyetlen adatot tartalmaz. Az elhagyható `ValueType` attribútum határozza meg, hogy milyen típusú adat, és hogyan kell megjeleníteni a felhasználónak a grafikus felületen. Ha nincs `ValueType` megadva, akkor alapértelmezetten `String` lesz.

Nézze meg az *Értéktípusok* fejezetet a különböző értéktípusok meghatározásához.

```
<Setting Name="Ticket::Hook" Required="1" Valid="1">
    <Description Translatable="1">The identifier for a ticket, e.g. Ticket#,␣
↪Call#, MyTicket#. The default is Ticket#.</Description>
    <Navigation>Core::Ticket</Navigation>
    <Value>
        <Item ValueType="String" ValueRegex="">Ticket#</Item>
    </Value>
</Setting>
```

#### Array

Ezzel a beállítási elemmel tömbök jeleníthetők meg.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Array>
            <Item Translatable="1">Value 1</Item>
            <Item Translatable="1">Value 2</Item>
            ...
        </Array>
    </Value>
</Setting>
```

#### Hash

Ezzel a beállítási elemmel kivonatok jeleníthetők meg.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Hash>
            <Item Key="One" Translatable="1">First</Item>
            <Item Key="Two" Translatable="1">Second</Item>
            ...
        </Hash>
    </Value>
</Setting>
```

Lehetőség van egymásba ágyazott tömb vagy kivonat elemek meglétére is (mint például tömbök kivonata, kivonatok tömbje, tömbök kivonatainak tömbje, stb.). Lent van egy példa a kivonatok tömbjére.

```
<Setting Name="ExampleAoH">
    ...
    <Value>
        <Array>
            <DefaultItem>
                <Hash>
                    <Item></Item>
                </Hash>
```

(continues on next page)

(folytatás az előző oldalról)

```
            </DefaultItem>
            <Item>
                <Hash>
                    <Item Key="One">1</Item>
                    <Item Key="Two">2</Item>
                </Hash>
            </Item>
            <Item>
                <Hash>
                    <Item Key="Three">3</Item>
                    <Item Key="Four">4</Item>
                </Hash>
            </Item>
        </Array>
    </Value>
</Setting>
```

### Értéktípusok

Az XML konfigurációs beállítások különféle típusú beállítási változókat támogatnak.

#### String

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="String" ValueRegex=""></Item>
    </Value>
</Setting>
```

Egy beállítási elem számokhoz és egysoros karakterláncokhoz. Ellenőrizhető az érvényesség egy reguláris kifejezéssel, ha lehetséges (elhagyható). Ez az alapértelmezett `ValueType`.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="String" ValueRegex="" Translatable="1">Value</Item>
    </Value>
</Setting>
```

Az opcionális `Translatable` attribútum jelöli meg ezt a beállítást fordíthatóként, amely azt fogja eredményezni, hogy fel lesz véve az OTRS fordítási fájljaiba. Ezt az attribútumot bármely címkére el lehet helyezni (lásd lent is).

#### Password

Egy beállítási elem jelszavakhoz. Továbbra is egyszerű szövegként van tárolva az XML-ben, de el van fedve a grafikus felhasználói felületen.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Password">Secret</Item>
    </Value>
</Setting>
```

#### PerlModule

Egy beállítási elem Perl-modulhoz. Rendelkezik egy `ValueFilter` attribútummal, amely szűri a lehetséges értékeket a kiválasztásnál. A lenti példában a felhasználó a `Kernel::System::Log::SysLog` vagy a `Kernel::System::Log::File` Perl-modult választhatja.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="PerlModule" ValueFilter="Kernel/System/Log/*.pm">
↪Kernel::System::Log::SysLog</Item>
    </Value>
</Setting>
```

#### Textarea

Egy beállítási elem többsoros szöveghez.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Textarea"></Item>
    </Value>
</Setting>
```

#### Select

Ez a beállítási elem előre beállított értékeket nyújt egy legördülő menüként. A `SelectedID` vagy a `SelectedValue` attribútumok előre kiválaszthatnak egy alapértelmezett értéket.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Select" SelectedID="Queue">
            <Item ValueType="Option" Value="Queue" Translatable="1">Queue</
↪Item>
            <Item ValueType="Option" Value="SystemAddress" Translatable="1">
↪System address</Item>
        </Item>
    </Value>
</Setting>
```

#### Checkbox

Ennek a jelölőnégyzet beállítási elemnek két állapota van: 0 vagy 1.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Checkbox">0</Item>
    </Value>
</Setting>
```

#### Date

Ez a beállítási elem egy dátumértéket tartalmaz.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Date">2016-02-02</Item>
    </Value>
</Setting>
```

#### DateTime

Ez a beállítási elem egy dátumot és egy időértéket tartalmaz.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="DateTime">2016-12-08 01:02:03</Item>
    </Value>
</Setting>
```

#### Directory

Ez a beállítási elem egy könyvtárat tartalmaz.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Directory">/etc</Item>
    </Value>
</Setting>
```

#### File

Ez a beállítási elem egy fájlútvonalat tartalmaz.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="File">/etc/hosts</Item>
    </Value>
</Setting>
```

#### Entity

Ez a beállítási elem egy bizonyos entitás értékét tartalmazza. A `ValueEntityType` attribútum határozza meg az entitás típusát. Támogatott entitások: `DynamicField`, `Queue`, `Priority`, `State` és `Type`. A következetesség-ellenőrzések fogják biztosítani, hogy csak érvényes entitások legyenek beállíthatók, és hogy azok az entitások, amelyeket a beállításokban használnak, ne legyenek érvénytelenre állíthatók. Amikor egy entitást átneveznek, akkor az összes hivatkozó konfigurációs beállítás frissítve lesz.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Entity" ValueEntityType="Queue">Junk</Item>
    </Value>
</Setting>
```

#### TimeZone

Ez a beállítási elem egy időzóna értéket tartalmaz.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="TimeZone">UTC</Item>
    </Value>
</Setting>
```

#### VacationDays

Ez a beállítási elem meghatározásokat tartalmaz az olyan munkaszüneti napokhoz, amelyek minden évben ismétlődnek. A következő attribútumok kötelezőek: `ValueMonth`, `ValueDay`.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="VacationDays">
            <DefaultItem ValueType="VacationDays"></DefaultItem>
            <Item ValueMonth="1" ValueDay="1" Translatable="1">New Year's Day
↪</Item>
            <Item ValueMonth="5" ValueDay="1" Translatable="1">International␣
↪Workers' Day</Item>
            <Item ValueMonth="12" ValueDay="24" Translatable="1">Christmas Eve
↪</Item>
```

(continues on next page)

(folytatás az előző oldalról)

```
        </Item>
    </Value>
</Setting>
```

#### VacationDaysOneTime

Ez a beállítási elem meghatározásokat tartalmaz az olyan munkaszüneti napokhoz, amelyek csak egyszer fordulnak elő. A következő attribútumok kötelezőek: `ValueMonth`, `ValueDay` és `ValueYear`.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="VacationDaysOneTime">
            <Item ValueYear="2004" ValueMonth="1" ValueDay="1">test</Item>
        </Item>
    </Value>
</Setting>
```

#### WorkingHours

Ez a beállítási elem meghatározásokat tartalmaz a munkaidőhöz.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="WorkingHours">
            <Item ValueType="Day" ValueName="Mon">
                <Item ValueType="Hour">8</Item>
                <Item ValueType="Hour">9</Item>
            </Item>
            <Item ValueType="Day" ValueName="Tue">
                <Item ValueType="Hour">8</Item>
                <Item ValueType="Hour">9</Item>
            </Item>
        </Item>
    </Value>
</Setting>
```

### Előtétprogram regisztráció

Modul regisztráció az ügyintézői felülethez:

```
<Setting Name="SettiFrontend::Module###AgentModuleName">
    ...
    <Value>
        <Item ValueType="FrontendRegistration">
            <Hash>
                <Item Key="Group">
```

(continues on next page)

(folytatás az előző oldalról)

```
                    <Array>
                    </Array>
                </Item>
                <Item Key="GroupRo">
                    <Array>
                    </Array>
                </Item>
                <Item Key="Description" Translatable="1">Phone Call.</Item>
                <Item Key="Title" Translatable="1">Phone-Ticket</Item>
                <Item Key="NavBarName">Ticket</Item>
            </Hash>
        </Item>
    </Value>
</Setting>
```

### Alapértelmezett elem tömbben és kivonatban

Az új XML szerkezet lehetővé teszi számunkra az összetett szerkezetek létrehozását. Ennélfogva szükségünk van alapértelmezett `DefaultItem` bejegyzésekre a tömb vagy kivonat szerkezetének leírásához. Ha ez nincs megadva, akkor a rendszer úgy tekinti, hogy egyszerű tömböt vagy kivonatot szeretne skalár értékekkel. A `DefaultItem` használható sablonként, amikor új elemeket adunk hozzá, így tartalmazhat további attribútumokat, mint például a `ValueType`, és meghatározhat alapértelmezett értékeket.

Itt van néhány példa:

#### Tömb tömbje `Select` elemekkel

```
<Array>
    <DefaultItem>
        <Array>
            <DefaultItem ValueType="Select" SelectedID='option-2'>
                <Item ValueType="Option" Value="option-1">Option 1</Item>
                <Item ValueType="Option" Value="option-2">Option 2</Item>
            </DefaultItem>
        </Array>
    </DefaultItem>
    ...
</Array>
```

#### Kivonat kivonata `Date` elemekkel

```
<Hash>
    <DefaultItem>
        <Hash>
            <DefaultItem ValueType="Date">2017-01-01</DefaultItem>
        </Hash>
    </DefaultItem>
    ...
</Hash>
```

### 2.1.4 Hozzáférés a beállítási lehetőségekhez futási időben

Olvashatja és írhatja (egy kérésnél) a beállítási lehetőségeket a `Kernel::Config` alapmodulon keresztül.

Ha egy beállítási lehetőséget szeretne olvasni:

```
my $ConfigOption = $Kernel::OM->Get('Kernel::Config')->Get('Prefix::Option');
```

Ha meg szeretne változtatni egy beállítási lehetőséget futási időben, és csak ennél az egy kérésnél/folyamatnál:

```
$Kernel::OM->Get('Kernel::Config')->Set(
    Key => 'Prefix::Option'
    Value => 'SomeNewValue',
);
```

## 2.2 Adatbázis mechanizmus

Az OTRS olyan adatbázis réteggel érkezik, amely különböző adatbázisokat támogat.

Az adatbázis rétegnek (`Kernel::System::DB`) két bemeneti lehetősége van: *SQL* és *XML*.

### 2.2.1 SQL

Az SQL felületet kell használni a normál adatbázis-műveleteknél (`SELECT`, `INSERT`, `UPDATE`, stb.). Úgy használható mint egy normál Perl DBI felület.

#### INSERT/UPDATE/DELETE utasítások

```
$Kernel::OM->Get('Kernel::System::DB')->Do(
    SQL=> "INSERT INTO table (name, id) VALUES ('SomeName', 123)",
);

$Kernel::OM->Get('Kernel::System::DB')->Do(
    SQL=> "UPDATE table SET name = 'SomeName', id = 123",
);

$Kernel::OM->Get('Kernel::System::DB')->Do(
    SQL=> "DELETE FROM table WHERE id = 123",
);
```

#### SELECT utasítás

```
my $SQL = "SELECT id FROM table WHERE tn = '123'";

$Kernel::OM->Get('Kernel::System::DB')->Prepare(SQL => $SQL, Limit => 15);

while (my @Row = $Kernel::OM->Get('Kernel::System::DB')->FetchrowArray()) {
    $Id = $Row[0];
```

(continues on next page)

(folytatás az előző oldalról)

```
}
return $Id;
```

**Megjegyzés:** Vigyázzon arra, hogy a `Limit` megadását paraméterként használja, és ne az SQL karakterláncban, mert nem minden adatbázis támogatja a `LIMIT` kulcsszót az SQL karakterláncokban.

```
my $SQL = "SELECT id FROM table WHERE tn = ? AND group = ?";

$Kernel::OM->Get('Kernel::System::DB')->Prepare(
    SQL   => $SQL,
    Limit => 15,
    Bind  => [ $Tn, $Group ],
);

while (my @Row = $Kernel::OM->Get('Kernel::System::DB')->FetchrowArray()) {
    $Id = $Row[0];
}
return $Id;
```

**Megjegyzés:** Használja a `Bind` attribútumot, ahol csak tudja, különösen a hosszú utasításoknál. Ha a `Bind` attribútumot használja, akkor nincs szükség a `Quote()` függvényre.

#### QUOTE

Szöveg:

```
my $QuotedString = $Kernel::OM->Get('Kernel::System::DB')->Quote("It's a␣
↪problem!");
```

Egész:

```
my $QuotedInteger = $Kernel::OM->Get('Kernel::System::DB')->Quote('123',
↪'Integer');
```

Szám:

```
my $QuotedNumber = $Kernel::OM->Get('Kernel::System::DB')->Quote('21.35',
↪'Number');
```

**Megjegyzés:** A `Bind` attribútumot használja a `Quote()` függvény helyett, ahol csak tudja.

### 2.2.2 XML

Az XML felületet kell használni `INSERT`, `CREATE TABLE`, `DROP TABLE` és `ALTER TABLE` utasításoknál. Mivel ez a szintaxis adatbázisról adatbázisra eltérő, ezért ennek használata gondoskodik arról, hogy olyan alkalmazásokat írjon, amelyek az összesnél használhatók.

### INSERT

```
<Insert Table="some_table">
    <Data Key="id">1</Data>
    <Data Key="description" Type="Quote">exploit</Data>
</Insert>
```

### CREATE TABLE

A lehetséges adattípusok a következők: `BIGINT`, `SMALLINT`, `INTEGER`, `VARCHAR` (méret = 1-1000000), `DATE` (formátum: yyyy-mm-dd hh:mm:ss) és `LONGBLOB`.

```
<TableCreate Name="calendar_event">
    <Column Name="id" Required="true" PrimaryKey="true" AutoIncrement="true"␣
↪Type="BIGINT"/>
    <Column Name="title" Required="true" Size="250" Type="VARCHAR"/>
    <Column Name="content" Required="false" Size="250" Type="VARCHAR"/>
    <Column Name="start_time" Required="true" Type="DATE"/>
    <Column Name="end_time" Required="true" Type="DATE"/>
    <Column Name="owner_id" Required="true" Type="INTEGER"/>
    <Column Name="event_status" Required="true" Size="50" Type="VARCHAR"/>
    <Index Name="calendar_event_title">
        <IndexColumn Name="title"/>
    </Index>
    <Unique Name="calendar_event_title">
        <UniqueColumn Name="title"/>
    </Unique>
    <ForeignKey ForeignTable="users">
        <Reference Local="owner_id" Foreign="id"/>
    </ForeignKey>
</TableCreate>
```

A `LONGBLOB` oszlopok különleges bánásmódot igényelnek. A tartalmukat base64-re kell átkódolni, ha az adatbázis-meghajtó nem támogatja a `DirectBlob` funkciót. Nézze meg a következő példát:

```
my $Content = $StorableContent;
if ( !$DBObject->GetDatabaseFunction('DirectBlob') ) {
    $Content = MIME::Base64::encode_base64($StorableContent);
}
```

Hasonlóan, amikor egy ilyen oszlopból olvas, akkor a tartalmat tilos automatikusan UTF-8-ként visszaalakítani az `Encode => 0` jelző átadásával a `Prepare()` függvénynek:

```
return if !$DBObject->Prepare(
    SQL => '
        SELECT content_type, content, content_id, content_alternative,␣
↪disposition, filename
        FROM article_data_mime_attachment
        WHERE id = ?',
    Bind   => [ \$AttachmentID ],
    Encode => [ 1, 0, 0, 0, 1, 1 ],
);
```

(continues on next page)

(folytatás az előző oldalról)

```
while ( my @Row = $DBObject->FetchrowArray() ) {

    $Data{ContentType} = $Row[0];

    # Decode attachment if it's e. g. a postgresql backend.
    if ( !$DBObject->GetDatabaseFunction('DirectBlob') ) {
        $Data{Content} = decode_base64( $Row[1] );
    }
    else {
        $Data{Content} = $Row[1];
    }
    $Data{ContentID}          = $Row[2] || '';
    $Data{ContentAlternative} = $Row[3] || '';
    $Data{Disposition}        = $Row[4];
    $Data{Filename}           = $Row[5];
}
```

### DROP TABLE

```
<TableDrop Name="calendar_event"/>
```

### ALTER TABLE

A következő az oszlopok hozzáadásának, megváltoztatásának és eldobásának példáját jeleníti meg.

```
<TableAlter Name="calendar_event">
    <ColumnAdd Name="test_name" Type="varchar" Size="20" Required="true"/>

    <ColumnChange NameOld="test_name" NameNew="test_title" Type="varchar"␣
↪Size="30" Required="true"/>

    <ColumnChange NameOld="test_title" NameNew="test_title" Type="varchar"␣
↪Size="100" Required="false"/>

    <ColumnDrop Name="test_title"/>

    <IndexCreate Name="index_test3">
        <IndexColumn Name="test3"/>
    </IndexCreate>

    <IndexDrop Name="index_test3"/>

    <UniqueCreate Name="uniq_test3">
        <UniqueColumn Name="test3"/>
    </UniqueCreate>

    <UniqueDrop Name="uniq_test3"/>
</TableAlter>
```

A következő egy olyan példát jelenít meg, hogy hogyan nevezhető át egy tábla.

```
<TableAlter NameOld="calendar_event" NameNew="calendar_event_new"/>
```

#### Kód az XML feldolgozásához

```
my @XMLARRAY = @{$Self->ParseXML(String => $XML)};

my @SQL = $Kernel::OM->Get('Kernel::System::DB')->SQLProcessor(
    Database => \@XMLARRAY,
);
push(@SQL, $Kernel::OM->Get('Kernel::System::DB')->SQLProcessorPost());

for (@SQL) {
    $Kernel::OM->Get('Kernel::System::DB')->Do(SQL => $_);
}
```

### 2.2.3 Adatbázis-meghajtók

Az adatbázis-meghajtók az `$OTRS_HOME/Kernel/System/DB/*.pm` alatt találhatók.

### 2.2.4 Támogatott adatbázisok

- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server (csak külső adatbázis-kapcsolatokhoz, nem OTRS adatbázisként)

## 2.3 Naplózó mechanizmus

### 2.3.1 Rendszernapló

Az OTRS egy olyan rendszernaplózó háttérprogrammal érkezik, amely használható az alkalmazás naplózásánál és a hibakeresésnél.

A `Log` objektum az objektumkezelőn keresztül érhető el és használható ehhez hasonlóan:

```
$Kernel::OM->Get('Kernel::System::Log')->Log(
    Priority => 'error',
    Message  => 'Need something!',
);
```

A rendszerbeállításokban a `MinimumLogLevel` beállításon keresztül beállított naplózási szinttől függően a naplózott üzenetek a `Priority` jelzőjük alapján mentésre kerülnek vagy sem.

Ha `error` van beállítva, akkor csak a hibák lesznek naplózva. A `debug` értékkel megkapja az összes naplózási üzenetet. A naplózási szintek sorrendje a következő:

- `debug`

- info
- notice
- error

A rendszernapló kimenete a rendszerbeállításokban lévő LogModule beállításban megadottak alapján átirányítható egy rendszernaplózó démonba vagy naplófájlba.

### 2.3.2 Kommunikációs napló

A rendszernapló mellett az OTRS különleges naplózó háttérprogramot biztosít az összes, a kommunikációra vonatkozó naplózáshoz. Az OTRS 6 óta a rendszer dedikált táblázatokkal és előtétprogramokkal érkezik a kommunikációs naplók követéséhez és megjelenítéséhez, hogy egyszerűbb legyen hibát keresni és a működést áttekinteni.

Az új rendszer előnyeinek kihasználásához először hozzon létre egy nem egyke példányt a kommunikációs napló objektumból:

```
my $CommunicationLogObject = $Kernel::OM->Create(
    'Kernel::System::CommunicationLog',
    ObjectParams => {
        Transport   => 'Email',      # Transport log module
        Direction   => 'Incoming',   # Incoming|Outgoing
        AccountType => 'POP3',       # Mail account type
        AccountID   => 1,            # Mail account ID
    },
);
```

Amikor megvan a kommunikációs napló objektum példánya, akkor elindíthat egy objektumnaplózást az egyéni üzenetek naplózásához. Jelenleg két objektumnapló van megvalósítva: Connection és Message.

A Connection objektumnaplót kell használni minden, a kommunikációra vonatkozó üzenet naplózásához (például hitelesítés egy kiszolgálón vagy bejövő üzenetek fogadása).

Egyszerűen indítsa el az objektumnaplót a típusának meghatározásával, és már azonnal használhatja:

```
$CommunicationLogObject->ObjectLogStart(
    ObjectLogType => 'Connection',
);

$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Connection',
    Priority      => 'Debug',                                # Trace, Debug,␣
↪Info, Notice, Warning or Error
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => "Open connection to 'host.example.com' (user-1).",
);
```

A kommunikációs napló objektum példánya kezeli a jelenleg elindított objektumnaplókat, így nem kell megjegyeznie és előhoznia mindenhol, de ez azt is jelenti, hogy csak egyetlen objektumot indíthat típusonként.

Ha javíthatatlan hibával találkozik, akkor választhatja az objektumnapló bezárását és sikertelenként való megjelölését:

```
$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Connection',
    Priority      => 'Error',
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => 'Something went wrong!',
);

$CommunicationLogObject->ObjectLogStop(
    ObjectLogType => 'Connection',
    Status        => 'Failed',
);
```

Viszont megjelölheti a kommunikációs naplót sikertelennek is:

```
$CommunicationLogObject->CommunicationStop(
    Status => 'Failed',
);
```

Egyébként állítsa le az objektumnaplót és jelölje a kommunikációs naplót sikeresként:

```
$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Connection',
    Priority      => 'Debug',
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => "Connection to 'host.example.com' closed.",
);

$CommunicationLogObject->ObjectLogStop(
    ObjectLogType => 'Connection',
    Status        => 'Successful',
);

$CommunicationLogObject->CommunicationStop(
    Status => 'Successful',
);
```

A `Message` objektumnaplót kell használni minden olyan naplóbejegyzésnél, amelyek bizonyos üzenetek és azok feldolgozását érintik. Hasonló módon használható, egyszerűen győződjön meg arról, hogy elindította-e a használta előtt:

```
$CommunicationLogObject->ObjectLogStart(
    ObjectLogType => 'Message',
);

$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Message',
    Priority      => 'Error',
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => "Could not process message. Raw mail saved (report it on␣
↪http://bugs.otrs.org/)!",
);

$CommunicationLogObject->ObjectLogStop(
```

(continues on next page)

(folytatás az előző oldalról)

```
    ObjectLogType => 'Message',
    Status        => 'Failed',
);

$CommunicationLogObject->CommunicationStop(
    Status => 'Failed',
);
```

Lehetősége van hivatkozni a naplóobjektumra és később megkeresni a kommunikációkat egy bizonyos objektumtípus vagy azonosító esetén:

```
$CommunicationLogObject->ObjectLookupSet(
    ObjectLogType    => 'Message',
    TargetObjectType => 'Article',
    TargetObjectID   => 2,
);

my $LookupInfo = $CommunicationLogObject->ObjectLookupGet(
    TargetObjectType => 'Article',
    TargetObjectID   => 2,
);
```

Meg kell győződnie arról, hogy mindig leállította-e a kommunikációt és megjelölte-e sikertelenként, ha valamelyik naplóobjektum sikertelen lenne. Ez lehetővé fogja tenni az adminisztrátorok számára, hogy megtekintsék a sikertelen kommunikációkat az áttekintőben, és megtegyék a szükséges lépéseket.

Fontos megőrizni a kommunikációs naplót egyetlen folyamat időtartamára. Ha munkája több modult is érint, és bármelyikük kihasználhatja a naplózás előnyeit, akkor győződjön meg arról, hogy átadta-e a létező kommunikációs napló példányát, így az összes metódus ugyanazt tudja használni. Ezzel a megközelítéssel biztosíthatja, hogy az azonos folyamathoz tartozó naplóbejegyzések egyetlen kommunikációban legyenek tárolva.

Ha a kommunikációs napló példányának átadása nem lehetséges (aszinkron feladatok!), akkor választhatja a kommunikációs napló objektum újra létrehozását ugyanabban az állapotban, amiben az előző lépésben volt. Egyszerűen fogja a kommunikációs azonosítót, és adja át az új kódnak, majd hozza létre az új példányt ezzel a megadott paraméterrel:

```
# Get communication ID in parent code.
my $CommunicationID = $CommunicationLogObject->CommunicationIDGet();

# Somehow pass communication ID to child code.
# ...

# Recreate the instance in child code by using same communication ID.
my $CommunicationLogObject = $Kernel::OM->Create(
    'Kernel::System::CommunicationLog',
    ObjectParams => {
        CommunicationID => $CommunicationID,
    },
);
```

Ezután folytathatja ennek a példánynak a használatát, amint azt korábban említettük. Indítson el valamilyen objektumnaplót, ha szükséges, adjon hozzá bejegyzéseket és állítsa be az állapotot a végén.

Ha a kommunikációs napló adatainak lekérésére van szüksége vagy valami mást szeretne csinálni vele,

akkor vessen egy pillantást a `Kernel::System::CommunicationLog::DB.pm` fájlra.

## 2.4 Dátum és idő

Az OTRS saját csomaggal érkezik a dátum és idő kezeléséhez, amely a dátum és idő helyes kiszámítását és tárolását biztosítja.

### 2.4.1 Bevezetés

A dátum és idő egy `Kernel::System::DateTime` objektummal van ábrázolva. Minden `DateTime` objektum tartalmazza a saját dátumát, idejét és időzóna információját. A mostanra elavult `Kernel::System::Time` csomaggal szemben ez azt jelenti, hogy létrehozhat és létre kell hoznia egy `DateTime` objektumot minden egyes dátumhoz vagy időhöz, amelyet használni szeretne.

### 2.4.2 Egy `DateTime` objektum létrehozása

Az OTRS objektumkezelője kibővítésre került egy `Create` metódussal, hogy támogassa azokat a csomagokat, amelyekhez egynél több példány hozható létre:

```
my $DateTimeObject = $Kernel::OM->Create(
    'Kernel::System::DateTime',
    ObjectParams => {
        TimeZone => 'Europe/Berlin'
    },
);
```

A fenti példa létrehoz egy `DateTime` objektumot a jelenlegi dátumhoz és időhöz az *Európa/Berlin* időzónában. További lehetőségek is vannak egy `DateTime` objektum létrehozására (időösszetevők, szövegek, időbélyegek, klónozás), nézze meg a `Kernel::System::DateTime` POD-ját.

---

**Megjegyzés:** Hibát fog kapni, ha megpróbál egy `DateTime` objektumot lekérni a `$Kernel::OM->Get('Kernel::System::DateTime')` használatával.

---

### 2.4.3 Időzónák

Az órákban lévő időeltolások (+2, -10, stb.) le lettek cserélve az időzónákkal (Európa/Berlin, Amerika/New_York, stb.). Az időzónák közti átalakítások teljes mértékben a `DateTime` objektumon belül vannak megvalósítva. Ha egy másik időzónára szeretne átalakítani, akkor egyszerűen használja a következő kódot:

```
$DateTimeObject->ToTimeZone( TimeZone => 'Europe/Berlin' );
```

Van egy új `OTRSTimeZone` rendszerbeállítási lehetőség. Ez a beállítás határozza meg azt az időzónát, amelyet az OTRS belsőleg használ a dátum és az idő tárolásához az adatbázison belül.

---

**Megjegyzés:** Biztosítania kell, hogy egy `DateTime` objektum át legyen alakítva az OTRS időzónájára, mielőtt az eltárolásra kerülne az adatbázisba (van egy kényelmes módszer erre: `ToOTRSTimeZone()`). Kivétel lehet, hogy ha kifejezetten egy olyan adatbázisoszlopot szeretne, amely egy dátum/idő értéket tárol

egy bizonyos időzónában. Ne feledje azonban, hogy maga az adatbázis önmagában nem fog időzóna-információkat biztosítani, amikor lekérik azt.

**Megjegyzés:** A `Kernel::System::DateTime TimeZoneList()` metódusa biztosítja az elérhető időzónák listáját.

### 2.4.4 Metódus összefoglaló

A `Kernel::System::DateTime` csomag biztosítja a következő metódusokat (ez csak egy kiválasztás, a részletekért nézze meg a forráskódot).

#### Objektumlétrehozási metódusok

Egy `DateTime` objektum létrehozható az objektumkezelő `Create()` metódusával vagy egy másik `DateTime` objektum leklónozásával a `Clone()` metódusának használatával.

#### Lekérő metódus

A `Get()` metódussal egy `DateTime` objektum összes adata vissza lesz adva kivonatként (dátum és idő összetevők, beleértve a nap nevét, stb., valamint az időzónát).

#### Beállító metódus

A `Set()` metódussal megváltoztathatja a `DateTime` objektum bizonyos összetevőit (év, hónap, nap, óra, perc, másodperc) vagy beállíthat egy dátumot és időt egy adott szöveg alapján (*2016-05-24 23:04:12*). Ne feledje, hogy nem tudja megváltoztatni az időzónát ezzel a metódussal.

#### Időzóna metódusok

Egy `DateTime` objektum időzónájának megváltoztatásához használja a `ToTimeZone()` metódust vagy egyszerűsítésként a `ToOTRSTimeZone()` metódust az OTRS időzóna átalakításához.

A beállított OTRS időzóna vagy a felhasználó alapértelmezett időzónájának lekéréséhez mindig az `OTRSTimeZoneGet()` vagy a `UserDefaultTimeZoneGet()` metódusokat használja. Sose kérje le ezeket kézzel a `Kernel::Config` használatával.

#### Összehasonlító operátorok és metódusok

A `Kernel::System::DateTime` operátortúlterhelést használ az összehasonlításhoz. Így egyszerűen összehasonlíthat két `DateTime` objektumot a <, <=, ==, !=, >= és > operátorokkal. A `Compare()` metódus használható a Perl rendezési környezetében, mivel -1, 0 vagy 1 a visszatérési értéke.

### 2.4.5 Elavult `Kernel::System::Time` csomag

A mostanra elavult `Kernel::System::Time` csomag ki lett bővítve, hogy teljesen támogassa az időzónákat az időeltolások helyett. Ez azért történt, hogy biztosítsa a meglévő kód működését (nagyobb) módosítások nélkül.

Azonban van egy eset, amikor meg kell változtatnia a meglévő kódot. Ha olyan kódja van, amely a régi időeltolásokat használja egy új dátum és idő kiszámításához vagy különbségéhez, akkor át kell költöztetnie ezt a kódot, hogy az új `DateTime` objektumot használja.

Példa (régi kód):

```
my $TimeObject     = $Kernel::OM->Get('Kernel::System::Time'); # Assume a␣
↪time offset of 0 for this time object
my $SystemTime     = $TimeObject->TimeStamp2SystemTime( String => '2004-08-14␣
↪22:45:00' );
my $UserTimeZone   = '+2'; # normally retrieved via config or param
my $UserSystemTime = $SystemTime + $UserTimeZone * 3600;
my $UserTimeStamp  = $TimeObject->SystemTime2TimeStamp( SystemTime =>
↪$UserSystemTime );
```

Példa (új kód):

```
my $DateTimeObject = $Kernel::OM->Create('Kernel::System::DateTime'); # This␣
↪implicitly sets the configured OTRS time zone
my $UserTimeZone   = 'Europe/Berlin'; # normally retrieved via config or param
$DateTimeObject->ToTimeZone( TimeZone => $UserTimeZone );
my $SystemTime    = $DateTimeObject->ToEpoch(); # note that the epoch is␣
↪independent from the time zone, it's always calculated for UTC
my $UserTimeStamp = $DateTimeObject->ToString();
```

## 2.5 Felszínek

Az OTRS látható megjelenése *felszínekkel* szabályozható.

Egy felszín CSS-fájlok és képfájlok halmaza, amelyek együtt azt vezérlik, hogy a grafikus felhasználói felület hogyan jelenjen meg a felhasználónak. A felszínek nem változtatják meg az OTRS által előállított HTML tartalmát (ez az, amit a *témák* csinálnak), hanem azt szabályozzák, hogy az hogyan jelenjen meg. A modern CSS szabványok segítségével lehetőség van a megjelenítés teljes megváltoztatására (például párbeszédablakok egyes részeinek áthelyezésére, elemek elrejtésére, stb.).

### 2.5.1 Felszín alapok

Az összes felszín az `$OTRS_HOME/var/httpd/htdocs/skins/Agent/$SKIN_NAME` mappában van.

Az ügyintézők mindegyike egyénileg választhatja ki, hogy melyik telepített ügyintézői felszínt szeretnék *viselni*.

---

**Megjegyzés:** A felszíntámogatás az ügyfélfelületnél el lett dobva az új külső felülettel. A külső felület személyre szabott elrendezésének létrehozásához használja az adminisztrációs területen lévő *Megjelenés* modult.

---

### 2.5.2 Hogyan töltődnek be a felszínek

Fontos megjegyezni, hogy **mindig** a `default` felszín fog **először** betöltődni. Ha az ügyintéző egy másik felszínt választott az „alapértelmezett” helyett, akkor a másik felszín csak az alapértelmezett felszín **után** lesz betöltve. A betöltésen itt azt értjük, hogy az OTRS olyan címkéket fog elhelyezni a HTML tartalmában, amelyek a CSS-fájlok betöltését idézik elő a böngészőnél. Nézzünk egy példát erre:

```
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css-cache/
↪CommonCSS_179376764084443c181048401ae0e2ad.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/ivory/css-cache/CommonCSS_
↪e0783e0c2445ad9cc59c35d6e4629684.css" />
```

Itt tisztán látható, hogy a `default` felszín töltődik be először, majd ezután az ügyintéző által megadott egyéni felszín. Ebben a példában a bekapcsolt betöltő (a `Loader::Enabled` 1-re állítva) eredményét látjuk, amely begyűjti az összes CSS-fájlt, összefűzi és minimalizálja azokat, majd egyetlen nagy egységként szolgálja ki a böngészőnek. Ezzel sávszélességet spórol, és csökkenti a HTTP-kérések számát is. Nézzük meg ugyanezt a példát kikapcsolt betöltővel:

```
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Reset.css
↪" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Default.
↪css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Header.css
↪" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.
↪OverviewControl.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.
↪OverviewSmall.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.
↪OverviewMedium.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.
↪OverviewLarge.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Footer.css
↪" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Grid.css"␣
↪/>
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Form.css"␣
↪/>
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Table.css
↪" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Widget.css
↪" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.
↪WidgetMenu.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.
↪TicketDetail.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Tooltip.
↪css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Dialog.css
↪" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Print.css
↪" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Agent.
↪CustomerUser.GoogleMaps.css" />
```

(continues on next page)

(folytatás az előző oldalról)

```
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Agent.
↪CustomerUser.OpenTicket.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/ivory/css/Core.Dialog.css"␣
↪/>
```

Itt jobban láthatjuk az egyes fájlokat, amelyek a felszínekből jönnek.

Különböző típusú CSS-fájlok vannak: közös fájlok, amelyeket mindig be kell tölteni, és modul specifikus fájlok, amelyek csak az OTRS keretrendszeren belüli speciális moduloknál vannak betöltve.

Továbbá lehetséges olyan CSS-fájlok megadása, amelyeket csak IE7 vagy IE8 böngészőknél kell betölteni (az ügyfélfelület esetén IE6 böngészőnél is). Ez szerencsétlen ugyan, de ezeknél a böngészőknél nem volt lehetséges egy modern grafikus felhasználói felület kifejlesztése a hozzájuk elkészített speciális CSS nélkül.

A CSS-fájltípusokra vonatkozó részletekért nézze meg a *A CSS és JavaScript betöltő* szakaszt.

Minden egyes HTML-oldal előállításához a betöltő először a `default` felszínből fogja az összes beállított CSS-fájlt venni, és ezután az egyes fájlok kinézetéhez, ha az egy egyéni felszínben is elérhető (ha egy egyéni felszín ki lett választva), majd betölti azokat az alapértelmezett fájlok után.

Ez azt jelenti, hogy az egyéni felszínekben lévő CSS-fájloknak ugyanolyan nevűeknek kell lenniük mint az alapértelmezett felszínekben, és hogy egy egyéni felszínnek nem kell az alapértelmezett felszín összes fájljával rendelkeznie. Ez a nagy előnye az alapértelmezett felszín elsőként való betöltésének: egy egyéni felszínben az összes alapértelmezett CSS-szabály jelen van, és csak azokat szükséges megváltoztatni, amelyeknek eltérő megjelenítést kell eredményezniük. Ez gyakran egyetlen fájllal elvégezhető, mint a fenti példában látható.

Ennek másik hatása, hogy figyelmesnek kell lennie az egyéni felszínekben lévő összes olyan alapértelmezett CSS-szabály felülírásánál, amelyeken változtatni szeretne. Nézzünk egy példát:

```
.Header h1 {
    font-weight: bold;
    color: #000;
}
```

Ez speciális címsorokat határoz meg a `.Header` elemen belül félkövér, fekete szöveggel. Ha most azt szeretné megváltoztatni, hogy a felszínben más színnel és normál szöveggel jelenjen meg, akkor nem elég ezt írni:

```
.Header h1 {
    color: #F00;
}
```

Ugyanis az eredeti `font-weight` szabály még mindig alkalmazva lesz. Határozottan felül kell írnia:

```
.Header h1 {
    font-weight: normal;
    color: #F00;
}
```

### 2.5.3 Új felszín létrehozása

Ebben a szakaszban egy új ügyintézői felszínt fogunk létrehozni, amely lecseréli az alapértelmezett (fehér) OTRS háttérszínt egy egyéni (világos szürke) vállalati színre és az alapértelmezett logót egy egyénire. Azt is be fogjuk állítani, hogy ez a felszín legyen az, amelyet az összes ügyintéző alapértelmezetten látni fog.

Csak három egyszerű lépést kell megtennünk a cél eléréséhez:

- a felszínfájlok létrehozását
- az új logó beállítását és
- a felszín megismertetését az OTRS rendszerrel

Kezdjük az új felszínünkhöz szükséges fájlok létrehozásával. Először is létre kell hoznunk egy új mappát ehhez a felszínhez (ezt `custom` néven fogjuk hívni). Ez a mappa a következő lesz: `$OTRS_HOME/var/httpd/htdocs/skins/Agent/custom`.

Ebben el kell helyeznünk az új CSS-fájlt egy új `css` könyvtárban, amely az új felszín megjelenését fogja meghatározni. Ezt `Core.Default.css` néven fogjuk hívni (emlékezzen arra, hogy ugyanolyan névvel kell rendelkeznie mint az alapértelmezett felszínben lévő fájlok egyike). Ez a CSS-fájlhoz szükséges kód:

```
body {
    background-color: #c0c0c0; /* not very beautiful but it meets our purpose␣
↪*/
}
```

Most következik a második lépés egy új logó hozzáadásával, és az új felszín megismertetésével az OTRS rendszer számára. Ehhez először el kell helyeznünk az egyéni logónkat (például `logo.png`) egy új könyvtárban (például `img`) a saját felszín könyvtárunkban. Ezután létre kell hoznunk egy új `$OTRS_HOME/Kernel/Config/Files/XML/CustomSkin.xml` beállítófájlt, amely tartalmazni fogja a szükséges beállításokat az alábbiak szerint:

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="1.0" init="Changes">
    <ConfigItem Name="AgentLogo" Required="0" Valid="1">
        <Description Translatable="1">The logo shown in the header of the␣
↪agent interface. The URL to the image must be a relative URL to the skin␣
↪image directory.</Description>
        <Group>Framework</Group>
        <SubGroup>Frontend::Agent</SubGroup>
        <Setting>
            <Hash>
                <Item Key="URL">skins/Agent/custom/img/logo.png</Item>
                <Item Key="StyleTop">13px</Item>
                <Item Key="StyleRight">75px</Item>
                <Item Key="StyleHeight">67px</Item>
                <Item Key="StyleWidth">244px</Item>
            </Hash>
        </Setting>
    </ConfigItem>
    <ConfigItem Name="Loader::Agent::Skin###100-custom" Required="0" Valid="1">
        <Description Translatable="1">Custom skin for the development manual.
↪</Description>
        <Group>Framework</Group>
        <SubGroup>Frontend::Agent</SubGroup>
        <Setting>
            <Hash>
                <Item Key="InternalName">custom</Item>
                <Item Key="VisibleName">Custom</Item>
                <Item Key="Description">Custom skin for the development␣
↪manual.</Item>
```

(continues on next page)

(folytatás az előző oldalról)

```
                <Item Key="HomePage">www.yourcompany.com</Item>
            </Hash>
        </Setting>
    </ConfigItem>
</otrs_config>
```

A beállítás aktívvá tételéhez el kell navigálnunk az OTRS adminisztrációs területén lévő rendszerbeállítás modulra. Alternatív esetben lefuttathatja a következő parancsfájlt:

```
$OTRS_HOME/bin/otrs.Console.pl Maint::Config::Rebuild
```

Ez újra elő fogja állítani az XML beállítófájlok Perl gyorsítótárát azért, hogy az új felszínünk most már ismert legyen, és kiválasztható legyen a rendszeren. Ennek alapértelmezett felszínné tételéhez, amelyet az új ügyintézők azelőtt láthatnak, mielőtt a saját felszínválasztásukat megtennék, szerkessze a `Loader::Agent::DefaultSelectedSkin` rendszerbeállítási paramétert, és állítsa *custom* értékre.

Következtetésképpen: egy új felszín létrehozásához az OTRS-ben el kellett helyeznünk az új logófájlt, és létre kellett hoznunk egy CSS-fájlt és egy XML-fájlt, amely három új fájlt eredményezett:

```
$OTRS_HOME/Kernel/Config/Files/XML/CustomSkin.xml
$OTRS_HOME/var/httpd/htdocs/skins/Agent/custom/img/custom-logo.png
$OTRS_HOME/var/httpd/htdocs/skins/Agent/custom/css/Core.Header.css
```

## 2.6 A CSS és JavaScript betöltő

Az OTRS-ben lévő CSS és JavaScript kód hatalmas mennyiségre nőtt. Hogy képes legyen kielégíteni mind a fejlesztői szempontokat (jó karbantarthatóság különálló fájlok nagy mennyiségével), mind a teljesítmény problémákat (kevés HTTP-kérés megtétele és minimalizált tartalom kiszolgálása felesleges üres karakterek és dokumentáció nélkül), foglalkozni kellett ezzel. A célok eléréséhez kitalálták a betöltőt.

### 2.6.1 Hogyan működik

Leegyszerűsítve, a betöltő:

- minden egyes kérésnél pontosan meghatározza, hogy mely CSS és JavaScript fájlok szükségesek a kliens oldalhoz a jelenlegi alkalmazásmodulnál
- összegyűjti az összes ide vonatkozó adatot
- minimalizálja az adatokat a felesleges üres karakterek és dokumentáció eltávolításával
- kiszolgálja a kliens oldalnak mindössze néhány HTTP-kérésben a sok egyedüli helyett, lehetővé téve a kliensnek, hogy a gyorstárazza ezeket a töredékeket a böngésző gyorsítótárába
- végrehajtja ezeket a feladatokat egy jól teljesítő módon az OTRS gyorsítótárazó mechanizmusait használva

Természetesen van egy kicsivel részletesebb magyarázat is, de ennek elegendőnek kell lennie első áttekintésként.

### 2.6.2 Alapvető működés

A `Loader::Enabled::CSS` és a `Loader::Enabled::JavaScript` konfigurációs beállításokkal kapcsolható be és ki a betöltő a CSS-t és a JavaScriptet illetőleg (alapértelmezetten be van kapcsolva).

**Figyelem:** Az Internet Explorer böngészőben lévő megjelenítési problémák miatt a betöltőt nem lehet kikapcsolni a CSS-fájlokhoz ennél a kliens böngészőnél (a konfigurációs beállítás felül lesz bírálva). A 8-as verzióig az Internet Explorer nem tud 32 CSS-fájlnál többet kezelni egy oldalon.

Ha többet szeretne megtudni arról, hogy a betöltő hogyan működik, akkor kapcsolja ki az OTRS telepítésében a fent említett konfigurációs beállításokkal. Most nézze meg annak az alkalmazásmodulnak a forráskódját, amelyet jelenleg használ ezen az OTRS rendszeren (természetesen egy újratöltés után). Látni fogja, hogy számos CSS-fájl töltődött be az oldal `<head>` szakaszában, és sok JavaScript fájl van az oldal alján közvetlenül a lezáró `</body>` elem előtt.

Ehhez hasonlóan számos egyedülálló fájlban lévő olvashatóan formázott tartalom megléte sokkal egyszerűbbé teszi a fejlesztést, és akár egyáltalán lehetségessé téve azt. Azonban ennek megvan az a hátránya, hogy nagyszámú TTP-kérést (a hálózati késleltetésnek nagy hatása van) és felesleges tartalmakat (üres karaktereket és dokumentációt) szükséges átvinni a kliensnek.

A betöltő megoldja ezt a problémát a fenti rövid leírásban felvázolt lépések végrehajtásával. Kapcsolja be ismét a betöltőt, és most töltse újra az oldalt. Most azt láthatja, hogy csak nagyon kevés CSS és JavaScript címke van a HTML kódban ehhez hasonlóan:

```
<script type="text/javascript" src="/otrs30-dev-web/js/js-cache/CommonJS_
↪d16010491cbd4faaaeb740136a8ecbfd.js"></script>

<script type="text/javascript" src="/otrs30-dev-web/js/js-cache/ModuleJS_
↪b54ba9c085577ac48745f6849978907c.js"></script>
```

Mi történt most? Ennél az oldalnál a HTML kódot előállító eredeti kérés közben a betöltő előállította ezt a két fájlt (vagy kivette azokat a gyorsítótárból), és betette a látható `<script>` címkébe azon az oldalon, amely ezekhez a fájlokhoz van kapcsolva, ahelyett, hogy az összes ide vonatkozó JavaScript fájlt különállóan kapcsolta volna hozzá (amint a bekapcsolt betöltő nélkül láthatta).

A CSS szakasz egy kicsivel bonyolultabbnak tűnik:

```
    <link rel="stylesheet" type="text/css" href="/otrs30-dev-web/skins/Agent/
↪default/css-cache/CommonCSS_00753c78c9be7a634c70e914486bfbad.css" />

<!--[if IE 7]>
    <link rel="stylesheet" type="text/css" href="/otrs30-dev-web/skins/Agent/
↪default/css-cache/CommonCSS_IE7_59394a0516ce2e7359c255a06835d31f.css" />
<![endif]-->

<!--[if IE 8]>
    <link rel="stylesheet" type="text/css" href="/otrs30-dev-web/skins/Agent/
↪default/css-cache/CommonCSS_IE8_ff58bd010ef0169703062b6001b13ca9.css" />
<![endif]-->
```

Ennek az az oka, hogy az Internet Explorer 7 és 8 esetén speciális bánásmód szükséges az alapértelmezett CSS mellett a webes szabványtechnológiák hiányos támogatásuk miatt. Ezért van néhány normál CSS-fájlunk, amelyek minden böngészőben betöltődnek, és néhány speciális CSS-fájl az úgynevezett *feltételes*

*megjegyzéseken* belül, amely azt idézi elő, hogy **csak** az Internet Explorer 7/8 töltse be azokat. Az összes többi böngésző figyelmen kívül fogja hagyni.

Most felvázoltuk, hogy a betöltő hogyan működik. Nézzük meg, hogy hogyan hasznosíthatja azt a saját OTRS kiterjesztéseiben a betöltőhöz történő konfigurációs adatok hozzáadásával, azt mondva neki, hogy további vagy alternatív CSS vagy JavaScript tartalmat töltsön be.

### 2.6.3 A betöltő beállítása: JavaScript

Hogy képes legyen helyesen működni, a betöltőnek tudnia kell, hogy mely tartalmat kell betöltenie egy bizonyos OTRS alkalmazásmodulnál. Először olyan JavaScript fájlokat fog keresni, amelyeket *mindig* be kell tölteni, és ezután keres olyan speciális fájlokat, amelyek csak a jelenlegi alkalmazásmodulnál fontosak.

#### Közös JavaScript

A betöltendő JavaScript fájlok listája a `Loader::Agent::CommonJS` (az ügyintézői felülethez) és a `Loader::Customer::CommonJS` (az ügyfélfelülethez) konfigurációs beállításokban állítható be.

Ezek a beállítások kivonatokként vannak tervezve azért, hogy az OTRS kiterjesztések hozzáadhassák a saját kivonatkulcsaikat a további betöltendő tartalomhoz. Nézzünk egy példát:

```
<Setting Name="Loader::Agent::CommonJS###000-Framework" Required="1" Valid="1
↪">
    <Description Translatable="1">List of JS files to always be loaded for the␣
↪agent interface.</Description>
    <Navigation>Frontend::Base::Loader</Navigation>
    <Value>
        <Array>
            <Item>thirdparty/jquery-3.2.1/jquery.js</Item>
            <Item>thirdparty/jquery-browser-detection/jquery-browser-
↪detection.js</Item>

            ...

            <Item>Core.Agent.Header.js</Item>
            <Item>Core.UI.Notification.js</Item>
            <Item>Core.Agent.Responsive.js</Item>
        </Array>
    </Value>
</Setting>
```

Ez azon JavaScript fájlok listája, amelyeket mindig be kell tölteni az OTRS ügyintézői felületénél.

Olyan új tartalom hozzáadásához, amelyet mindig be kellene tölteni az ügyintézői felületen, egyszerűen adjon hozzá egy XML beállítófájlt egy másik kivonatbejegyzéssel:

```
<Setting Name="Loader::Agent::CommonJS###000-Framework" Required="1" Valid="1
↪">
    <Description Translatable="1">List of JS files to always be loaded for the␣
↪agent interface.</Description>
    <Navigation>Frontend::Base::Loader</Navigation>
    <Value>
        <Array>
```

(continues on next page)

(folytatás az előző oldalról)

```
            <Item>thirdparty/jquery-3.2.1/jquery.js</Item>
        </Array>
    </Value>
</Setting>
```

Egyszerű, nemde?

### Modulspecifikus JavaScript

Nem minden JavaScript használható az OTRS összes alkalmazásmoduljánál. Ezért lehetséges modulspecifikus JavaScript fájlok megadása. Amikor egy bizonyos modult használnak (mint például `AgentDashboard`), akkor ennek a modulnak a modulspecifikus JavaScript fájlja is be lesz töltve. A beállítás az XML beállításokban lévő előtétprogram-modul regisztrációban történik. Ismét egy példa:

```
<Setting Name="Loader::Module::AgentDashboard###001-Framework" Required="0"␣
↪Valid="1">
    <Description Translatable="1">Loader module registration for the agent␣
↪interface.</Description>
    <Navigation>Frontend::Agent::ModuleRegistration::Loader</Navigation>
    <Value>
        <Hash>
            <Item Key="CSS">
                <Array>
                    <Item>Core.Agent.Dashboard.css</Item>

                    ...

                </Array>
            </Item>
            <Item Key="JavaScript">
                <Array>
                    <Item>thirdparty/momentjs-2.18.1/moment.min.js</Item>
                    <Item>thirdparty/fullcalendar-3.4.0/fullcalendar.min.js</
↪Item>
                    <Item>thirdparty/d3-3.5.6/d3.min.js</Item>
                    <Item>thirdparty/nvd3-1.7.1/nvd3.min.js</Item>
                    <Item>thirdparty/nvd3-1.7.1/models/OTRSLineChart.js</Item>
                    <Item>thirdparty/nvd3-1.7.1/models/OTRSMultiBarChart.js</
↪Item>
                    <Item>thirdparty/nvd3-1.7.1/models/OTRSStackedAreaChart.js
↪</Item>
                    <Item>thirdparty/canvg-1.4/rgbcolor.js</Item>
                </Array>
            </Item>
        </Hash>
    </Value>
</Setting>
```

Lehetséges egy `<Item Key="JavaScript">` címke elhelyezése az előtétprogram-modul regisztrációkban, amely tartalmazhat `<Array>` tömböt és egy `<Item>` címkéket minden egyes JavaScript fájlhoz, amelyet be kellene tölteni ennél az alkalmazásmodulnál.

Most már rendelkezik az összes olyan információval, amely annak a módszernek a beállításhoz szükséges, hogy a betöltő kezelje a JavaScript kódot.

### 2.6.4 A betöltő beállítása: CSS

A betöltő a CSS-fájlokat nagyon hasonlóan kezeli a JavaScript fájlokhoz, ahogy az előző szakaszban le van írva, és a beállítások kiterjesztése is ugyanolyan módon működik.

#### Közös CSS

Annak a módja, ahogy a közös CSS-t kezelik, nagyon hasonló a *Közös JavaScript* betöltésének módjához.

## 2.7 Sablonozó mechanizmus

Belsőleg az OTRS egy sablonozó mechanizmust használ a HTML oldalak (és egyéb tartalom) dinamikus előállításához, miközben szétválasztva tartja a program logikáját (Perl) és a megjelenítést (HTML). Tipikusan egy előtétprogram modul egy saját sablonfájlt fog használni, át fog adni néhány adatot annak, és vissza fogja adni a megjelenített eredményt a felhasználónak.

A sablonfájlok itt találhatók: `$OTRS_HOME/Kernel/Output/HTML/Standard/*.tt`.

Az OTRS a Template::Toolkit megjelenítő motorra támaszkodik. A teljes Template::Toolkit szintaxis használható az OTRS sablonokban. Ez a szakasz néhány példa használati esetet és OTRS kiterjesztést mutat be a Template::Toolkit szintaxishoz.

### 2.7.1 Sablonparancsok

#### Dinamikus adatok beszúrása

A sablonokba dinamikus adatokat kell beszúrni, idézni, stb. Ez a szakasz sorolja fel a fontos parancsokat ennek elvégzéséhez.

```
[% Data.Name %]
```

Ha az alkalmazásmodul adatparamétereket ad meg a sablonoknak, akkor ezeket az adatokat ki lehet íratni a sablonra. A `[% Data.Name %]` a legegyszerűbb, de a legveszélyesebb is. Azt az adatparamétert fogja további feldolgozás nélkül beszúrni a sablonba úgy ahogy van, amely neve `Name`.

> **Figyelem:** A hiányzó HTML idézés miatt ez biztonsági problémákat eredményezhet. Sose írasson ki olyan adatokat, amelyeket egy felhasználó adott meg, anélkül, hogy idézné azokat a HTML környezetben. A felhasználó például egyszerűen beszúrhat egy `<script>` címkét, és az kiíródhat az OTRS által előállított HTML oldalon.
>
> Amikor csak lehetséges, használjon `[% Data.Name | html %]` (HTML-ben) vagy `[% Data.Name | uri %]` (hivatkozásokban) paramétert helyette.

Példa: Amikor HTML kimenetet állítunk elő az alkalmazásban, akkor HTML idézés nélkül kell kiíratnunk azt a sablonba, mint például a `<select>` elemeket, amelyeket a `Layout::BuildSelection()` függvény állít elő az OTRS-ben.

```
<label for="Dropdown">Example Dropdown</label>
[% Data.DropdownString"]
```

Ha speciális karaktereket tartalmazó, összetett nevű adatbejegyzései vannak, akkor nem használhatja a pont (.) jelölést az adathoz való hozzáféréshez. Az `item()` függvényt használja helyette: `[% Data.item('Összetett-név') %]`.

#### [% Data.Name | html %]

Ennek a parancsnak ugyanaz a funkciója mint az előzőnek, de HTML idézést hajt végre az adatokon, amint beszúrásra kerülnek a sablonba.

```
The name of the author is [% Data.Name | html %].
```

Lehetséges az érték legnagyobb hosszának megadása is. Ha például egy változónak csak 9 karakterét szeretné megjeleníteni (az eredmény *ValamiNév[···]* lesz), akkor használja a következőt:

```
The first 20 characters of the author's name: [% Data.Name | truncate(20) |␣
↪html %].
```

#### [% Data.Name | uri %]

Ez a parancs URL-kódolást hajt végre az adatokon, amint az beszúrásra kerül a sablonba. Ezt kell használni az URL-ek egyedülálló paraméternevei vagy értékei kiíratásánál a biztonsági problémák megakadályozásához. Nem használható teljes URL-eknél, mert ki fogja maszkolni például az = karaktert is.

```
<a href="[% Env("Baselink") %];Location=[% Data.File | uri %]">[% Data.File |␣
↪truncate(110) | html %]</a>
```

#### [% Data.Name | JSON %]

Ez a parancs JavaScript JSON szövegként írat ki egy szöveget vagy más adatszerkezetet.

```
var Text = [% Data.Text | JSON %];
```

Vegye figyelembe, hogy a szűrőjelölés csak egyszerű szövegeknél fog működni. Összetett adatok JSON szövegként való kiíratásához függvényként használja azt:

```
var TreeData = [% JSON(Data.TreeData) %];
```

#### [% Env() %]

A `LayoutObject` által szolgáltatott környezeti változókat szúrja be. Néhány példa:

```
The current user name is: [% Env("UserFullname") %]
```

Néhány egyéb gyakori előre meghatározott változó:

- `[% Env("Action") %]`: a jelenlegi művelet

- `[% Env("Baselink") %]`: az alaphivatkozás, például `index.pl?SessionID=...`
- `[% Env("CGIHandle") %]`: a jelenlegi CGI-kezelő, például `index.pl`
- `[% Env("SessionID") %]`: a jelenlegi munkamenet-azonosító
- `[% Env("Time") %]`: a jelenlegi idő, például *Thu Dec 27 16:00:55 2001*
- `[% Env("UserFullname") %]`: például Kovács János
- `[% Env("UserIsGroup[admin]") %]`: igen
- `[% Env("UserIsGroup[users]") %]`: igen, felhasználócsoportok (hasznos saját hivatkozásoknál)
- `[% Env("UserLogin") %]`: például mgg@x11.org

**Figyelem:** A hiányzó HTML idézés miatt ez biztonsági problémákat eredményezhet. Sose írasson ki olyan adatokat, amelyeket egy felhasználó adott meg, anélkül, hogy idézné azokat a HTML környezetben. A felhasználó például egyszerűen beszúrhat egy `<script>` címkét, és az kiíródhat az OTRS által előállított HTML oldalon.

Ne felejtse el a `| html` szűrőt hozzáadni, ahol az helyénvaló.

**`[% Config() %]`**

Beállítási változókat szúr be a sablonba. Nézzünk egy példa `Kernel/Config.pm` fájlt:

```
[Kernel/Config.pm]
    # FQDN
    # (Full qualified domain name of your system.)
    $Self->{FQDN} = 'otrs.example.com';
    # AdminEmail
    # (Email of the system admin.)
    $Self->{AdminEmail} = 'admin@example.com';
[...]
```

Változók kiíratásához ebből fájlból a sablonba a következőt használja:

```
The hostname is '$Config{"FQDN"}'
The admin email address is '[% Config("AdminEmail") %]'
```

**Figyelem:** A hiányzó HTML idézés miatt ez biztonsági problémákat eredményezhet.

Ne felejtse el a `| html` szűrőt hozzáadni, ahol az helyénvaló.

### Honosítási parancsok

**`[% Translate() %]`**

Lefordít egy szöveget a felhasználó által kiválasztott jelenlegi nyelve. Ha nem található fordítás, akkor az eredeti szöveget fogja használni.

```
Translate this text: [% Translate("Help") | html %]
```

Lefordíthat dinamikus adatokat is a `Translate` szűrőként való használatával:

```
Translate data from the application: [% Data.Type | Translate | html %]
```

Egy vagy több paramétert (`%s`) is megadhat a szövegen belül, amelyeket dinamikus adatokkal kell kicserélni:

```
Translate this text and insert the given data: [% Translate("Change %s␣
↪settings", Data.Type) | html %]
```

A JavaScriptben lévő szövegek is lefordíthatók és feldolgozhatók a JSON szűrővel.

```
var Text = [% Translate("Change %s settings", Data.Type) | JSON %];
```

#### [% Localize() %]

Kiírja az adatokat a jelenlegi nyelv vagy területi beállítás szerint.

Különböző kulturális területeken különböző egyezményt használnak a dátum és idő formázásához. Például ami Németországban 01.02.2010 formátum, annak az USA-ban 02/01/2010 formátumban kellene lennie. A `[% Localize() %]` függvény elvonatkoztatja ezt a sablontól. Nézzünk egy példát:

```
[% Data.CreateTime  Localize("TimeLong") %]
# Result for US English locale:
06/09/2010 15:45:41
```

Először is az adatok a `Data` segítségével kerülnek beszúrásra az alkalmazásmodulból. Itt mindig egy ISO UTC időbéleget (2010-06-09 15:45:41) kell átadni adatként a `[% Localize() %]` függvénynek. Ezután lesz kiíratva a jelenlegi területi beállítás dátum és idő meghatározása szerint.

A `[% Localize() %]` függvénynek átadott adatoknak UTC formátumban kell lenniük. Ha időzóna-eltolás van meghatározva a jelenlegi ügyintézőnél, akkor az alkalmazva lesz az UTC időbélyegen a kimenet előállítása előtt.

Különböző lehetséges dátum és idő kimeneti formátumok léteznek: `TimeLong` (teljes dátum és idő), `TimeShort` (nincsenek másodpercek) és `Date` (nincs idő).

```
[% Data.CreateTime  Localize("TimeLong") %]
# Result for US English locale:
06/09/2010 15:45:41

[% Data.CreateTime  Localize("TimeShort") %]
# Result for US English locale:
06/09/2010 15:45

[% Data.CreateTime  Localize("Date") %]
# Result for US English locale:
06/09/2010
```

Ember által olvasható fájlméretek kimenete is elérhető lehetőségként `Localize('Filesize')` (egyszerűen adja át a nyers fájlméretet bájtokban).

```
[% Data.Filesize  Localize("Filesize") %]
# Result for US English locale:
23 MB
```

**[% ReplacePlaceholders() %]**

Kicseréli a helykitöltőket (`%s`) a szövegekben az átadott paraméterekre.

Bizonyos esetekben érdemes lehet HTML kódot beszúrni a fordításokba a helykitöltők helyett. Másrészt viszont gondoskodnia kellfertőtlenítésről, mivel a lefordított szövegekben nem szabad megbízni úgy, ahogy vannak. Ehhez először fordítsa le a szöveget, küldje át a HTML szűrőn, és végül cserélje ki a helykitöltőket statikus (biztonságos) HTML kóddal.

```
[% Translate("This is %s.") | html | ReplacePlaceholders('<strong>bold text</
↪strong>') %]
```

A `ReplacePlaceholders()` szűrőnek átadott paraméterek számának meg kell egyeznie az eredeti szövegben lévő helykitöltők számával.

Használhatja a `[% ReplacePlaceholders() %]` szűrőt függvény formában is abban az esetben, ha nem fordít le semmit sem. Ebben az esetben az első paraméter a célszöveg, és a benne talált bármilyen helykitöltő helyettesítve lesz az azt követő paraméterekkel.

```
[% ReplacePlaceholders("This string has both %s and %s.", '<strong>bold text</
↪strong>', '<em>italic text</em>') %]
```

### Sablonfeldolgozó parancsok

### Megjegyzés

Azok a sorok, amelyek # karakterrel kezdődnek az elején, nem lesznek láthatók a HTML kimeneten. Ez használható a sablonkód magyarázatához, vagy annak egyes részei letiltásához is.

```
# this section is temporarily disabled
# <div class="AsBlock">
#     <a href="...">link</a>
# </div>
```

**[% InsertTemplate("Copyright.tt") %]**

**Figyelem:** Felhívjuk a figyelmét, hogy az `InsertTemplate` parancs azért lett hozzáadva, hogy jobb visszafelé kompatibilitást nyújtson a régi DTL rendszerhez. Ez esetleg elavulttá válhat az OTRS jövőbeli verzióiban, és később eltávolításra kerülhet. Ha nem használ blokk parancsokat a felvett sablonjában, akkor nincs szüksége az `InsertTemplate` parancsra, és használhatja helyette a szabványos Template::Toolkit szintaxist, úgymint `INCLUDE/PROCESS`.

Felvesz egy másik sablonfájlt a jelenlegibe. A felvett fájl is tartalmazhat sablonparancsokat.

```
# include Copyright.tt
[% InsertTemplate("Copyright") %]
```

Felhívjuk a figyelmét, hogy ez nem ugyanaz mint a Template::Toolkit `[% INCLUDE %]` parancsa, amely csak feldolgozza a hivatkozott sablont. Az `[% InsertTemplate() %]` tulajdonképpen hozzáadja a hivatkozott sablon tartalmát a jelenlegi sablonhoz azért, hogy együtt legyenek feldolgozhatók. Ez lehetővé teszi a beágyazott sablon számára, hogy ugyanazon környezethez vagy adatokhoz férjen hozzá mint a fő sablon.

#### `[% RenderBlockStart %]` / `[% RenderBlockEnd %]`

**Figyelem:** Vegye figyelembe, hogy a blokk parancsok azért lettek hozzáadva, hogy jobb visszafelé kompatibilitást nyújtsanak a régi DTL rendszerhez. Ezek esetleg elavulttá válhatnak az OTRS jövőbeli verzióiban, és később eltávolításra kerülhetnek. Azt javasoljuk, hogy blokk parancsok használata nélkül fejlesszen bármilyen új kódot. Használhatja a szabványos Template::Toolkit szintaxist a feltételes sablonkimenethez, mint például `IF/ELSE`, `LOOP` és egyéb hasznos dolgok.

Ezzel a paranccsal lehet megadni egy sablonfájl részeit blokként. Ezt a blokkot határozottan ki kell tölteni egy függvényhívással az alkalmazásból, hogy megjelenjen az előállított kimeneten. Az alkalmazás meghívhatja a blokkot 0-szor (nem fog megjelenni a kimeneten), illetve 1 vagy többször (esetleg mindegyiket a sablonnak átadott adatparaméterek különböző halmazával).

Egy gyakori használati eset egy táblázat kitöltése dinamikus adatokkal:

```
<table class="DataTable">
    <thead>
        <tr>
            <th>[% Translate("Name") | html %]</th>
            <th>[% Translate("Type") | html %]</th>
            <th>[% Translate("Comment") | html %]</th>
            <th>[% Translate("Validity") | html %]</th>
            <th>[% Translate("Changed") | html %]</th>
            <th>[% Translate("Created") | html %]</th>
        </tr>
    </thead>
    <tbody>
[% RenderBlockStart("NoDataFoundMsg") %]
        <tr>
            <td colspan="6">
                [% Translate("No data found.") | html %]
            </td>
        </tr>
[% RenderBlockEnd("NoDataFoundMsg") %]
[% RenderBlockStart("OverviewResultRow") %]
        <tr>
            <td><a class="AsBlock" href="[% Env("Baselink") %]Action=[% Env(
↪"Action") %];Subaction=Change;ID=[% Data.ID | uri %]">[% Data.Name | html %]
↪</a></td>
            <td>[% Translate(Data.TypeName) | html %]</td>
            <td title="[% Data.Comment | html %]">[% Data.Comment |␣
↪truncate(20) | html %]</td>
```

(continues on next page)

(folytatás az előző oldalról)

```
            <td>[% Translate(Data.Valid) | html %]</td>
            <td>[% Data.ChangeTime | Localize("TimeShort") %]</td>
            <td>[% Data.CreateTime | Localize("TimeShort") %]</td>
        </tr>
[% RenderBlockEnd("OverviewResultRow") %]
    </tbody>
</table>
```

A körülvevő táblázat a fejléccel mindig elő lesz állítva. Ha nem található adat, akkor a `NoDataFoundMsg` blokk egyszer lesz meghívva egy olyan táblázatot eredményezve, amelynek egy adatsora van a *Nem található adat* üzenettel.

Ha találhatók adatok, akkor minden egyes sornál egy függvényhívás történik az `OverViewResultRow` blokknál (minden alkalommal átadva az adatokat ehhez a bizonyos sorhoz) egy olyan táblázatot eredményezve, amelynek annyi sora van, ahány eredmény található.

Nézzük meg, hogyan vannak meghívva a blokkok az alkalmazásmodulból:

```
my %List = $Kernel::OM->Get('Kernel::System::State)->StateList(
    UserID => 1,
    Valid  => 0,
);

# if there are any states, they are shown
if (%List) {

    # get valid list
    my %ValidList = $Kernel::OM->Get('Kernel::System::Valid')->ValidList();
    for my $ListKey ( sort { $List{$a} cmp $List{$b} } keys %List ) {

        my %Data = $Kernel::OM->Get('Kernel::System::State)->StateGet( ID =>
↪$ListKey );
        $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Block(
            Name => 'OverviewResultRow',
            Data => {
                Valid => $ValidList{ $Data{ValidID} },
                %Data,
            },
        );
    }
}

# otherwise a no data found msg is displayed
else {
    $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Block(
        Name => 'NoDataFoundMsg',
        Data => {},
    );
}
```

Figyelje meg, hogy a blokkoknak hogyan kell átadniuk mind a nevüket, mind egy opcionális adathalmazt különálló paraméterekként a blokkfüggvény hívásnak. Az adatbeszúró parancsoknak egy blokkon belül mindig az ezen blokk blokkfüggvény hívásához megadott adatokra van szükségük, nem az általános sablonmegjelenítő híváshoz.

További információkért nézze meg a dokumentációs portált.

### [% WRAPPER JSOnDocumentComplete %]...[% END %]

Megjelöli azt a JavaScript kódot, amelyet azután kell lefuttatni, miután az összes CSS, JavaScript és egyéb külső tartalom betöltődött, és az alapvető JavaScript előkészítés befejeződött. Vessünk egy pillantást ismét egy példára:

```
<form action="[% Env("CGIHandle") %]" method="post" enctype="multipart/form-
↪data" name="MoveTicketToQueue" class="Validate PreventMultipleSubmits" id=
↪"MoveTicketToQueue">
    <input type="hidden" name="Action"       value="[% Env("Action") %]"/>
    <input type="hidden" name="Subaction"    value="MoveTicket"/>

    ...

    <div class="Content">
        <fieldset class="TableLike FixedLabel">
            <label class="Mandatory" for="DestQueueID"><span class="Marker">*
↪</span> [% Translate("New Queue") | html %]:</label>
            <div class="Field">
                [% Data.MoveQueuesStrg %]
                <div id="DestQueueIDError" class="TooltipErrorMessage" ><p>[%␣
↪Translate("This field is required.") | html %]</p></div>
                <div id="DestQueueIDServerError" class="TooltipErrorMessage">
↪<p>[% Translate("This field is required.") | html %]</p></div>
[% WRAPPER JSOnDocumentComplete %]
<script type="text/javascript">
    $('#DestQueueID').bind('change', function (Event) {
        $('#NoSubmit').val('1');
        Core.AJAX.FormUpdate($('#MoveTicketToQueue'), 'AJAXUpdate',
↪'DestQueueID', ['NewUserID', 'OldUserID', 'NewStateID', 'NewPriorityID' [%␣
↪Data.DynamicFieldNamesStrg %]]);
    });
</script>
[% END %]
                    </div>
                    <div class="Clear"></div>
```

Ez a kódrészlet egy kicsi űrlapot hoz létre, és rátesz egy `onchange` kezelőt a `<select>` elemre, amely aktivál egy AJAX-alapú űrlapfrissítést.

Miért van szükség a JavaScript kód körbezárására a `[% WRAPPER JSOnDocumentComplete %]...[% END %]` blokkban? A JavaScript betöltést teljesítmény okok miatt áthelyezték az oldal lábrészébe. Ez azt jelenti, hogy az oldal `<body>` részén belül még nincsenek JavaScript programkönyvtárak betöltve. A `[% WRAPPER JSOnDocumentComplete %]...[% END %]` blokkal lehet biztos abban, hogy ez a JavaScript áthelyezésre kerül a végső HTML dokumentumnak egy olyan részébe, ahol csak akkor kerül végrehajtásra, miután a teljes külső JavaScript és CSS tartalom sikeresen be lett töltve és elő lett készítve.

A `[% WRAPPER JSOnDocumentComplete %]...[% END %]` blokkon belül használhatja a `<script>` címkéket a JavaScript kód körbezárásához, de ezt nem kell megtennie. Előnyös lehet, mert engedélyezni fogja a helyes szintaxis-kiemelést az olyan integrált fejlesztői környezetekben, amelyek támogatják azt.

### 2.7.2 Egy sablonfájl használata

Rendben, de tulajdonképpen hogyan kell egy sablonfájlt feldolgozni és az eredményt előállítani? Ez igazán egyszerű:

```
# render AdminState.tt
$Output .= $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Output(
    TemplateFile => 'AdminState',
    Data         => \%Param,
);
```

Az előtétprogram-modulokban a `Kernel::Output::HTML::Layout` objektum `Output()` függvénye lesz meghívva (miután az összes szükséges blokk meg lett hívva ebben a sablonban) a végső kimenet előállításához. Adatparaméterek opcionális halmaza is átadásra kerül a sablonnak minden olyan adatbeszúró parancsnál, amelyek nincsenek egy blokk belsejében.

## 2.8 Saját témák létrehozása

Létrehozhatja a saját témáit, így olyan elrendezést használhat az OTRS webes előtétprogramján, amilyet csak szeretne. Egyéni témák létrehozásához személyre kell szabnia a kimeneti sablonokat az igényei szerint. A kimeneti sablonok szerkezetéről és szintaxisáról további információk találhatók a *Sablonozó mechanizmus* szakaszban.

Példaként hajtsa végre a következő lépéseket egy új *Company* nevű téma létrehozásához:

1. Hozzon létre egy `Kernel/Output/HTML/Templates/Company` nevű könyvtárat, és másoljon át a `Kernel/Output/HTML/Templates/Standard` mappából minden olyan fájlt az új mappába, amelyet meg szeretne változtatni.

   ---

   **Megjegyzés:** Csak azokat a fájlokat másolja át, amelyeket meg szeretne változtatni. Az OTRS automatikusan be fogja szerezni a hiányzó fájlokat a szabványos témából. Ez sokkal könnyebbé fogja tenni a későbbiekben a frissítést.

   ---

2. Szabja személyre a `Kernel/Output/HTML/Templates/Company` könyvtárban lévő fájlokat, és változtassa meg az elrendezést az igényei szerint.
3. Az új téma bekapcsolásához adja hozzá azokat a rendszerbeállításokban a `Frontend::Themes` alatt.

Most az új témának használhatónak kell lennie. A személyes beállítások oldalán keresztül tudja kiválasztani.

**Figyelem:** Ne változtassa meg az OTRS-sel szállított témafájlokat, mivel azok a változtatások el fognak veszni egy frissítés után. Csak a fent leírt lépések végrehajtásával hozzon létre saját témákat.

## 2.9 Honosítási és fordítási mechanizmus

Négy lépés szükséges a szoftver lefordításához és honosításához: a honosítható szövegek megjelölése a forrásfájlokban, a fordítási adatbázis/fájl előállítása, maga a fordítási folyamat, és a lefordított adatok használata a kódon belül.

### 2.9.1 Lefordítható szövegek megjelölése a forrásfájlokban

A Perl-kódban az összes lefordítandó literál szöveg automatikusan meg van jelölve a fordításhoz:

```
$LanguageObject->Translate('My string %s', $Data)
```

Ez fogja megjelölni a *My string %s* szöveget a fordításhoz. Ha arra van szüksége, hogy a kódban megjelölje a szövegeket, de még NE fordítsa le azokat, akkor használhatja a `Kernel::Langauge::Translatable()` NOOP metódust.

```
package MyPackage;

use strict;
use warnings;

use Kernel::Language (qw(Translatable));

...

my $UntranslatedString = Translatable('My string %s');
```

A sablonfájlokban az összes olyan literál szöveg automatikusan meg van jelölve a kigyűjtéshez, amelyek `Translate()` címkével vannak körbezárva: `[% Translate('My string %s', Data.Data )%]`.

A rendszerbeállítás és az adatbázis XML-fájlokban a `Translatable="1"` attribútummal jelölheti meg a szövegeket a kigyűjtéshez.

```
# Database XML
<Insert Table="groups">
    <Data Key="id" Type="AutoIncrement">1</Data>
    ...
    <Data Key="comments" Type="Quote" Translatable="1">Group for default␣
↪access.</Data>
    ...
</Insert>

# SysConfig XML
<Setting>
    <Option SelectedID="0">
        <Item Key="0" Translatable="1">No</Item>
        <Item Key="1" Translatable="1">Yes</Item>
    </Option>
</Setting>
```

### 2.9.2 Lefordítható szövegek összegyűjtése a fordítási adatbázisba

Az `otrs.Console.pl Dev::Tools::TranslationsUpdate` konzolparancs használható az összes lefordítható szöveg kigyűjtéséhez a forrásfájlokból. Ezek össze lesznek gyűjtve, és ki lesznek írva a fordítási fájlokba.

Az OTRS keretrendszerhez és az összes kiterjesztőmodulhoz `.pot` és `.po` fájlok kerülnek kiírásra. Ezeket a fájlokat használják a fordítók a szoftver honosításához.

De az OTRS-nek sebességi okok miatt a fordításokra Perl-fájlokban van szüksége. Az `otrs.Console.pl Dev::Tools::TranslationsUpdate` parancs ezeket a fájlokat is elő fogja állítani. Két különböző

fordítási gyorsítótár fájltípus létezik, amelyek a következő sorrendben kerülnek felhasználásra. Ha egy szó vagy mondat újra meg van adva egy fordítási fájlban, akkor a legutolsó meghatározást fogja használni.

1. Alapértelmezett keretrendszer fordítási fájl: `Kernel/Language/$Language.pm`
2. Egyéni fordítási fájl: `Kernel/Language/$Language_Custom.pm`

### Alapértelmezett keretrendszer fordítási fájl

Az alapértelmezett keretrendszer fordítási fájl tartalmazza az alapvető fordításokat. Az alábbi egy alapértelmezett keretrendszer fordítási fájl példája.

```
package Kernel::Language::de;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub Data {
    my $Self = shift;

    # $$START$$

    # possible charsets
    $Self->{Charset} = ['iso-8859-1', 'iso-8859-15', ];
    # date formats (%A=WeekDay;%B=LongMonth;%T=Time;%D=Day;%M=Month;%Y=Jear;)
    $Self->{DateFormat} = '%D.%M.%Y %T';
    $Self->{DateFormatLong} = '%A %D %B %T %Y';
    $Self->{DateFormatShort} = '%D.%M.%Y';
    $Self->{DateInputFormat} = '%D.%M.%Y';
    $Self->{DateInputFormatLong} = '%D.%M.%Y - %T';

    $Self->{Translation} = {
    # Template: AAABase
    'Yes' => 'Ja',
    'No' => 'Nein',
    'yes' => 'ja',
    'no' => 'kein',
    'Off' => 'Aus',
    'off' => 'aus',
    };
    # $$STOP$$
    return 1;
}

1;
```

### Egyéni fordítási fájl

Az egyéni fordítási fájl kerül beolvasásra legutoljára, és így annak fordítása, amely használva lesz. Ha saját megfogalmazást szeretne hozzáadni a telepítéshez, akkor hozza létre ezt a fájlt a nyelvéhez.

```
package Kernel::Language::xx_Custom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub Data {
    my $Self = shift;

    # $$START$$

    # own translations
    $Self->{Translation}->{'Lock'} = 'Lala';
    $Self->{Translation}->{'Unlock'} = 'Lulu';

    # $$STOP$$
    return 1;
}

1;
```

**Megjegyzés:** Az új felület nyelvi fájljai mostantól az összeállított alkalmazás (statikus JSON) része. Ha egyéni nyelvi fájlt ad hozzá a fájlrendszerhez, akkor újra össze kell állítania az alkalmazást a változtatások figyelembe vételéhez. Az ismételt összeállítás aktiválásához indítsa újra a kiszolgálót a `--deploy-assets` kapcsolóval:

```
otrs> /opt/otrs/bin/otrs.WebServer.pl --deploy-assets
```

Az összeállítási folyamat során a nyelvi fájlok frissítve lesznek, és az összes `*_Custom.pm` fájl is életbe fog lépni.

### 2.9.3 Maga a fordítási folyamat

Az OTRS a Weblate fordítószoftvert használja a fordítási folyamat kezeléséhez. A részletekért nézze meg a *Fordítás* szakaszt.

### 2.9.4 A lefordított adatok használata a kódból

Használhatja a `$LanguageObject->Translate()` metódust a szövegek lefordításához futási időben a Perl-kódból, és a `Translate()` címkét a *Sablonozó mechanizmus* esetén.

FEJEZET 3

# Hogyan bővíthető az OTRS

## 3.1 Egy új OTRS előtétprogram-modul írása

Ebben a fejezetben egy új OTRS modul írása van szemléltetve egy egyszerű kis program alapján. A szükséges előkövetelmény egy olyan OTRS fejlesztői környezet, amely a hasonló nevű fejezetben van megadva.

### 3.1.1 Mit szeretnénk írni

Szeretnénk írni egy olyan kis OTRS modult, amely a „Hello World" szöveget jeleníti meg, amikor előhívják. Mindenek előtt fel kell építenünk a `/Hello World` könyvtárat a modulhoz a fejlesztői könyvtárban. Ebben a könyvtárban létrehozható az OTRS-ben meglévő összes könyvtár. Minden modulnak legalább a következő könyvtárakat kell tartalmaznia:

```
Kernel
Kernel/System
Kernel/Modules
Kernel/Output/HTML/Templates/Standard
Kernel/Config
Kernel/Config/Files
Kernel/Config/Files/XML/
Kernel/Language
```

### 3.1.2 Alapértelmezett beállítófájl

Egy modulregisztráció létrehozása megkönnyíti az új modul megjelenítését az OTRS-ben. Ezért létrehozunk egy `/Kernel/Config/Files/XML/HelloWorld.xml` fájlt. Ebben a fájlban létrehozunk egy új beállítási elemet. A különféle beállítások hatása a *Beállítási mechanizmus* fejezetben van leírva.

```
<?xml version="1.0" encoding="UTF-8" ?>
<otrs_config version="2.0" init="Application">
    <Setting Name="Frontend::Module###AgentHelloWorld" Required="1" Valid="1">
        <Description Translatable="1">FrontendModuleRegistration for␣
↪HelloWorld module.</Description>
        <Navigation>Frontend::Agent::ModuleRegistration</Navigation>
        <Value>
            <Item ValueType="FrontendRegistration">
                <Hash>
                    <Item Key="Group">
                        <Array>
                            <Item>users</Item>
                        </Array>
                    </Item>
                    <Item Key="GroupRo">
                        <Array>
                        </Array>
                    </Item>
                    <Item Key="Description" Translatable="1">HelloWorld.</
↪Item>
                    <Item Key="Title" Translatable="1">HelloWorld</Item>
                    <Item Key="NavBarName">HelloWorld</Item>
                </Hash>
            </Item>
        </Value>
    </Setting>
    <Setting Name="Loader::Module::AgentHelloWorld###002-Filename" Required="0
↪" Valid="1">
        <Description Translatable="1">Loader module registration for the␣
↪agent interface.</Description>
        <Navigation>Frontend::Agent::ModuleRegistration::Loader</Navigation>
        <Value>
            <Hash>
                <Item Key="CSS">
                    <Array>
                    </Array>
                </Item>
                <Item Key="JavaScript">
                    <Array>
                    </Array>
                </Item>
            </Hash>
        </Value>
    </Setting>
    <Setting Name="Frontend::Navigation###AgentHelloWorld###002-Filename"␣
↪Required="0" Valid="1">
        <Description Translatable="1">Main menu item registration.</
↪Description>
        <Navigation>Frontend::Agent::ModuleRegistration::MainMenu</Navigation>
        <Value>
            <Array>
                <DefaultItem ValueType="FrontendNavigation">
```

(continues on next page)

(folytatás az előző oldalról)

```
                    <Hash>
                    </Hash>
                </DefaultItem>
                <Item>
                    <Hash>
                        <Item Key="Group">
                            <Array>
                                <Item>users</Item>
                            </Array>
                        </Item>
                        <Item Key="GroupRo">
                            <Array>
                            </Array>
                        </Item>
                        <Item Key="Description" Translatable="1">HelloWorld.</
↪Item>
                        <Item Key="Name" Translatable="1">HelloWorld</Item>
                        <Item Key="Link">Action=AgentHelloWorld</Item>
                        <Item Key="LinkOption"></Item>
                        <Item Key="NavBar">HelloWorld</Item>
                        <Item Key="Type">Menu</Item>
                        <Item Key="Block"></Item>
                        <Item Key="AccessKey"></Item>
                        <Item Key="Prio">8400</Item>
                    </Hash>
                </Item>
            </Array>
        </Value>
    </Setting>
</otrs_config>
```

### 3.1.3 Előtétprogram-modul

A hivatkozások létrehozása és a rendszerbeállítások végrehajtása után megjelenik egy új modul „Hello-World” néven. Előhívásakor egy hibaüzenet jelenik meg, mivel az OTRS még nem találja a hozzá tartozó előtétprogram modult. Ez a következő dolog, amit létre kell hozni. Ehhez hozzuk létre az alábbi fájlt:

```
# --
# Copyright (C) (year) (name of author) (email of author)
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Modules::AgentHelloWorld;

use strict;
use warnings;

# Frontend modules are not handled by the ObjectManager.
```

(continues on next page)

(folytatás az előző oldalról)

```
our $ObjectManagerDisabled = 1;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless ($Self, $Type);

    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;
    my %Data = ();

    my $HelloWorldObject = $Kernel::OM->Get('Kernel::System::HelloWorld');
    my $LayoutObject     = $Kernel::OM->Get('Kernel::Output::HTML::Layout');

    $Data{HelloWorldText} = $HelloWorldObject->GetHelloWorldText();

    # build output
    my $Output = $LayoutObject->Header(Title => "HelloWorld");
    $Output   .= $LayoutObject->NavigationBar();
    $Output   .= $LayoutObject->Output(
        Data         => \%Data,
        TemplateFile => 'AgentHelloWorld',
    );
    $Output   .= $LayoutObject->Footer();

    return $Output;
}

1;
```

### 3.1.4 Alapmodul

Ezután hozzunk létre egy fájlt a `/HelloWorld/Kernel/System/HelloWorld.pm` alapmodulhoz az alábbi tartalommal:

```
# --
# Copyright (C) (year) (name of author) (email of author)
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::HelloWorld;

use strict;
```

(continues on next page)

(folytatás az előző oldalról)

```
use warnings;

# list your object dependencies (e.g. Kernel::System::DB) here
our @ObjectDependencies = (
    # 'Kernel::System::DB',
);

=head1 NAME

HelloWorld - Little "Hello World" module

=head1 DESCRIPTION

Little OTRS module that displays the text 'Hello World' when called up.

=head2 new()

Create an object. Do not use it directly, instead use:

    my $HelloWorldObject = $Kernel::OM->Get('Kernel::System::HelloWorld');

=cut

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless ($Self, $Type);

    return $Self;
}

=head2 GetHelloWorldText()

Return the "Hello World" text.

    my $HelloWorldText = $HelloWorldObject->GetHelloWorldText();

=cut

sub GetHelloWorldText {
    my ( $Self, %Param ) = @_;

    # Get the DBObject from the central object manager
    # my $DBObject = $Kernel::OM->Get('Kernel::System::DB');

    my $HelloWorld = $Self->_FormatHelloWorldText(
        String => 'Hello World',
    );

    return $HelloWorld;
```

(continues on next page)

(folytatás az előző oldalról)

```
}

=begin Internal:

=head2 _FormatHelloWorldText()

Format "Hello World" text to uppercase

    my $HelloWorld = $Self->_FormatHelloWorldText(
        String => 'Hello World',
    );

=cut

sub _FormatHelloWorldText{
    my ( $Self, %Param ) = @_;

    my $HelloWorld = uc $Param{String};

    return $HelloWorld;

}

=end Internal:

1;
```

### 3.1.5 Sablonfájl

Az utolsó hiányzó dolog, mielőtt az új modul futtatható lenne, a hozzá tartozó HTML sablon. Ezért hozzuk létre az alábbi fájlt:

```
# --
# Copyright (C) (year) (name of author) (email of author)
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
<h1>[% Translate("Overview") | html %]: [% Translate("HelloWorld") %]</h1>
<p>
    [% Data.HelloWorldText | Translate() | html %]
</p>
```

A modul most már működik, és a meghívásakor megjeleníti a *Hello World* szöveget.

### 3.1.6 Nyelvi fájl

Ha a *Hello World!* szöveget le kell fordítani például magyarra, akkor létrehozhat egy fordítási fájlt ehhez a nyelvhez a `HelloWorld/Kernel/Language/hu_AgentHelloWorld.pm` helyre. Példa:

```
package Kernel::Language::de_AgentHelloWorld;

use strict;
use warnings;

sub Data {
    my $Self = shift;

    $Self->{Translation}->{'Hello World!'} = 'Hallo Welt!';

    return 1;
}
1;
```

### 3.1.7 Összefoglaló

A fent megadott példa azt mutatja be, hogy nem túl bonyolult egy új modult írni az OTRS-hez. Fontos azonban meggyőződni arról, hogy a modul és a fájlnév egyedi legyen, és ennélfogva ne ütközzenek a keretrendszerrel vagy egyéb kiterjesztő modulokkal. Amikor egy modul elkészült, akkor egy OPM csomagot kell előállítani belőle (lásd a *Csomagkészítés* fejezetet).

## 3.2 Egy új OTRS előtétprogram összetevő írása

Ebben a példában megpróbálunk megírni egy új OTRS előtétprogram összetevőt. Az OTRS 7-tel kezdve a keretrendszer támogatja a Vue.js-ben írt és egy új JavaScript eszközláncon alapuló egyoldalas alkalmazás előtétprogramokat. Az első megközelítés az új külső felületet tartalmazza, amelyhez megpróbálunk egy egyéni összetevőt írni. Szüksége lesz egy futó OTRS *Fejlesztői környezet* meglétére, amint az a hasonló nevű fejezetben meg van határozva.

### 3.2.1 A cél

Szeretnénk írni egy kis előtétprogram összetevőt, amely a *Helló, Világ!* szöveget jeleníti meg, amikor meghívják. Ez egy útvonal összetevő lesz, amely azt jelenti, hogy elérhető lesz a külső felületen, amikor meghívják egy gondosan kialakított URL-lel.

### 3.2.2 A csontváz parancs használata

A fejlesztés gyorsításához egy csontváz parancsot kell használnunk egy gyakran használt sablonfájl beszerzéséhez, amelyre építkezhetünk.

Egy futó OTRS példányon hívja meg a következő parancsot a sablon előállításához. A `HelloWorld` értéket fogjuk használni az új összetevőnk neveként:

```
bin/otrs.Console.pl Dev::Code::Generate::VueComponent --component-directory /
↪ws/MyPackage --component-subdirectory Apps/External/Components/Route --no-
↪docs HelloWorld
```

A parancsban a `--component-directory` a modul könyvtára, a `--component-subdirectory` a `Frontend/` mappa alatti útvonal, amely otthont ad az összetevőfájlnak. Most használja a `--no-docs` kapcsolót, hogy kihagyja a dokumentációs összetevő létrehozását a megjelenítési rendszernél.

Ez a parancs két fájlt fog előállítani a következő útvonalakkal:

```
Generated: /ws/MyPackage/Frontend/Apps/External/Components/Route/HelloWorld.
↪vue
Generated: /ws/MyPackage/Frontend/Tests/Apps/External/Components/Route/
↪HelloWorld.js
Skipped creating documentation component.
```

### 3.2.3 Az útvonal beállítása

Ahhoz, hogy lehetővé tegye az útvonalat a külső felület alkalmazásának, hozzá kell adnunk egy helyes útvonal-beállítást, amely az összetevőnkre mutat. Emiatt létrehozunk egy `Kernel/Config/Files/XML/HelloWorld.xml` fájlt a következő meghatározásokkal:

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="2.0" init="Application">
    <Setting Name="ExternalFrontend::Route###420-HelloWorld" Required="0"␣
↪Valid="1">
        <Description Translatable="1">Defines the application routes for the␣
↪external interface. Additional routes are defined by adding new items and␣
↪specifying their parameters. 'Group' and 'GroupRo' arrays can be used to␣
↪limit access of the route to members of certain groups with RW and RO␣
↪permissions respectively. 'Path' defines the relative path of the route, and
↪'Alias' can be used for specifying an alternative path. 'Component' is the␣
↪path of the Vue component responsible for displaying the route content,␣
↪relative to the Components/Route folder in the app. 'IsPublic' defines if␣
↪the route will be accessible for unauthenticated users and in case this is␣
↪set to '1', 'Group' and 'GroupRo' parameters will be ignored. 'Props' can␣
↪be used to signal that the path contain dynamic segments, and that their␣
↪values should be bound to the component as props (use '1' to turn on this␣
↪feature).</Description>
        <Navigation>Frontend::External::Route</Navigation>
        <Value>
            <Array>
                <DefaultItem ValueType="ApplicationRoute">
                    <Hash>
                    </Hash>
                </DefaultItem>
                <Item>
                    <Hash>
                        <Item Key="Group">
                            <Array>
                            </Array>
                        </Item>
                        <Item Key="GroupRo">
                            <Array>
                            </Array>
                        </Item>
                        <Item Key="Path">/hello-world/:headingText?</Item>
```

(continues on next page)

(folytatás az előző oldalról)

```
                        <Item Key="Alias"></Item>
                        <Item Key="Component">HelloWorld</Item>
                        <Item Key="IsPublic">1</Item>
                        <Item Key="Props">1</Item>
                    </Hash>
                </Item>
            </Array>
        </Value>
    </Setting>
</otrs_config>
```

- A `Group` és `GroupRo` használható az útvonal képernyő korlátozásához azokra a felhasználókra, akik bizonyos csoportba tartoznak. Ne feledje, hogy ez csak a hitelesített ügyfél-felhasználókra vonatkozik.
- A `Path` tulajdonképpen a rövid útvonal, ami alatt az útvonal összetevő elérhető lesz. A teljes URL ebben az esetben `/external/hello-world` lesz, és bármely további útvonal összetevő egy `headingText` nevű paraméterként kerül átadásra. Ha a rendszeren beállították a `Frontend::PrefixPath` értéket, akkor a teljes URL elé lesz fűzve.
- Az `Alias` használható egy álnév adásához ugyanarra az útvonalra (például `/hello-world-alt`). Ugyanarra az összetevőre fog mutatni.
- A `Component` az összetevő azonosítója, a fájlnév első része a `.vue` kiterjesztés nélkül. Összetevőmappák esetén ez a gyökér mappa neve. További információkért nézze meg az *Összetevőmappák* fejezetet.
- Az `IsPublic` határozza meg, hogy az útvonal elérhető legyen-e a nem hitelesített felhasználóknak (0 vagy üres – nem érhető el, 1 – elérhető).
- A `Props` határozza meg, hogy az útvonal át fog-e adni URI paramétereket prop értékekként (0 vagy üres – nincs átadva, 1 – átadva). További információkért nézze meg a *Paraméterek átadása az útvonal összetevőnek* fejezetet.

### 3.2.4 Összetevő sablon kód

Most indítsuk el a kódszerkesztőt, és vessünk közelebbi pillantást a `HelloWorld.vue` fájlra, amit a csontváz parancsunk létrehozott.

A fájl felső része egy sablonszakaszt tartalmaz, amely Vue.js sablonkódot kell tartalmazzon. Például módosítsuk azt, hogy megjelenítsen egy címsort egy szöveges változóval:

```
<template>
    <main class="HelloWorld">
        <b-container>
            <b-row>
                <b-col>
                    <h1 class="HelloWorld__Heading">
                        {{ headingText | translate }}
                    </h1>
                </b-col>
            </b-row>
        </b-container>
    </main>
</template>
```

Az OTRS sok szűrőt támogat, ezek egyike a `translate` szűrő. Akkor is támogatja a szövegliterálok lefordítását, ha helykitöltő értékeket használ. Használhatja ehhez hasonlóan:

```
{{ 'This is a %s.' | translate('string') }}
```

### 3.2.5 Összetevő alapkódja

Ezután támogatást adunk egy prophoz az összetevő alapkód blokkjánál. A következő egy módosított és rövidített verzió, amely megfelel példaként:

```
<script>
export default {
    name: 'HelloWorld',

    props: {
        headingText: {
            type: String,
            default: translatable('Hello, world!'),
        },
    },
};
</script>
```

Ez hozzáad egy `headingText` nevű propot az összetevőnkhöz, amely szöveg típusú és van egy érzékeny alapértelmezett értéke.

A `translatable()` művelet nélküli metódus használata csak arra van korlátozva, hogy megjelölje azokat a lefordítható szövegeket, amelyek megjelennek a kódban. Ne feledje, hogy ez nem szükséges az olyan szövegliteráloknál, amelyek át lesznek küldve a fordítási szűrőn, mivel ezt a kezdettől fogva feltételezzük. Ökölszabályként használja a jelölőt minden olyan helyen, ahol a szöveg nincs azon a helyen lefordítva, ahol meghatározták.

### 3.2.6 Összetevő stílus kód

Végül, de nem utoljára, lehetőségünk van az összetevő által használt stílus meghatározásához. Ehhez hozzá kell férnünk az SCSS-hez, amely a SASS CSS-kiterjesztés egyik változata. A kihasználásához egyszerűen adjon hozzá egy stílus címkét az összetevő fájl végénél:

```
<style lang="scss">
.HelloWorld {
    &__Heading {
        color: $primary;
    }
}
</style>
```

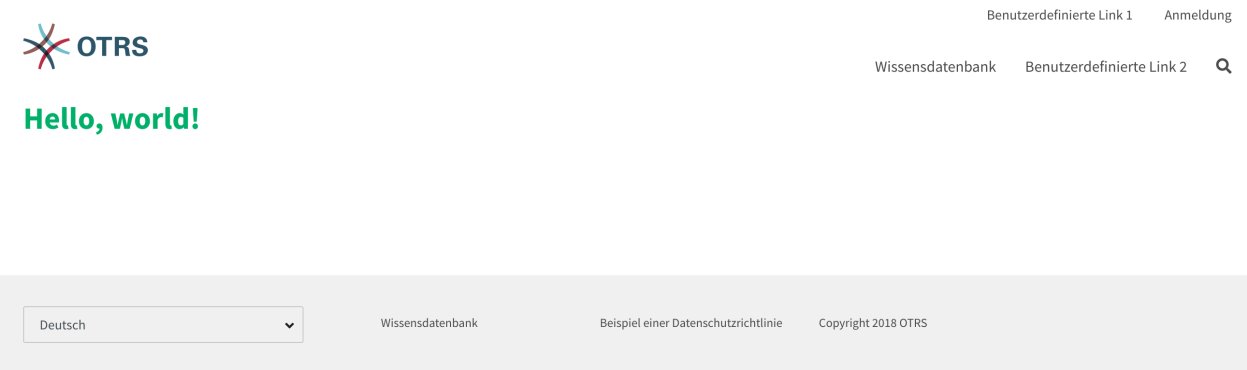
A stílusblokkon belül hozzá kell férnie globális változók és mixinek bizonyos halmazához. Nézze meg a keretrendszer kódját a részletekért (vessen egy pillantást a `Frontend/Styles/globals.js` fájlra).

Ne feledje, hogy míg a stílusok csak akkor lesznek betöltve, amikor az összetevőre hivatkoznak, addig ezek globálisan elérhetők lesznek azután, mivel a CSS eredendően globális ugyanannál az oldalnál. Van lehetőség a stílus hatókörének beállításához kizárólag az összetevőre, ezt a `scoped` attribútumon keresztül teheti meg a stílus címkén, de ez nem mindig szükséges a blokkelem módosító megközelítés okosabb használatával az osztálynevek tervezésekor.

### 3.2.7 Paraméterek átadása az útvonal összetevőnek

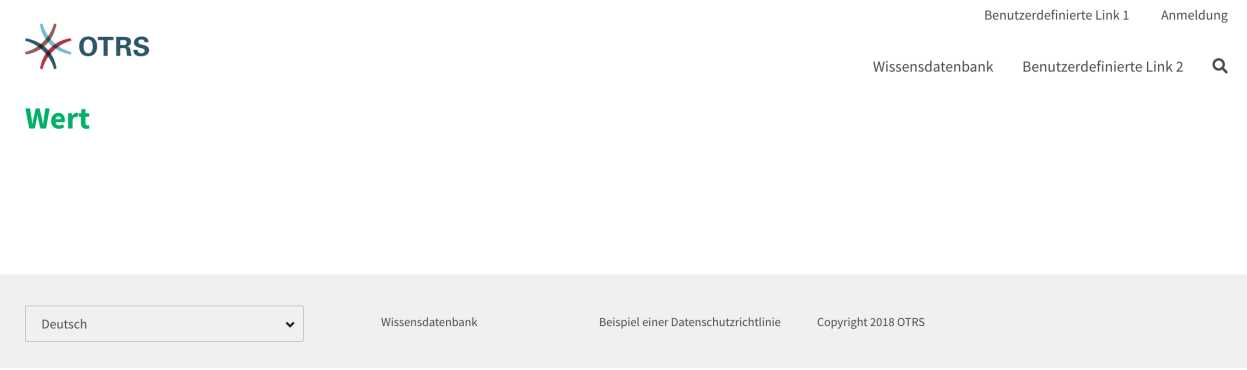
A fenti útvonal beállításban meghatároztuk az útvonal elérési útját, amely egy paraméter helykitöltőt (`headingText`) tartalmaz. A `Props` jelző aktiválásával meggyőződtünk arról, hogy ennek a paraméternek az értéke hozzá lesz kötve az összetevő prophoz ugyanazzal a névvel minden esetben, amikor az útvonalra belépnek.

Például ha belépünk az útvonalra az `/external/hello-world` URL-en keresztül, akkor az összetevő propunk meghatározatlan lesz és ennélfogva megkapja az alapértelmezett értékét.




1. ábra: Paraméterek átadása – alapértelmezett prop érték

Azonban ha az `/external/hello-world/Value` URL-en keresztül lépünk be az útvonalra, akkor a prop beállításra kerül a `Value` szöveggel, és még automatikusan le is lesz fordítva az aktuális felhasználó nyelvére (ahol alkalmazható).




2. ábra: Paraméterek átadása – lefordított prop érték

### 3.2.8 Összetevőmappák

Az önmagát tartalmazó összetevők esetén érdemes lehet néhány további fájlt is szállítani vele együtt. Néha jobb modulárissá tenni a kódbázist, mivel úgy egyszerűbb karbantartani. Az előtétprogram összetevők esetén egy igazán egyszerű módja van ennek: összetevőmappák. Egy összetevő önálló `.vue` fájlja helyett helyezze el az `index.vue` nevű fájlt egy olyan mappába, amelyet úgy nevez el, mint az összetevőt. Valahogy így:

```
HelloWorld/
HelloWorld/index.vue
```

Ezután egyszerűen adjon hozzá új fájlokat ugyanabba a mappába egy észszerű szerkezetet követve:

```
HelloWorld/
HelloWorld/index.vue
HelloWorld/Styles/_mystyles.scss
HelloWorld/Images/foobar.png
HelloWorld/Fonts/awesome-font.woff
HelloWorld/Fonts/awesome-font.woff2
HelloWorld/ChildComponent1.vue
HelloWorld/ChildComponent2/index.vue
HelloWorld/ChildComponent2/Styles/_childstyles2.scss
```

Megvan az ötlet. Ezután lehetőség van hivatkozni az új fájlokra relatív elérési úton keresztül, azért hogy valami ehhez hasonlót érjen el a szülő összetevőben (`index.vue`):

```
<template>
    <img src="./Images/foobar.png" alt="Foobar" />
</template>
```

Vagy ehhez hasonlóan:

```
<script>
export default {
    name: 'HelloWorld',

    components: {
        ChildComponent1: () => import('./ChildComponent1'),
        ChildComponent2: () => import('./ChildComponent2'),
    },
...
```

Még a külső stílusok és hivatkozhatók a helyes blokkban:

```
<style lang="scss">
@import './Styles/mystyles';
</style>
```

Ezzel a megközelítéssel otthagyhatja a csomagolt összetevőt egy önálló mappában, amely a logikus faszerkezetet követi, és egyszerűen megtalálhatóvá teszi az összes erőforrást, amikor szükséges.

### 3.2.9 További gyártók moduljainak csomagolása

Bizonyos esetekben szükség lehet további Node.js modulokat is szállítani a csomaggal. Sajnálatos módon sem az NPM, sem az OTRS nem támogatja a modulok egyszerű hozzáadását a gyökér `node_modules/` mappához, azonban létezik egy mechanizmus előre csomagolt modulfájlok biztosításához.

Egyszerűen hozzon létre egy `Frontend/Vendor` mappát a csomagjában, és adja hozzá a modul erőforrásait almappákon belül.

Például tegyük fel, hogy szállítani szeretnénk a hasznos `vue-full-calendar` összetevőt és annak függőségeit a csomagunk részeként. Ennek az összetevőnek a következő NPM függőségei vannak:

```
$ npm view vue-full-calendar dependencies
{ 'babel-plugin-transform-runtime': '^6.23.0', fullcalendar: '^3.4.0',
↪'lodash.defaultsdeep': '^4.6.0' }
```

Azonban néhány függőségének még további függőségei is lehetnek, és megvizsgálhatjuk azokat is:

```
$ npm view babel-plugin-transform-runtime dependencies
{ 'babel-runtime': '^6.22.0' }

$ npm view fullcalendar dependencies
{ jquery: '2 - 3', moment: '^2.20.1' }

$ npm view lodash.defaultsdeep dependencies
```

Egy gyors ellenőrzés tájékoztatni fog minket, hogy mind a `babel-runtime`, mind a `moment` már valójában része az OTRS keretrendszer függőségeinek:

```
/opt/otrs $ npm list babel-runtime
otrs-frontend@7.0.0-dev /ws/otrs7-mojo
  bootstrap-vue@2.0.0-rc.11
    opencollective@1.0.3
      babel-polyfill@6.23.0
        babel-runtime@6.26.0  deduped
  esdoc2@2.1.5
    babel-generator@6.26.0
      babel-messages@6.23.0
        babel-runtime@6.26.0  deduped
...

/opt/otrs $ npm list moment
otrs-frontend@7.0.0-dev /ws/otrs7-mojo
  moment-timezone@0.5.21
    moment@2.22.2
```

Ez azt jelenti, hogy nem kell ezeket a modulokat is szállítanunk, mivel már elérhetők alapértelmezetten. Miközben nehézkes az összes függőséget ellenőrizni, mégis érdemes, mert a csomagunk kisebb lesz. Ezzel meg fogjuk akadályozni az olyan problémákat, ami a keretrendszer függőségeinek felülírásából adódik, mivel a `Frontend/Vendor` nyer mindig.

Most telepítsük, amire szükségünk van, és dobjuk el, amire nincs szükségünk. A legegyszerűbb mód ennek elvégzéséhez a következő NPM parancs futtatása:

```
/ws/MyPackage $ npm install vue-full-calendar --no-save
+ vue-full-calendar@2.7.0
added 9 packages from 14 contributors in 1.883s

/ws/MyPackage $ ls -1 node_modules/
babel-plugin-transform-runtime
babel-runtime
core-js
fullcalendar
jquery
lodash.defaultsdeep
moment
regenerator-runtime
vue-full-calendar
```

Most eltávolítjuk azokat a modulokat, amelyekről tudjuk, hogy a keretrendszer biztosítja:

```
/ws/MyPackage $ rm -rf node_modules/babel-runtime node_modules/core-js node_
↪modules/moment node_modules/regenerator-runtime

/ws/MyPackage $ ls -1 node_modules/
babel-plugin-transform-runtime
fullcalendar
jquery
lodash.defaultsdeep
vue-full-calendar
```

Sokkal jobb. Most áthelyezzük a modulokat a megfelelő helyre:

```
/ws/MyPackage $ mkdir -p Frontend/Vendor
/ws/MyPackage $ mv node_modules/* Frontend/Vendor/
/ws/MyPackage $ rmdir node_modules/
```

A végső optimalizáció az lenne, hogy eltávolítja a szükségtelen fájlokat az adott modulmappából. Ez bonyolultnak bizonyulhat, de megéri, mivel tovább csökkenti a modul méretét és a fájlok számát, amelyeket fel kell venni a csomagba.

Például távolítsuk el a minimalizált JavaScript fájlokat a `fullcalendar` modulból, mert azt vettük észre, hogy a Vue összetevő csak a teljes disztribúciófájlokat használja:

```
/ws/MyPackage $ rm Frontend/Vendor/fullcalendar/dist/*.min.*
```

Biztonságosan eltávolíthatók a jQuery forrás és a minimalizált fájlok is, mivel a `fullcalendar` az eredeti disztribúciófájlokat is használja:

```
/ws/MyPackage $ rm Frontend/Vendor/jquery/dist/*.min.*
/ws/MyPackage $ rm Frontend/Vendor/jquery/external/sizzle/dist/*.min.*
/ws/MyPackage $ rm -rf Frontend/Vendor/jquery/src
```

Maradt körülbelül 100+ fájl, amelyet fel kell vennünk az SOPM fájlunkba, úgymint bármilyen egyéb megszokott csomagfájlba. Amikor ezt megtesszük, akkor ezek a függőségek jelen lesznek és feloldhatóvá válnak a célrendszeren:

```
/ws/MyPackage $ ls -la Frontend/Vendor
Frontend/Vendor
Frontend/Vendor/vue-full-calendar
Frontend/Vendor/vue-full-calendar/.babelrc
Frontend/Vendor/vue-full-calendar/LICENSE
Frontend/Vendor/vue-full-calendar/tests
Frontend/Vendor/vue-full-calendar/tests/fullcalendar.spec.js
Frontend/Vendor/vue-full-calendar/index.js
...
```

## 3.3 Az OTRS modulrétegek erejének használata

Az OTRS nagyszámú úgynevezett *modulréteggel* rendelkezik, amely nagyon egyszerűvé teszi a rendszer kibővítését a meglévő kód foltozása nélkül. Egy példa erre a számelőállító mechanizmus a jegyeknél. Ez egy csatlakoztatható modulokkal rendelkező *modulréteg*, és ha szeretné, hozzáadhatja a saját egyéni számelőállító moduljait is. Nézzük meg részletesen a különböző rétegeket!

### 3.3.1 Ügyintézői hitelesítő modul

Számos ügyintézői hitelesítő modul létezik (DB, LDAP és HTTPBasicAuth), amelyek az OTRS keretrendszerrel érkeznek. Lehetőség van saját hitelesítő modulok fejlesztésére is. Az ügyintézői hitelesítő modulok a `Kernel/System/Auth/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. Ezt követően egy egyszerű ügyintézői hitelesítő modul példája található. Mentse el a `Kernel/System/Auth/Simple.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `GetOption()` és `Auth()`. Adja vissza az uid-t, és ezután a hitelesítés rendben van.

#### Ügyintézői hitelesítő modul kódpélda

A felületosztály neve `Kernel::System::Auth`. A példa ügyintézői hitelesítés hívható `Kernel::System::Auth::CustomAuth` néven. Lent találhat egy példát.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Auth::CustomAuth;

use strict;
use warnings;

use Authen::CustomAuth;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(LogObject ConfigObject DBObject)) {
        $Self->{$_} = $Param{$_} || die "No $_!";
    }

    # Debug 0=off 1=on
    $Self->{Debug} = 0;

    # get config
    $Self->{Die} = $Self->{ConfigObject}->Get( 'AuthModule::CustomAuth::Die' .
↪$Param{Count} );

    # get user table
    $Self->{CustomAuthHost} = $Self->{ConfigObject}->Get(
↪'AuthModule::CustomAuth::Host' . $Param{Count} )
        || die "Need AuthModule::CustomAuth::Host$Param{Count}.";
    $Self->{CustomAuthSecret}
```

(continues on next page)

(folytatás az előző oldalról)

```
        = $Self->{ConfigObject}->Get( 'AuthModule::CustomAuth::Password' .
↪$Param{Count} )
        || die "Need AuthModule::CustomAuth::Password$Param{Count}.";

    return $Self;
}

sub GetOption {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{What} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need What!"␣
↪);
        return;
    }

    # module options
    my %Option = ( PreAuth => 0, );

    # return option
    return $Option{ $Param{What} };
}

sub Auth {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{User} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need User!"␣
↪);
        return;
    }

    # get params
    my $User       = $Param{User}       || '';
    my $Pw         = $Param{Pw}         || '';
    my $RemoteAddr = $ENV{REMOTE_ADDR} || 'Got no REMOTE_ADDR env!';
    my $UserID     = '';
    my $GetPw      = '';

    # just in case for debug!
    if ( $Self->{Debug} > 0 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: '$User' tried to authenticate with Pw: '$Pw' (
↪$RemoteAddr)",
        );
    }

    # just a note
    if ( !$User ) {
```

(continues on next page)

(folytatás az előző oldalról)

```
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "No User given!!! (REMOTE_ADDR: $RemoteAddr)",
        );
        return;
    }

    # just a note
    if ( !$Pw ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User authentication without Pw!!! (REMOTE_
↪ADDR: $RemoteAddr)",
        );
        return;
    }

    # Create a RADIUS object
    my $CustomAuth = Authen::CustomAuth->new(
        Host   => $Self->{CustomAuthHost},
        Secret => $Self->{CustomAuthecret},
    );
    if ( !$CustomAuth ) {
        if ( $Self->{Die} ) {
            die "Can't connect to $Self->{CustomAuthHost}: $@";
        }
        else {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Can't connect to $Self->{CustomAuthHost}: $@",
            );
            return;
        }
    }
    my $AuthResult = $CustomAuth->check_pwd( $User, $Pw );

    # login note
    if ( defined($AuthResult) && $AuthResult == 1 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User authentication ok (REMOTE_ADDR:
↪$RemoteAddr).",
        );
        return $User;
    }

    # just a note
    else {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User authentication with wrong Pw!!! (REMOTE_
↪ADDR: $RemoteAddr)"
```

(continues on next page)

(folytatás az előző oldalról)

```
        );
        return;
    }
}

1;
```

#### Ügyintézői hitelesítő modul beállítási példa

Szükség van az egyéni ügyintézői hitelesítés modul bekapcsolására. Ezt a lenti Perl beállítás használatával lehet megtenni. Nem ajánlott az XML beállítás használata, mert kizárhatja magát a rendszerbeállításokon keresztül.

```
$Self->{'AuthModule'} = 'Kernel::System::Auth::CustomAuth';
```

#### Ügyintézői hitelesítő modul használati eset példa

Egy hitelesítési megvalósítás hasznos példája lehet egy SOAP háttérprogram.

### 3.3.2 Hitelesítés szinkronizációs modul

Létezik egy LDAP hitelesítés szinkronizációs modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját hitelesítés modulok fejlesztésére is. A hitelesítés szinkronizációs modulok a `Kernel/System/Auth/Sync/*.pm` alatt találhatók. A beállításaikkal kapcsolatban további információkért nézze meg az adminisztrációs kézikönyvet. A következőkben egy hitelesítés szinkronizációs modul példája található. Mentse el a `Kernel/System/Auth/Sync/CustomAuthSync.pm` fájlba. Mindössze két függvényre van szüksége: `new()` és `Sync()`. Adjon vissza 1-et, és ezután a szinkronizáció rendben van.

#### Hitelesítés szinkronizációs modul kódpélda

A felületosztály neve `Kernel::System::Auth`. A példa ügyintézői hitelesítés hívható `Kernel::System::Auth::Sync::CustomAuthSync` néven. Lent találhat egy példát.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Auth::Sync::CustomAuthSync;

use strict;
use warnings;
use Net::LDAP;

sub new {
```

(continues on next page)

(folytatás az előző oldalról)

```
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(LogObject ConfigObject DBObject UserObject GroupObject␣
↪EncodeObject)) {
        $Self->{$_} = $Param{$_} || die "No $_!";
    }

    # Debug 0=off 1=on
    $Self->{Debug} = 0;

...

    return $Self;
}

sub Sync {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(User)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!
↪" );
            return;
        }
    }
...
    return 1;
}
```

### Hitelesítés szinkronizációs modul beállítási példa

Be kell kapcsolnia az egyéni szinkronizációs modult. Ezt a lenti Perl beállítás használatával lehet megtenni. Nem ajánlott az XML beállítás használata, mert az lehetővé teheti, hogy kizárja magát a rendszerbeállításokon keresztül.

```
$Self->{'AuthSyncModule'} = 'Kernel::System::Auth::Sync::LDAP';
```

### Hitelesítés szinkronizációs modul használati eset példa

Hasznos szinkronizációs megvalósítás lehet egy SOAP vagy egy RADIUS háttérprogram.

### 3.3.3 Ügyfél hitelesítő modul

Számos ügyfél hitelesítő modul létezik (DB, LDAP és HTTPBasicAuth), amelyek az OTRS keretrendszerrel érkeznek. Lehetőség van saját hitelesítő modulok fejlesztésére is. Az ügyfél hitelesítő modulok a `Kernel/System/CustomerAuth/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. Ezt követően egy egyszerű ügyfél hitelesítő modul példája található. Mentse el a `Kernel/System/CustomerAuth/Simple.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `GetOption()` és `Auth()`. Adja vissza az uid-t, és ezután a hitelesítés rendben van.

#### Ügyfél hitelesítő modul kódpélda

A felületosztály neve `Kernel::System::CustomerAuth`. A példa ügyfél hitelesítés hívható `Kernel::System::CustomerAuth::CustomAuth` néven. Lent találhat egy példát.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::CustomerAuth::CustomAuth;

use strict;
use warnings;

use Authen::CustomAuth;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(LogObject ConfigObject DBObject)) {
        $Self->{$_} = $Param{$_} || die "No $_!";
    }

    # Debug 0=off 1=on
    $Self->{Debug} = 0;

    # get config
    $Self->{Die}
        = $Self->{ConfigObject}->Get( 'Customer::AuthModule::CustomAuth::Die' .
↪ $Param{Count} );

    # get user table
    $Self->{CustomAuthHost}
        = $Self->{ConfigObject}->Get( 'Customer::AuthModule::CustomAuth::Host'␣
↪. $Param{Count} )
```

(continues on next page)

(folytatás az előző oldalról)

```
        || die "Need Customer::AuthModule::CustomAuth::Host$Param{Count} in␣
↪Kernel/Config.pm";
    $Self->{CustomAuthSecret}
        = $Self->{ConfigObject}->Get(
↪'Customer::AuthModule::CustomAuth::Password' . $Param{Count} )
        || die "Need Customer::AuthModule::CustomAuth::Password$Param{Count}␣
↪in Kernel/Config.pm";

    return $Self;
}

sub GetOption {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{What} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need What!"␣
↪);
        return;
    }

    # module options
    my %Option = ( PreAuth => 0, );

    # return option
    return $Option{ $Param{What} };
}

sub Auth {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{User} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need User!"␣
↪);
        return;
    }

    # get params
    my $User       = $Param{User}      || '';
    my $Pw         = $Param{Pw}        || '';
    my $RemoteAddr = $ENV{REMOTE_ADDR} || 'Got no REMOTE_ADDR env!';
    my $UserID     = '';
    my $GetPw      = '';

    # just in case for debug!
    if ( $Self->{Debug} > 0 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: '$User' tried to authentificate with Pw: '$Pw' (
↪$RemoteAddr)",
        );
```

(continues on next page)

(folytatás az előző oldalról)

```
    }

    # just a note
    if ( !$User ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "No User given!!! (REMOTE_ADDR: $RemoteAddr)",
        );
        return;
    }

    # just a note
    if ( !$Pw ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User Authentication without Pw!!! (REMOTE_
↪ADDR: $RemoteAddr)",
        );
        return;
    }

    # Create a custom object
    my $CustomAuth = Authen::CustomAuth->new(
        Host   => $Self->{CustomAuthHost},
        Secret => $Self->{CustomAuthSecret},
    );
    if ( !$CustomAuth ) {
        if ( $Self->{Die} ) {
            die "Can't connect to $Self->{CustomAuthHost}: $@";
        }
        else {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Can't connect to $Self->{CustomAuthHost}: $@",
            );
            return;
        }
    }
    my $AuthResult = $CustomAuth->check_pwd( $User, $Pw );

    # login note
    if ( defined($AuthResult) && $AuthResult == 1 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User Authentication ok (REMOTE_ADDR:
↪$RemoteAddr).",
        );
        return $User;
    }

    # just a note
    else {
```

(continues on next page)

(folytatás az előző oldalról)

```
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User Authentication with wrong Pw!!! (REMOTE_
↪ADDR: $RemoteAddr)"
        );
        return;
    }
}

1;
```

#### Ügyfél hitelesítő modul beállítási példa

Szükség van az egyéni ügyfél hitelesítő modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="AuthModule" Required="1" Valid="1">
    <Description Translatable="1">Module to authenticate customers.</
↪Description>
    <Group>Framework</Group>
    <SubGroup>Frontend::CustomerAuthAuth</SubGroup>
    <Setting>
        <Option Location="Kernel/System/CustomerAuth/*.pm" SelectedID=
↪"Kernel::System::CustomerAuth::CustomAuth"></Option>
    </Setting>
</ConfigItem>
```

#### Ügyfél hitelesítő modul használati eset példa

Hasznos hitelesítés megvalósítás lehet egy SOAP háttérprogram.

### 3.3.4 Ügyfél-felhasználó beállítások modul

Létezik egy DB ügyfél-felhasználó beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját ügyfél-felhasználó beállítási modulok fejlesztésére is. Az ügyfél-felhasználó beállítási modulok a `Kernel/System/CustomerUser/Preferences/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy ügyfél-felhasználó beállítások modul példája található. Mentse el a `Kernel/System/CustomerUser/Preferences/Custom.pm` helyre. Mindössze négy függvényre van szüksége: `new()`, `SearchPreferences()`, `SetPreferences()` és `GetPreferences()`.

#### Ügyfél-felhasználó beállítások modul kódpélda

A felületosztály neve `Kernel::System::CustomerUser`. A példa ügyfél-felhasználó beállítások hívhatók `Kernel::System::CustomerUser::Preferences::Custom` néven. Lent találhat egy példát.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
```

(continues on next page)

(folytatás az előző oldalról)

```
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::CustomerUser::Preferences::Custom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for my $Object (qw(DBObject ConfigObject LogObject)) {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }

    # preferences table data
    $Self->{PreferencesTable} = $Self->{ConfigObject}->Get('CustomerPreferences
↪')->{Params}->{Table}
        || 'customer_preferences';
    $Self->{PreferencesTableKey}
        = $Self->{ConfigObject}->Get('CustomerPreferences')->{Params}->
↪{TableKey}
        || 'preferences_key';
    $Self->{PreferencesTableValue}
        = $Self->{ConfigObject}->Get('CustomerPreferences')->{Params}->
↪{TableValue}
        || 'preferences_value';
    $Self->{PreferencesTableUserID}
        = $Self->{ConfigObject}->Get('CustomerPreferences')->{Params}->
↪{TableUserID}
        || 'user_id';

    return $Self;
}

sub SetPreferences {
    my ( $Self, %Param ) = @_;

    my $UserID = $Param{UserID} || return;
    my $Key    = $Param{Key}    || return;
    my $Value = defined( $Param{Value} ) ? $Param{Value} : '';
```

(continues on next page)

(folytatás az előző oldalról)

```
    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . " $Self->{PreferencesTableUserID} = ? AND $Self->
↪{PreferencesTableKey} = ?",
        Bind => [ \$UserID, \$Key ],
    );

    $Value .= 'Custom';

    # insert new data
    return if !$Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->
↪{PreferencesTableUserID}, "
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue})
↪"
            . " VALUES (?, ?, ?)",
        Bind => [ \$UserID, \$Key, \$Value ],
    );

    return 1;
}

sub GetPreferences {
    my ( $Self, %Param ) = @_;

    my $UserID = $Param{UserID} || return;
    my %Data;

    # get preferences

    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->
↪{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->
↪{PreferencesTableUserID} = ?",
        Bind => [ \$UserID ],
    );
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
    return %Data;
}

sub SearchPreferences {
    my ( $Self, %Param ) = @_;

    my %UserID;
    my $Key   = $Param{Key}   || '';
    my $Value = $Param{Value} || '';
```

(continues on next page)

(folytatás az előző oldalról)

```
    # get preferences
    my $SQL = "SELECT $Self->{PreferencesTableUserID}, $Self->
↪{PreferencesTableValue} "
        . " FROM "
        . " $Self->{PreferencesTable} "
        . " WHERE "
        . " $Self->{PreferencesTableKey} = '"
        . $Self->{DBObject}->Quote($Key) . "'" . " AND "
        . " LOWER($Self->{PreferencesTableValue}) LIKE LOWER('"
        . $Self->{DBObject}->Quote( $Value, 'Like' ) . "')";

    return if !$Self->{DBObject}->Prepare( SQL => $SQL );
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $UserID{ $Row[0] } = $Row[1];
    }

    # return data
    return %UserID;
}

1;
```

### Ügyfél-felhasználó beállítások modul beállítási példa

Szükség van az egyéni ügyfél-felhasználó beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="CustomerPreferences" Required="1" Valid="1">
    <Description Translatable="1">Parameters for the customer preference␣
↪table.</Description>
    <Group>Framework</Group>
    <SubGroup>Frontend::Customer::Preferences</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">
↪Kernel::System::CustomerUser::Preferences::Custom</Item>
            <Item  Key="Params">
                <Hash>
                    <Item Key="Table">customer_preferences</Item>
                    <Item Key="TableKey">preferences_key</Item>
                    <Item Key="TableValue">preferences_value</Item>
                    <Item Key="TableUserID">user_id</Item>
                </Hash>
            </Item>
        </Hash>
    </Setting>
</ConfigItem>
```

#### Ügyfél-felhasználó beállítások modul használati eset példa

Hasznos beállítások megvalósítás lehet egy SOAP vagy egy LDAP háttérprogram.

### 3.3.5 Várólista beállítások modul

Létezik egy DB várólista beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját várólista beállítási modulok fejlesztésére is. A várólista beállítási modulok a `Kernel/System/Queue/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy várólista beállítások modul példája található. Mentse el a `Kernel/System/Queue/PreferencesCustom.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `QueuePreferencesSet()` és `QueuePreferencesGet()`. Adjon vissza 1-et, és ezután a szinkronizáció rendben van.

#### Várólista beállítások kódpélda

A felületosztály neve `Kernel::System::Queue`. A példa várólista beállítások hívhatók `Kernel::System::Queue::PreferencesCustom` néven. Lent találhat egy példát.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Queue::PreferencesCustom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(DBObject ConfigObject LogObject)) {
        $Self->{$_} = $Param{$_} || die "Got no $_!";
    }

    # preferences table data
    $Self->{PreferencesTable}        = 'queue_preferences';
    $Self->{PreferencesTableKey}     = 'preferences_key';
    $Self->{PreferencesTableValue}   = 'preferences_value';
    $Self->{PreferencesTableQueueID} = 'queue_id';
```

(continues on next page)

(folytatás az előző oldalról)

```
    return $Self;
}

sub QueuePreferencesSet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(QueueID Key Value)) {
        if ( !defined( $Param{$_} ) ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!
↪" );
            return;
        }
    }

    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . "$Self->{PreferencesTableQueueID} = ? AND $Self->
↪{PreferencesTableKey} = ?",
        Bind => [ \$Param{QueueID}, \$Param{Key} ],
    );

    $Self->{PreferencesTableValue} .= 'PreferencesCustom';

    # insert new data
    return $Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->
↪{PreferencesTableQueueID}, "
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue})
↪"
            . " VALUES (?, ?, ?)",
        Bind => [ \$Param{QueueID}, \$Param{Key}, \$Param{Value} ],
    );
}

sub QueuePreferencesGet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(QueueID)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!
↪" );
            return;
        }
    }

    # check if queue preferences are available
    if ( !$Self->{ConfigObject}->Get('QueuePreferences') ) {
        return;
    }
```

(continues on next page)

(folytatás az előző oldalról)

```
    # get preferences
    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->
↪{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->
↪{PreferencesTableQueueID} = ?",
        Bind => [ \$Param{QueueID} ],
    );
    my %Data;
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
    return %Data;
}

1;
```

#### Várólista beállítások beállítási példa

Szükség van az egyéni várólista beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="Queue::PreferencesModule" Required="1" Valid="1">
    <Description Translatable="1">Default queue preferences module.</
↪Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Queue::Preferences</SubGroup>
    <Setting>
        <String Regex="">Kernel::System::Queue::PreferencesCustom</String>
    </Setting>
</ConfigItem>
```

#### Várólista beállítások használati eset példa

Hasznos beállítások megvalósítás lehet egy SOAP vagy egy RADIUS háttérprogram.

### 3.3.6 Szolgáltatás beállítások modul

Létezik egy DB szolgáltatás beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját szolgáltatás beállítási modulok fejlesztésére is. A szolgáltatás beállítási modulok a `Kernel/System/Service/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy szolgáltatás beállítások modul példája található. Mentse el a `Kernel/System/Service/PreferencesCustom.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `ServicePreferencesSet()` és `ServicePreferencesGet()`. Adjon vissza 1-et, és ezután a szinkronizáció rendben van.

### Szolgáltatás beállítások kódpélda

A felületosztály neve `Kernel::System::Service`. A példa szolgáltatás beállítások hívhatók `Kernel::System::Service::PreferencesCustom` néven. Lent találhat egy példát.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Service::PreferencesCustom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(DBObject ConfigObject LogObject)) {
        $Self->{$_} = $Param{$_} || die "Got no $_!";
    }

    # preferences table data
    $Self->{PreferencesTable}          = 'service_preferences';
    $Self->{PreferencesTableKey}       = 'preferences_key';
    $Self->{PreferencesTableValue}     = 'preferences_value';
    $Self->{PreferencesTableServiceID} = 'service_id';

    return $Self;
}

sub ServicePreferencesSet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(ServiceID Key Value)) {
        if ( !defined( $Param{$_} ) ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!
↪" );
            return;
        }
    }
```

(continues on next page)

(folytatás az előző oldalról)

```
    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . "$Self->{PreferencesTableServiceID} = ? AND $Self->
↪{PreferencesTableKey} = ?",
        Bind => [ \$Param{ServiceID}, \$Param{Key} ],
    );

$Self->{PreferencesTableValue} .= 'PreferencesCustom';

    # insert new data
    return $Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->
↪{PreferencesTableServiceID}, "
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue})
↪"
            . " VALUES (?, ?, ?)",
        Bind => [ \$Param{ServiceID}, \$Param{Key}, \$Param{Value} ],
    );
}

sub ServicePreferencesGet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(ServiceID)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!
↪" );
            return;
        }
    }

    # check if service preferences are available
    if ( !$Self->{ConfigObject}->Get('ServicePreferences') ) {
        return;
    }

    # get preferences
    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->
↪{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->
↪{PreferencesTableServiceID} = ?",
        Bind => [ \$Param{ServiceID} ],
    );
    my %Data;
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
```

(continues on next page)

(folytatás az előző oldalról)

```
    return %Data;
}

1;
```

#### Szolgáltatás beállítások beállítási példa

Szükség van az egyéni szolgáltatás beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="Service::PreferencesModule" Required="1" Valid="1">
    <Description Translatable="1">Default service preferences module.</
↪Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Service::Preferences</SubGroup>
    <Setting>
        <String Regex="">Kernel::System::Service::PreferencesCustom</String>
    </Setting>
</ConfigItem>
```

#### Szolgáltatás beállítások használati eset példa

Hasznos beállítások megvalósítás lehet egy SOAP vagy egy RADIUS háttérprogram.

### 3.3.7 SLA beállítások modul

Létezik egy DB SLA beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját SLA beállítási modulok fejlesztésére is. Az SLA beállítási modulok a `Kernel/System/SLA/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy SLA beállítások modul példája található. Mentse el a `Kernel/System/SLA/PreferencesCustom.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `SLAPreferencesSet()` és `SLAPreferencesGet()`. Győződjön meg arról, hogy a függvény 1-et adjon vissza.

#### SLA beállítások kódpélda

A felületosztály neve `Kernel::System::SLA`. A példa SLA beállítások hívhatók `Kernel::System::SLA::PreferencesCustom` néven. Lent találhat egy példát.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::SLA::PreferencesCustom;
```

(continues on next page)

(folytatás az előző oldalról)

```
use strict;
use warnings;

use vars qw(@ISA);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(DBObject ConfigObject LogObject)) {
        $Self->{$_} = $Param{$_} || die "Got no $_!";
    }

    # preferences table data
    $Self->{PreferencesTable}      = 'sla_preferences';
    $Self->{PreferencesTableKey}   = 'preferences_key';
    $Self->{PreferencesTableValue} = 'preferences_value';
    $Self->{PreferencesTableSLAID} = 'sla_id';

    return $Self;
}

sub SLAPreferencesSet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(SLAID Key Value)) {
        if ( !defined( $Param{$_} ) ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!
↪" );
            return;
        }
    }

    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . "$Self->{PreferencesTableSLAID} = ? AND $Self->
↪{PreferencesTableKey} = ?",
        Bind => [ \$Param{SLAID}, \$Param{Key} ],
    );

$Self->{PreferencesTableValue} .= 'PreferencesCustom';

    # insert new data
    return $Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->
↪{PreferencesTableSLAID}, "
```

(continues on next page)

(folytatás az előző oldalról)

```
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue})
↪"
            . " VALUES (?, ?, ?)",
        Bind => [ \$Param{SLAID}, \$Param{Key}, \$Param{Value} ],
    );
}

sub SLAPreferencesGet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(SLAID)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!
↪" );
            return;
        }
    }

    # check if SLA preferences are available
    if ( !$Self->{ConfigObject}->Get('SLAPreferences') ) {
        return;
    }

    # get preferences
    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->
↪{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->
↪{PreferencesTableSLAID} = ?",
        Bind => [ \$Param{SLAID} ],
    );
    my %Data;
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
    return %Data;
}

1;
```

### SLA beállítások beállítási példa

Szükség van az egyéni SLA beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="SLA::PreferencesModule" Required="1" Valid="1">
    <Description Translatable="1">Default SLA preferences module.</
↪Description>
```

(continues on next page)

(folytatás az előző oldalról)

```
    <Group>Ticket</Group>
    <SubGroup>Frontend::SLA::Preferences</SubGroup>
    <Setting>
        <String Regex="">Kernel::System::SLA::PreferencesCustom</String>
    </Setting>
</ConfigItem>
```

#### SLA beállítások használati eset példa

Hasznos beállítások megvalósítás lehet további értékek tárolása az SLA-khoz.

### 3.3.8 Naplózás modul

Létezik egy globális naplózó felület az OTRS-hez, amely a saját naplózó háttérprogramok létrehozásának lehetőségét biztosítja.

Egy saját naplózó háttérprogram írása olyan egyszerű, mint újra megvalósítani a `Kernel::System::Log::Log()` metódust.

#### Naplózás modul kódpélda

Ebben a kis példában írni fogunk egy kicsi fájl naplózó háttérprogramot, amely hasonlóan működik mint a `Kernel::System::Log::File`, de egy szöveget fűz minden naplóbejegyzés elé.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Log::CustomFile;

use strict;
use warnings;

umask "002";

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # get needed objects
    for (qw(ConfigObject EncodeObject)) {
        if ( $Param{$_} ) {
            $Self->{$_} = $Param{$_};
```

(continues on next page)

(folytatás az előző oldalról)

```
        }
        else {
            die "Got no $_!";
        }
    }

    # get logfile location
    $Self->{LogFile} = '/var/log/CustomFile.log';

    # set custom prefix
    $Self->{CustomPrefix} = 'CustomFileExample';

    # Fixed bug# 2265 - For IIS we need to create a own error log file.
    # Bind stderr to log file, because IIS do print stderr to web page.
    if ( $ENV{SERVER_SOFTWARE} && $ENV{SERVER_SOFTWARE} =~ /^microsoft\-iis/i␣
↪) {
        if ( !open STDERR, '>>', $Self->{LogFile} . '.error' ) {
            print STDERR "ERROR: Can't write $Self->{LogFile}.error: $!";
        }
    }

    return $Self;
}

sub Log {
    my ( $Self, %Param ) = @_;

    my $FH;

    # open logfile
    if ( !open $FH, '>>', $Self->{LogFile} ) {

        # print error screen
        print STDERR "\n";
        print STDERR " >> Can't write $Self->{LogFile}: $! <<\n";
        print STDERR "\n";
        return;
    }

    # write log file
    $Self->{EncodeObject}->SetIO($FH);
    print $FH '[' . localtime() . ']';
    if ( lc $Param{Priority} eq 'debug' ) {
        print $FH "[Debug][$Param{Module}][$Param{Line}] $Self->{CustomPrefix}
↪$Param{Message}\n";
    }
    elsif ( lc $Param{Priority} eq 'info' ) {
        print $FH "[Info][$Param{Module}]  $Self->{CustomPrefix} $Param
↪{Message}\n";
    }
    elsif ( lc $Param{Priority} eq 'notice' ) {
        print $FH "[Notice][$Param{Module}] $Self->{CustomPrefix} $Param
↪{Message}\n";
```

(continues on next page)

(folytatás az előző oldalról)

```
    }
    elsif ( lc $Param{Priority} eq 'error' ) {
        print $FH "[Error][$Param{Module}][$Param{Line}] $Self->{CustomPrefix}
↪$Param{Message}\n";
    }
    else {

        # print error messages to STDERR
        print STDERR
            "[Error][$Param{Module}] $Self->{CustomPrefix} Priority: '$Param
↪{Priority}' not defined! Message: $Param{Message}\n";

        # and of course to logfile
        print $FH
            "[Error][$Param{Module}] $Self->{CustomPrefix} Priority: '$Param
↪{Priority}' not defined! Message: $Param{Message}\n";
    }

    # close file handle
    close $FH;
    return 1;
}

1;
```

### Naplózás modul beállítási példa

Az egyéni naplózómodul bekapcsolásához az adminisztrátor beállíthatja kézzel a meglévő `LogModule` konfigurációs elemet a `Kernel::System::Log::CustomFile` osztályhoz. Ennek automatikus megvalósításához megadhat egy XML beállítófájlt, amely felülbírálja az alapértelmezett beállítást.

```
<ConfigItem Name="LogModule" Required="1" Valid="1">
    <Description Translatable="1">Set Kernel::System::Log::CustomFile as␣
↪default logging backend.</Description>
    <Group>Framework</Group>
    <SubGroup>Core::Log</SubGroup>
    <Setting>
        <Option Location="Kernel/System/Log/*.pm" SelectedID=
↪"Kernel::System::Log::CustomFile"></Option>
    </Setting>
</ConfigItem>
```

### Naplózás modul használati eset példa

Hasznos naplózó háttérprogram lehet egy webszolgáltatásba vagy egy titkosított fájlba történő naplózás.

---

**Megjegyzés:** A `Kernel::System::Log` a `Log()` metóduson kívül egyéb metódusokkal is rendelkezik, amelyeket nem lehet újra megvalósítani, például az osztott memóriaszakaszokkal történő munkához tartozó kód és a naplóadatok gyorsítótárazása.

---

### 3.3.9 Kimenetszűrő

A kimenetszűrők lehetővé teszik a HTML módosítását röptében. A bevált gyakorlat a kimenetszűrők használata a `.tt` fájlok közvetlen módosítása helyett. Három jó ok létezik erre. Amikor ugyanazt az átdolgozást kell alkalmazni számos előtétprogram modulon, akkor az átdolgozást csak egyszer kell megvalósítani. A második előnye, hogy amikor az OTRS-t frissítik, akkor megvan az esély arra, hogy a szűrőt nem kell frissíteni, ha a hozzá tartozó minta nem változott. Amikor két kiterjesztés ugyanazt a fájlt módosítja, akkor ütközés lép fel a második csomag telepítése során. Ez az ütközés feloldható két kimenetszűrő használatával, amelyek ugyanazt az előtétprogram modult módosítják.

Három különböző fajta kimenetszűrő létezik. Ezek a HTML tartalom előállításának különböző szakaszaiban aktívak.

#### FilterElementPost

Ezek a szűrők lehetővé teszik egy sablon kimenetének módosítást, miután az megjelenítésre került.

A tartalom lefordításához futtathatja közvetlenül a `$LayoutObject->Translate()` függvényt. Ha egyéb sablonszolgáltatásokra van szüksége, akkor egyszerűen határozzon meg egy kis sablonfájlt a kimenetszűrőhöz, és használja azt a tartalom megjelenítéséhez, mielőtt beültetné azt a fő adatokba. Néhány esetben hasznos lehet a jQuery DOM műveletek használata is a képernyőn lévő tartalom sorrendjének megváltoztatásához vagy cseréjéhez a reguláris kifejezések használata helyett. Ebben az esetben láthatatlan tartalomként kellene beültetnie az új kódot valahova az oldalba (például a `Hidden` osztállyal), majd ezután áthelyezni a jQuery használatával a megfelelő helyre a DOM-ban, és megjeleníteni azt.

Az utó-kimenetszűrők használatának megkönnyítéséhez létezik egy mechanizmus is a HTML megjegyzéshorgok lekéréséhez bizonyos sablonoknál vagy blokkoknál. Hozzáadhatja a modulbeállító XML-be a következőhöz hasonlóan:

```
<Setting Name="Frontend::Template::GenerateBlockHooks###100-OTRSBusiness-
↪ContactWithData" Required="1" Valid="1">
    <Description Translatable="1">Generate HTML comment hooks for the␣
↪specified blocks so that filters can use them.</Description>
    <Navigation>Frontend::Base::OutputFilter</Navigation>
    <Value>
        <Hash>
            <Item Key="AgentTicketZoom">
                <Array>
                    <Item>CustomerTable</Item>
                </Array>
            </Item>
        </Hash>
    </Value>
</Setting>
```

Ez azt fogja okozni, hogy az `AgentTicketZoom.tt` fájlban lévő `CustomerTable` blokk át lesz alakítva a HTML megjegyzésekben minden alkalommal, amikor megjelenítésre kerül:

```
<!--HookStartCustomerTable-->
... block output ...
<!--HookEndCustomerTable-->
```

Ezzel a mechanizmussal minden csomag csak azokat a blokkhorgokat kérheti, amelyekre szüksége van, és következetesen kerülnek megjelenítésre. Ezek a HTML megjegyzések használhatók ezután a kimenetszűrőben az egyszerű reguláris kifejezés illesztéshez.

#### FilterContent

Ez a fajta szűrő lehetővé teszi a teljes HTML kimenet feldolgozását a kérésnél közvetlenül azelőtt, hogy kiküldésre kerül a böngészőnek. Ez globális átalakításokhoz használható.

#### FilterText

Ez a fajta kimenetszűrő egy bővítmény a `Kernel::Output::HTML::Layout::Ascii2HTML()` metódushoz, és csak akkor aktív, amikor a `LinkFeature` paraméter 1-re van állítva. Így a `FilterText` kimenetszűrők jelenleg csak az egyszerű szöveges bejegyzések törzsének megjelenítésénél aktívak. Az egyszerű szöveges bejegyzéseket a bejövő nem HTML levelek állítják elő, illetve amikor az OTRS úgy van beállítva, hogy ne használja a Rich Text szolgáltatást az előtétprogramon.

### Kimenetszűrő kódpélda

Nézze meg a TemplateModule csomagot.

### Kimenetszűrő beállítási példa

Nézze meg a TemplateModule csomagot.

### Kimenetszűrő használati eset példa

**További jegyattribútumok megjelenítése az `AgentTicketZoom` képernyőn** Ez egy `FilterElementPost` kimenetszűrővel valósítható meg.

**A szolgáltatásválasztás megjelenítése többszintű menüként** Használjon egy `FilterElementPost` szűrőt ehhez a szolgáltatáshoz. A választható szolgáltatások listája a feldolgozott sablonkimenetből dolgozható fel. A többszintű választás a szolgáltatáslistából építhető fel, és szúrható be a sablontartalomba. Egy `FilterElementPost` kimenetszűrőt kell használni ehhez.

**Hivatkozások létrehozása az egyszerű szöveges bejegyzés törzseiben** Egy biotechnológiai vállalat IPI00217472 formátumú génneveket használ az egyszerű szöveges bejegyzésekben. Egy `FilterText` kimenetszűrő használható a szekvencia-adatbázisra mutató hivatkozások létrehozásához a génneveknél, például http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-e+{[}IPI-acc:IPI00217472{]}+-vn+2 formában.

**Az aktív tartalom megtiltása** Van egy olyan tűzfalszabály, amely megtiltja az összes aktív tartalmat. Azért, hogy elkerüljük a tűzfal visszautasítását, az `<applet>` HTML-címke kiszűrhető egy `FilterContent` kimenetszűrővel.

---

**Megjegyzés:** Minden `FilterElementPost` kimenetszűrő felépítésre és futtatásra kerül minden olyan beállított sablonnál, amely szükséges az aktuális kéréshez. Így a kimenetszűrő alacsony teljesítménye vagy a szűrők nagy száma komolyan csökkentheti a teljesítményt.

---

### Bevált gyakorlatok

A rugalmasság növelésének érdekében az érintett sablonok listáját be kell állítani a rendszerbeállításokban.

### 3.3.10 Statisztikák modul

A belső statisztikamoduloknak két különböző típusa létezik - dinamikus és statikus. Ez a szakasz azt írja le, hogy az ilyen statisztikamodulok hogyan fejleszthetők.

#### Dinamikus statisztikák

A statikus statisztikamodulokkal ellentétben a dinamikus statisztikák beállíthatók az OTRS webes felületén keresztül. Ebben a szakaszban egy egyszerű statisztikamodul kerül fejlesztésre. Minden egyes dinamikus statisztikamodulnak meg kell valósítania ezeket a szubrutinokat:

- `new`
- `GetObjectName`
- `GetObjectAttributes`
- `ExportWrapper`
- `ImportWrapper`

Továbbá a modulnak meg kell valósítania vagy a `GetStatElement`, vagy a `GetStatTable` rutint. És ha az eredménytábla fejlécsorát is meg kell változtatni, akkor egy úgynevezett `GetHeaderLine` szubrutint is fejleszteni kell.

#### Statisztikák kódpélda

Ebben a szakaszban egy minta statisztikamodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Stats::Dynamic::DynamicStatsTemplate;

use strict;
use warnings;

use Kernel::System::Queue;
use Kernel::System::State;
use Kernel::System::Ticket;
```

Ez egy gyakori sabloncsomag, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva. Ezután a szükséges modulok használatának megadása következik a `use` kulcsszóval.

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
```

(continues on next page)

(folytatás az előző oldalról)

```
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for my $Object (
        qw(DBObject ConfigObject LogObject UserObject TimeObject MainObject␣
↪EncodeObject)
        )
    {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }

    # created needed objects
    $Self->{QueueObject}     = Kernel::System::Queue->new( %{$Self} );
    $Self->{TicketObject}    = Kernel::System::Ticket->new( %{$Self} );
    $Self->{StateObject}     = Kernel::System::State->new( %{$Self} );

    return $Self;
}
```

A `new` a statisztikamodul konstruktora. Ez hozza létre az osztály új példányát. A kódolási irányelveknek megfelelően az ebben a modulban szükséges más osztályok objektumait is a `new` konstruktorban kell létrehozni. A 27-29. sorban van létrehozva a statisztikák modul objektuma. A 31-37. sorban azt ellenőrzik, hogy az ebben a kódban szükséges objektumok - vagy más objektumok létrehozásánál, vagy ebben a modulban - át vannak-e adva. Ezután a többi objektum kerül létrehozásra.

```
sub GetObjectName {
    my ( $Self, %Param ) = @_;

    return 'Sample Statistics';
}
```

A `GetObjectName` visszaad egy nevet a statisztikák modulhoz. Ez az a címke, amely a lenyíló menüben jelenik meg a beállításokban, valamint a meglévő statisztikák listájában (az *objektum* oszlopban).

```
sub GetObjectAttributes {
    my ( $Self, %Param ) = @_;

    # get state list
    my %StateList = $Self->{StateObject}->StateList(
        UserID => 1,
    );

    # get queue list
    my %QueueList = $Self->{QueueObject}->GetAllQueues();

    # get current time to fix bug#3830
    my $TimeStamp = $Self->{TimeObject}->CurrentTimestamp();
    my ($Date) = split /\s+/, $TimeStamp;
    my $Today = sprintf "%s 23:59:59", $Date;

    my @ObjectAttributes = (
```

(continues on next page)

(folytatás az előző oldalról)

```
        {
            Name             => 'State',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'StateIDs',
            Block            => 'MultiSelectField',
            Values           => \%StateList,
        },
        {
            Name             => 'Created in Queue',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreatedQueueIDs',
            Block            => 'MultiSelectField',
            Translation      => 0,
            Values           => \%QueueList,
        },
        {
            Name             => 'Create Time',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreateTime',
            TimePeriodFormat => 'DateInputFormat',     # 'DateInputFormatLong',
            Block            => 'Time',
            TimeStop         => $Today,
            Values           => {
                TimeStart => 'TicketCreateTimeNewerDate',
                TimeStop  => 'TicketCreateTimeOlderDate',
            },
        },
    );

    return @ObjectAttributes;
}
```

Ebben a minta statisztikák modulban három olyan attribútumot szeretnénk szolgáltatni, amelyből a felhasználó választhat: a várólisták listáját, az állapotok listáját és egy idő legördülőt. A legördülőben megjelenített értékek lekéréséhez szükséges néhány művelet. Ebben az esetben a `StateList` és a `GetAllQueues` kerül meghívásra.

Ezután az attribútumok listája kerül létrehozásra. Minden egyes attribútum egy kivonathivatkozáson keresztül van meghatározva. Ezeket a kulcsokat használhatja:

**`Name`** A címke a webes felületen.

**`UseAsXvalue`** Ez az attribútum használható az X-tengelyen.

**`UseAsValueSeries`** Ez az attribútum használható az Y-tengelyen.

**`UseAsRestriction`** Ez az attribútum használható a korlátozásokhoz.

**`Element`** A HTML mező neve.

**Block** A blokknév a sablonfájlban (például `<OTRS_HOME>/Kernel/Output/HTML/Standard/AgentStatsEditXaxis.tt`).

**Values** Az attribútumban megjelenített értékek.

Tipp: Ha telepíti ezt a mintát, és beállít egy statisztikát néhány várólistával (mondjuk „A várólista" és „B várólista" ), akkor ezek a várólisták az egyetlenek, amelyek láthatóak lesznek a felhasználónak, amikor elindítja a statisztikát. Néha egy dinamikus legördülő vagy többválasztós mező szükséges. Ebben az esetben beállíthatja a `SelectedValues` kulcsot az attribútum meghatározásában:

```
{
    Name             => 'Created in Queue',
    UseAsXvalue      => 1,
    UseAsValueSeries => 1,
    UseAsRestriction => 1,
    Element          => 'CreatedQueueIDs',
    Block            => 'MultiSelectField',
    Translation      => 0,
    Values           => \%QueueList,
    SelectedValues   => [ @SelectedQueues ],
},
```

```
sub GetStatElement {
    my ( $Self, %Param ) = @_;

    # search tickets
    return $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'COUNT',
        Permission => 'ro',
        Limit      => 100_000_000,
        %Param,
    );
}
```

A `GetStatElement` kerül meghívásra minden egyes cellánál az eredménytáblában. Így annak számszerű értéknek kell lennie. Ebben a mintában egy egyszerű jegykeresést hajt végre. A `%Param` kivonat tartalmaz információkat a *jelenlegi* X-értékről és Y-értékről, valamint bármely korlátozásról. Így egy olyan cellánál, amelynek össze kell számolnia a *nyitott* állapotban lévő létrehozott jegyeket a *Misc* várólistánál, az átadott paraméter kivonat valahogy így néz ki:

```
'CreatedQueueIDs' => [
    '4'
],
'StateIDs' => [
    '2'
]
```

Ha a *cellánkénti* számítást el kellene kerülni, akkor a `GetStatTable` egy alternatíva. A `GetStatTable` visszaadja a sorok listáját, amely ezentúl egy tömbhivatkozások tömbje. Ez ugyanahhoz az eredményhez vezet mint a `GetStatElement` használata.

```
sub GetStatTable {
    my ( $Self, %Param ) = @_;
```

(continues on next page)

(folytatás az előző oldalról)

```
    my @StatData;

    for my $StateName ( keys %{ $Param{TableStructure} } ) {
        my @Row;
        for my $Params ( @{ $Param{TableStructure}->{$StateName} } ) {
            my $Tickets = $Self->{TicketObject}->TicketSearch(
                UserID     => 1,
                Result     => 'COUNT',
                Permission => 'ro',
                Limit      => 100_000_000,
                %{$Params},
            );

            push @Row, $Tickets;
        }

        push @StatData, [ $StateName, @Row ];
    }

    return @StatData;
}
```

A `GetStatTable` az összes olyan információt lekéri a statisztikák lekérdezéssel kapcsolatban, amelyek szükségesek. Az átadott paraméterek információkat tartalmaznak az attribútumokról (`Restrictions`, olyan attribútumok, amelyek az X/Y-tengelynél vannak használva) és a táblaszerkezetről. A táblaszerkezet egy olyan kivonathivatkozás, ahol a kulcsok az Y-tengely értékei, és azok értékei kivonathivatkozások a `GetStatElement` szubrutinokhoz használt paraméterekkel.

```
'Restrictions' => {},
'TableStructure' => {
    'closed successful' => [
        {
            'CreatedQueueIDs' => [
                '3'
            ],
            'StateIDs' => [
                '2'
            ]
        },
    ],
    'closed unsuccessful' => [
        {
            'CreatedQueueIDs' => [
                '3'
            ],
            'StateIDs' => [
                '3'
            ]
        },
    ],
},
```

(continues on next page)

(folytatás az előző oldalról)

```
'ValueSeries' => [
    {
        'Block' => 'MultiSelectField',
        'Element' => 'StateIDs',
        'Name' => 'State',
        'SelectedValues' => [
            '5',
            '3',
            '2',
            '1',
            '4'
        ],
        'Translation' => 1,
        'Values' => {
            '1' => 'new',
            '10' => 'closed with workaround',
            '2' => 'closed successful',
            '3' => 'closed unsuccessful',
            '4' => 'open',
            '5' => 'removed',
            '6' => 'pending reminder',
            '7' => 'pending auto close+',
            '8' => 'pending auto close-',
            '9' => 'merged'
        }
    }
],
'XValue' => {
    'Block' => 'MultiSelectField',
    'Element' => 'CreatedQueueIDs',
    'Name' => 'Created in Queue',
    'SelectedValues' => [
        '3',
        '4',
        '1',
        '2'
    ],
    'Translation' => 0,
    'Values' => {
        '1' => 'Postmaster',
        '2' => 'Raw',
        '3' => 'Junk',
        '4' => 'Misc'
    }
}
```

Néha a táblázat fejléceit meg kell változtatni. Ebben az esetben egy `GetHeaderLine` nevű szubrutint kell megvalósítani. Ennek a szubrutinnak egy tömbhivatkozást kell visszaadnia az oszlopfejlécekkel mint elemekkel. Ez információkat kap az átadott X-értékekkel kapcsolatban.

```
sub GetHeaderLine {
    my ( $Self, %Param ) = @_;
```

(continues on next page)

(folytatás az előző oldalról)

```
    my @HeaderLine = ('');
    for my $SelectedXValue ( @{ $Param{XValue}->{SelectedValues} } ) {
        push @HeaderLine, $Param{XValue}->{Values}->{$SelectedXValue};
    }

    return \@HeaderLine;
}
```

```
sub ExportWrapper {
    my ( $Self, %Param ) = @_;

    # wrap ids to used spelling
    for my $Use (qw(UseAsValueSeries UseAsRestriction UseAsXvalue)) {
        ELEMENT:
        for my $Element ( @{ $Param{$Use} } ) {
            next ELEMENT if !$Element || !$Element->{SelectedValues};
            my $ElementName = $Element->{Element};
            my $Values      = $Element->{SelectedValues};

            if ( $ElementName eq 'QueueIDs' || $ElementName eq
↪'CreatedQueueIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $Self->{QueueObject}->QueueLookup(␣
↪QueueID => $ID->{Content} );
                }
            }
            elsif ( $ElementName eq 'StateIDs' || $ElementName eq
↪'CreatedStateIDs' ) {
                my %StateList = $Self->{StateObject}->StateList( UserID => 1␣
↪);
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $StateList{ $ID->{Content} };
                }
            }
        }
    }
    return \%Param;
}
```

A beállított statisztikák exportálhatók XML-formátumba. De ahogy a várólistáknál, ahol ugyanazok a várólistanevek rendelkezhetnek különböző azonosítókkal a különböző OTRS példányoknál, különösen fájdalmas lehet az azonosítók exportálása (a statisztikák ekkor rossz számokat számítanának ki). Ezért egy exportálási átalakítót kell írni, hogy az azonosítók helyett neveket használjon. Ezt a statisztikák modul minden egyes *dimenziójánál* el kell végezni (X-tengely, Y-tengely és korlátozások).

Az `ImportWrapper` fordítva működik - átalakítja a nevet az azonosítóra abban a példányban, ahova a beállítások importálásra kerülnek.

Ez egy minta exportálás:

```
<?xml version="1.0" encoding="utf-8"?>

<otrs_stats>
<Cache>0</Cache>
<Description>Sample stats module</Description>
<File></File>
<Format>CSV</Format>
<Format>Print</Format>
<Object>DeveloperManualSample</Object>
<ObjectModule>Kernel::System::Stats::Dynamic::DynamicStatsTemplate</
↪ObjectModule>
<ObjectName>Sample Statistics</ObjectName>
<Permission>stats</Permission>
<StatType>dynamic</StatType>
<SumCol>0</SumCol>
<SumRow>0</SumRow>
<Title>Sample 1</Title>
<UseAsValueSeries Element="StateIDs" Fixed="1">
<SelectedValues>removed</SelectedValues>
<SelectedValues>closed unsuccessful</SelectedValues>
<SelectedValues>closed successful</SelectedValues>
<SelectedValues>new</SelectedValues>
<SelectedValues>open</SelectedValues>
</UseAsValueSeries>
<UseAsXvalue Element="CreatedQueueIDs" Fixed="1">
<SelectedValues>Junk</SelectedValues>
<SelectedValues>Misc</SelectedValues>
<SelectedValues>Postmaster</SelectedValues>
<SelectedValues>Raw</SelectedValues>
</UseAsXvalue>
<Valid>1</Valid>
</otrs_stats>
```

Most, hogy az összes szubrutin elmagyarázásra került, itt a teljes minta statisztikák modul.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Stats::Dynamic::DynamicStatsTemplate;

use strict;
use warnings;

use Kernel::System::Queue;
use Kernel::System::State;
use Kernel::System::Ticket;

```

(continues on next page)

(folytatás az előző oldalról)

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for my $Object (
        qw(DBObject ConfigObject LogObject UserObject TimeObject MainObject␣
↪EncodeObject)
        )
    {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }

    # created needed objects
    $Self->{QueueObject}     = Kernel::System::Queue->new( %{$Self} );
    $Self->{TicketObject}    = Kernel::System::Ticket->new( %{$Self} );
    $Self->{StateObject}     = Kernel::System::State->new( %{$Self} );

    return $Self;
}

sub GetObjectName {
    my ( $Self, %Param ) = @_;

    return 'Sample Statistics';
}

sub GetObjectAttributes {
    my ( $Self, %Param ) = @_;

    # get state list
    my %StateList = $Self->{StateObject}->StateList(
        UserID => 1,
    );

    # get queue list
    my %QueueList = $Self->{QueueObject}->GetAllQueues();

    # get current time to fix bug#3830
    my $TimeStamp = $Self->{TimeObject}->CurrentTimestamp();
    my ($Date) = split /\s+/, $TimeStamp;
    my $Today = sprintf "%s 23:59:59", $Date;

    my @ObjectAttributes = (
        {
            Name             => 'State',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
```

(continues on next page)

(folytatás az előző oldalról)

```
            Element          => 'StateIDs',
            Block            => 'MultiSelectField',
            Values           => \%StateList,
        },
        {
            Name             => 'Created in Queue',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreatedQueueIDs',
            Block            => 'MultiSelectField',
            Translation      => 0,
            Values           => \%QueueList,
        },
        {
            Name             => 'Create Time',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreateTime',
            TimePeriodFormat => 'DateInputFormat',    # 'DateInputFormatLong',
            Block            => 'Time',
            TimeStop         => $Today,
            Values           => {
                TimeStart => 'TicketCreateTimeNewerDate',
                TimeStop  => 'TicketCreateTimeOlderDate',
            },
        },
    );

    return @ObjectAttributes;
}

sub GetStatElement {
    my ( $Self, %Param ) = @_;

    # search tickets
    return $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'COUNT',
        Permission => 'ro',
        Limit      => 100_000_000,
        %Param,
    );
}

sub ExportWrapper {
    my ( $Self, %Param ) = @_;

    # wrap ids to used spelling
    for my $Use (qw(UseAsValueSeries UseAsRestriction UseAsXvalue)) {
        ELEMENT:
```

(continues on next page)

(folytatás az előző oldalról)

```
        for my $Element ( @{ $Param{$Use} } ) {
            next ELEMENT if !$Element || !$Element->{SelectedValues};
            my $ElementName = $Element->{Element};
            my $Values      = $Element->{SelectedValues};

            if ( $ElementName eq 'QueueIDs' || $ElementName eq
↪'CreatedQueueIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $Self->{QueueObject}->QueueLookup(␣
↪QueueID => $ID->{Content} );
                }
            }
            elsif ( $ElementName eq 'StateIDs' || $ElementName eq
↪'CreatedStateIDs' ) {
                my %StateList = $Self->{StateObject}->StateList( UserID => 1␣
↪);
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $StateList{ $ID->{Content} };
                }
            }
        }
    }
    return \%Param;
}

sub ImportWrapper {
    my ( $Self, %Param ) = @_;

    # wrap used spelling to ids
    for my $Use (qw(UseAsValueSeries UseAsRestriction UseAsXvalue)) {
        ELEMENT:
        for my $Element ( @{ $Param{$Use} } ) {
            next ELEMENT if !$Element || !$Element->{SelectedValues};
            my $ElementName = $Element->{Element};
            my $Values      = $Element->{SelectedValues};

            if ( $ElementName eq 'QueueIDs' || $ElementName eq
↪'CreatedQueueIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    if ( $Self->{QueueObject}->QueueLookup( Queue => $ID->
↪{Content} ) ) {
                        $ID->{Content}
                            = $Self->{QueueObject}->QueueLookup( Queue => $ID-
↪>{Content} );
                    }
                    else {
```

(continues on next page)

(folytatás az előző oldalról)

```
                        $Self->{LogObject}->Log(
                            Priority => 'error',
                            Message  => "Import: Can' find the queue $ID->
↪{Content}!"
                        );
                        $ID = undef;
                    }
                }
            }
            elsif ( $ElementName eq 'StateIDs' || $ElementName eq
↪'CreatedStateIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;

                    my %State = $Self->{StateObject}->StateGet(
                        Name  => $ID->{Content},
                        Cache => 1,
                    );
                    if ( $State{ID} ) {
                        $ID->{Content} = $State{ID};
                    }
                    else {
                        $Self->{LogObject}->Log(
                            Priority => 'error',
                            Message  => "Import: Can' find state $ID->{Content}
↪!"
                        );
                        $ID = undef;
                    }
                }
            }
        }
    }
    return \%Param;
}

1;
```

### Statisztikák beállítási példa

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="1.0" init="Config">
    <ConfigItem Name="Stats::DynamicObjectRegistration###DynamicStatsTemplate"␣
↪Required="0" Valid="1">
        <Description Translatable="1">Here you can decide if the common stats␣
↪module may generate stats about the number of default tickets a requester␣
↪created.</Description>
        <Group>Framework</Group>
        <SubGroup>Core::Stats</SubGroup>
```

(continues on next page)

(folytatás az előző oldalról)

```
        <Setting>
            <Hash>
                <Item Key="Module">
↪Kernel::System::Stats::Dynamic::DynamicStatsTemplate</Item>
            </Hash>
        </Setting>
    </ConfigItem>
</otrs_config>
```

**Megjegyzés:** Ha nagyon sok cellája van az eredménytáblázatban és a `GetStatElement` meglehetősen összetett, akkor a kérés eltarthat egy ideig.

### Statikus statisztikák

A következő bekezdések a statikus statisztikákat írják le. A statikus statisztikákat nagyon könnyű létrehozni, mivel ezeknek a moduloknak csak három szubrutint kell megvalósítaniuk.

- `new`
- `Param`
- `Run`

### Statikus statisztikák kódpélda

A következő bekezdések a statikus statisztikákban szükséges szubrutinokat mutatják be.

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless( $Self, $Type );

    # check all needed objects
    for my $Needed (
        qw(DBObject ConfigObject LogObject
        TimeObject MainObject EncodeObject)
        )
    {
        $Self->{$Needed} = $Param{$Needed} || die "Got no $Needed";
    }

    # create needed objects
    $Self->{TypeObject}   = Kernel::System::Type->new( %{$Self} );
    $Self->{TicketObject} = Kernel::System::Ticket->new( %{$Self} );
    $Self->{QueueObject}  = Kernel::System::Queue->new( %{$Self} );

    return $Self;
}
```

A `new` hozza létre a statikus statisztikák osztályának egy új példányát. Először létrehoz egy új objektumot, és azután ellenőrzi a szükséges objektumokat.

```
sub Param {
    my $Self = shift;

    my %Queues = $Self->{QueueObject}->GetAllQueues();
    my %Types  = $Self->{TypeObject}->TypeList(
        Valid => 1,
    );

    my @Params = (
        {
            Frontend  => 'Type',
            Name      => 'TypeIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Types,
        },
        {
            Frontend  => 'Queue',
            Name      => 'QueueIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Queues,
        },
    );

    return @Params;
}
```

A `Param` metódus biztosítja az összes olyan paraméter és attribútum listáját, amelyek kiválaszthatók egy statikus statisztika létrehozásához. Megkap néhány átadott paramétert: az értékeket egy kérésben szolgáltatott statisztikák attribútumaihoz, a statisztikák formátumát és az objektum nevét (a modul nevét).

A paramétereknek és az attribútumoknak kivonathivatkozásoknak kell lenniük ezekkel a kulcs-érték párokkal:

**`Frontend`** A címke a webes felületen.

**`Name`** A HTML mező neve.

**`Data`** Az attribútumban megjelenített értékek.

Egyéb paraméterek is használhatók a `LayoutObject BuildSelection` metódusánál, ahogy az a `Size` és `Multiple` paraméterekkel történik ebben a minta modulban.

```
sub Run {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for my $Needed (qw(TypeIDs QueueIDs)) {
        if ( !$Param{$Needed} ) {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Need $Needed!",
```

(continues on next page)

(folytatás az előző oldalról)

```
            );
            return;
        }
    }

    # set report title
    my $Title = 'Tickets per Queue';

    # table headlines
    my @HeadData = (
        'Ticket Number',
        'Queue',
        'Type',
    );

    my @Data;
    my @TicketIDs = $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'ARRAY',
        Permission => 'ro',
        %Param,
    );

    for my $TicketID ( @TicketIDs ) {
        my %Ticket = $Self->{TicketObject}->TicketGet(
            UserID => 1,
            TicketID => $TicketID,
        );
        push @Data, [ $Ticket{TicketNumber}, $Ticket{Queue}, $Ticket{Type} ];
    }

    return ( [$Title], [@HeadData], @Data );
}
```

Tulajdonképpen a `Run` metódus állítja elő a táblázat adatait a statisztikákhoz. Megkapja az ennél a statisztikánál átadott attribútumokat. Ebben a mintában a `%Param` paraméterben egy `TypeIDs` kulcs és egy `QueueIDs` kulcs létezik (lásd a `Param` metódusban lévő attribútumokat), és ezek értékei tömbhivatkozások. A visszaadott adatok három részből állnak: két tömbhivatkozásból és egy tömbből. Az első tömbhivatkozásban a statisztika címe van eltárolva, a második tömbhivatkozás tartalmazza a táblázatban lévő oszlopok címsorait. És ezután a táblázattörzs adatai következnek.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Stats::Static::StaticStatsTemplate;

use strict;
```

(continues on next page)

(folytatás az előző oldalról)

```
use warnings;

use Kernel::System::Type;
use Kernel::System::Ticket;
use Kernel::System::Queue;

=head1 NAME

StaticStatsTemplate.pm - the module that creates the stats about tickets in a␣
↪queue

=head1 SYNOPSIS

All functions

=head1 PUBLIC INTERFACE

=over 4

=cut

=item new()

create an object

    use Kernel::Config;
    use Kernel::System::Encode;
    use Kernel::System::Log;
    use Kernel::System::Main;
    use Kernel::System::Time;
    use Kernel::System::DB;
    use Kernel::System::Stats::Static::StaticStatsTemplate;

    my $ConfigObject = Kernel::Config->new();
    my $EncodeObject = Kernel::System::Encode->new(
        ConfigObject => $ConfigObject,
    );
    my $LogObject    = Kernel::System::Log->new(
        ConfigObject => $ConfigObject,
    );
    my $MainObject = Kernel::System::Main->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
    );
    my $TimeObject = Kernel::System::Time->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
    );
    my $DBObject = Kernel::System::DB->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
        MainObject   => $MainObject,
```

(continues on next page)

(folytatás az előző oldalról)

```
    );
    my $StatsObject = Kernel::System::Stats::Static::StaticStatsTemplate->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
        MainObject   => $MainObject,
        TimeObject   => $TimeObject,
        DBObject     => $DBObject,
        EncodeObject => $EncodeObject,
    );

=cut

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless( $Self, $Type );

    # check all needed objects
    for my $Needed (
        qw(DBObject ConfigObject LogObject
        TimeObject MainObject EncodeObject)
        )
    {
        $Self->{$Needed} = $Param{$Needed} || die "Got no $Needed";
    }

    # create needed objects
    $Self->{TypeObject}   = Kernel::System::Type->new( %{$Self} );
    $Self->{TicketObject} = Kernel::System::Ticket->new( %{$Self} );
    $Self->{QueueObject}  = Kernel::System::Queue->new( %{$Self} );

    return $Self;
}

=item Param()

Get all parameters a user can specify.

    my @Params = $StatsObject->Param();

=cut

sub Param {
    my $Self = shift;

    my %Queues = $Self->{QueueObject}->GetAllQueues();
    my %Types  = $Self->{TypeObject}->TypeList(
        Valid => 1,
    );
```

(continues on next page)

(folytatás az előző oldalról)

```
    my @Params = (
        {
            Frontend  => 'Type',
            Name      => 'TypeIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Types,
        },
        {
            Frontend  => 'Queue',
            Name      => 'QueueIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Queues,
        },
    );

    return @Params;
}

=item Run()

generate the statistic.

    my $StatsInfo = $StatsObject->Run(
        TypeIDs  => [
            1, 2, 4
        ],
        QueueIDs => [
            3, 4, 6
        ],
    );

=cut

sub Run {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for my $Needed (qw(TypeIDs QueueIDs)) {
        if ( !$Param{$Needed} ) {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Need $Needed!",
            );
            return;
        }
    }

    # set report title
    my $Title = 'Tickets per Queue';
```

(continues on next page)

(folytatás az előző oldalról)

```
    # table headlines
    my @HeadData = (
        'Ticket Number',
        'Queue',
        'Type',
    );

    my @Data;
    my @TicketIDs = $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'ARRAY',
        Permission => 'ro',
        %Param,
    );

    for my $TicketID ( @TicketIDs ) {
        my %Ticket = $Self->{TicketObject}->TicketGet(
            UserID => 1,
            TicketID => $TicketID,
        );
        push @Data, [ $Ticket{TicketNumber}, $Ticket{Queue}, $Ticket{Type} ];
    }

    return ( [$Title], [@HeadData], @Data );
}

1;
```

#### Statikus statisztikák beállítási példa

Nincs szükség beállításokra. Közvetlenül telepítés után a modul elérhető egy statisztika létrehozásához ennél a modulnál.

### 3.3.11 Jegyszám előállító modulok

A jegyszám előállítókat elkülönülő azonosítók létrehozásához használják az új jegyekhez, amelyeket jegyszámnak neveznek. Bármilyen metódus lehetséges a számok karakterláncainak létrehozásához, de a józan ész határain belül kell maradnia az eredményül kapott szöveg hosszával kapcsolatban (irányelv: 5-10).

Egy jegyszám létrehozásakor győződjön meg arról, hogy az eredmény megkapta-e a `SystemID` rendszerbeállítási változót előtagként annak érdekében, hogy engedélyezze a jegyszámok felismerését a bejövő e-mail válaszoknál. Egy jegyszám előállító modulnak a következő két függvényre van szüksége: `TicketCreateNumber()` és `GetTNByString()`.

A `TicketCreateNumber()` metódus paraméterek nélkül kerül meghívásra, és az új jegyszámot adja vissza.

A `GetTNByString()` metódus egy olyan `String` paraméterrel kerül meghívásra, amely a feldolgozandó szöveget tartalmazza a jegyszámnál, és visszaadja a jegyszámot, ha megtalálta.

#### Jegyszám előállító kódpélda

Nézze meg a forráskód `Kernel/System/Ticket/Number` mappájában lévő fájlokat.

#### Jegyszám előállító beállítási példa

Nézze meg a `Kernel/Config/Files/XML/Ticket.xml` fájlban azokat a beállításokat, amelyek neve `Ticket::NumberGenerator` előtaggal kezdődik.

#### Jegyszám előállító használati eset példa

**A jegyszámoknak egy bizonyos sémát kell követniük** Akkor kell majd egy új jegyszám előállítót létrehoznia, ha az alapértelmezett modulok nem biztosítják azt a jegyszám sémát, amelyet használni szeretne.

---

**Megjegyzés:** Ragaszkodnia kell a meglévő jegyszám előállítókban használt `GetTNByString()` kódjához, hogy megelőzze a jegyszám feldolgozással kapcsolatos problémákat. A `TicketCreateNumber()` metódusban lévő hurok felismeréséhez használt rutint is érintetlenül kell hagynia a kettőzött jegyszámok megelőzéséhez.

---

### 3.3.12 Jegyesemény modul

A jegyesemény modulok közvetlenül azután futnak le, amikor egy jegyművelet megtörténik. Megegyezés szerint ezek a modulok a `Kernel/System/Ticket/Event` könyvtárban találhatók. Egy jegyesemény modulnak mindössze két függvényre van szüksége: `new()` és `Run()`. A `Run()` metódus legalább az `Event`, a *UserID*' és a `Data` paramétereket fogadja. A `Data` a jegy adatait tartalmazó kivonathivatkozás, és a bejegyzésre vonatkozó események esetében a bejegyzés adatait is tartalmazza.

#### Jegyesemény modul kódpélda

Nézze meg a forráskód `Kernel/System/Ticket/Event` mappájában lévő fájlokat.

#### Jegyesemény modul beállítási példa

Nézze meg a `Kernel/Config/Files/XML/Ticket.xml` fájlban azokat a beállításokat, amelyek neve `Ticket::EventModulePost###` előtaggal kezdődik.

#### Jegyesemény modul használati eset példa

**Egy jegyet fel kell oldani egy áthelyezés művelet után** Ez a szabványos funkció a `Kernel::System::Ticket::Event::ForceUnlock` jegyesemény modullal lett megvalósítva. Amikor erre a funkcióra nincs szükség, akkor az kikapcsolható a `Ticket::EventModulePost###910-ForceUnlockOnMove` rendszerbeállítási bejegyzés beállításának törlésével.

**További tisztítóművelet végrehajtása egy jegy törlésekor** Egy személyre szabott OTRS tarthat nem szabványos adatokat további adatbázistáblákban. Amikor egy jegyet törölnek, akkor ezeket a további adatokat is törölni kell. Ez a funkcionalitás elérhető egy olyan jegyesemény modullal, amely a `TicketDelete` eseményekre figyel.

**Az új jegyeket közzé kell tenni a Twitteren** Egy `TicketCreate` eseményre figyelő jegyesemény modul képes üzeneteket kiküldeni a Twitterre.

### Jegy- és bejegyzésesemények

Elérhető jegyesemények:

- `TicketCreate`
- `TicketDelete`
- `TicketTitleUpdate`
- `TicketUnlockTimeoutUpdate`
- `TicketQueueUpdate`
- `TicketTypeUpdate`
- `TicketServiceUpdate`
- `TicketSLAUpdate`
- `TicketCustomerUpdate`
- `TicketPendingTimeUpdate`
- `TicketLockUpdate`
- `TicketArchiveFlagUpdate`
- `TicketStateUpdate`
- `TicketOwnerUpdate`
- `TicketResponsibleUpdate`
- `TicketPriorityUpdate`
- `HistoryAdd`
- `HistoryDelete`
- `TicketAccountTime`
- `TicketMerge`
- `TicketSubscribe`
- `TicketUnsubscribe`
- `TicketFlagSet`
- `TicketFlagDelete`
- `EscalationResponseTimeNotifyBefore`
- `EscalationUpdateTimeNotifyBefore`
- `EscalationSolutionTimeNotifyBefore`
- `EscalationResponseTimeStart`
- `EscalationUpdateTimeStart`
- `EscalationSolutionTimeStart`
- `EscalationResponseTimeStop`

- EscalationUpdateTimeStop
- EscalationSolutionTimeStop
- NotificationNewTicket
- NotificationFollowUp
- NotificationLockTimeout
- NotificationOwnerUpdate
- NotificationResponsibleUpdate
- NotificationAddNote
- NotificationMove
- NotificationPendingReminder
- NotificationEscalation
- NotificationEscalationNotifyBefore
- NotificationServiceUpdate

Elérhető bejegyzésesemények:

- ArticleCreate
- ArticleUpdate
- ArticleSend
- ArticleBounce
- ArticleAgentNotification
- ArticleCustomerNotification
- ArticleAutoResponse
- ArticleFlagSet
- ArticleFlagDelete
- ArticleAgentNotification
- ArticleCustomerNotification

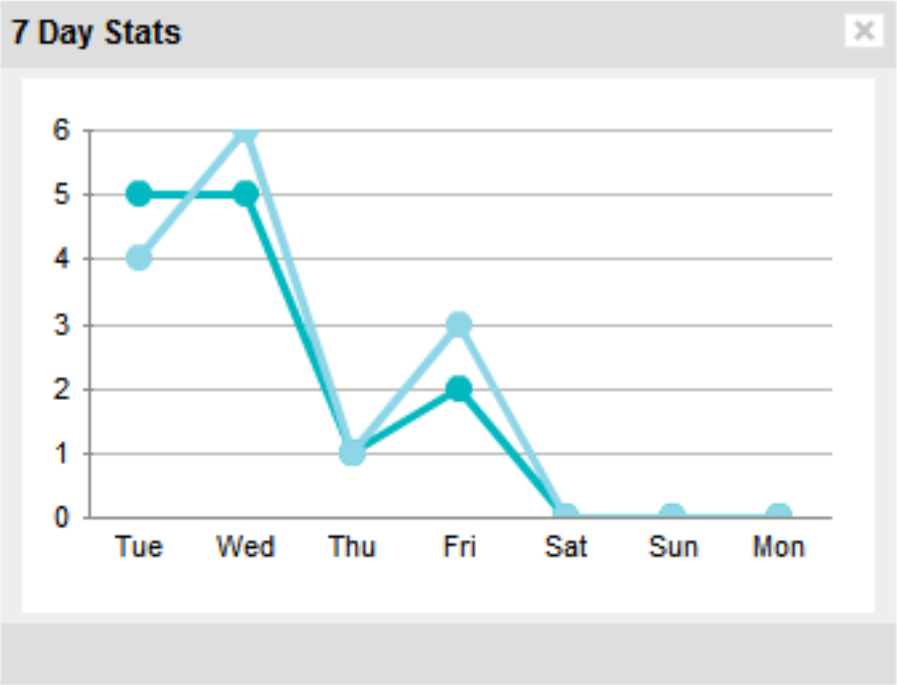
### 3.3.13 Vezérlőpult modul

Vezérlőpult modul statisztikák megjelenítéséhez vonaldiagram formájában.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::DashboardTicketStatsGeneric;
```

(continues on next page)




3. ábra: Vezérlőpult felületi elem

(folytatás az előző oldalról)

```
use strict;
use warnings;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless( $Self, $Type );

    # get needed objects
    for (
        qw(Config Name ConfigObject LogObject DBObject LayoutObject ParamObject␣
↪TicketObject UserID)
        )
    {
        die "Got no $_!" if !$Self->{$_};
    }

    return $Self;
}

sub Preferences {
    my ( $Self, %Param ) = @_;

    return;
```

(continues on next page)

(folytatás az előző oldalról)

```
}

sub Config {
    my ( $Self, %Param ) = @_;

    my $Key = $Self->{LayoutObject}->{UserLanguage} . '-' . $Self->{Name};
    return (
        %{ $Self->{Config} },
        CacheKey => 'TicketStats' . '-' . $Self->{UserID} . '-' . $Key,
    );

}

sub Run {
    my ( $Self, %Param ) = @_;

    my %Axis = (
        '7Day' => {
            0 => { Day => 'Sun', Created => 0, Closed => 0, },
            1 => { Day => 'Mon', Created => 0, Closed => 0, },
            2 => { Day => 'Tue', Created => 0, Closed => 0, },
            3 => { Day => 'Wed', Created => 0, Closed => 0, },
            4 => { Day => 'Thu', Created => 0, Closed => 0, },
            5 => { Day => 'Fri', Created => 0, Closed => 0, },
            6 => { Day => 'Sat', Created => 0, Closed => 0, },
        },
    );

    my @Data;
    my $Max = 1;
    for my $Key ( 0 .. 6 ) {

        my $TimeNow = $Self->{TimeObject}->SystemTime();
        if ($Key) {
            $TimeNow = $TimeNow - ( 60 * 60 * 24 * $Key );
        }
        my ( $Sec, $Min, $Hour, $Day, $Month, $Year, $WeekDay )
            = $Self->{TimeObject}->SystemTime2Date(
            SystemTime => $TimeNow,
            );

        $Data[$Key]->{Day} = $Self->{LayoutObject}->{LanguageObject}->Get(
            $Axis{'7Day'}->{$WeekDay}->{Day}
        );

        my $CountCreated = $Self->{TicketObject}->TicketSearch(

            # cache search result 20 min
            CacheTTL => 60 * 20,

            # tickets with create time after ... (ticket newer than this␣
↪date) (optional)
```

(continues on next page)

(folytatás az előző oldalról)

```
            TicketCreateTimeNewerDate => "$Year-$Month-$Day 00:00:00",

            # tickets with created time before ... (ticket older than this␣
↪date) (optional)
            TicketCreateTimeOlderDate => "$Year-$Month-$Day 23:59:59",

            CustomerID => $Param{Data}->{UserCustomerID},
            Result     => 'COUNT',

            # search with user permissions
            Permission => $Self->{Config}->{Permission} || 'ro',
            UserID => $Self->{UserID},
        );
        $Data[$Key]->{Created} = $CountCreated;
        if ( $CountCreated > $Max ) {
            $Max = $CountCreated;
        }

        my $CountClosed = $Self->{TicketObject}->TicketSearch(

            # cache search result 20 min
            CacheTTL => 60 * 20,

            # tickets with create time after ... (ticket newer than this␣
↪date) (optional)
            TicketCloseTimeNewerDate => "$Year-$Month-$Day 00:00:00",

            # tickets with created time before ... (ticket older than this␣
↪date) (optional)
            TicketCloseTimeOlderDate => "$Year-$Month-$Day 23:59:59",

            CustomerID => $Param{Data}->{UserCustomerID},
            Result     => 'COUNT',

            # search with user permissions
            Permission => $Self->{Config}->{Permission} || 'ro',
            UserID => $Self->{UserID},
        );
        $Data[$Key]->{Closed} = $CountClosed;
        if ( $CountClosed > $Max ) {
            $Max = $CountClosed;
        }
    }

    @Data = reverse @Data;
    my $Source = $Self->{LayoutObject}->JSONEncode(
        Data => \@Data,
    );

    my $Content = $Self->{LayoutObject}->Output(
        TemplateFile => 'AgentDashboardTicketStats',
        Data         => {
```

(continues on next page)

(folytatás az előző oldalról)

```
            %{ $Self->{Config} },
            Key    => int rand 99999,
            Max    => $Max,
            Source => $Source,
        },
    );

    return $Content;
}

1;
```

Ezen modul használatához adja hozzá a következőket a `Kernel/Config.pm` fájlhoz, és indítsa újra a web-kiszolgálóját (ha a `mod_perl` modult használja).

```
<ConfigItem Name="DashboardBackend###0250-TicketStats" Required="0" Valid="1">
    <Description Translatable="1">Parameters for the dashboard backend. "Group
↪" are used to restricted access to the plugin (e. g. Group: admin;group1;
↪group2;). "Default" means if the plugin is enabled per default or if the␣
↪user needs to enable it manually. "CacheTTL" means the cache time in␣
↪minutes for the plugin.</Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Agent::Dashboard</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">
↪Kernel::Output::HTML::DashboardTicketStatsGeneric</Item>
            <Item Key="Title">7 Day Stats</Item>
            <Item Key="Created">1</Item>
            <Item Key="Closed">1</Item>
            <Item Key="Permission">rw</Item>
            <Item Key="Block">ContentSmall</Item>
            <Item Key="Group"></Item>
            <Item Key="Default">1</Item>
            <Item Key="CacheTTL">45</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

**Megjegyzés:** A napok vagy az önálló sorok túlzott száma teljesítmény-csökkenéshez vezethet.

### 3.3.14 Értesítési modul

Az értesítési modulokat egy értesítés megjelenítéséhez használják a fő navigáció alatt. Megírhatja és regisztrálhatja a saját értesítési modulját. Jelenleg 5 jegymenü van az OTRS keretrendszerben.

- `AgentOnline`
- `AgentTicketEscalation`
- `CharsetCheck`

- CustomerOnline
- UIDCheck

### Értesítési modul kódpélda

Az értesítési modulok a `Kernel/Output/HTML/TicketNotification*.pm` alatt találhatók. Ezt követően egy értesítőmodul példája található. Mentse el a `Kernel/Output/HTML/TicketNotificationCustom.pm` fájlba. Mindössze két függvényre van szüksége: `new()` és `Run()`.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::NotificationCustom;

use strict;
use warnings;

use Kernel::System::Custom;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # get needed objects
    for my $Object (qw(ConfigObject LogObject DBObject LayoutObject TimeObject␣
↪UserID)) {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }
    $Self->{CustomObject} = Kernel::System::Custom->new(%Param);
    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;

    # get session info
    my %CustomParam      = ();
    my @Customs     = $Self->{CustomObject}->GetAllCustomIDs();
    my $IdleMinutes = $Param{Config}->{IdleMinutes} || 60 * 2;
    for (@Customs) {
        my %Data = $Self->{CustomObject}->GetCustomIDData( CustomID => $_, );
        if (
            $Self->{UserID} ne $Data{UserID}
            && $Data{UserType} eq 'User'
```

(continues on next page)

(folytatás az előző oldalról)

```
            && $Data{UserLastRequest}
            && $Data{UserLastRequest} + ( $IdleMinutes * 60 ) > $Self->
↪{TimeObject}->SystemTime()
            && $Data{UserFirstname}
            && $Data{UserLastname}
            )
        {
            $CustomParam{ $Data{UserID} } = "$Data{UserFirstname} $Data
↪{UserLastname}";
            if ( $Param{Config}->{ShowEmail} ) {
                $CustomParam{ $Data{UserID} } .= " ($Data{UserEmail})";
            }
        }
    }
    for ( sort { $CustomParam{$a} cmp $CustomParam{$b} } keys %CustomParam ) {
        if ( $Param{Message} ) {
            $Param{Message} .= ', ';
        }
        $Param{Message} .= "$CustomParam{$_}";
    }
    if ( $Param{Message} ) {
        return $Self->{LayoutObject}->Notify( Info => 'Custom Message: %s', "
↪' . $Param{Message} );
    }
    else {
        return '';
    }
}

1;
```

### Értesítési modul beállítási példa

Szükség van az egyéni értesítési modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni. Lehetnek további paraméterek is a beállítás kivonatában az értesítési moduljánál.

```
<ConfigItem Name="Frontend::NotifyModule###3-Custom" Required="0" Valid="0">
    <Description Translatable="1">Module to show custom message in the agent␣
↪interface.</Description>
    <Group>Framework</Group>
    <SubGroup>Frontend::Agent::ModuleNotify</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::Output::HTML::NotificationCustom</Item>
            <Item Key="Key1">1</Item>
            <Item Key="Key2">2</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

#### Értesítési modul használati eset példa

Hasznos jegymenü megvalósítás lehet egy hivatkozás egy külső eszközre, ha a paraméterek (például `FreeTextField`) be lettek állítva.

### 3.3.15 Jegymenü modul

A jegymenü modulokat egy további hivatkozás megjelenítéséhez használják a jegy fölött lévő menüben. Megírhatja és regisztrálhatja a saját jegymenü moduljait. Négy jegymenü létezik (*Általános*, *Zárolás*, *Felelős* és *Jegymegfigyelő*), amely az OTRS keretrendszerrel érkezik. További információkért nézzen bele az OTRS adminisztrációs kézikönyvébe.

#### Jegymenü modul kódpélda

A jegymenü modulok a `Kernel/Output/HTML/TicketMenu*.pm` fájlokban találhatók. A következőkben egy jegymenü modul példája található. Mentse el a `Kernel/Output/HTML/TicketMenuCustom.pm` helyre. Mindössze két függvényre van szüksége: `new()` és `Run()`.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::TicketMenuCustom;

use strict;
use warnings;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # get needed objects
    for my $Object (qw(ConfigObject LogObject DBObject LayoutObject UserID␣
↪TicketObject)) {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }

    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{Ticket} ) {
```

(continues on next page)

(folytatás az előző oldalról)

```
        $Self->{LogObject}->Log(
            Priority => 'error',
            Message  => 'Need Ticket!'
        );
        return;
    }

    # check if frontend module registered, if not, do not show action
    if ( $Param{Config}->{Action} ) {
        my $Module = $Self->{ConfigObject}->Get('Frontend::Module')->{ $Param
↪{Config}->{Action} };
        return if !$Module;
    }

    # check permission
    my $AccessOk = $Self->{TicketObject}->Permission(
        Type     => 'rw',
        TicketID => $Param{Ticket}->{TicketID},
        UserID   => $Self->{UserID},
        LogNo    => 1,
    );
    return if !$AccessOk;

    # check permission
    if ( $Self->{TicketObject}->CustomIsTicketCustom( TicketID => $Param
↪{Ticket}->{TicketID} ) ) {
        my $AccessOk = $Self->{TicketObject}->OwnerCheck(
            TicketID => $Param{Ticket}->{TicketID},
            OwnerID  => $Self->{UserID},
        );
        return if !$AccessOk;
    }

    # check acl
    return
        if defined $Param{ACL}->{ $Param{Config}->{Action} }
            && !$Param{ACL}->{ $Param{Config}->{Action} };

    # if ticket is customized
    if ( $Param{Ticket}->{Custom} eq 'lock' ) {

        # if it is locked for somebody else
        return if $Param{Ticket}->{OwnerID} ne $Self->{UserID};

        # show custom action
        return {
            %{ $Param{Config} },
            %{ $Param{Ticket} },
            %Param,
            Name        => 'Custom',
            Description => 'Custom to give it back to the queue!',
            Link        => 'Action=AgentTicketCustom;Subaction=Custom;
↪TicketID=$QData{"TicketID"}',
```

(continues on next page)

(folytatás az előző oldalról)

```
        };
    }

    # if ticket is customized
    return {
        %{ $Param{Config} },
        %{ $Param{Ticket} },
        %Param,
        Name        => 'Custom',
        Description => 'Custom it to work on it!',
        Link        => 'Action=AgentTicketCustom;Subaction=Custom;TicketID=
↪$QData{"TicketID"}',
    };
}

1;
```

### Jegymenü modul beállítási példa

Szükség van az egyéni jegymenü modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni. Lehetnek további paraméterek is a beállítás kivonatában a jegymenü moduljánál.

```
<ConfigItem Name="Ticket::Frontend::MenuModule###110-Custom" Required="0"␣
↪Valid="1">
    <Description Translatable="1">Module to show custom link in menu.</
↪Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Agent::Ticket::MenuModule</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::Output::HTML::TicketMenuCustom</Item>
            <Item Key="Name">Custom</Item>
            <Item Key="Action">AgentTicketCustom</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

### Jegymenü modul használati eset példa

Hasznos jegymenü megvalósítás lehet egy hivatkozás egy külső eszközre, ha a paraméterek (például `FreeTextField`) be lettek állítva.

---

**Megjegyzés:** A jegymenü egy olyan URL-re irányít, amely kezelhető. Ha ezt a kérést az OTRS keretrendszeren keresztül szeretné kezelni, akkor meg kell írnia a saját előtétprogram modulját.

---

### 3.3.16 Hálózati átvitel

A hálózati átvitelt használják az információk küldésének és fogadásának módszereként az OTRS és egy távoli rendszer között. Az általános felület beállításai lehetővé teszik egy webszolgáltatásnak, hogy különböző hálózati átviteli modulokat használjon a szolgáltatónál és a kérelmezőnél, de a leggyakoribb forgatókönyv az, hogy ugyanazt az átviteli modult használják mindkettőnél.

**OTRS mint szolgáltató** Az OTRS arra használja a hálózati átviteli modulokat, hogy lekérje az adatokat a távoli rendszertől, valamint lekérje a végrehajtandó műveleteket. A művelet végrehajtása után az OTRS ismét azokat használja a válasz visszaküldéséhez a távoli rendszernek.

**OTRS mint kérelmező** Az OTRS arra használja a hálózati átviteli modulokat, hogy kérelmeket küldjön a távoli rendszernek egy távoli művelet végrehajtásához a szükséges adatok mellett. Az OTRS várakozik a távoli rendszer válaszára, és visszaküldi azt a kérelmező modulnak.

Mindkét irányban a hálózati átviteli modulok foglalkoznak a távoli rendszer formátumában lévő adatokkal. Nem ajánlott semmilyen adatátalakítás végrehajtása sem ezekben a modulokban, mivel a leképező réteg felelős a kommunikáció során szükséges bármilyen adatátalakítás végrehajtásáért. Egy kivétel erre az olyan adatátalakítás, amely kifejezetten az átvitelnél szükséges, például XML vagy JSON és Perl átalakítások.

#### Átviteli háttérprogram

Ezután be fogjuk mutatni, hogy hogyan kell egy új átviteli háttérprogramot kifejleszteni. Minden egyes átviteli háttérprogramnak meg kell valósítania ezeket a szubrutinokat:

- `new`
- `ProviderProcessRequest`
- `ProviderGenerateResponse`
- `RequesterPerformRequest`

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk mindkét irányban helyesen kommunikálni egy távoli rendszerrel. Az összes átviteli háttérprogramot az átviteli modul kezeli (`Kernel/GenericInterface/Transport.pm`).

Jelenleg az általános felület megvalósítja a HTTP SOAP és a HTTP REST átviteleket. Ha a tervezett webszolgáltatás használhat HTTP SOAP vagy HTTP SOAP átviteleket, akkor nincs szükség egy új hálózati átviteli modul létrehozására, hanem ahelyett azt ajánljuk, hogy vessen egy pillantást a HTTP SOAP vagy HTTP REST konfigurációira a beállításaik ellenőrzéséhez, valamint hogy hogyan hangolhatók a távoli rendszernek megfelelően.

#### Hálózati átvitel kódpélda

Abban az esetben, ha a biztosított hálózati átvitelek nem illeszkednek a webszolgáltatás igényeire, akkor ebben a szakaszban egy minta hálózati átviteli modul van bemutatva, és minden egyes szubrutin elmagyarázásra kerül. Normális esetben az átviteli modulok CPAN modulokat használnak háttérprogramokként. Például a HTTP SOAP átviteli modulok a `SOAP::Lite` modult használják háttérprogramként.

Ennél a példánál egy egyéni csomagot használnak az adatok visszaadásához,anélkül hogy valódi hálózati kérést intéznének egy távoli rendszerhez, ehelyett ez az egyéni modul működik visszacsatolási felületként.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
```

(continues on next page)

(folytatás az előző oldalról)

```
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::GenericInterface::Transport::HTTP::Test;

use strict;
use warnings;

use HTTP::Request::Common;
use LWP::UserAgent;
use LWP::Protocol;

# prevent 'Used once' warning for Kernel::OM
use Kernel::System::ObjectManager;

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva. Az átvitelek nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {};
    bless( $Self, $Type );

    for my $Needed (qw( DebuggerObject TransportConfig)) {
        $Self->{$Needed} = $Param{$Needed} || return {
            Success      => 0,
            ErrorMessage => "Got no $Needed!"
        };
    }

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a `new` konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub ProviderProcessRequest {
    my ( $Self, %Param ) = @_;

    if ( $Self->{TransportConfig}->{Config}->{Fail} ) {

        return {
            Success      => 0,
            ErrorMessage => "HTTP status code: 500",
            Data         => {},
        };
    }
```

(continues on next page)

(folytatás az előző oldalról)

```
    my $ParamObject = $Kernel::OM->Get('Kernel::System::Web::Request');

    my %Result;
    for my $ParamName ( $ParamObject->GetParamNames() ) {
        $Result{$ParamName} = $ParamObject->GetParam( Param => $ParamName );
    }

    # special handling for empty post request
    if ( scalar keys %Result == 1 && exists $Result{POSTDATA} && !$Result
↪{POSTDATA} ) {
        %Result = ();
    }

    if ( !%Result ) {

        return $Self->{DebuggerObject}->Error(
            Summary => 'No request data found.',
        );
    }

    return {
        Success   => 1,
        Data      => \%Result,
        Operation => 'test_operation',
    };
}
```

A `ProviderProcessRequest` függvény megkapja a kérést a távoli kiszolgálótól (ebben az esetben ugyanaz az OTRS), és kibontja az adatokat és a műveletet a kérésből a végrehajtáshoz. Ennél a példánál a művelet mindig a `test_operation`.

Annak a módja, ahogy ez a függvény feldolgozza a kérést az adatok és a művelet nevének lekéréséhez, az teljes egészében a megvalósítandó protokolltól és azon külső moduloktól függ, amelyekhez használják azokat.

```
sub ProviderGenerateResponse {
    my ( $Self, %Param ) = @_;

    if ( $Self->{TransportConfig}->{Config}->{Fail} ) {

        return {
            Success      => 0,
            ErrorMessage => 'Test response generation failed',
        };
    }

    my $Response;

    if ( !$Param{Success} ) {
        $Response
            = HTTP::Response->new( 500 => ( $Param{ErrorMessage} || 'Internal␣
↪Server Error' ) );
```

(continues on next page)

(folytatás az előző oldalról)

```
        $Response->protocol('HTTP/1.0');
        $Response->content_type("text/plain; charset=UTF-8");
        $Response->date(time);
    }
    else {

        # generate a request string from the data
        my $Request
            = HTTP::Request::Common::POST( 'http://testhost.local/', Content␣
↪=> $Param{Data} );

        $Response = HTTP::Response->new( 200 => "OK" );
        $Response->protocol('HTTP/1.0');
        $Response->content_type("text/plain; charset=UTF-8");
        $Response->add_content_utf8( $Request->content() );
        $Response->date(time);
    }

    $Self->{DebuggerObject}->Debug(
        Summary => 'Sending HTTP response',
        Data    => $Response->as_string(),
    );

    # now send response to client
    print STDOUT $Response->as_string();

    return {
        Success => 1,
    };
}
```

Ez a függvény visszaküldi a választ a távoli rendszernek a kért művelethez.

Ennél a bizonyos példánál minden esetben egy szabványos sikeres HTTP-választ adunk vissza (200) vagy nem (500) a szükséges adatok mellett.

```
sub RequesterPerformRequest {
    my ( $Self, %Param ) = @_;

    if ( $Self->{TransportConfig}->{Config}->{Fail} ) {

        return {
            Success      => 0,
            ErrorMessage => "HTTP status code: 500",
            Data         => {},
        };
    }

    # use custom protocol handler to avoid sending out real network requests
    LWP::Protocol::implementor(
        testhttp =>
↪'Kernel::GenericInterface::Transport::HTTP::Test::CustomHTTPProtocol'
    );
```

(continues on next page)

(folytatás az előző oldalról)

```
    my $UserAgent = LWP::UserAgent->new();
    my $Response = $UserAgent->post( 'testhttp://localhost.local/', Content =>
↪ $Param{Data} );

    return {
        Success => 1,
        Data    => {
            ResponseContent => $Response->content(),
        },
    };
}
```

Ez az egyetlen olyan függvény, amelyet az OTRS mint kérelmező használ. Elküldi a kérést a távoli rendszernek, és várakozik annak válaszára.

Ennél a példánál egy egyéni protokollkezelőt használunk a valódi hálózatra történő kérésküldés elkerüléséhez. Ez az egyéni protokoll az alábbiakban van megadva.

```
package Kernel::GenericInterface::Transport::HTTP::Test::CustomHTTPProtocol;

use base qw(LWP::Protocol);

sub new {
    my $Class = shift;

    return $Class->SUPER::new(@_);
}

sub request {    ## no critic
    my $Self = shift;

    my ( $Request, $Proxy, $Arg, $Size, $Timeout ) = @_;

    my $Response = HTTP::Response->new( 200 => "OK" );
    $Response->protocol('HTTP/1.0');
    $Response->content_type("text/plain; charset=UTF-8");
    $Response->add_content_utf8( $Request->content() );
    $Response->date(time);

    #print $Request->as_string();
    #print $Response->as_string();

    return $Response;
}
```

Ez a kód ahhoz az egyéni protokollhoz van, amelyet használunk. Ez a megközelítés csak gyakorlásnál vagy olyan tesztelési környezeteknél hasznos, ahol a távoli rendszerek nem érhetők el.

Egy új modul kifejlesztéséhez nem ajánljuk ezen megközelítés használatát, egy valódi protokollt kell megvalósítani.

#### Hálózati átvitel beállítási példa

Szükség van ezen hálózati átviteli modul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Transport::Module###HTTP::Test" Required="0
↪" Valid="1">
    <Description Translatable="1">GenericInterface module registration for␣
↪the transport layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Transport::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Name">Test</Item>
            <Item Key="Protocol">HTTP</Item>
            <Item Key="ConfigDialog">AdminGenericInterfaceTransportHTTPTest</
↪Item>
        </Hash>
    </Setting>
</ConfigItem>
```

### 3.3.17 Leképezés

A leképezést adatok átalakításához használják az OTRS és a távoli rendszer között, illetve fordítva. Ezek az adatok kulcs => érték párokként vannak ábrázolva. Leképező modulok fejleszthetők ki nem csak az értékek, hanem a kulcsok átalakításához is.

Például:

| Leképezés erről | Leképezés erre |
|---|---|
| Prio => Warning | PriorityID => 3 |

A leképező réteg nem feltétlenül szükséges, egy webszolgáltatás teljesen kihagyhatja azt a webszolgáltatás beállításaitól, valamint a meghívók és műveletek megvalósításának módjától függően. De ha egy kis átalakítás szükséges, akkor erősen ajánlott egy meglévő leképezőmodul használata, vagy egy új létrehozása.

A leképező modulok egynél több alkalommal is meghívhatók egy normál kommunikáció közben. Vessen egy pillantást a következő példákra.

**OTRS mint szolgáltató példa**

1. A távoli rendszer elküldi a kérést az adatokkal a távoli rendszer formátumában.
2. Az adatok leképezésre kerülnek a távoli rendszer formátumáról az OTRS formátumára.
3. Az OTRS végrehajtja a műveletet, és visszaadja a választ az OTRS formátumában.
4. Az adatok leképezésre kerülnek az OTRS formátumáról a távoli rendszer formátumára.
5. A válasz a távoli rendszer formátumában lévő adatokkal elküldésre kerül a távoli rendszernek.

**OTRS mint kérelmező példa**

1. Az OTRS előkészíti a kérést a távoli rendszerhez az OTRS formátumában használt adatokkal.
2. Az adatok leképezésre kerülnek az OTRS formátumáról a távoli rendszer formátumára.

3. A kérés elküldésre kerül a távoli rendszernek, amely végrehajtja a műveletet, és visszaküldi a választ az OTRS-nek a távoli rendszer formátumában lévő adatokkal.
4. Az adatok (ismét) leképezésre kerülnek a távoli rendszer formátumáról az OTRS formátumára.
5. Az OTRS feldolgozza a választ.

#### Leképező háttérprogram

Az általános felület biztosít egy *Simple* nevű leképezőmodult. Ezzel a modullal a legtöbb adatátalakítás (beleértve a kulcs és érték leképezést) elvégezhető, és szabályokat is meghatároz az alapértelmezett leképezések kezeléséhez mind a kulcsoknál, mind az értékeknél.

Ezért erősen valószínű, hogy nem lesz szüksége egy egyéni leképezőmodul kifejlesztésére. A folytatás előtt nézze meg a *Simple* leképezőmodult (`Kernel/GenericInterface/Mapping/Simple.pm`) és annak internetes dokumentációját.

Ha a *Simple* leképezőmodul nem felel meg az igényeinek, akkor meg fogjuk mutatni, hogy hogyan lehet kifejleszteni egy új leképező háttérprogramot. Minden egyes leképező háttérprogramnak meg kell valósítania ezeket a szubrutinokat:

- `new`
- `Map`

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk az adatok leképezésére a kommunikációban, amelyet vagy a kérelmező, vagy a szolgáltató kezel. Az összes leképező háttérprogramot a leképezőmodul kezeli (`Kernel/GenericInterface/Mapping.pm`).

#### Leképezés kódpélda

Ebben a szakaszban egy minta leképezőmodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::GenericInterface::Mapping::Test;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(IsHashRefWithData IsStringWithData);

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

Felveszünk egy `VariableCheck` modult is bizonyos ellenőrzések végrehajtásához néhány változón. A leképezések nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed params
    for my $Needed (qw(DebuggerObject MappingConfig)) {
        if ( !$Param{$Needed} ) {

            return {
                Success      => 0,
                ErrorMessage => "Got no $Needed!"
            };
        }
        $Self->{$Needed} = $Param{$Needed};
    }

    # check mapping config
    if ( !IsHashRefWithData( $Param{MappingConfig} ) ) {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Got no MappingConfig as hash ref with content!',
        );
    }

    # check config - if we have a map config, it has to be a non-empty hash ref
    if (
        defined $Param{MappingConfig}->{Config}
        && !IsHashRefWithData( $Param{MappingConfig}->{Config} )
        )
    {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Got MappingConfig with Data, but Data is no hash ref␣
↪with content!',
        );
    }

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a `new` konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub Map {
    my ( $Self, %Param ) = @_;

    # check data - only accept undef or hash ref
    if ( defined $Param{Data} && ref $Param{Data} ne 'HASH' ) {
```

(continues on next page)

(folytatás az előző oldalról)

```
        return $Self->{DebuggerObject}->Error(
            Summary => 'Got Data but it is not a hash ref in Mapping Test␣
↪backend!'
        );
    }

    # return if data is empty
    if ( !defined $Param{Data} || !%{ $Param{Data} } ) {

        return {
            Success => 1,
            Data    => {},
        };
    }

    # no config means that we just return input data
    if (
        !defined $Self->{MappingConfig}->{Config}
        || !defined $Self->{MappingConfig}->{Config}->{TestOption}
        )
    {

        return {
            Success => 1,
            Data    => $Param{Data},
        };
    }

    # check TestOption format
    if ( !IsStringWithData( $Self->{MappingConfig}->{Config}->{TestOption} ) ) {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Got no TestOption as string with value!',
        );
    }

    # parse data according to configuration
    my $ReturnData = {};
    if ( $Self->{MappingConfig}->{Config}->{TestOption} eq 'ToUpper' ) {
        $ReturnData = $Self->_ToUpper( Data => $Param{Data} );
    }
    elsif ( $Self->{MappingConfig}->{Config}->{TestOption} eq 'ToLower' ) {
        $ReturnData = $Self->_ToLower( Data => $Param{Data} );
    }
    elsif ( $Self->{MappingConfig}->{Config}->{TestOption} eq 'Empty' ) {
        $ReturnData = $Self->_Empty( Data => $Param{Data} );
    }
    else {
        $ReturnData = $Param{Data};
    }

    # return result
```

(continues on next page)

(folytatás az előző oldalról)

```
    return {
        Success => 1,
        Data    => $ReturnData,
    };
}
```

A `Map` függvény az egyes leképezőmodulok fő része. Fogadja a leképezési beállításokat (szabályokat) és az eredeti formátumban lévő adatokat (vagy az OTRS vagy a távoli rendszer formátumában lévőket), és átalakítja azokat egy új formátumra még akkor is, ha az adatok szerkezete megváltozhat a leképezési folyamat során.

Ebben a bizonyos példában három szabály van az értékek leképezéséhez. Ezek a szabályok a leképezési beállítások `TestOption` kulcsában vannak beállítva, és a következők: `ToUpper`, `ToLower` és `Empty`.

- `ToUpper`: nagybetűsre alakít át minden egyes adatértéket.
- `ToLower`: kisbetűsre alakít át minden egyes adatértéket.
- `Empty`: egy üres szövegre alakít át minden egyes adatértéket.

Ebben a példában nem lettek adatkulcs átalakítások megvalósítva.

```
sub _ToUpper {
    my ( $Self, %Param ) = @_;

    my $ReturnData = {};
    for my $Key ( sort keys %{ $Param{Data} } ) {
        $ReturnData->{$Key} = uc $Param{Data}->{$Key};
    }

    return $ReturnData;
}

sub _ToLower {
    my ( $Self, %Param ) = @_;

    my $ReturnData = {};
    for my $Key ( sort keys %{ $Param{Data} } ) {
        $ReturnData->{$Key} = lc $Param{Data}->{$Key};
    }

    return $ReturnData;
}

sub _Empty {
    my ( $Self, %Param ) = @_;

    my $ReturnData = {};
    for my $Key ( sort keys %{ $Param{Data} } ) {
        $ReturnData->{$Key} = '';
    }

    return $ReturnData;
}
```

Ezek azok a segédfüggvények, amelyek ténylegesen végrehajtják a szövegátalakításokat.

#### Leképezés beállítási példa

Szükség van ezen leképezőmodul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Mapping::Module###Test" Required="0" Valid=
↪"1">
    <Description Translatable="1">GenericInterface module registration for␣
↪the mapping layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Mapping::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="ConfigDialog"></Item>
        </Hash>
    </Setting>
</ConfigItem>
```

### 3.3.18 Meghívó

A meghívót arra használják, hogy egy kérést hozzon létre az OTRS-ből egy távoli rendszerhez. Az általános ügyintéző ezen része felelős a szükséges feladatok végrehajtásáért az OTRS oldalán, illetve a szükséges adatok begyűjtéséért a kérés felépítésének érdekében.

#### Meghívó háttérprogram

Ezután be fogjuk mutatni, hogy hogyan kell egy új meghívót kifejleszteni. Minden egyes meghívónak meg kell valósítania ezeket a szubrutinokat:

- `new`
- `PrepareRequest`
- `HandleResponse`

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk végrehajtani egy kérést a kéréskezelő használatával (`Kernel/GenericInterface/Requester.pm`).

#### Meghívó kódpélda

Ebben a szakaszban egy minta meghívómodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
```

(continues on next page)

(folytatás az előző oldalról)

```
# --

package Kernel::GenericInterface::Invoker::Test::Test;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(IsString IsStringWithData);

# prevent 'Used once' warning for Kernel::OM
use Kernel::System::ObjectManager;

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva. A meghívók nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed params
    if ( !$Param{DebuggerObject} ) {
        return {
            Success      => 0,
            ErrorMessage => "Got no DebuggerObject!"
        };
    }

    $Self->{DebuggerObject} = $Param{DebuggerObject};

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a `new` konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub PrepareRequest {
    my ( $Self, %Param ) = @_;

    # we need a TicketNumber
    if ( !IsStringWithData( $Param{Data}->{TicketNumber} ) ) {
        return $Self->{DebuggerObject}->Error( Summary => 'Got no TicketNumber
↪' );
    }

    my %ReturnData;
```

(continues on next page)

(folytatás az előző oldalról)

```
    $ReturnData{TicketNumber} = $Param{Data}->{TicketNumber};

    # check Action
    if ( IsStringWithData( $Param{Data}->{Action} ) ) {
        $ReturnData{Action} = $Param{Data}->{Action} . 'Test';
    }

    # check request for system time
    if ( IsStringWithData( $Param{Data}->{GetSystemTime} ) && $Param{Data}->
↪{GetSystemTime} ) {
        $ReturnData{SystemTime} = $Kernel::OM->Get('Kernel::System::Time')->
↪SystemTime();
    }

    return {
        Success => 1,
        Data    => \%ReturnData,
    };
}
```

A `PrepareRequest` függvényt használják a kérésbe küldendő összes szükséges adat kezeléséhez és összegyűjtéséhez. Itt fogadhatunk adatokat a kéréskezelőtől, használhatjuk azokat, kiterjeszthetjük azokat, új adatokat állíthatunk elő, és ezután átvihetjük az eredményeket a leképező réteghez.

Ennél a példánál azt várjuk, hogy kapunk egy jegyszámot. Ha nem, akkor az `Error()` hibakeresési metódust használjuk, amely létrehoz egy bejegyzést a hibakeresési naplóban, és visszaad egy szerkezetet is a `Success` paraméterrel 0-ként, és egy hibaüzenetet az átadott `Summary` értékként.

Ez a példa hozzáfűzi a *Test* szót is az `Action` paraméterhez, és ha a `GetSystemTime` kérve volt, akkor ki fogja tölteni a `SystemTime` paramétert az aktuális rendszeridővel. A kód ezen része azért van, hogy előkészítse az elküldendő adatokat. Egy valódi meghívónál itt kell elvégezni néhány hívást az alapmodulokhoz (`Kernel/System/*.pm`).

Ha a kérést a `PrepareRequest` függvény bármely része közben le kell állítani a hibakeresési naplóba való bejegyzés előállítása és hibajelzése nélkül, akkor a következő kód használható:

```
# stop requester communication
return {
    Success           => 1,
    StopCommunication => 1,
};
```

Ennek használatával a kérelmező meg foga érteni, hogy a kérést nem szabad folytatni (nem kerül elküldésre a leképező réteghez, és nem kerül elküldésre a hálózati átvitelhez sem). A kérelmező nem fog hibát küldeni a hibakeresési naplóba, hanem csak csendben le fog állni.

```
sub HandleResponse {
    my ( $Self, %Param ) = @_;

    # if there was an error in the response, forward it
    if ( !$Param{ResponseSuccess} ) {
        if ( !IsStringWithData( $Param{ResponseErrorMessage} ) ) {

            return $Self->{DebuggerObject}->Error(
```

(continues on next page)

(folytatás az előző oldalról)

```
                Summary => 'Got response error, but no response error message!
↪',
            );
        }

        return {
            Success      => 0,
            ErrorMessage => $Param{ResponseErrorMessage},
        };
    }

    # we need a TicketNumber
    if ( !IsStringWithData( $Param{Data}->{TicketNumber} ) ) {

        return $Self->{DebuggerObject}->Error( Summary => 'Got no␣
↪TicketNumber!' );
    }

    # prepare TicketNumber
    my %ReturnData = (
        TicketNumber => $Param{Data}->{TicketNumber},
    );

    # check Action
    if ( IsStringWithData( $Param{Data}->{Action} ) ) {
        if ( $Param{Data}->{Action} !~ m{ \A ( .*? ) Test \z }xms ) {

            return $Self->{DebuggerObject}->Error(
                Summary => 'Got Action but it is not in required format!',
            );
        }
        $ReturnData{Action} = $1;
    }

    return {
        Success => 1,
        Data    => \%ReturnData,
    };
}
```

A `HandleResponse` függvényt használják az előző kérésből származó adatok fogadásához és feldolgozásához, amelyet a távoli rendszernek készítettek. Ezeket az adatokat már átadta a leképező réteg, hogy átalakítsa azokat a távoli rendszer formátumáról az OTRS formátumára (ha szükséges).

Ennél a bizonyos példánál ismét ellenőrzi a jegyszámot, és azt is ellenőrzi, hogy a művelet a *Test* szóval végződik-e (amint az a `PrepareRequest` függvényben történt).

---

**Megjegyzés:** Ez a meghívó csak tesztelésekhez van, egy valódi meghívó ellenőrizni fogja, hogy a válasz a távoli rendszer által leírt formátumban volt-e, és végrehajthat néhány műveletet, mint például: egy másik meghívó meghívása, egy hívás végrehajtása egy alapmodulhoz, az adatbázis frissítése, hiba küldése, stb.

---

#### Meghívó beállítási példa

Szükség van ezen meghívómodul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Invoker::Module###Test::Test" Required="0"␣
↪Valid="1">
    <Description Translatable="1">GenericInterface module registration for␣
↪the invoker layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Invoker::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Name">Test</Item>
            <Item Key="Controller">Test</Item>
            <Item Key="ConfigDialog">AdminGenericInterfaceInvokerDefault</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

### 3.3.19 Művelet

A műveletet egy tevékenység végrehajtásához használják az OTRS-en belül. Ezt a tevékenységet a távoli rendszer kéri, és tartalmazhat különleges paramétereket azért, hogy helyesen végrehajtsa a tevékenységet. A tevékenység végrehajtása után az OTRS elküld egy meghatározott megerősítést a távoli rendszernek.

#### Műveleti háttérprogram

Ezután be fogjuk mutatni, hogy hogyan kell egy új műveletet kifejleszteni. Minden egyes műveletnek meg kell valósítania ezeket a szubrutinokat:

- `new`
- `Run`

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk végrehajtani a szolgáltató által kezelt műveletet (`Kernel/GenericInterface/Provider.pm`).

#### Művelet kódpélda

Ebben a szakaszban egy minta műveletmodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::GenericInterface::Operation::Test::Test;
```

(continues on next page)

(folytatás az előző oldalról)

```
use strict;
use warnings;

use Kernel::System::VariableCheck qw(IsHashRefWithData);

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

Felveszünk egy `VariableCheck` modult is bizonyos ellenőrzések végrehajtásához néhány változón. A műveletek nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for my $Needed (qw(DebuggerObject)) {
        if ( !$Param{$Needed} ) {
            return {
                Success      => 0,
                ErrorMessage => "Got no $Needed!"
            };
        }

        $Self->{$Needed} = $Param{$Needed};
    }

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a `new` konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub Run {
    my ( $Self, %Param ) = @_;

    # check data - only accept undef or hash ref
    if ( defined $Param{Data} && ref $Param{Data} ne 'HASH' ) {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Got Data but it is not a hash ref in Operation Test␣
↪backend)!'
        );
    }

    if ( defined $Param{Data} && $Param{Data}->{TestError} ) {

        return {
```

(continues on next page)

(folytatás az előző oldalról)

```
            Success      => 0,
            ErrorMessage => "Error message for error code: $Param{Data}->
↪{TestError}",
            Data         => {
                ErrorData => $Param{Data}->{ErrorData},
            },
        };
    }

    # copy data
    my $ReturnData;

    if ( ref $Param{Data} eq 'HASH' ) {
        $ReturnData = \%{ $Param{Data} };
    }
    else {
        $ReturnData = undef;
    }

    # return result
    return {
        Success => 1,
        Data    => $ReturnData,
    };
}
```

A `Run` függvény az egyes műveletek fő része. Fogadja az összes belsőleg leképezett adatot a távoli rendszertől, amelyre a szolgáltatónak szüksége van a művelet végrehajtásához, végrehajtja a műveletet, és visszaadja az eredményt a szolgáltatónak a külső leképezéshez, valamint visszaszállítja a távoli rendszerhez.

Ez a bizonyos példa ugyanúgy adja vissza az adatokat, ahogy azok a távoli rendszertől jönnek, hacsak a `TestError` paraméter át nincs adva. Ebben az esetben egy hibát ad vissza.

#### Művelet beállítási példa

Szükség van ezen műveletmodul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Operation::Module###Test::Test" Required="0
↪" Valid="1">
    <Description Translatable="1">GenericInterface module registration for␣
↪the operation layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Operation::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Name">Test</Item>
            <Item Key="Controller">Test</Item>
            <Item Key="ConfigDialog">AdminGenericInterfaceOperationDefault</
↪Item>
        </Hash>
```

(continues on next page)

(folytatás az előző oldalról)

```
    </Setting>
</ConfigItem>
```

### Egységteszt példa

Az általános felület műveleteinek egységtesztjei nem különböznek más egységtesztektől, de fontolóra kell venni a helyi tesztelést, viszont egy távoli kapcsolatot is szimulálni kell. Egy jó bevált gyakorlat mindkettőt különválasztva tesztelni, mivel az eredmények némileg különbözők lehetnek.

**Lásd még:**

Ha többet szeretne megtudni az egységtesztekről, akkor vessen egy pillantást az *Egységtesztek* fejezetre.

A következő csak a kezdési pont egy egységteszthez:

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

## no critic (Modules::RequireExplicitPackage)
use strict;
use warnings;
use utf8;

use vars (qw($Self));

use Kernel::GenericInterface::Debugger;
use Kernel::GenericInterface::Operation::Test::Test;

use Kernel::System::VariableCheck qw(:all);

# Skip SSL certificate verification (RestoreDatabase must not be used in this␣
↪test).
$Kernel::OM->ObjectParamAdd(
    'Kernel::System::UnitTest::Helper' => {
        SkipSSLVerify => 1,
    },
);
my $Helper = $Kernel::OM->Get('Kernel::System::UnitTest::Helper');

# get a random number
my $RandomID = $Helper->GetRandomNumber();

# create a new user for current test
my $UserLogin = $Helper->TestUserCreate(
    Groups => ['users'],
);
my $Password = $UserLogin;
```

(continues on next page)

(folytatás az előző oldalról)

```
my $UserID = $Kernel::OM->Get('Kernel::System::User')->UserLookup(
    UserLogin => $UserLogin,
);

# set web-service name
my $WebserviceName = '-Test-' . $RandomID;

# create web-service object
my $WebserviceObject = $Kernel::OM->Get(
↪'Kernel::System::GenericInterface::Webservice');
$Self->Is(
    'Kernel::System::GenericInterface::Webservice',
    ref $WebserviceObject,
    "Create web service object",
);

my $WebserviceID = $WebserviceObject->WebserviceAdd(
    Name   => $WebserviceName,
    Config => {
        Debugger => {
            DebugThreshold => 'debug',
        },
        Provider => {
            Transport => {
                Type => '',
            },
        },
    },
    ValidID => 1,
    UserID  => 1,
);
$Self->True(
    $WebserviceID,
    "Added Web Service",
);

# get remote host with some precautions for certain unit test systems
my $Host = $Helper->GetTestHTTPHostname();

my $ConfigObject = $Kernel::OM->Get('Kernel::Config');

# prepare web-service config
my $RemoteSystem =
    $ConfigObject->Get('HttpType')
    . '://'
    . $Host
    . '/'
    . $ConfigObject->Get('ScriptAlias')
    . '/nph-genericinterface.pl/WebserviceID/'
    . $WebserviceID;
```

(continues on next page)

(folytatás az előző oldalról)

```
my $WebserviceConfig = {
    Description =>
        'Test for Ticket Connector using SOAP transport backend.',
    Debugger => {
        DebugThreshold => 'debug',
        TestMode       => 1,
    },
    Provider => {
        Transport => {
            Type   => 'HTTP::SOAP',
            Config => {
                MaxLength => 10000000,
                NameSpace => 'http://otrs.org/SoapTestInterface/',
                Endpoint  => $RemoteSystem,
            },
        },
        Operation => {
            Test => {
                Type => 'Test::Test',
            },
        },
    },
    Requester => {
        Transport => {
            Type   => 'HTTP::SOAP',
            Config => {
                NameSpace => 'http://otrs.org/SoapTestInterface/',
                Encoding  => 'UTF-8',
                Endpoint  => $RemoteSystem,
            },
        },
        Invoker => {
            Test => {
                Type => 'Test::TestSimple'
                ,    # requester needs to be Test::TestSimple in order to␣
↪simulate a request to a remote system
            },
        },
    },
};

# update web-service with real config
# the update is needed because we are using
# the WebserviceID for the Endpoint in config
my $WebserviceUpdate = $WebserviceObject->WebserviceUpdate(
    ID      => $WebserviceID,
    Name    => $WebserviceName,
    Config  => $WebserviceConfig,
    ValidID => 1,
    UserID  => $UserID,
);
$Self->True(
```

(continues on next page)

(folytatás az előző oldalról)

```
    $WebserviceUpdate,
    "Updated Web Service $WebserviceID - $WebserviceName",
);

# debugger object
my $DebuggerObject = Kernel::GenericInterface::Debugger->new(
    DebuggerConfig => {
        DebugThreshold => 'debug',
        TestMode       => 1,
    },
    WebserviceID      => $WebserviceID,
    CommunicationType => 'Provider',
);
$Self->Is(
    ref $DebuggerObject,
    'Kernel::GenericInterface::Debugger',
    'DebuggerObject instantiate correctly',
);

# define test cases
my @Tests = (
    {
        Name           => 'Test case name',
        SuccessRequest => 1,                    # 1 or 0
        RequestData    => {

            # ... add test data
        },
        ExpectedReturnLocalData => {
            Data => {

                # ... add expected local results
            },
            Success => 1,                       # 1 or 0
        },
        ExpectedReturnRemoteData => {
            Data => {

                # ... add expected remote results
            },
            Success => 1,                       # 1 or 0
        },
        Operation => 'Test',
    },

    # ... add more test cases
);

TEST:
for my $Test (@Tests) {

    # create local object
```

(continues on next page)

(folytatás az előző oldalról)

```
    my $LocalObject = "Kernel::GenericInterface::Operation::Test::$Test->
↪{Operation}"->new(
        DebuggerObject => $DebuggerObject,
        WebserviceID   => $WebserviceID,
    );

    $Self->Is(
        "Kernel::GenericInterface::Operation::Test::$Test->{Operation}",
        ref $LocalObject,
        "$Test->{Name} - Create local object",
    );

    my %Auth = (
        UserLogin => $UserLogin,
        Password  => $Password,
    );
    if ( IsHashRefWithData( $Test->{Auth} ) ) {
        %Auth = %{ $Test->{Auth} };
    }

    # start requester with our web-service
    my $LocalResult = $LocalObject->Run(
        WebserviceID => $WebserviceID,
        Invoker      => $Test->{Operation},
        Data         => {
            %Auth,
            %{ $Test->{RequestData} },
        },
    );

    # check result
    $Self->Is(
        'HASH',
        ref $LocalResult,
        "$Test->{Name} - Local result structure is valid",
    );

    # create requester object
    my $RequesterObject = $Kernel::OM->Get(
↪'Kernel::GenericInterface::Requester');
    $Self->Is(
        'Kernel::GenericInterface::Requester',
        ref $RequesterObject,
        "$Test->{Name} - Create requester object",
    );

    # start requester with our web-service
    my $RequesterResult = $RequesterObject->Run(
        WebserviceID => $WebserviceID,
        Invoker      => $Test->{Operation},
        Data         => {
            %Auth,
```

(continues on next page)

(folytatás az előző oldalról)

```
            %{ $Test->{RequestData} },
        },
    );

    # check result
    $Self->Is(
        'HASH',
        ref $RequesterResult,
        "$Test->{Name} - Requester result structure is valid",
    );

    $Self->Is(
        $RequesterResult->{Success},
        $Test->{SuccessRequest},
        "$Test->{Name} - Requester successful result",
    );

    # ... add tests for the results
}

# delete web service
my $WebserviceDelete = $WebserviceObject->WebserviceDelete(
    ID     => $WebserviceID,
    UserID => $UserID,
);
$Self->True(
    $WebserviceDelete,
    "Deleted Web Service $WebserviceID",
);

# also delete any other added data during the this test, since␣
↪RestoreDatabase must not be used.

1;
```

### WSDL-kiterjesztés példa

A WSDL-fájlok tartalmazzák a webszolgáltatás és a SOAP-üzenethez tartozó műveleteinek meghatározásait abban az esetben, a kibővítjük a `development/webservices/GenericTickeConnectorSOAP.wsdl` fájlt néhány helyen:

Port típusa:

```
<wsdl:portType name="GenericTicketConnector_PortType">
    <!-- ... -->
    <wsdl:operation name="Test">
        <wsdl:input message="tns:TestRequest"/>
        <wsdl:output message="tns:TestResponse"/>
    </wsdl:operation>
<!-- ... -->
```

Kötés:

```
<wsdl:binding name="GenericTicketConnector_Binding" type=
↪"tns:GenericTicketConnector_PortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/
↪http"/>
    <!-- ... -->
    <wsdl:operation name="Test">
        <soap:operation soapAction="http://www.otrs.org/TicketConnector/Test"/
↪>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <!-- ... -->
</wsdl:binding>
```

Típus:

```
<wsdl:types>
    <xsd:schema targetNamespace="http://www.otrs.org/TicketConnector/"␣
↪xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!-- ... -->
        <xsd:element name="Test">
            <xsd:complexType>
                <xsd:sequence>
                        <xsd:element minOccurs="0" name="Param1" type=
↪"xsd:string"/>
                        <xsd:element minOccurs="0" name="Param2" type=
↪"xsd:positiveInteger"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="TestResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element maxOccurs="unbounded" minOccurs="1" name=
↪"Attribute1" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    <!-- ... -->
    </xsd:schema>
</wsdl:types>
```

Üzenet:

```
<!-- ... -->
<wsdl:message name="TestRequest">
    <wsdl:part element="tns:Test" name="parameters"/>
</wsdl:message>
<wsdl:message name="TestResponse">
```

(continues on next page)

(folytatás az előző oldalról)

```
    <wsdl:part element="tns:TestResponse" name="parameters"/>
</wsdl:message>
<!-- ... -->
```

### WADL-kiterjesztés példa

A WADL-fájlok tartalmazzák a webszolgáltatás és a REST-felülethez tartozó műveleteinek meghatározásait. Adjon hozzá egy új erőforrást a `development/webservices/GenericTickeConnectorREST.wadl` fájlhoz.

```
<resources base="http://localhost/otrs/nph-genericinterface.pl/Webservice/
↪GenericTicketConnectorREST">
  <!-- ... -->
  <resource path="Test" id="Test">
    <doc xml:lang="en" title="Test"/>
      <param name="Param1" type="xs:string" required="false" default="" style=
↪"query" xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      <param name="Param2" type="xs:string" required="false" default="" style=
↪"query" xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      <method name="GET" id="GET_Test">
        <doc xml:lang="en" title="GET_Test"/>
        <request/>
        <response status="200">
          <representation mediaType="application/json; charset=UTF-8"/>
        </response>
      </method>
    </resource>
  </resource>
  <!-- ... -->
</resources>
```

### Webszolgáltatás SOAP-kiterjesztés példa

A webszolgáltatások importálhatók az OTRS-be egy előre meghatározott szerkezettel rendelkező YAML-fájllal. Ebben az esetben a `development/webservices/GenericTickeConnectorSOAP.yml` fájlt bővítjük ki egy SOAP webszolgáltatáshoz.

```
Provider:
  Operation:
    # ...
    Test:
      Description: This is only a test
      MappingInbound: {}
      MappingOutbound: {}
      Type: Test::Test
```

#### Webszolgáltatás REST-kiterjesztés példa

A webszolgáltatások importálhatók az OTRS-be egy előre meghatározott szerkezettel rendelkező YAML-fájllal. Ebben az esetben a `development/webservices/GenericTickeConnectorREST.yml` fájlt bővítjük ki egy REST webszolgáltatáshoz.

```
Provider:
  Operation:
    # ...
    Test:
      Description: This is only a test
      MappingInbound: {}
      MappingOutbound: {}
      Type: Test::Test
  # ...
  Transport:
    Config:
      # ...
      RouteOperationMapping:
        # ..
        Test:
          RequestMethod:
          - GET
          Route: /Test
```

### 3.3.20 OTRS démon

Az OTRS démon egy elkülönített folyamat, amely segít az OTRS-nek bizonyos műveleteket aszinkron módon és a webszolgáltatás folyamattól leválasztva végrehajtani, de ugyanazt az adatbázist megosztva.

#### OTRS démonmodulok

A `bin/otrs.Daemon.pl` OTRS démon fő célja, hogy meghívja (démonizálja) a rendszerbeállításokban lévő összes regisztrált démonmodult.

Minden egyes démonmodulnak meg kell valósítania egy közös API-t annak érdekében, hogy az OTRS démon helyesen tudja meghívni, és félig állandó folyamat legyen a rendszeren. Az állandó folyamat megnövelheti a méretét és memóriahasználatát az idő múlásával, és normális esetben nem válaszolnak a beállításokban lévő változásokra. Ezért kell a démonmoduloknak megvalósítaniuk egy eldobási mechanizmust, hogy leállíthatók és újra meghívhatók legyenek időről időre, felszabadítva a rendszer erőforrásait és újraolvasva a beállításokat.

Egy démonmodul lehet mindenre jó megoldás bizonyos feladat végrehajtásánál, de lehet olyan eset is, amikor egy megoldás különböző démonmodulokat igényel az összetettsége miatt. Pontosan ez az eset az OTRS ütemező démonjával, amely fel van osztva számos démonmodulra, beleértve a feladatkezeléshez és a feladat-végrehajtáshoz szükséges néhány démonmodult.

Nem szükséges mindig új démonmodult létrehozni bizonyos feladatok végrehajtásához. Általában az OTRS ütemező démon elboldogul ezek jelentős részével – akár ha egy olyan OTRS függvényről van szó, amelyet rendszeresen végre kell hajtani (CRON-szerűen), vagy ha egy OTRS esemény aktiválta azt – az OTRS ütemezőnek képesnek kell lenni kezelnie mindenféle beállítás nélkül, vagy egy új ütemező feladatelvégző modul hozzáadásával.

### Új démonmodul létrehozása

Az összes démonmodulnak regisztrálva kell lennie a rendszerbeállításokban azért, hogy a fő OTRS démon meg tudja hívni azokat.

### Démonmodul regisztrációs kódpélda

```
<Setting Name="DaemonModules###TestDaemon" Required="1" Valid="1">
    <Description Translatable="1">The daemon registration for the scheduler␣
↪generic agent task manager.</Description>
    <Navigation>Daemon::ModuleRegistration</Navigation>
    <Value>
        <Hash>
            <Item Key="Module">
↪Kernel::System::Daemon::DaemonModules::TestDaemon</Item>
        </Hash>
    </Value>
</Setting>
```

### Démonmodul kódpélda

A következő kód egy olyan démonmodult valósít meg, amely megjeleníti a rendszeridőt 2 másodpercenként.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Daemon::DaemonModules::TestDaemon;

use strict;
use warnings;
use utf8;

use Kernel::System::VariableCheck qw(:all);

use parent qw(Kernel::System::Daemon::BaseDaemon);

our @ObjectDependencies = (
    'Kernel::Config',
    'Kernel::System::Cache',
    'Kernel::System::DB',
);
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

Ebben az esetben a `BaseDaemon` osztályból származtatunk le, és az objektumkezelő függőségei be vannak állítva.

```
sub new {
    my ( $Type, %Param ) = @_;

    # Allocate new hash for object.
    my $Self = {};
    bless $Self, $Type;

    # Get objects in constructor to save performance.
    $Self->{ConfigObject} = $Kernel::OM->Get('Kernel::Config');
    $Self->{CacheObject}  = $Kernel::OM->Get('Kernel::System::Cache');
    $Self->{DBObject}     = $Kernel::OM->Get('Kernel::System::DB');

    # Disable in memory cache to be clusterable.
    $Self->{CacheObject}->Configure(
        CacheInMemory  => 0,
        CacheInBackend => 1,
    );

    $Self->{SleepPost} = 2;          # sleep 2 seconds after each loop
    $Self->{Discard}   = 60 * 60;    # discard every hour

    $Self->{DiscardCount} = $Self->{Discard} / $Self->{SleepPost};

    $Self->{Debug}      = $Param{Debug};
    $Self->{DaemonName} = 'Daemon: TestDaemon';

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát. Néhány felhasznált objektum is itt lesz létrehozva. Erősen ajánlott a memóriába történő gyorsítótárazás letiltása a démonmodulokban, különösen akkor, ha az OTRS fürtözött környezetben fut.

Azért, hogy ez a démonmodul minden második másodpercben végrehajtható legyen, egy alvási idő meghatározása szükséges annak megfelelően, egyébként azonnal végrehajtásra kerül, amint lehetséges.

A démonmodul frissítése időről időre azért szükséges, hogy meghatározható legyen, mikor kell eldobni.

A következő függvényeknél (`PreRun`, `Run` és `PostRun`) ha azok hamis értékkel térnek vissza, akkor a fő OTRS démon el fogja dobni az objektumot, és egy újat hoz létre, amint lehetséges.

```
sub PreRun {
    my ( $Self, %Param ) = @_;

    # Check if database is on-line.
    return 1 if $Self->{DBObject}->Ping();

    sleep 10;

    return;
}
```

A `PreRun` metódus a fő démonmodul metódusa előtt kerül végrehajtásra, és a célja néhány teszt elvégzése a valódi műfelet előtt. Ebben az esetben az adatbázis ellenőrzése készen van (mindig javasolt), egyébként 10 másodpercet alszik. Ez azért szükséges, hogy megvárja az adatbázis-kapcsolat ismételt felépítését.

```
sub Run {
    my ( $Self, %Param ) = @_;

    print "Current time " . localtime . "\n";

    return 1;
}
```

A `Run` metódus az, ahol a fő démonmodul kódja található. Ebben az esetben csak az aktuális időt írja ki.

```
sub PostRun {
    my ( $Self, %Param ) = @_;
    sleep $Self->{SleepPost};
    $Self->{DiscardCount}--;

    if ( $Self->{Debug} ) {
        print "  $Self->{DaemonName} Discard Count: $Self->{DiscardCount}\n";
    }

    return if $Self->{DiscardCount} <= 0;

    return 1;
}
```

A `PostRun` metódus használható az alvások végrehajtásához (annak megakadályozásához, hogy a démonmodul túl gyakran legyen végrehajtva), valamint az objektum biztonságos eldobásának kezeléséhez is. Egyéb műveletek is elvégezhetők itt, mint például ellenőrzés vagy tisztítás.

```
sub Summary {
    my ( $Self, %Param ) = @_;

    my %Summary = (
        Header => 'Test Daemon Summary:',
        Column => [
            {
                Name        => 'SomeColumn',
                DisplayName => 'Some Column',
                Size        => 15,
            },
            {
                Name        => 'AnotherColumn',
                DisplayName => 'Another Column',
                Size        => 15,
            },
            # ...
        ],
        Data => [
            {
                SomeColumn    => 'Some Data 1',
                AnotherColumn => 'Another Data 1',
            },
            {
                SomeColumn    => 'Some Data 2',
```

(continues on next page)

(folytatás az előző oldalról)

```
                AnotherColumn => 'Another Data 2',
            },
            # ...
        ],
        NoDataMesssage => '',
    );

    return \%Summary;
}
```

A `Summary` metódust a `Maint::Daemon::Summary` konzolparancs hívja meg, és `Header`, `Column`, `Data` és `NoDataMessages` kulcsokat kell visszaadnia. A `Column` és a `Data` kulcsoknak tömböknek vagy kivonatoknak kell lenniük. Arra használható, hogy hasznos információkat jelenítsen meg arról, amit a démonmodul éppen csinál, vagy ami eddig történt. Ez a metódus elhagyható.

```
1;
```

Fájl vége.

### 3.3.21 OTRS ütemező

Az OTRS ütemező a démonmodulok és a feladatelvégzők együttese, amelyek együtt futnak azért, hogy az összes szükséges OTRS feladatot aszinkron módon végrehajtsák a webkiszolgáló folyamatából.

#### OTRS ütemező feladatkezelők

**`SchedulerCronTaskManager`** Ez kiolvassa a regisztrált cron-feladatokat az OTRS rendszerbeállításaiból, és meghatározza a helyes időt a végrehajtandó feladat létrehozásához.

**`SchedulerFutureTaskManager`** Ez ellenőrzi azokat a feladatokat, amelyek úgy vannak beállítva, hogy csak egy alkalommal fussanak a jövőben, és beállítja, hogy a feladat időben kerüljön végrehajtásra. Például amikor egy általános felület meghívónak nem sikerül elérnie a távoli kiszolgálót, akkor ütemezni tudja magát, hogy 5 perccel később újra fusson.

**`SchedulerGenericAgentTaskManager`** Ez folyamatosan olvassa az általános ügyintéző feladatokat, amelyek rendszeres időközönkénti futáshoz vannak beállítva, és annak megfelelően állítja be azok végrehajtását.

Amikor ezek a feladatkezelők nem elegendőek, akkor egy új démonmodul hozható létre. Egy feladat regisztrálásához a `Run()` metódusuk egy bizonyos pontján meg kell hívni a `TaskAdd()` függvényt a `chedulerDB` objektumból, és amint regisztrálva lett, akkor a `SchedulerTaskWorker` végrehajtja a következő szabad időszeletben.

#### OTRS ütemező feladatelvégzők

**`SchedulerTaskWorker`** Ez az aszinkron végrehajtó használatával végrehajtja az előző feladatkezelő által tervezett összes feladatot, és még azokat is, amelyek közvetlenül a kódból jönnek.

Annak érdekében, hogy az összes feladatot végrehajtsa, a `SchedulerTaskWorker` meghív egy háttérprogram-modult (feladatelvégzőt) az adott feladat végrehajtásához. Az elvégző modult a feladat típusa határozza meg. Ha új feladattípus kerül hozzáadásra, akkor az új feladatelvégzőt is igényel.

### Új ütemező feladatelvégző létrehozása

A `Kernel/System/Daemon/DaemonModules/SchedulerTaskWorker` mappa alatt elhelyezett összes fájl lehet potenciális feladatelvégző, és azok nem igényelnek semmilyen regisztrációt a rendszerbeállításokban.

### Ütemező feladatelvégző kódpélda

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package␣
 ↪Kernel::System::Daemon::DaemonModules::SchedulerTaskWorker::TestWorker;

use strict;
use warnings;

use parent qw(Kernel::System::Daemon::DaemonModules::BaseTaskWorker);

our @ObjectDependencies = (
    'Kernel::System::Log',
);
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

Ebben az esetben a `BaseTaskWorker` osztályból származtatunk le, és az objektumkezelő függőségei be vannak állítva.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {};
    bless( $Self, $Type );

    $Self->{Debug}      = $Param{Debug};
    $Self->{WorkerName} = 'Worker: Test';

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát.

```
sub Run {
    my ( $Self, %Param ) = @_;

    # Check task params.
```

(continues on next page)

(folytatás az előző oldalról)

```
    my $CheckResult = $Self->_CheckTaskParams(
        %Param,
        NeededDataAttributes => [ 'NeededAtrribute1', 'NeededAtrribute2' ],
        DataParamsRef        => 'HASH', # or 'ARRAT'
    );

    # Stop execution if an error in params is detected.
    return if !$CheckResult;

    my $Success;
    my $ErrorMessage;

    if ( $Self->{Debug} ) {
        print "    $Self->{WorkerName} executes task: $Param{TaskName}\n";
    }

    do {

        # Localize the standard error.
        local *STDERR;

        # Redirect the standard error to a variable.
        open STDERR, ">>", \$ErrorMessage;

        $Success = $Kernel::OM->Get('Kernel::System::MyPackage')->Run(
            Param1 => 'someparam',
        );
    };

    if ( !$Success ) {

        $ErrorMessage ||= "$Param{TaskName} execution failed without an error␣
↪message!";

        $Self->_HandleError(
            TaskName     => $Param{TaskName},
            TaskType     => 'Test',
            LogMessage   => "There was an error executing $Param{TaskName}:
↪$ErrorMessage",
            ErrorMessage => "$ErrorMessage",
        );
    }

    return $Success;
}
```

A `Run` a fő metódus. Egy `_CheckTaskParams()` hívás az alaposztályból megspórol néhány kódsort. A feladat végrehajtása a szabványos hibakimenet megszerzése közben nagyon jó gyakorlat, mivel az OTRS ütemező normális esetben felügyelet nélkül fut, és az összes hiba egy változóba történő mentése lehetővé fogja tenni a későbbi feldolgozást. A `_HandleError()` közös felületet biztosít a hibaüzenetek e-mailben történő küldéséhez a rendszerbeállításokban megadott címzetteknek.

```
1;
```

Fájl vége.

### 3.3.22 Áttekintés

A dinamikus mezők olyan egyéni mezők, amelyek hozzáadhatók egy képernyőhöz, hogy javítsák és információkat adjanak hozzá egy objektumhoz (például egy jegyhez vagy egy bejegyzéshez).

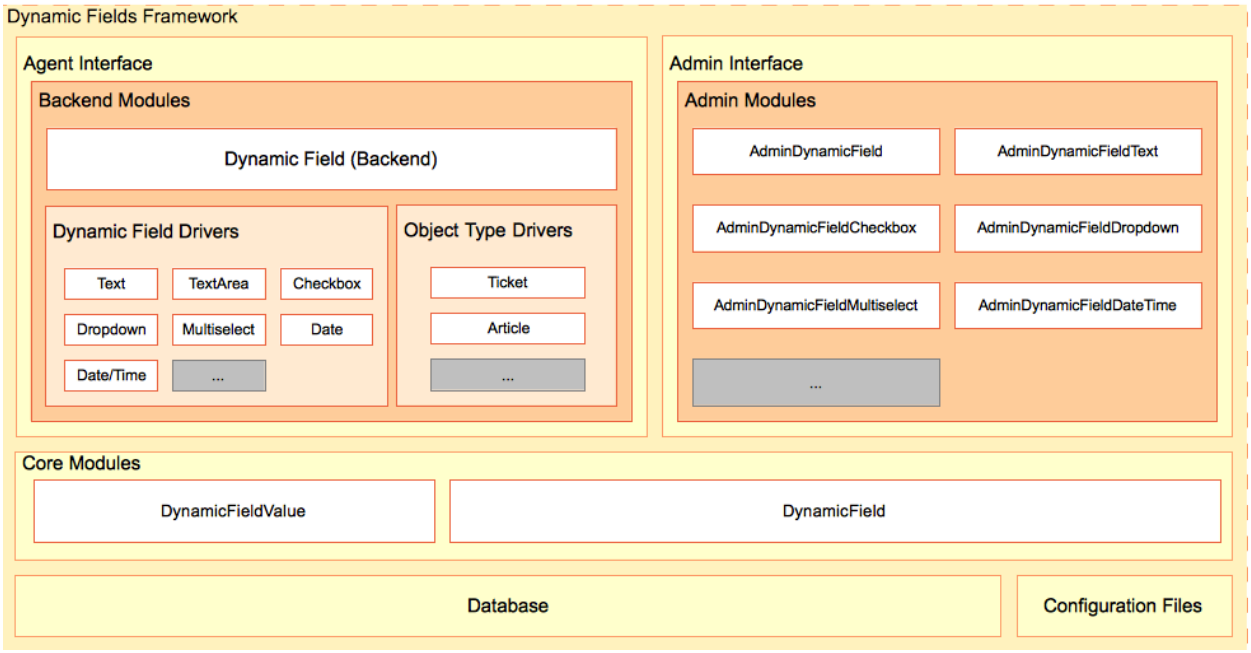
A jegyek vagy a bejegyzések annyi mezővel rendelkezhetnek, amennyi szükséges. Lehetséges a dinamikus mezők keretrendszerének használata egyéb objektumoknál is, ahelyett hogy csak a jegynél vagy a bejegyzésnél lenne használható.

A moduláris tervezésének köszönhetően az egyes dinamikus mező típusok egy keretrendszerhez tartozó bővítményként láthatók, és ez a bővítmény lehet egy szabványos OTRS csomag is a dinamikus mezők elérhető típusainak kiterjesztéséhez, vagy akár a jelenlegi dinamikus mező további függvényekkel való kiterjesztéséhez.

### 3.3.23 Dinamikus mezők keretrendszer

Az új dinamikus mezők létrehozása előtt szükséges megérteni azok keretrendszerét, és hogy az OTRS képernyők hogyan lépnek kölcsönhatásba azokkal, valamint a mögöttes API-t.

A következő kép a dinamikus mezők keretrendszer szerkezetét mutatja be.




4. ábra: Dinamikus mezők keretrendszer

#### Dinamikus mező háttérprogram-modulok

### Dinamikus mező (háttérprogram)

Az előtétprogram modulokban normális esetben a `BackendObject` nevű objektum a közvetítő az előtétprogram modulok és az egyes konkrét dinamikus mező megvalósítás vagy illesztőprogram között. Ez határoz meg egy általános közbenső API-t az összes dinamikus mező illesztőprogramhoz, és az egyes illesztőprogramok felelőssége a közbenső API megvalósítása a mező sajátos szükségleteihez.

A dinamikus mező háttérprogram az összes illesztőprogram fő vezérlője. Ebben a modulban minden egyes függvény felelős a szükséges paraméterek ellenőrzéséért, és ugyanazon függvény meghívásáért az adott illesztőprogramban a kapott dinamikus mező beállítási paraméter szerint.

Ez a modul felelős bizonyos függvények meghívásáért is minden egyes objektumtípus delegáltnál (úgymint `Ticket` vagy `Article`). Például egy előzmény bejegyzés hozzáadásához vagy egy esemény elsütéséhez.

Ez a modul az `$OTRS_HOME/Kernel/System/DynamicField/Backend.pm` fájlban található.

### Dinamikus mező illesztőprogramok

Egy dinamikus mező illesztőprogram a dinamikus mező megvalósítása. Minden egyes illesztőprogramnak meg kell valósítania a háttérprogramban meghatározott összes kötelező függvényt (van néhány olyan függvény, amely egy viselkedéstől függ, és nem szükséges megvalósítani azokat, ha a dinamikus mező nem rendelkezik azzal a bizonyos viselkedéssel).

Egy illesztőprogram felelős annak ismeretéért, hogy hogyan kérje le a saját értékét vagy értékeit egy webkérésből vagy egy profilból (mint például egy keresési profilból). Szükséges tudnia a HTML kódot is a szerkesztő vagy megjelenítő képernyőkön lévő mező megjelenítéséhez, vagy hogy hogyan lépjen kölcsönhatásba a statisztikák modullal, többek között a függvényekkel.

Ezek a modulok az `$OTRS_HOME/Kernel/System/DynamicField/Driver/*.pm` fájlokban találhatók.

Létezik néhány alap illesztőprogram, úgymint `Base.pm`, `BaseText.pm`, `BaseSelect.pm` és `BaseDateTime.pm`, amely gyakori függvényeket valósít meg bizonyos illesztőprogramokhoz (például a `TextArea.pm` illesztőprogram a `BaseText.pm` fájlt használja, amely a `Base.pm` fájlt használja, ekkor a `TextArea` csak azon függvények megvalósítását igényli, amelyek hiányoznak a `Base.pm` és `BateText.pm` fájlokból, vagy azokat, amelyek különleges esetek).

A következő az illesztőprogramok öröklődési fája:

- `Base.pm`
  - `BaseText.pm`
    - `Text.pm`
    - `TextArea.pm`
  - `BaseSelect.pm`
    - `Dropdown.pm`
    - `Multiselect.pm`
  - `BaseDateTime.pm`
    - `DateTime.pm`
    - `Date.pm`
  - `Checkbox.pm`

#### Objektumtípus delegált

Egy objektumtípus delegált felelős bizonyos függvények végrehajtásáért a dinamikus mezőhöz kapcsolt objektumon. Ezeket a függvényeket a háttérprogram objektum aktiválja, amint szükség van rájuk.

Ezek a modulok az `$OTRS_HOME/Kernel/System/DynamicField/ObjectType/*.pm` fájlokban találhatók.

#### Dinamikus mezők adminisztrátori moduljai

A dinamikus mezők kezeléséhez (hozzáadás, szerkesztés és felsorolás) már egy csomó modul van kifejlesztve. Van egy bizonyos fő modul (`AdminDynamicField.pm`), amely megjeleníti a meghatározott dinamikus mezők listáját, és más modulokon belülről hívják meg új dinamikus mezők létrehozásához vagy a meglévők módosításához.

Normális esetben egy dinamikus mező illesztőprogramnak saját adminisztrátori modulra van szüksége (adminisztrátori párbeszédablak) a tulajdonságai meghatározásához. Ez a párbeszédablak esetleg eltérhet a többi illesztőprogramtól. De ez nem kötelező, az illesztőprogramok megoszthatják az adminisztrátori párbeszédablakokat, ha szükséges információkat biztosíthatnak az összes olyan illesztőprogramhoz, amelyek hozzájuk vannak kapcsolva, nem számít, hogy eltérő típusból származnak. Ami kötelező, hogy minden egyes illesztőprogramnak hozzákapcsolva kell lennie egy adminisztrátori párbeszédablakhoz (például a szöveg és a szövegterület illesztőprogramok megosztják az `AdminDynamicFieldText.pm` adminisztrátori párbeszédablakot, és a dátum és a dátum/idő illesztőprogramok megosztják az `AdminDynamicFieldDateTime.pm` adminisztrátori párbeszédablakot).

Az adminisztrátori párbeszédablakok a normál OTRS adminisztrátori modulszabályokat és szerkezetet követik. De a szabványosításhoz az összes beállítás közös részének az összes dinamikus mezőnél ugyanolyan megjelenésűnek kell lennie az összes adminisztrátori párbeszédablaknál.

Ezek a modulok az `$OTRS_HOME/Kernel/Modules/*.pm` fájlokban találhatók.

---

**Megjegyzés:** Minden adminisztrátori párbeszédablaknak szüksége van a neki megfelelő HTML sablonfájlra (`.tt`).

---

#### Dinamikus mezők alapmoduljai

Ezek a modulok olvassák és írják a dinamikus mezők információit az adatbázistáblákban.

**`DynamicField.pm`** Ez a modul felelős a dinamikus mező meghatározások kezeléséért. Ez biztosítja az alap API-t a hozzáadáshoz, megváltoztatáshoz, törléshez, felsoroláshoz és a dinamikus mezők lekéréséhez. Ez a modul az `$OTRS_HOME/Kernel/System/DynamicField.pm` fájlban található.

**`DynamicFieldValue.pm`** Ez a modul felelős a dinamikus mező értékeinek olvasásáért és írásáért az űrlapon és az adatbázisban. Ezt a modult erősen használják az illesztőprogramok, és az `$OTRS_HOME/Kernel/System/DynamicFieldValue.pm` fájlban található.

#### Dinamikus mezők adatbázistáblái

Két tábla van az adatbázisban a dinamikus mező információinak tárolásához:

**`dynamic_field`** A `DynamicField.pm` alapmodul használja, és a dinamikus mező meghatározásokat tárolja.

**dynamic_field_value** A `DynamicFieldValue.pm` alapmodul használja a dinamikus mező értékeinek mentéséhez minden egyes dinamikus mező és minden egyes objektumtípus példánynál.

#### Dinamikus mezők beállítófájljai

A háttérprogram-modulnak szüksége van egy módra megtudni azt, hogy mely illesztőprogramok léteznek, mivel az illesztőprogramok mennyisége egyszerűen kiterjeszthető. Ezek kezelésének legegyszerűbb módja a rendszerbeállítás használata, ahol a dinamikus mező illesztőprogramok és az objektumtípus illesztőprogramok információi eltárolhatók és kiterjeszthetők.

A fő adminisztrátori modulnak is szükséges tudnia ezeket az információkat az elérhető dinamikus mező illesztőprogramokról a hozzájuk kapcsolt adminisztrátori párbeszédablakok használatához, a dinamikus mezők létrehozásához vagy módosításához.

Az előtétprogram-moduloknak szükségük van a rendszerbeállítások olvasására megtudni azt, hogy mely dinamikus mezők vannak bekapcsolva az egyes képernyőknél, és melyek kötelezőek. Például a `Ticket::Frontend::AgentTicketPhone###DynamicField` tárolja az aktív, kötelező és inaktív dinamikus mezőket az *Új telefonos jegy* képernyőnél.

### 3.3.24 Dinamikus mező kölcsönhatása az előtétprogram-modulokkal

Ismerve azt, hogy az előtétprogram-modulok hogyan lépnek kölcsönhatásba a dinamikus mezőkkel, nem feltétlenül szükséges a dinamikus mezők kiterjesztése a jegy vagy bejegyzés objektumokhoz, mivel már elő van készítve az összes olyan képernyő, amely dinamikus mezőket tud használni. De egyéni fejlesztések esetén vagy a dinamikus mezők más objektumokhoz történő kiterjesztéséhez nagyon hasznos tudni, hogy a dinamikus mezők keretrendszere hogyan érhető el egy előtétprogram-modulból.

A következő kép egy egyszerű példáját mutatja be annak, hogy a dinamikus mezők hogyan lépnek kölcsönhatásba az OTRS keretrendszer többi részével.

Az első lépés, hogy az előtétprogram-modul beolvassa a beállított dinamikus mezőket. Például az `AgentTicketNote` modulnak be kell olvasnia a `Ticket::Frontend::AgentTicketNote###DynamicField` beállítást. Ez a beállítás használható szűrőparaméterként a `DynamicFieldListGet()` `DynamicField` alapmodul függvénynél. A képernyő tárolhatja ennek a függvénynek az eredményeit, hogy meglegyen az aktivált dinamikus mezők listája ennél a bizonyos képernyőnél.

Ezután a képernyőnek meg kell próbálnia lekérni az értékeket a webkérésből. Erre a célra használhatja az `EditFieldValueGet()` háttérprogram-objektum függvényt, és használhatja ezeket az értékeket az ACL-ek aktiválásához. A háttérprogram-objektum minden egyes illesztőprogramot használni fog a különleges műveletek végrehajtásához az összes függvénynél.

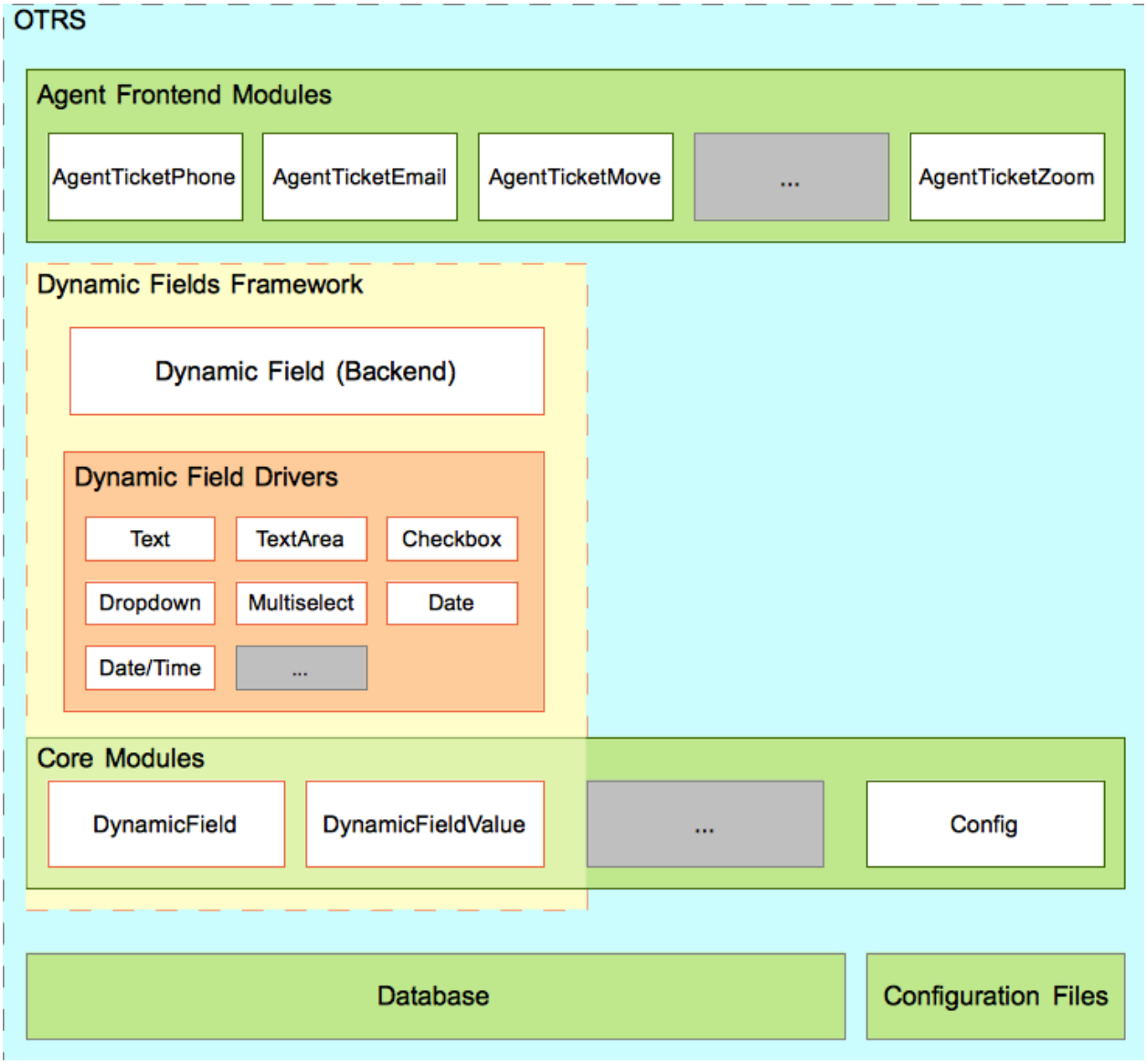
A folytatáshoz a képernyőnek le kell kérnie a HTML-t minden egyes mezőhöz annak megjelenítéséhez. Az `EditFieldRender()` háttérprogram-objektum függvény használható ezen művelet és az ACL-ek korlátozásának végrehajtásához, valamint a webkérésből származó értékek átadhatók ennek a függvénynek azért, hogy jobb eredményeket kapjon. Egy elküldés esetén a képernyő használhatja az `EditFieldValueValidate()` háttérprogram-objektum függvényt is a kötelező mezők ellenőrzéséhez.

---

**Megjegyzés:** A többi képernyő használhatja a `DisplayFieldRender()` függvényt az `EditFieldRender()` helyett, ha a képernyő csak a mezőértéket jeleníti meg, és ilyen esetben nincs szükség értékellenőrzésre.

---

A dinamikus mező értékének tárolásához szükséges az objektumazonosító lekérése. Ennél a példánál ha a dinamikus mező hozzá van kapcsolva egy jegy objektumhoz, akkor a képernyőnek már rendelkeznie




5. ábra: Dinamikus mezők kölcsönhatása

kell a `TicketID` azonosítóval, egyébként ha a mező hozzá van kapcsolva egy bejegyzés objektumhoz azért, hogy beállítsa a mező értékét, akkor először a bejegyzés létrehozása szükséges. A háttérprogram-objektumból a `ValueSet()` függvény használható a dinamikus mező értékének beállításához.

Összefoglalva, az előtétprogram-moduloknak nem szükséges tudniuk, hogy az egyes dinamikus mezők hogyan működnek belsőleg azért, hogy lekérjék vagy beállítsák az értékeiket vagy megjelenítsék azokat. Egyszerűen csak meg kell hívnia a háttérprogram-objektum modult, és általános módon kell használnia a mezőket.

### 3.3.25 Hogyan lehet kiterjeszteni a dinamikus mezőket

Számos módszer létezik a dinamikus mezők kiterjesztésére. A következő szakaszok meg fogják próbálni a leggyakoribb forgatókönyveket bemutatni.

#### Egy új dinamikus mező típus létrehozása (a jegy vagy bejegyzés objektumokhoz)

Egy új dinamikus mező típus létrehozásához a következők szükségesek:

1. Hozzon létre egy dinamikus mező illesztőprogramot. Ez az új mező fő modulja.
2. Hozzon létre vagy használjon egy meglévő adminisztrátori párbeszédablakot egy kezelőfelület megszerzéséhez, és a konfigurációs beállításainak megadásához.
3. Hozzon létre egy beállítófájlt az új mező regisztrálásához a háttérprogramban (vagy a keretrendszerben lévő új adminisztrátori párbeszédablakokban, ha szükséges), valamint hogy képes legyen példányokat vagy azt létrehozni.

#### Egy új dinamikus mező típus létrehozása (egyéb objektumokhoz)

Egy új dinamikus mező típus létrehozásához más objektumoknál a következők szükségesek:

1. Hozzon létre egy dinamikus mező illesztőprogramot. Ez az új mező fő modulja.
2. Hozzon létre egy objektumtípus delegáltat. Ez akkor is szükséges, ha a másik objektum nem igényel semmilyen különleges adatkezelést a függvényeiben (például egy érték beállítása után). Az összes objektumtípus delegáltnak meg kell valósítania azokat a függvényeket, amelyeket a háttérprogram igényel.

   Vessen egy pillantást a jelenlegi objektumtípus delegáltakra ugyanazon függvények megvalósításához még akkor is, ha azok csak egy sikeres értéket adnak vissza a másik objektumnál.
3. Hozzon létre vagy használjon egy meglévő adminisztrátori párbeszédablakot egy kezelőfelület megszerzéséhez, és a konfigurációs beállításainak megadásához.
4. Valósítsa meg a dinamikus mezőket az előtétprogram-modulokban, hogy képes legyen használni a dinamikus mezőket.
5. Hozzon létre egy beállítófájlt az új mező regisztrálásához a háttérprogramban (vagy a keretrendszerben lévő új adminisztrátori párbeszédablakokban, ha szükséges), valamint hogy képes legyen példányokat vagy azt létrehozni.

   És végezze el a szükséges beállításokat az új képernyőkön történő megjelenítéshez, elrejtéshez vagy a dinamikus mezők kötelezőként való megjelenítéséhez.

#### Egy új csomag létrehozása a dinamikus mezők használatához

Egy csomag létrehozásához a meglévő dinamikus mezők használata érdekében a következők szükségesek:

1. Valósítsa meg a dinamikus mezőket az előtétprogram-modulokban, hogy képes legyen használni a dinamikus mezőket.
2. Hozzon létre egy beállítófájlt, hogy lehetőséget adjon a végfelhasználónak az új képernyőkön történő megjelenítéshez, elrejtéshez vagy a dinamikus mezők kötelezőként való megjelenítéséhez.

#### A háttérprogram és az illesztőprogramok funkcionalitásainak kiterjesztése

Lehetséges lehet, hogy a háttérprogram objektum nem rendelkezik egy szükséges függvénnyel az egyéni fejlesztésekhez, vagy az is előfordulhat, hogy megvan ugyan a szükséges függvénye, de a visszatérési formátum nem felel meg az egyéni fejlesztés szükségleteinek, vagy hogy egy új viselkedés az új vagy a régi függvények végrehajtását igényli.

A legegyszerűbb mód ennek elvégzéséhez a jelenlegi mezőfájlok kiterjesztése. Ehhez egy olyan új háttérprogram kiterjesztésfájlt szükséges létrehozni, amely meghatározza az új függvényeket, és olyan illesztőprogram kiterjesztéseket is létre kell hozni, amelyek megvalósítják ezeket az új függvényeket minden egyes mezőnél. Ezeknek az új illesztőprogramoknak csak az új függvényeket kell majd megvalósítaniuk, mivel az eredeti illesztőprogramok törődnek a szabványos függvényekkel. Ezen új fájlok egyikének sincs szüksége konstruktorra, mivel ezek egy alapként lesznek betöltve a háttérprogram objektumhoz és az illesztőprogramokhoz.

Az egyetlen korlátozás, hogy a függvényeket eltérően kell elnevezni a háttérprogramnál és az illesztőprogramnál lévőknél, különben felül fognak íródni a jelenlegi objektumokkal.

Tegye az új háttérprogram kiterjesztést a `DynamicField` könyvtárba (például `/$OTRS_HOME/Kernel/System/DynamicField/NewPackageBackend.pm` és az illesztőprogramjait a `/$OTRS_HOME/Kernel/System/DynamicField/Driver/NewPackage*.pm` fájlokba).

Az új viselkedéseknek csak egy kis beállítás szükséges a kiterjesztések beállítófájljában.

Az új háttérprogram függvények létrehozásához a következők szükségesek:

1. Hozzon létre egy új háttérprogram kiterjesztés modult csak az új függvények meghatározásához.
2. Hozza létre a dinamikus mezők illesztőprogram kiterjesztéseit csak az új függvények megvalósításához.
3. Valósítsa meg az új dinamikus mezők függvényeit az előtétprogram-modulokban, hogy képes legyen használni az új dinamikus mezők függvényeit.
4. Hozzon létre egy beállítófájlt az új háttérprogram és az illesztőprogramok kiterjesztéseinek és viselkedéseinek regisztrálásához.

#### Egyéb kiterjesztések

Egyéb kiterjesztések lehetnek a fenti példák kombinációi.

### 3.3.26 Egy új dinamikus mező létrehozása

A folyamat bemutatásához egy új *Jelszó* dinamikus mező lesz létrehozva. Ez az új dinamikus mező típus egy új jelszómezőt fog megjeleníteni a jegy és a bejegyzés objektumokhoz. Mivel nagyon hasonló egy szöveg dinamikus mezőhöz, ezért a `Base` és a `BaseText` illesztőprogramokat fogjuk használni alapként ezen új mező felépítéséhez.

**Figyelem:** Az új jelszómező megvalósítása csak oktatási célokra van, nem biztosít semmilyen biztonsági szintet, és nem ajánlott termelési rendszereknél.

A dinamikus mező létrehozásához négy fájlt fogunk létrehozni:

1. Egy beállítófájlt (XML) a modulok regisztrálásához.
2. Egy adminisztrátori párbeszédablak modult (Perl) a mezőlehetőségek beállításához.
3. Egy sablonmodult az adminisztrátori párbeszédablakhoz.
4. Egy dinamikus mező illesztőprogramot (Perl).

Fájlszerkezet:

```
$HOME (e. g. /opt/otrs/)
|
...
|--/Kernel/
|   |--/Config/
|   |   |--/Files/
|   |   |   |--/XML/
|   |   |   |   |DynamicFieldPassword.xml
...
|   |--/Modules/
|   |   |AdminDynamicFieldPassword.pm
...
|   |--/Output/
|   |   |--/HTML/
|   |   |   |--/Standard/
|   |   |   |   |AdminDynamicFieldPassword.tt
...
|   |--/System/
|   |   |--/DynamicField/
|   |   |   |--/Driver/
|   |   |   |   |Password.pm
...
```

### Dinamikus mező jelszó fájlok

### Dinamikus mező beállítófájl példa

A beállítófájlokat használják a dinamikus mező típusok (illesztőprogram) és az objektumtípus illesztőprogramok regisztrálásához a háttérprogram-objektum számára. Ezek szabványos regisztrációkat is tárolnak az adminisztrátori modulokhoz a keretrendszerben.

Ebben a szakaszban a jelszó dinamikus mezőhöz egy beállítófájl van megjelenítve és elmagyarázva.

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="2.0" init="Application">
```

Ez a normál fejléc egy beállítófájlhoz.

```
<ConfigItem Name="DynamicFields::Driver###Password" Required="0" Valid="1">
    <Description Translatable="1">DynamicField backend registration.</
↪Description>
    <Group>DynamicFieldPassword</Group>
    <SubGroup>DynamicFields::Backend::Registration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="DisplayName" Translatable="1">Password</Item>
            <Item Key="Module">Kernel::System::DynamicField::Driver::Password
↪</Item>
            <Item Key="ConfigDialog">AdminDynamicFieldPassword</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

Ez a beállítás regisztrálja a jelszó dinamikus mező illesztőprogramot a háttérprogram-modulhoz, így az felvehető az elérhető dinamikus mezők típusainak listájába. A saját adminisztrátori párbeszédablakát is meghatározza a `ConfigDialog` kulcsban. Ezt a kulcsot a fő dinamikus mező adminisztrátori modul használja ennek az új dinamikus mező típusának a kezeléséhez.

```
<ConfigItem Name="Frontend::Module###AdminDynamicFieldPassword" Required="0"␣
↪Valid="1">
    <Description Translatable="1">Frontend module registration for the agent␣
↪interface.</Description>
    <Group>DynamicFieldPassword</Group>
    <SubGroup>Frontend::Admin::ModuleRegistration</SubGroup>
    <Setting>
        <FrontendModuleReg>
            <Group>admin</Group>
            <Description>Admin</Description>
            <Title Translatable="1">Dynamic Fields Text Backend GUI</Title>
            <Loader>
                <JavaScript>Core.Agent.Admin.DynamicField.js</JavaScript>
            </Loader>
        </FrontendModuleReg>
    </Setting>
</ConfigItem>
```

Ez egy szabványos modulregisztráció a jelszó adminisztrátori párbeszédablakhoz az adminisztrátori felületen.

```
</otrs_config>
```

Egy beállítófájl szabványos lezárása.

#### Dinamikus mező adminisztrátori párbeszédablak példa

Az adminisztrátori párbeszédablakok szabványos adminisztrátori modulok a dinamikus mezők kezeléséhez (hozzáadás vagy szerkesztés).

Ebben a szakaszban a jelszó dinamikus mezőhöz egy adminisztrátori párbeszédablak van megjelenítve és elmagyarázva.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Modules::AdminDynamicFieldPassword;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);
use Kernel::System::Valid;
use Kernel::System::CheckItem;
use Kernel::System::DynamicField;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {%Param};
    bless( $Self, $Type );

    for (qw(ParamObject LayoutObject LogObject ConfigObject)) {
        if ( !$Self->{$_} ) {
            $Self->{LayoutObject}->FatalError( Message => "Got no $_!" );
        }
    }

    # create additional objects
    $Self->{ValidObject} = Kernel::System::Valid->new( %{$Self} );

    $Self->{DynamicFieldObject} = Kernel::System::DynamicField->new( %{$Self}␣
↪);

    # get configured object types
    $Self->{ObjectTypeConfig} = $Self->{ConfigObject}->Get(
↪'DynamicFields::ObjectType');

    # get the fields config
    $Self->{FieldTypeConfig} = $Self->{ConfigObject}->Get(
↪'DynamicFields::Backend') || {};

    $Self->{DefaultValueMask} = '****';
    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint más osztályoknak azon objektumait kell a `new` konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub Run {
    my ( $Self, %Param ) = @_;

    if ( $Self->{Subaction} eq 'Add' ) {
        return $Self->_Add(
            %Param,
        );
    }
    elsif ( $Self->{Subaction} eq 'AddAction' ) {

        # challenge token check for write action
        $Self->{LayoutObject}->ChallengeTokenCheck();

        return $Self->_AddAction(
            %Param,
        );
    }
    if ( $Self->{Subaction} eq 'Change' ) {

        return $Self->_Change(
            %Param,
        );
    }
    elsif ( $Self->{Subaction} eq 'ChangeAction' ) {

        # challenge token check for write action
        $Self->{LayoutObject}->ChallengeTokenCheck();

        return $Self->_ChangeAction(
            %Param,
        );
    }

    return $Self->{LayoutObject}->ErrorScreen(
        Message => "Undefined subaction.",
    );
}
```

A `Run` a webkérés által meghívott alapértelmezett függvény. Megpróbáljuk ezt a függvényt annyira egyszerűvé tenni, amennyire csak lehetséges, és lehetővé tenni a segítő függvényeknek a kemény munka elvégzését.

```
sub _Add {
    my ( $Self, %Param ) = @_;

    my %GetParam;
    for my $Needed (qw(ObjectType FieldType FieldOrder)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed␣
↪);
        if ( !$Needed ) {

            return $Self->{LayoutObject}->ErrorScreen(
                Message => "Need $Needed",
```

(continues on next page)

(folytatás az előző oldalról)

```
            );
        }
    }

    # get the object type and field type display name
    my $ObjectTypeName = $Self->{ObjectTypeConfig}->{ $GetParam{ObjectType} }->
↪{DisplayName} || '';
    my $FieldTypeName  = $Self->{FieldTypeConfig}->{ $GetParam{FieldType} }->
↪{DisplayName}   || '';

    return $Self->_ShowScreen(
        %Param,
        %GetParam,
        Mode           => 'Add',
        ObjectTypeName => $ObjectTypeName,
        FieldTypeName  => $FieldTypeName,
    );
}
```

Az `_Add` függvény is nagyon egyszerű, csak lekér néhány paramétert a webkérésből, és meghívja a `_ShowScreen()` függvényt. Normális esetben ezt a függvényt nem szükséges módosítani.

```
sub _AddAction {
    my ( $Self, %Param ) = @_;

    my %Errors;
    my %GetParam;

    for my $Needed (qw(Name Label FieldOrder)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed␣
↪);
        if ( !$GetParam{$Needed} ) {
            $Errors{ $Needed . 'ServerError' }        = 'ServerError';
            $Errors{ $Needed . 'ServerErrorMessage' } = 'This field is␣
↪required.';
        }
    }

    if ( $GetParam{Name} ) {

        # check if name is alphanumeric
        if ( $GetParam{Name} !~ m{\A ( ?: [a-zA-Z] | \d )+ \z}xms ) {

            # add server error error class
            $Errors{NameServerError} = 'ServerError';
            $Errors{NameServerErrorMessage} =
                'The field does not contain only ASCII letters and numbers.';
        }

        # check if name is duplicated
        my %DynamicFieldsList = %{
            $Self->{DynamicFieldObject}->DynamicFieldList(
```

(continues on next page)

(folytatás az előző oldalról)

```
                Valid      => 0,
                ResultType => 'HASH',
                )
        };

        %DynamicFieldsList = reverse %DynamicFieldsList;

        if ( $DynamicFieldsList{ $GetParam{Name} } ) {

            # add server error error class
            $Errors{NameServerError}        = 'ServerError';
            $Errors{NameServerErrorMessage} = 'There is another field with the␣
↪same name.';
        }
    }

    if ( $GetParam{FieldOrder} ) {

        # check if field order is numeric and positive
        if ( $GetParam{FieldOrder} !~ m{\A ( ?: \d )+ \z}xms ) {

            # add server error error class
            $Errors{FieldOrderServerError}        = 'ServerError';
            $Errors{FieldOrderServerErrorMessage} = 'The field must be numeric.
↪';
        }
    }

    for my $ConfigParam (
        qw(
        ObjectType ObjectTypeName FieldType FieldTypeName DefaultValue␣
↪ValidID ShowValue
        ValueMask
        )
        )
    {
        $GetParam{$ConfigParam} = $Self->{ParamObject}->GetParam( Param =>
↪$ConfigParam );
    }

    # uncorrectable errors
    if ( !$GetParam{ValidID} ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Need ValidID",
        );
    }

    # return to add screen if errors
    if (%Errors) {

        return $Self->_ShowScreen(
```

(continues on next page)

(folytatás az előző oldalról)

```
            %Param,
            %Errors,
            %GetParam,
            Mode => 'Add',
        );
    }

    # set specific config
    my $FieldConfig = {
        DefaultValue => $GetParam{DefaultValue},
        ShowValue    => $GetParam{ShowValue},
        ValueMask    => $GetParam{ValueMask} || $Self->{DefaultValueMask},
    };

    # create a new field
    my $FieldID = $Self->{DynamicFieldObject}->DynamicFieldAdd(
        Name       => $GetParam{Name},
        Label      => $GetParam{Label},
        FieldOrder => $GetParam{FieldOrder},
        FieldType  => $GetParam{FieldType},
        ObjectType => $GetParam{ObjectType},
        Config     => $FieldConfig,
        ValidID    => $GetParam{ValidID},
        UserID     => $Self->{UserID},
    );

    if ( !$FieldID ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Could not create the new field",
        );
    }

    return $Self->{LayoutObject}->Redirect(
        OP => "Action=AdminDynamicField",
    );
}
```

Az `_AddAction` függvény kéri le a beállítási paramétereket egy új dinamikus mezőből, és ellenőrzi, hogy a dinamikus mező neve csak betűket és számokat tartalmaz-e. Ez a függvény képes ellenőrizni bármilyen egyéb paramétert is.

A `Name`, `Label`, `FieldOrder`, `Validity` közös paraméterek az összes dinamikus mezőnél, és ezek kötelezők. Minden egyes dinamikus mezőnek megvan a saját különleges beállítása, amelynek tartalmaznia kell legalább a `DefaultValue` paramétert. Ebben az esetben a `ShowValue` és a `ValueMask` paraméterekkel is rendelkezik a jelszómezőnél.

Ha a mező rendelkezik egy rögzített listájú értékek tárolásának képességével, akkor azokat a `PossibleValues` paraméterben kell tárolni a különleges beállítási kivonaton belül.

Mint más adminisztrátori modulokban, ha egy paraméter nem érvényes, akkor ez a függvény visszatér a hozzáadás képernyőre, kiemelve a hibás űrlapmezőket.

Ha az összes paraméter helyes, akkor létrehoz egy új dinamikus mezőt.

```
sub _Change {
    my ( $Self, %Param ) = @_;

    my %GetParam;
    for my $Needed (qw(ObjectType FieldType)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed␣
↪);
        if ( !$Needed ) {

            return $Self->{LayoutObject}->ErrorScreen(
                Message => "Need $Needed",
            );
        }
    }

    # get the object type and field type display name
    my $ObjectTypeName = $Self->{ObjectTypeConfig}->{ $GetParam{ObjectType} }->
↪{DisplayName} || '';
    my $FieldTypeName  = $Self->{FieldTypeConfig}->{ $GetParam{FieldType} }->
↪{DisplayName}   || '';

    my $FieldID = $Self->{ParamObject}->GetParam( Param => 'ID' );

    if ( !$FieldID ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Need ID",
        );
    }

    # get dynamic field data
    my $DynamicFieldData = $Self->{DynamicFieldObject}->DynamicFieldGet(
        ID => $FieldID,
    );

    # check for valid dynamic field configuration
    if ( !IsHashRefWithData($DynamicFieldData) ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Could not get data for dynamic field $FieldID",
        );
    }

    my %Config = ();

    # extract configuration
    if ( IsHashRefWithData( $DynamicFieldData->{Config} ) ) {
        %Config = %{ $DynamicFieldData->{Config} };
    }

    return $Self->_ShowScreen(
        %Param,
```

(continues on next page)

(folytatás az előző oldalról)

```
        %GetParam,
        %${DynamicFieldData},
        %Config,
        ID             => $FieldID,
        Mode           => 'Change',
        ObjectTypeName => $ObjectTypeName,
        FieldTypeName  => $FieldTypeName,
    );
}
```

A `_Change` függvény nagyon hasonló az `_Add` függvényhez, de mivel ezt a függvényt egy meglévő mező szerkesztéséhez használják, ellenőriznie kell a `FieldID` paramétert, és be kell gyűjtenie a jelenlegi dinamikus mező adatait.

```
sub _ChangeAction {
    my ( $Self, %Param ) = @_;

    my %Errors;
    my %GetParam;

    for my $Needed (qw(Name Label FieldOrder)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed␣
↪);
        if ( !$GetParam{$Needed} ) {
            $Errors{ $Needed . 'ServerError' }        = 'ServerError';
            $Errors{ $Needed . 'ServerErrorMessage' } = 'This field is␣
↪required.';
        }
    }

    my $FieldID = $Self->{ParamObject}->GetParam( Param => 'ID' );
    if ( !$FieldID ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Need ID",
        );
    }

    if ( $GetParam{Name} ) {

        # check if name is lowercase
        if ( $GetParam{Name} !~ m{\A ( ?: [a-zA-Z] | \d )+ \z}xms ) {

            # add server error error class
            $Errors{NameServerError} = 'ServerError';
            $Errors{NameServerErrorMessage} =
                'The field does not contain only ASCII letters and numbers.';
        }

        # check if name is duplicated
        my %DynamicFieldsList = %{
            $Self->{DynamicFieldObject}->DynamicFieldList(
```

(continues on next page)

(folytatás az előző oldalról)

```
                Valid      => 0,
                ResultType => 'HASH',
                )
        };

        %DynamicFieldsList = reverse %DynamicFieldsList;

        if (
            $DynamicFieldsList{ $GetParam{Name} } &&
            $DynamicFieldsList{ $GetParam{Name} } ne $FieldID
            )
        {

            # add server error class
            $Errors{NameServerError}        = 'ServerError';
            $Errors{NameServerErrorMessage} = 'There is another field with the␣
↪same name.';
        }
    }

    if ( $GetParam{FieldOrder} ) {

        # check if field order is numeric and positive
        if ( $GetParam{FieldOrder} !~ m{\A ( ?: \d )+ \z}xms ) {

            # add server error error class
            $Errors{FieldOrderServerError}        = 'ServerError';
            $Errors{FieldOrderServerErrorMessage} = 'The field must be numeric.
↪';
        }
    }

    for my $ConfigParam (
        qw(
        ObjectType ObjectTypeName FieldType FieldTypeName DefaultValue␣
↪ValidID ShowValue
        ValueMask
        )
        )
    {
        $GetParam{$ConfigParam} = $Self->{ParamObject}->GetParam( Param =>
↪$ConfigParam );
    }

    # uncorrectable errors
    if ( !$GetParam{ValidID} ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Need ValidID",
        );
    }
```

(continues on next page)

(folytatás az előző oldalról)

```
    # get dynamic field data
    my $DynamicFieldData = $Self->{DynamicFieldObject}->DynamicFieldGet(
        ID => $FieldID,
    );

    # check for valid dynamic field configuration
    if ( !IsHashRefWithData($DynamicFieldData) ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Could not get data for dynamic field $FieldID",
        );
    }

    # return to change screen if errors
    if (%Errors) {

        return $Self->_ShowScreen(
            %Param,
            %Errors,
            %GetParam,
            ID   => $FieldID,
            Mode => 'Change',
        );
    }

    # set specific config
    my $FieldConfig = {
        DefaultValue => $GetParam{DefaultValue},
        ShowValue    => $GetParam{ShowValue},
        ValueMask    => $GetParam{ValueMask},
    };

    # update dynamic field (FieldType and ObjectType cannot be changed; use␣
↪old values)
    my $UpdateSuccess = $Self->{DynamicFieldObject}->DynamicFieldUpdate(
        ID         => $FieldID,
        Name       => $GetParam{Name},
        Label      => $GetParam{Label},
        FieldOrder => $GetParam{FieldOrder},
        FieldType  => $DynamicFieldData->{FieldType},
        ObjectType => $DynamicFieldData->{ObjectType},
        Config     => $FieldConfig,
        ValidID    => $GetParam{ValidID},
        UserID     => $Self->{UserID},
    );

    if ( !$UpdateSuccess ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Could not update the field $GetParam{Name}",
        );
    }
```

(continues on next page)

(folytatás az előző oldalról)

```
    return $Self->{LayoutObject}->Redirect(
        OP => "Action=AdminDynamicField",
    );
}
```

A `_ChangeAction()` nagyon hasonló az `_AddAction()` függvényhez, de át van írva egy meglévő mező frissítéséhez egy új létrehozása helyett.

```
sub _ShowScreen {
    my ( $Self, %Param ) = @_;

    $Param{DisplayFieldName} = 'New';

    if ( $Param{Mode} eq 'Change' ) {
        $Param{ShowWarning}      = 'ShowWarning';
        $Param{DisplayFieldName} = $Param{Name};
    }

    # header
    my $Output = $Self->{LayoutObject}->Header();
    $Output .= $Self->{LayoutObject}->NavigationBar();

    # get all fields
    my $DynamicFieldList = $Self->{DynamicFieldObject}->DynamicFieldListGet(
        Valid => 0,
    );

    # get the list of order numbers (is already sorted).
    my @DynamicfieldOrderList;
    for my $Dynamicfield ( @{$DynamicFieldList} ) {
        push @DynamicfieldOrderList, $Dynamicfield->{FieldOrder};
    }

    # when adding we need to create an extra order number for the new field
    if ( $Param{Mode} eq 'Add' ) {

        # get the last element from the order list and add 1
        my $LastOrderNumber = $DynamicfieldOrderList[-1];
        $LastOrderNumber++;

        # add this new order number to the end of the list
        push @DynamicfieldOrderList, $LastOrderNumber;
    }

    my $DynamicFieldOrderSrtg = $Self->{LayoutObject}->BuildSelection(
        Data          => \@DynamicfieldOrderList,
        Name          => 'FieldOrder',
        SelectedValue => $Param{FieldOrder} || 1,
        PossibleNone  => 0,
        Class         => 'W50pc Validate_Number',
    );
```

(continues on next page)

(folytatás az előző oldalról)

```
    my %ValidList = $Self->{ValidObject}->ValidList();

    # create the Validity select
    my $ValidityStrg = $Self->{LayoutObject}->BuildSelection(
        Data         => \%ValidList,
        Name         => 'ValidID',
        SelectedID   => $Param{ValidID} || 1,
        PossibleNone => 0,
        Translation  => 1,
        Class        => 'W50pc',
    );

    # define config field specific settings
    my $DefaultValue = ( defined $Param{DefaultValue} ? $Param{DefaultValue} :
↪'' );

    # create the Show value select
    my $ShowValueStrg = $Self->{LayoutObject}->BuildSelection(
        Data => [ 'No', 'Yes' ],
        Name => 'ShowValue',
        SelectedValue => $Param{ShowValue} || 'No',
        PossibleNone  => 0,
        Translation   => 1,
        Class         => 'W50pc',
    );

    # generate output
    $Output .= $Self->{LayoutObject}->Output(
        TemplateFile => 'AdminDynamicFieldPassword',
        Data         => {
            %Param,
            ValidityStrg          => $ValidityStrg,
            DynamicFieldOrderSrtg => $DynamicFieldOrderSrtg,
            DefaultValue          => $DefaultValue,
            ShowValueStrg         => $ShowValueStrg,
            ValueMask             => $Param{ValueMask} || $Self->
↪{DefaultValueMask},
        },
    );

    $Output .= $Self->{LayoutObject}->Footer();

    return $Output;
}

1;
```

A `_ShowScreen` függvényt használják egy sablonból történő HTML elemek és blokkok beállítására és meghatározására az adminisztrátori párbeszédablak HTML kódjának előállításához.

### Dinamikus mező sablon az adminisztrátori párbeszédablak példához

A sablon az a hely, ahol a párbeszédablak HTML kódja el van tárolva.

Ebben a szakaszban a jelszó dinamikus mezőhöz egy adminisztrátori párbeszédablak sablon van megjelenítve és elmagyarázva.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban.

```
<div class="MainBox ARIARoleMain LayoutFixedSidebar SidebarFirst">
    <h1>[% Translate("Dynamic Fields") | html %] - [% Translate(Data.
↪ObjectTypeName) | html %]: [% Translate(Data.Mode) | html %] [%␣
↪Translate(Data.FieldTypeName) | html %] [% Translate("Field") | html %]</h1>

    <div class="Clear"></div>

    <div class="SidebarColumn">
        <div class="WidgetSimple">
            <div class="Header">
                <h2>[% Translate("Actions") | html %]</h2>
            </div>
            <div class="Content">
                <ul class="ActionList">
                    <li>
                        <a href="[% Env("Baselink") %]Action=AdminDynamicField
↪" class="CallForAction"><span>[% Translate("Go back to overview") | html %]
↪</span></a>
                    </li>
                </ul>
            </div>
        </div>
    </div>
```

A kód ezen része rendelkezik a fő dobozzal és a műveletek oldalsávval is. Nincs szükség módosításokra ebben a szakaszban.

```
<div class="ContentColumn">
    <form action="[% Env("CGIHandle") %]" method="post" class="Validate␣
↪PreventMultipleSubmits">
        <input type="hidden" name="Action" value="AdminDynamicFieldPassword" /
↪>
        <input type="hidden" name="Subaction" value="[% Data.Mode | html
↪%]Action" />
        <input type="hidden" name="ObjectType" value="[% Data.ObjectType |␣
↪html %]" />
        <input type="hidden" name="FieldType" value="[% Data.FieldType | html
↪%]" />
```

(continues on next page)

(folytatás az előző oldalról)

```
        <input type="hidden" name="ID" value="[% Data.ID | html %]" />
```

A kód ezen szakaszában van meghatározva a párbeszédablak jobboldali része. Figyelje meg, hogy a rejtett `Action` beviteli mező értékének egyeznie kell az adminisztrátori párbeszédablak nevével.

```
<div class="WidgetSimple">
    <div class="Header">
        <h2>[% Translate("General") | html %]</h2>
    </div>
    <div class="Content">
        <div class="LayoutGrid ColumnsWithSpacing">
            <div class="Size1of2">
                <fieldset class="TableLike">
                    <label class="Mandatory" for="Name"><span class="Marker">*
↪</span> [% Translate("Name") | html %]:</label>
                    <div class="Field">
                        <input id="Name" class="W50pc [% Data.NameServerError␣
↪| html %] [% Data.ShowWarning | html %]  Validate_Alphanumeric" type="text"␣
↪maxlength="200" value="[% Data.Name | html %]" name="Name"/>
                        <div id="NameError" class="TooltipErrorMessage"><p>[%␣
↪Translate("This field is required, and the value should be alphabetic and␣
↪numeric characters only.") | html %]</p></div>
                        <div id="NameServerError" class="TooltipErrorMessage">
↪<p>[% Translate(Data.NameServerErrorMessage) | html %]</p></div>
                        <p class="FieldExplanation">[% Translate("Must be␣
↪unique and only accept alphabetic and numeric characters.") | html %]</p>
                        <p class="Warning Hidden">[% Translate("Changing this␣
↪value will require manual changes in the system.") | html %]</p>
                    </div>
                    <div class="Clear"></div>

                    <label class="Mandatory" for="Label"><span class="Marker">
↪*</span> [% Translate("Label") | html %]:</label>
                    <div class="Field">
                        <input id="Label" class="W50pc [% Data.
↪LabelServerError | html %] Validate_Required" type="text" maxlength="200"␣
↪value="[% Data.Label | html %]" name="Label"/>
                        <div id="LabelError" class="TooltipErrorMessage"><p>[
↪% Translate("This field is required.") | html %]</p></div>
                        <div id="LabelServerError" class="TooltipErrorMessage
↪"><p>[% Translate(Data.LabelServerErrorMessage) | html %]</p></div>
                        <p class="FieldExplanation">[% Translate("This is the␣
↪name to be shown on the screens where the field is active.") | html %]</p>
                    </div>
                    <div class="Clear"></div>

                    <label class="Mandatory" for="FieldOrder"><span class=
↪"Marker">*</span> [% Translate("Field order") | html %]:</label>
                    <div class="Field">
                        [% Data.DynamicFieldOrderSrtg %]
                        <div id="FieldOrderError" class="TooltipErrorMessage">
↪<p>[% Translate("This field is required and must be numeric.") | html %]</p>
↪</div>
```

(continues on next page)

(folytatás az előző oldalról)

```
                        <div id="FieldOrderServerError" class=
↪"TooltipErrorMessage"><p>[% Translate(Data.FieldOrderServerErrorMessage) |␣
↪html %]</p></div>
                        <p class="FieldExplanation">[% Translate("This is the␣
↪order in which this field will be shown on the screens where is active.") |␣
↪html %]</p>
                    </div>
                    <div class="Clear"></div>
                </fieldset>
            </div>
            <div class="Size1of2">
                <fieldset class="TableLike">
                    <label for="ValidID">[% Translate("Validity") | html %]:</
↪label>
                    <div class="Field">
                        [% Data.ValidityStrg %]
                    </div>
                    <div class="Clear"></div>

                    <div class="SpacingTop"></div>
                    <label for="FieldTypeName">[% Translate("Field type") |␣
↪html %]:</label>
                    <div class="Field">
                        <input id="FieldTypeName" readonly="readonly" class=
↪"W50pc" type="text" maxlength="200" value="[% Data.FieldTypeName | html %]"␣
↪name="FieldTypeName"/>
                        <div class="Clear"></div>
                    </div>

                    <div class="SpacingTop"></div>
                    <label for="ObjectTypeName">[% Translate("Object type") |␣
↪html %]:</label>
                    <div class="Field">
                        <input id="ObjectTypeName" readonly="readonly" class=
↪"W50pc" type="text" maxlength="200" value="[% Data.ObjectTypeName | html %]
↪" name="ObjectTypeName"/>
                        <div class="Clear"></div>
                    </div>
                </fieldset>
            </div>
        </div>
    </div>
</div>
```

Ez az első felületi elem tartalmazza a szokásos űrlapattribútumokat a dinamikus mezőkhöz. A többi dinamikus mezővel való következetességért javasolt a kód ezen részének változatlanul hagyása.

```
<div class="WidgetSimple">
    <div class="Header">
        <h2>[% Translate(Data.FieldTypeName) | html %] [% Translate("Field␣
↪Settings") | html %]</h2>
    </div>
```

(continues on next page)

(folytatás az előző oldalról)

```
    <div class="Content">
        <fieldset class="TableLike">

            <label for="DefaultValue">[% Translate("Default value") | html %]:
↪</label>
            <div class="Field">
                <input id="DefaultValue" class="W50pc" type="text" maxlength=
↪"200" value="[% Data.DefaultValue | html %]" name="DefaultValue"/>
                <p class="FieldExplanation">[% Translate("This is the default␣
↪value for this field.") | html %]</p>
            </div>
            <div class="Clear"></div>

            <label for="ShowValue">[% Translate("Show value") | html %]:</
↪label>
            <div class="Field">
                [% Data.ShowValueStrg %]
                <p class="FieldExplanation">
                    [% Translate("To reveal the field value in non edit␣
↪screens ( e.g. Ticket Zoom Screen )") | html %]
                </p>
            </div>
            <div class="Clear"></div>

            <label for="ValueMask">[% Translate("Hidden value mask") | html
↪%]:</label>
            <div class="Field">
                <input id="ValueMask" class="W50pc" type="text" maxlength="200
↪" value="[% Data.ValueMask | html %]" name="ValueMask"/>
                <p class="FieldExplanation">
                    [% Translate("This is the alternate value to show if Show␣
↪value is set to \"No\" ( Default: **** ).") | html %]
                </p>
            </div>
            <div class="Clear"></div>

        </fieldset>
    </div>
</div>
```

A második felületi elem a dinamikus mezőre jellemző űrlapattribútumokkal rendelkezik. Ez az a hely, ahol az új attribútumok beállíthatók, és használhatnak JavaScript és AJAX technológiákat, hogy egyszerűbbé és barátságosabbá tegyék a végfelhasználó számára.

```
            <fieldset class="TableLike">
                <div class="Field SpacingTop">
                    <button type="submit" class="Primary" value="[% Translate(
↪"Save") | html %]">[% Translate("Save") | html %]</button>
                    [% Translate("or") | html %]
                    <a href="[% Env("Baselink") %]Action=AdminDynamicField">[
↪% Translate("Cancel") | html %]</a>
                </div>
```

(continues on next page)

(folytatás az előző oldalról)

```
                <div class="Clear"></div>
            </fieldset>
        </form>
    </div>
</div>
[% WRAPPER JSOnDocumentComplete %]
<script type="text/javascript">//<![CDATA[
$('.ShowWarning').bind('change keyup', function (Event) {
    $('p.Warning').removeClass('Hidden');
});

Core.Agent.Admin.DynamicField.ValidationInit();
//]]></script>
[% END %]
```

A fájl utolsó része tartalmazza a *Mentés* gombot és a *Mégse* hivatkozást, valamint az egyéb szükséges JavaScript kódot.

### Dinamikus mező illesztőprogram példa

Az illesztőprogram *maga a* dinamikus mező. Számos olyan függvényt tartalmaz, amelyet széles körben használnak az OTRS keretrendszerben. Egy illesztőprogram örökölhet néhány függvényt az alaposztályokból, például a `TextArea` illesztőprogram a `Base.pm` és a `BaseText.pm` fájlokból örökli a függvények legnagyobb részét, és csak azokat a függvényeket valósítja meg, amelyek eltérő logikát vagy eredményeket igényelnek. A jelölőnégyzet mező illesztőprogram csak a `Base.pm` fájlból örököl, mivel az összes többi függvény nagyon eltérő bármely más alap illesztőprogramtól.

**Lásd még:**

Nézze meg a `/Kernel/System/DynmicField/Backend.pm` modul Perl On-line Dokumentációját (POD) az összes attribútum listája és a lehetséges visszatérési adatok megismeréséhez az egyes függvényeknél.

Ebben a szakaszban a jelszó dinamikus mező illesztőprogram van bemutatva és elmagyarázva. Ez az illesztőprogram örököl néhány függvényt a `Base.pm` és a `BaseText.pm` fájlokból, és csak azokat a függvényeket valósítja meg, amelyek eltérő eredményeket igényelnek.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::DynamicField::Driver::Password;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);
use Kernel::System::DynamicFieldValue;
```

(continues on next page)

(folytatás az előző oldalról)

```
use base qw(Kernel::System::DynamicField::Driver::BaseText);

our @ObjectDependencies = (
    'Kernel::Config',
    'Kernel::System::DynamicFieldValue',
    'Kernel::System::Main',
);
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva. Figyelje meg, hogy a `BaseText` osztályt használják alaposztályként.

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # set field behaviors
    $Self->{Behaviors} = {
        'IsACLReducible'               => 0,
        'IsNotificationEventCondition' => 1,
        'IsSortable'                   => 0,
        'IsFiltrable'                  => 0,
        'IsStatsCondition'             => 1,
        'IsCustomerInterfaceCapable'   => 1,
    };

    # get the Dynamic Field Backend custom extensions
    my $DynamicFieldDriverExtensions
        = $Kernel::OM->Get('Kernel::Config')->Get(
↪'DynamicFields::Extension::Driver::Password');

    EXTENSION:
    for my $ExtensionKey ( sort keys %{$DynamicFieldDriverExtensions} ) {

        # skip invalid extensions
        next EXTENSION if !IsHashRefWithData( $DynamicFieldDriverExtensions->{
↪$ExtensionKey} );

        # create a extension config shortcut
        my $Extension = $DynamicFieldDriverExtensions->{$ExtensionKey};

        # check if extension has a new module
        if ( $Extension->{Module} ) {

            # check if module can be loaded
            if (
                !$Kernel::OM->Get('Kernel::System::Main')->RequireBaseClass(
↪$Extension->{Module} )
                )
```

(continues on next page)

(folytatás az előző oldalról)

```
            {
                die "Can't load dynamic fields backend module"
                    . " $Extension->{Module}! $@";
            }
        }

        # check if extension contains more behaviors
        if ( IsHashRefWithData( $Extension->{Behaviors} ) ) {

            %{ $Self->{Behaviors} } = (
                %{ $Self->{Behaviors} },
                %{ $Extension->{Behaviors} }
            );
        }
    }

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint más osztályoknak azon objektumait kell a `new` konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

Fontos a viselkedéseket helyesen meghatározni, mivel a mező lehet használható vagy lehet nem használható bizonyos képernyőkön. Azokat a függvényeket esetleg nem szükséges megvalósítani, amelyek olyan viselkedésektől függenek, amelyek nem aktívak ennél a bizonyos mezőnél.

---

**Megjegyzés:** Az illesztőprogramokat kizárólag a `BackendObject` hozza létre, és nem közvetlenül bármely egyéb modulból.

---

```
sub EditFieldRender {
    my ( $Self, %Param ) = @_;

    # take config from field config
    my $FieldConfig = $Param{DynamicFieldConfig}->{Config};
    my $FieldName   = 'DynamicField_' . $Param{DynamicFieldConfig}->{Name};
    my $FieldLabel  = $Param{DynamicFieldConfig}->{Label};

    my $Value = '';

    # set the field value or default
    if ( $Param{UseDefaultValue} ) {
        $Value = ( defined $FieldConfig->{DefaultValue} ? $FieldConfig->
↪{DefaultValue} : '' );
    }
    $Value = $Param{Value} if defined $Param{Value};

    # extract the dynamic field value from the web request
    my $FieldValue = $Self->EditFieldValueGet(
        %Param,
    );
```

(continues on next page)

(folytatás az előző oldalról)

```
    # set values from ParamObject if present
    if ( defined $FieldValue ) {
        $Value = $FieldValue;
    }

    # check and set class if necessary
    my $FieldClass = 'DynamicFieldText W50pc';
    if ( defined $Param{Class} && $Param{Class} ne '' ) {
        $FieldClass .= ' ' . $Param{Class};
    }

    # set field as mandatory
    $FieldClass .= ' Validate_Required' if $Param{Mandatory};

    # set error css class
    $FieldClass .= ' ServerError' if $Param{ServerError};

    my $HTMLString = <<"EOF";
<input type="password" class="$FieldClass" id="$FieldName" name="$FieldName"␣
↪title="$FieldLabel" value="$Value" />
EOF

    if ( $Param{Mandatory} ) {
        my $DivID = $FieldName . 'Error';

        # for client side validation
        $HTMLString .= <<"EOF";
    <div id="$DivID" class="TooltipErrorMessage">
        <p>
            \$Text{"This field is required."}
        </p>
    </div>
EOF
    }

    if ( $Param{ServerError} ) {

        my $ErrorMessage = $Param{ErrorMessage} || 'This field is required.';
        my $DivID = $FieldName . 'ServerError';

        # for server side validation
        $HTMLString .= <<"EOF";
    <div id="$DivID" class="TooltipErrorMessage">
        <p>
            \$Text{"$ErrorMessage"}
        </p>
    </div>
EOF
    }

    # call EditLabelRender on the common Driver
    my $LabelString = $Self->EditLabelRender(
```

(continues on next page)

(folytatás az előző oldalról)

```
        %Param,
        DynamicFieldConfig => $Param{DynamicFieldConfig},
        Mandatory          => $Param{Mandatory} || '0',
        FieldName          => $FieldName,
    );

    my $Data = {
        Field => $HTMLString,
        Label => $LabelString,
    };

    return $Data;
}
```

Ez a függvény felelős a mező és annak címkéje HTML ábrázolásának létrehozásáért, és olyan szerkesztő képernyőkön használják, mint például az `AgentTicketPhone`, `AgentTicketNote`, stb.

```
sub DisplayValueRender {
    my ( $Self, %Param ) = @_;

    # set HTMLOutput as default if not specified
    if ( !defined $Param{HTMLOutput} ) {
        $Param{HTMLOutput} = 1;
    }

    my $Value;
    my $Title;

    # check if field is set to show password or not
    if (
        defined $Param{DynamicFieldConfig}->{Config}->{ShowValue}
        && $Param{DynamicFieldConfig}->{Config}->{ShowValue} eq 'Yes'
        )
    {

        # get raw Title and Value strings from field value
        $Value = defined $Param{Value} ? $Param{Value} : '';
        $Title = $Value;
    }
    else {

        # show the mask and not the value
        $Value = $Param{DynamicFieldConfig}->{Config}->{ValueMask} || '';
        $Title = 'The value of this field is hidden.'
    }

    # HTMLOutput transformations
    if ( $Param{HTMLOutput} ) {
        $Value = $Param{LayoutObject}->Ascii2Html(
            Text => $Value,
            Max => $Param{ValueMaxChars} || '',
        );
```

(continues on next page)

(folytatás az előző oldalról)

```
        $Title = $Param{LayoutObject}->Ascii2Html(
            Text => $Title,
            Max => $Param{TitleMaxChars} || '',
        );
    }
    else {
        if ( $Param{ValueMaxChars} && length($Value) > $Param{ValueMaxChars}␣
↪) {
            $Value = substr( $Value, 0, $Param{ValueMaxChars} ) . '...';
        }
        if ( $Param{TitleMaxChars} && length($Title) > $Param{TitleMaxChars}␣
↪) {
            $Title = substr( $Title, 0, $Param{TitleMaxChars} ) . '...';
        }
    }

    # create return structure
    my $Data = {
        Value => $Value,
        Title => $Title,
    };

    return $Data;
}
```

A `DisplayValueRender()` függvény egyszerű szövegként adja vissza a mező értékét és annak feliratát (mindkettő lefordítható). Ennél a bizonyos példánál azt ellenőrizzük, hogy a jelszót meg kell-e mutatni, vagy egy beállítási paraméter által előre meghatározott maszkot kell-e megjeleníteni a dinamikus mezőben.

```
sub ReadableValueRender {
    my ( $Self, %Param ) = @_;

    my $Value;
    my $Title;

    # check if field is set to show password or not
    if (
        defined $Param{DynamicFieldConfig}->{Config}->{ShowValue}
        && $Param{DynamicFieldConfig}->{Config}->{ShowValue} eq 'Yes'
        )
    {

        # get raw Title and Value strings from field value
        $Value = $Param{Value} // '';
        $Title = $Value;
    }
    else {

        # show the mask and not the value
        $Value = $Param{DynamicFieldConfig}->{Config}->{ValueMask} || '';
        $Title = 'The value of this field is hidden.'
```

(continues on next page)

(folytatás az előző oldalról)

```
    }

    # cut strings if needed
    if ( $Param{ValueMaxChars} && length($Value) > $Param{ValueMaxChars} ) {
        $Value = substr( $Value, 0, $Param{ValueMaxChars} ) . '...';
    }
    if ( $Param{TitleMaxChars} && length($Title) > $Param{TitleMaxChars} ) {
        $Title = substr( $Title, 0, $Param{TitleMaxChars} ) . '...';
    }

    # create return structure
    my $Data = {
        Value => $Value,
        Title => $Title,
    };

    return $Data;
}
```

Ez a függvény hasonló a `DisplayValueRender()` függvényhez, de olyan helyeken használják, ahol nincs `LayoutObject`.

### Egyéb függvények

A következők olyan egyéb függvények, amelyek talán szükségesek lehetnek, ha egy új dinamikus mező nem örököl más osztályokból. Ezen függvények teljes kódjának megtekintéséhez nézzen bele közvetlenül a `Kernel/System/DynamicField/Driver/Base.pm` és a `Kernel/System/DynamicField/Driver/BaseText.pm` fájlokba.

```
sub ValueGet { ... }
```

Ez a függvény lekéri az értéket a mezőből egy adott objektumnál. Ebben az esetben az első szövegértéket adjuk vissza, mivel a mező egyszerre csak egy szövegértéket tárol.

```
sub ValueSet { ... }
```

Ez a függvényt egy dinamikus mező érték tárolásához használják. Ebben az esetben a mező csak egy szöveg típusú értéket tárol. Más mezők tárolhatnak egynél több értéket is a `ValueText`, a `ValueDateTime` vagy a `ValueInt` formátumnál.

```
sub ValueDelete { ... }
```

Ezt a függvényt egy mező értékének törléséhez használják, amely egy bizonyos objektumazonosítóhoz van csatolva. Például ha egy objektum példánynak törlésére készülnek, akkor nincs oka a mezőérték tárolásának az adatbázisban annál a bizonyos objektumpéldánynál.

```
sub AllValuesDelete { ... }
```

Ezt a függvényt az összes érték törléséhez használják egy bizonyos dinamikus mezőből. Ez a függvény nagyon hasznos, amikor egy dinamikus mező törlésre fog kerülni.

```
sub ValueValidate { ... }
```

Ezt a függvényt annak ellenőrzéséhez használják, hogy az érték megegyezik-e a típusának.

```
sub SearchSQLGet { ... }
```

Ezt a függvényt a `TicketSearch` alapmodul használja egy jegy kereséséhez szükséges belső lekérdezés felépítéséhez, ezt a mezőt alapul véve keresési paraméterként.

```
sub SearchSQLOrderFieldGet { ... }
```

Ez a függvény szintén egy segítő a `TicketSearch` modulhoz. A `$Param{TableAlias}` paramétert meg kell tartani, és a `value_text` lecserélhető a `value_date` vagy a `value_int` paraméterrel a mezőtől függően.

```
sub EditFieldValueGet { ... }
```

Ezt a függvényt az OTRS szerkesztőképernyőin használják, és a célja a mező értékének lekérése vagy egy sablonból (mint például az általános ügyintéző profilból), vagy egy webkérésből. Ez a függvény megkapja a webkérést a `$Param{ParamObject}` paraméterben, amely az előtétprogram-modul vagy a képernyő `ParamObject` objektumának egy másolata.

Két visszatérési formátum létezik ennél a függvénynél. A normál, amely egyszerűen a nyers érték, vagy egy szerkezet, amely a mezőnév => mezőérték páros. Például egy dátum dinamikus mező normális esetben a dátumot szövegként adja vissza, és ha egy szerkezetet kell visszaadnia, akkor a kivonatban egy párost ad vissza a dátum minden egyes részéhez.

Ha az eredménynek egy szerkezetnek kell lennie, akkor normális esetben ezt arra használják, hogy az értéket egy sablonban tárolja, mint például egy általános ügyintéző profilban. Például egy dátummező számos HTML összetevőt használ a mező felépítéséhez, mint például a használt jelölőnégyzetet és kiválasztókat az évhez, hónaphoz, naphoz, stb.

```
sub EditFieldValueValidate { ... }
```

Ennek a függvénynek biztosítania kell legalább egy metódust az ellenőrzéshez, ha a mező üres, és hibát kell visszaadnia, ha a mező üres és kötelező, de végrehajthat további ellenőrzéseket is a másfajta mezőknél, mint például ha a kiválasztott lehetőség érvényes, vagy ha egy dátumnak csak a múltban kell lennie, stb. Egy egyéni hibaüzenetet is biztosíthat.

```
sub SearchFieldRender { ... }
```

Ezt a függvényt a jegykeresés párbeszédablak használja, és hasonló a `EditFieldRander()` függvényhez, de normális esetben egy keresési képernyőn kisebb változtatásokat kell elvégezni az összes mezőnél. Például ebben az esetben egy HTML szöveges beviteli mező használunk egy jelszóbeviteli mező helyett. Más mezőkben (például legördülő lista) többválasztósként jelenik meg azért, hogy egyszerre több érték keresését tegye lehetővé a felhasználónak.

```
sub SearchFieldValueGet { ... }
```

Nagyon hasonló az `EditFieldValueGet()` függvényhez, de eltérő név előtagot használ a keresési párbeszédablak képernyőhöz átírva.

```
sub SearchFieldParameterBuild { ... }
```

Ezt a függvényt is a jegykeresés párbeszédablak használja a helyes operátor és érték beállításához, hogy elvégezze a keresést ezen a mezőn. Azt is visszaadja, hogy az értéket hogyan kell megjeleníteni a használt keresési attribútumokban a találatok oldalon.

```
sub StatsFieldParameterBuild { ... }
```

Ezt a függvényt a statisztikák moduljai használják. Tartalmazza a mező meghatározást a statisztikák formátumában. A rögzített értékekkel rendelkező mezőknél tartalmazza az összes lehetséges értéket is, és ha azok lefordíthatók, akkor nézze meg a `BaseSelect` illesztőprogram kódját példaként arra, hogy azokat hogyan kell megvalósítani.

```
sub StatsSearchFieldParameterBuild { ... }
```

Ez a függvény nagyon hasonló a `SearchFieldParameterBuild()` függvényhez. A különbség az, hogy az utóbbi a keresési profilból kapja meg az értéket, és ez pedig közvetlenül a paramétereiből kapja meg az értéket.

Ezt a függvényt a statisztikák modul használja.

```
sub TemplateValueTypeGet { ... }
```

Ezt a függvényt annak megismeréséhez használják, hogy a dinamikus mező értékei hogyan vannak eltárolva egy olyan profilnál, amelyet le kell kérni skalár vagy tömb formában, és meghatározza a mező helyes nevét is a profilban.

```
sub RandomValueSet { ... }
```

Ezt a függvényt az `otrs.FillDB.pl` parancsfájl használja, hogy feltöltse az adatbázist néhány teszt és véletlenszerű adattal. A függvény által beszúrt érték nem igazán fontos. Az egyetlen megkötés az, hogy az értéknek kompatibilisnek kell lennie a mezőérték típusával.

```
sub ObjectMatch { ... }
```

Az értesítési modulok használják. Ez a függvény 1-et ad vissza, ha a mező jelen van a `$Param{ObjectAttributes}` paraméterben, és ha megegyezik a megadott értékkel.

### 3.3.27 Egy dinamikus mező funkcionalitás kiterjesztés létrehozása

Ezen folyamat bemutatásához a `Foo` függvénynél egy új dinamikus mező funkcionalitás kiterjesztés lesz hozzáadva a háttérprogram objektumhoz, valamint a szövegmező illesztőprogramba.

A kiterjesztés létrehozásához három fájlt fogunk létrehozni:

1. Egy beállítófájlt (XML) a modulok regisztrálásához.
2. Egy háttérprogram kiterjesztést (Perl) az új függvény meghatározásához.
3. Egy szövegmező illesztőprogram kiterjesztést (Perl), amely megvalósítja az új függvényt a szövegmezőknél.

Fájlszerkezet:

```
$HOME (e. g. /opt/otrs/)
|
...
|--/Kernel/
|   |--/Config/
|   |   |--/Files/
|   |   |   |--/XML/
|   |   |   |DynamicFieldFooExtension.xml
```

(continues on next page)

(folytatás az előző oldalról)

```
...
|   |--/System/
|   |   |--/DynamicField/
|   |   |   FooExtensionBackend.pm
|   |   |   |--/Driver/
|   |   |   |   |FooExtensionText.pm
...
```

### Dinamikus mező Foo kiterjesztés fájlok

### Dinamikus mező kiterjesztés beállítófájl példa

A beállítófájlokat használják a kiterjesztések regisztrálásához a háttérprogramnál és az illesztőprogramoknál, valamint az egyes illesztőprogramok új viselkedéseihez.

---

**Megjegyzés:** Ha egy illesztőprogramot kiterjesztenek egy új függvénnyel, akkor a háttérprogramnak is szüksége lesz egy kiterjesztésre ahhoz a függvényhez.

---

Ebben a szakaszban a `Foo` kiterjesztéshez egy beállítófájl van megjelenítve és elmagyarázva.

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="2.0" init="Application">
```

Ez a normál fejléc egy beállítófájlhoz.

```
<ConfigItem Name="DynamicFields::Extension::Backend###100-Foo" Required="0"␣
↪Valid="1">
    <Description Translatable="1">Dynamic Fields Extension.</Description>
    <Group>DynamicFieldFooExtension</Group>
    <SubGroup>DynamicFields::Extension::Registration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">
↪Kernel::System::DynamicField::FooExtensionBackend</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

Ez a beállítás regisztrálja a kiterjesztést a háttérprogram objektumban. A modul alaposztályként lesz betöltve a `Backend` objektumból.

```
<ConfigItem Name="DynamicFields::Extension::Driver::Text###100-Foo" Required="0
↪" Valid="1">
    <Description Translatable="1">Dynamic Fields Extension.</Description>
    <Group>DynamicFieldFooExtension</Group>
    <SubGroup>DynamicFields::Extension::Registration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">
↪Kernel::System::DynamicField::Driver::FooExtensionText</Item>
```

(continues on next page)

(folytatás az előző oldalról)

```
            <Item Key="Behaviors">
                <Hash>
                    <Item Key="Foo">1</Item>
                </Hash>
            </Item>
        </Hash>
    </Setting>
</ConfigItem>
```

Ez egy kiterjesztés regisztrációja a szöveg dinamikus mező illesztőprogramban. A modul alaposztályként lesz betöltve az illesztőprogramban. Figyeljen arra is, hogy új viselkedések is megadhatók. Ezek a kiterjesztett viselkedések lesznek hozzáadva azokhoz a viselkedésekhez, amelyekkel az illesztőprogram eredetileg rendelkezik, ebből adódóan a `HasBehavior()` hívásával azt ellenőrizve, hogy ezeknél az új viselkedéseknél teljesen átlátszó legyen.

```
</otrs_config>
```

Egy beállítófájl szabványos lezárása.

### Dinamikus mező háttérprogram kiterjesztés példa

A háttérprogram kiterjesztések átláthatóan lesznek betöltve magába a háttérprogramba egy alaposztályként. A háttérprogramból az összes meghatározott objektum és tulajdonság elérhető lesz a kiterjesztésben.

---

**Megjegyzés:** A háttérprogram kiterjesztésben meghatározott összes új függvényt meg kell valósítani egy illesztőprogram kiterjesztésben.

---

Ebben a szakaszban a háttérprogramhoz készített `Foo` kiterjesztés van megjelenítve és elmagyarázva. A kiterjesztés csak a `Foo()` függvényt határozza meg.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::DynamicField::FooExtensionBackend;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);

=head1 NAME

Kernel::System::DynamicField::FooExtensionBackend

=head1 SYNOPSIS
```

(continues on next page)

(folytatás az előző oldalról)

```
DynamicFields Extension for Backend

=head1 PUBLIC INTERFACE

=over 4

=cut
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

```
=item Foo()

Testing function: returns 1 if function is available on a Dynamic Field␣
↪driver.

    my $Success = $BackendObject->Foo(
        DynamicFieldConfig   => $DynamicFieldConfig,      # complete config of␣
↪the DynamicField
    );

Returns:
    $Success = 1;       # or undef

=cut

sub Foo {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for my $Needed (qw(DynamicFieldConfig)) {
        if ( !$Param{$Needed} ) {
            $Kernel::OM->Get('Kernel::System::Log')->Log(
                Priority => 'error',
                Message  => "Need $Needed!",
            );

            return;
        }
    }

    # check DynamicFieldConfig (general)
    if ( !IsHashRefWithData( $Param{DynamicFieldConfig} ) ) {
        $Kernel::OM->Get('Kernel::System::Log')->Log(
            Priority => 'error',
            Message  => "The field configuration is invalid",
        );

        return;
    }
```

(continues on next page)

(folytatás az előző oldalról)

```
    # check DynamicFieldConfig (internally)
    for my $Needed (qw(ID FieldType ObjectType)) {
        if ( !$Param{DynamicFieldConfig}->{$Needed} ) {
            $Kernel::OM->Get('Kernel::System::Log')->Log(
                Priority => 'error',
                Message  => "Need $Needed in DynamicFieldConfig!",
            );

            return;
        }
    }

    # set the dynamic field specific backend
    my $DynamicFieldBackend = 'DynamicField' . $Param{DynamicFieldConfig}->
↪{FieldType} . 'Object';

    if ( !$Self->{$DynamicFieldBackend} ) {
        $Kernel::OM->Get('Kernel::System::Log')->Log(
            Priority => 'error',
            Message  => "Backend $Param{DynamicFieldConfig}->{FieldType} is␣
↪invalid!",
        );

        return;
    }

    # verify if function is available
    return if !$Self->{$DynamicFieldBackend}->can('Foo');

    # call HasBehavior on the specific backend
    return $Self->{$DynamicFieldBackend}->Foo(%Param);
}
```

A `Foo()` függvényt kizárólag tesztelési célokra használják. Először leellenőrzi a dinamikus mező beállításait, majd azt ellenőrzi, hogy a dinamikus mező illesztőprogram (típus) létezik-e, és betöltésre került-e. Annak megakadályozásához, hogy egy olyan illesztőprogramnál történjen függvényhívás, ahol nincs meghatározva, először azt ellenőrzi, hogy az illesztőprogram képes-e végrehajtani a függvényt, majd végrehajtja a függvényt az illesztőprogramban az összes paramétert átadva.

---

**Megjegyzés:** Lehetséges annak a lépésnek a kihagyása is, amely azt próbálja, hogy az illesztőprogram képes-e végrehajtani a függvényt. Ennek elvégzéséhez szükséges egy mechanizmus megvalósítása az előtétprogram-modulban egy különleges viselkedés megköveteléséhez a dinamikus mezőnél, és csak azután hívja meg a függvényt a háttérprogram objektumban.

---

#### Dinamikus mező illesztőprogram kiterjesztés példa

Az illesztőprogram kiterjesztések átláthatóan lesznek betöltve magába az illesztőprogramba egy alaposztályként. Az illesztőprogramból az összes meghatározott objektum és tulajdonság elérhető lesz a kiterjesztésben.

**Megjegyzés:** Az illesztőprogram kiterjesztésben megvalósított összes új függvényt meg kell határozni egy háttérprogram kiterjesztésben, mivel minden függvény a háttérprogram objektumból kerül meghívásra.

Ebben a szakaszban a szövegmező illesztőprogramhoz készített `Foo` kiterjesztés van megjelenítve és elmagyarázva. A kiterjesztés csak a `Foo()` függvényt valósítja meg.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::DynamicField::Driver::FooExtensionText;

use strict;
use warnings;

=head1 NAME

Kernel::System::DynamicField::Driver::FooExtensionText

=head1 SYNOPSIS

DynamicFields Text Driver Extension

=head1 PUBLIC INTERFACE

This module extends the public interface of L
↪<Kernel::System::DynamicField::Backend>.
Please look there for a detailed reference of the functions.

=over 4

=cut
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

```
sub Foo {
    my ( $Self, %Param ) = @_;
    return 1;
}
```

A `Foo()` függvénynek nincs különleges logikája. Csak tesztelésre van, és mindig 1-et ad vissza.

### 3.3.28 Jegy levelezési modul

A levelezési modulokat a levelezési folyamat során használják. Kétféle levelezési modul létezik:

- `PostMasterPre`: egy e-mail feldolgozása után használják.

- `PostMasterPost`: azután használják, amikor egy e-mail feldolgozásra került, és az adatbázisban van.

Ha saját levelezési szűrőket szeretne létrehozni, akkor egyszerűen hozza létre a saját modulját. Ezek a modulok a `Kernel/System/PostMaster/Filter/*.pm` fájlokban találhatók. Az alapértelmezett modulokért nézze meg az adminisztrátori kézikönyvet. Mindössze két függvényre van szüksége: `new()` és `Run()`.

A következőkben egy példaszerű modul található az e-mailek egyeztetéséhez és az X-OTRS fejlécek beállításához (további információkért nézze meg a `doc/X-OTRS-Headers.txt` fájlt).

`Kernel/Config/Files/XML/MyModule.xml`:

```
<ConfigItem Name="PostMaster::PreFilterModule###1-Example" Required="0" Valid=
↪"1">
    <Description Translatable="1">Example module to filter and manipulate␣
↪incoming messages.</Description>
    <Group>Ticket</Group>
    <SubGroup>Core::PostMaster</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::System::PostMaster::Filter::Example</
↪Item>
            <Item Key="Match">
                <Hash>
                    <Item Key="From">noreply@</Item>
                </Hash>
            </Item>
            <Item Key="Set">
                <Hash>
                    <Item Key="X-OTRS-Ignore">yes</Item>
                </Hash>
            </Item>
        </Hash>
    </Setting>
</ConfigItem>
```

És a tényleges szűrőkód a `Kernel/System/PostMaster/Filter/Example.pm` fájlban:

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::PostMaster::Filter::Example;

use strict;
use warnings;

our @ObjectDependencies = (
    'Kernel::System::Log',
);
```

(continues on next page)

(folytatás az előző oldalról)

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless ($Self, $Type);

    $Self->{Debug} = $Param{Debug} || 0;

    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;
    # get config options
    my %Config = ();
    my %Match = ();
    my %Set = ();
    if ($Param{JobConfig} && ref($Param{JobConfig}) eq 'HASH') {
        %Config = %{$Param{JobConfig}};
        if ($Config{Match}) {
            %Match = %{$Config{Match}};
        }
        if ($Config{Set}) {
            %Set = %{$Config{Set}};
        }
        }
        # match 'Match => ???' stuff
        my $Matched = '';
        my $MatchedNot = 0;
        for (sort keys %Match) {
        if ($Param{GetParam}->{$_} && $Param{GetParam}->{$_} =~ /$Match{$_}/
↪i) {
            $Matched = $1 || '1';
            if ($Self->{Debug} > 1) {
                $Kernel::OM->Get('Kernel::System::Log')->Log(
                    Priority => 'debug',
                    Message => "'$Param{GetParam}->{$_}' =~ /$Match{$_}/i␣
↪matched!",
                );
            }
        }
        else {
            $MatchedNot = 1;
            if ($Self->{Debug} > 1) {
                $Kernel::OM->Get('Kernel::System::Log')->Log(
                    Priority => 'debug',
                    Message => "'$Param{GetParam}->{$_}' =~ /$Match{$_}/i␣
↪matched NOT!",
                );
            }
```

(continues on next page)

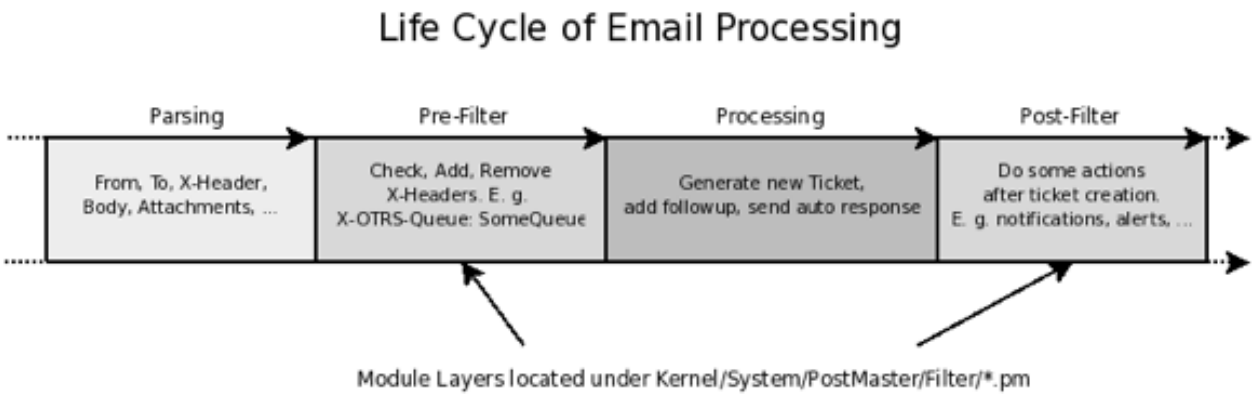
(folytatás az előző oldalról)

```
        }
        }
        # should I ignore the incoming mail?
        if ($Matched && !$MatchedNot) {
        for (keys %Set) {
            if ($Set{$_} =~ /\[\*\*\*\]/i) {
                $Set{$_} = $Matched;
            }
            $Param{GetParam}->{$_} = $Set{$_};
            $Kernel::OM->Get('Kernel::System::Log')->Log(
                Priority => 'notice',
                Message => "Set param '$_' to '$Set{$_}' (Message-ID: $Param
↪{GetParam}->{'Message-ID'}) ",
            );
        }
    }

    return 1;
}

1;
```

A következő kép az e-mail feldolgozási folyamatát mutatja be.




6. ábra: E-mail feldolgozási folyamat

### 3.3.29 Folyamatkezelés

Az OTRS 7-től kezdve a folyamatkezelés képes vezérelt modulokat használni a tevékenységek (vezérelt feladatok) és/vagy szekvenciafolyamok (korábban átmenet-műveteknek hívták) végrehajtásához. Ezek a modulok a `Kernel::System::ProcessManagement::Modules` helyen találhatók.

#### Folyamatkezelés modulok

A folyamatkezelés modulok Perl nyelven megírt parancsfájlok, amelyek a folyamatjegy során bizonyos művelet vagy műveletek végrehajtásához készültek, mint például egy dinamikus mező beállítása vagy egy várólista megváltoztatása. Vagy a rendszer bármely egyéb részéhez, mint például új jegyek létrehozásához.

Alapértelmezetten a modulok egy kulcs-érték készletet használnak, hogy paraméterként használják azokat a műveletekhez, például a folyamatjegy várólistájának megváltoztatásához a várólista vagy a várólista-azonosító, illetve a hozzá tartozó érték szükséges.

Néhány parancsfájl esetleg többet igényel egy egyszerű kulcs-érték párnál paraméterként, vagy a beállításának esetleg még felhasználóbarátabb grafikus felületre lenne szüksége. Ilyen esetekben az OTRS biztosít néhány beállítási mezőtípust, amely szintén kibővíthető, ha szükséges.

Jelenlegi mezőtípusok:

**Legördülő** Egy legördülő listát jelenít meg előre meghatározott értékekkel.

**Kulcs-érték lista** Egyszerű kulcs-érték párok listáját jeleníti meg (szöveges bevitelek). A párok hozzáadhatók vagy törölhetők.

**Többnyelvű Rich Text** Egy rendszernyelvhez rendelt Rich Text szerkesztőt jelenít meg, valamint egy nyelvválasztót is Rich Text mezők hozzáadásához a megfelelő kiválasztott nyelvhez.

**Címzettek** Ügyintézőkkel előre kitöltött többválasztós mezőt jelenít meg, akik használhatók e-mailek címzettjeinél, valamint egy szabad beviteli mezőt is megjelenít, amely használható külső e-mail címek megadásához és a címzettlistához adásához.

**Rich Text** Egy egyszerű Rich Text mezőt jelenít meg.

### Egy új folyamatkezelés modul létrehozása

A következő egy példa arra, hogy hogyan kell létrehozni egy folyamatkezelés modult. Tartalmaz egy szakaszt, ahol az összes lehetséges mező meg van határozva referenciaként. Egy új modul létrehozásához csak egy mezőtípus szükséges, de fontolja meg, hogy megegyezés szerint a felhasználó-azonosító paramétert használják az összes művelet megszemélyesítéséhez a modulban egy másik felhasználó nevében, ha az eltér attól, aki a műveletet aktiválta. Ekkor jó megoldás lehet a kulcs-érték pár lista mezőtípust minden esetben felvenni a többi szükséges mező mellett.

### Folyamatkezelés modul kódpélda

A következő kód egy üres folyamatkezelés modult valósít meg, amely használható vezérelt feladat tevékenységekben vagy szekvenciafolyam-műveletekben.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::ProcessManagement::Modules::Test;

use strict;
use warnings;
use utf8;

use Kernel::System::VariableCheck qw(:all);

use parent qw(Kernel::System::ProcessManagement::Modules::Base);
```

(continues on next page)

(folytatás az előző oldalról)

```
our @ObjectDependencies = ( );
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

Ebben az esetben a `Base` osztályból származtatunk le, és az objektumkezelő függőségei be vannak állítva.

```
sub new {
    my ( $Type, %Param ) = @_;

    # Allocate new hash for object.
    my $Self = {};
    bless( $Self, $Type );

    $Self->{Config} = {
        Description => 'Description for the module.',
        ConfigSets  => [
            {
                Type        => 'Dropdown',
                Description => 'Description for Dropdown.',
                Mandatory   => 1,
                Config      => {
                    Data => {
                        Internal1 => 'Display 1',
                        Internal2 => 'Display 2',
                    },
                    Name         => 'Test-DropDown',
                    Multiple     => 1,
                    PossibleNone => 1,
                    Sort         => 'AlphanumericValue',
                    Translation  => 1,
                },
            },
            {
                Type        => 'KeyValueList',
                Description => 'Description of the Key/Value pairs',
                Defaults    => [
                    {
                        Key       => 'Test-Param1',
                        Value     => 'Hello',
                        Mandatory => 1,
                    },
                    {
                        Key       => 'Test-Param2',
                        Mandatory => 1,
                    },
                    {
                        Key       => 'Test-Param3',
                        Value     => 'World',
                        Mandatory => 0,
                    },
                ],
```

(continues on next page)

(folytatás az előző oldalról)

```
            },
            {
                Type        => 'MultiLanguageRichText',
                Description => "Description for Mutli-Language Rich Text.",
                Defaults    => [
                    {
                        Key   => 'en_Subject',
                        Value => 'Hello',
                    },
                    {
                        Key   => 'en_Body',
                        Value => 'World',
                    },

                ],
            },
            {
                Type        => 'Recipients',
                Description => "Description for Recipients."
            },
            {
                Type        => 'RichText',
                Description => "Description for Rich Text.",
                Defaults    => [
                    {
                        Value     => 'Hello World',
                        Mandatory => 1,
                    },
                ],
            },
        ],
    };

    return $Self;
}
```

A `new()` konstruktor hozza létre az osztály új példányát. A beállítási mezők itt vannak meghatározva, és a `$Self->{Config}` változóban vannak beállítva.

A beállításnak két fő bejegyzése van:

**`Description`** Ezt arra használják, hogy elmagyarázzák az adminisztrátoroknak, hogy mit csinál a modul és/vagy a beállítására vonatkozó megfontolásokhoz.

**`ConfigSets`** Ez csak egy tároló a tényleges beállítási mezőkhöz.

Az összes beállítási mezőnek szüksége van egy típusra, amely meghatározza a mező fajtáját és tartalmazhat egy belső leírást is, amely a mező felületi elem címeként lesz használva. Ha ez nincs meghatározva, akkor egy alapértelmezett leírás lesz használva.

Minden egyes mező meghatározza a beállítási paramétereit és képességeit. A következő egy kis referencia azon mezőkhöz, amelyeket az OTRS alapértelmezetten biztosít.

- `Dropdown`

  **`Mandatory`** Annak meghatározásához használják, hogy egy értéket kötelező-e beállítani.

**Config** Információkat tárol a legördülő lista megjelenítéséhez.

**Data** Egyszerű kivonat, amely meghatározza a lehetőségeket a legördülőhöz. A kulcsok belsőleg lesznek használva és az értékek azok a lehetőségek, amelyeket a felhasználó a képernyőn lát.

**Name** A paraméter neve.

**Multiple** Annak meghatározásához, hogy csak egy vagy több értéket lehet kiválasztani.

**PossibleNone** Meghatározza, hogy az értékek listája ajánljon-e fel üres értéket vagy sem.

**Sort** Meghatározza, hogy a lehetőségeket hogyan kell rendezni, amikor a mező megjelenik. A lehetséges értékek: `AlphanumericValue`, `NumericValue`, `AlphanumericKey` és `NumericKey`.

**Translation:** Beállítja, hogy a megjelenített értékeket le kell-e fordítani.

- `KeyValueList`

  **Defaults** Kivonatok tömbje, amely az alapértelmezett beállítást tárolja a kulcs-érték párjához.

  **Key** Egy paraméter neve.

  **Value** A paraméter alapértelmezett értéke (elhagyható).

  **Mandatory** A kötelező paramétereket nem lehet átnevezni vagy eltávolítani (elhagyható).

- `MultiLanguageRichText`

  **Defaults** Kivonatok tömbje, amely az alapértelmezett beállítómezőt tárolja minden egyes nyelvhez és mezőrészhez.

  **Key** Ezt egy nyelv alkotja, mint például `en` vagy `es_MX`, amelyet egy _ (aláhúzás karakter) követ, majd a `Subject` vagy `Body` szó a mező megfelelő részéhez.

  **Value** A mezőrész alapértelmezett értéke (elhagyható).

- `Recipients`

  Nincs további beállítás megadva ennél a típusú mezőnél.

- `RichText`

  **Defaults** Kivonatok tömbje, amely az alapértelmezett beállítómezőt tárolja (csak az első elem lesz használva).

  **Value** A mező alapértelmezett értéke.

  **Mandatory** Annak meghatározásához használják, hogy egy értéket kötelező-e beállítani.

```
sub Run {
    my ( $Self, %Param ) = @_;

    # Define a common message to output in case of any error.
    my $CommonMessage = "Process: $Param{ProcessEntityID} Activity: $Param
↪{ActivityEntityID}";

    # Add SequenceFlowEntityID to common message if available.
```

(continues on next page)

(folytatás az előző oldalról)

```
    if ( $Param{SequenceFlowEntityID} ) {
        $CommonMessage .= " SequenceFlow: $Param{SequenceFlowEntityID}";
    }

    # Add SequenceFlowActionEntityID to common message if available.
    if ( $Param{SequenceFlowActionEntityID} ) {
        $CommonMessage .= " SequenceFlowAction: $Param
↪{SequenceFlowActionEntityID}";
    }

    # Check for missing or wrong params.
    my $Success = $Self->_CheckParams(
        %Param,
        CommonMessage => $CommonMessage,
    );
    return if !$Success;

    # Override UserID if specified as a parameter in the TA config.
    $Param{UserID} = $Self->_OverrideUserID(%Param);

    # Use ticket attributes if needed.
    $Self->_ReplaceTicketAttributes(%Param);
    $Self->_ReplaceAdditionalAttributes(%Param);

    # Get module configuration.
    my $ModuleConfig = $Param{Config};

    # Add module logic here!

    return 1;
}
```

A `Run` metódus a modul fő része. Először beállít egy szokásos üzenetet, amely hibanaplókban vagy egyéb célokra használható. A következetességért erősen ajánlott a használata, amint az fent bemutatásra került.

A következő lépés annak ellenőrzése, hogy a globális paraméterek helyesen lettek-e elküldve.

Megegyezés szerint az összes modulnak képesnek kell lennie felülírnia a jelenlegi felhasználó-azonosítót azzal, amit paraméterben adnak át (ha létezik). Ezt az átadott felhasználó-azonosítót kell használni minden függvényhívásban, amely igényli azt.

A felhasználó által meghatározott attribútumértékek használhatnak jelenlegi jegyértékeket az OTRS intelligens címkéivel. A `_ReplaceTicketAttributes` a normál szöveges attribútumoknál van használatban, míg a `_ReplaceAdditionalAttributes` a Rich Text szövegeknél. Az összetettebb paramétereknél szükséges lehet a függvények személyre szabása az intelligens címkék cseréjéhez.

Az ezután következő a modul megfelelő logikája.

Ha minden rendben volt, akkor 1-et kell visszaadnia.

### Egy új folyamatkezelés modul beállítási mező létrehozása

A következő egy példa arra, hogy hogyan kell létrehozni egy folyamatkezelés modul beállítási mezőt. Ezt a mezőt használhatja bármely folyamatkezelés modul a beállítása után.

### Folyamatkezelés modul beállítási mező kódpélda

A következő kód egy egyszerű beviteli folyamatkezelés modul beállítási mezőt (teszt) valósít meg. A mező neve és alapértelmezett értéke egy `ConfigSets` folyamatkezelés modulon keresztül állítható be.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::ProcessManagement::ModuleConfiguration::Test;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);
use Kernel::Language qw(Translatable);

our @ObjectDependencies = (
    'Kernel::Output::HTML::Layout',
    'Kernel::System::Web::Request',
);
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a `package` kulcsszón keresztül van deklarálva.

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless( $Self, $Type );

    return $Self;
}
```

A `new` konstruktor hozza létre az osztály új példányát.

Minden egyes beállítási mező legalább két fő metódus megvalósítását igényli: `Render` és `GetParams`.

```
sub Render {
    my ( $Self, %Param ) = @_;

    my $ConfigSet = $Param{ConfigSet} // {};
    my $EntityConfig = $Param{EntityData}->{Config}->{Config}->{ConfigTest} // {};

    my %Data;

    $Data{Description} = $ConfigSet->{Description} || 'Config Parameters (Test)
↪';
    $Data{Name} = $ConfigSet->{Config}->{Name} // 'Test';
```

(continues on next page)

(folytatás az előző oldalról)

```
    $Data{Value} = $EntityConfig->{ $ConfigSet->{Config}->{Name} } // $ConfigSet->
↪{Defaults}->[0]->{Value} // '';

    return {
        Success => 1,
        HTML    => $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Output(
            TemplateFile => 'ProcessManagement/ModuleConfiguration/Test',
            Data         => \%Data,
        ),
    };
}
```

A `Render` metódus felelős a szükséges HTML létrehozásáért a mezőnél.

Ebben a példában ez először behatárol néhány paramétert az egyszerűbb olvasásért és karbantartásért.

A következő sorok állítják be a megjelenítendő adatokat. A mező felületi elem `Description` címe a `ConfigSet` változóból lesz kinyerve, ha meg van határozva, egyébként egy alapértelmezett szöveget használ. A `Name` mezőhöz hasonlóan a `Value` értékénél először ellenőrzi, hogy a tevékenység vagy a szekvenciafolyam-művelet rendelkezik-e már egy eltárolt értékkel, és ha nem, akkor megpróbálja az alapértelmezett értéket használni a `ConfigSet` változóból, vagy üres értéket egyébként.

A végén visszaad egy szerkezetet a HTML-kóddal, amely a sablonból lett feltöltve a kinyert adatokkal.

```
sub GetParams {
    my ( $Self, %Param ) = @_;

    my %GetParams;
    my $Config = $Param{ConfigSet}->{Config} // 'Test';

    $GetParams{ $Config->{Name} } = $Kernel::OM->Get(
↪'Kernel::System::Web::Request')->GetParam( Param => $Config->{Name} );

    return \%GetParams;
}
```

Ennél a példánál a `GetParams` metódus nagyon egyértelmű. Lekéri a mező nevét a `ConfigSet` változóból vagy egy alapértelmezettet használ, és lekéri az értéket a webkérésből.

### Folyamatkezelés modul beállítási mező sablon példa

A következő kód egy alap HTML sablont valósít meg a teszt folyamatkezelés modul beállítási mezőhöz.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban.

```
<div id="TestConfig" class="WidgetSimple Expanded">
    <div class="Header">
        <div class="WidgetAction Toggle">
            <a href="#" title="[% Translate("Show or hide the content") |␣
↪html %]"><i class="fa fa-caret-right"></i><i class="fa fa-caret-down"></i></
↪a>
        </div>
        <h2 for="Config">[% Translate(Data.Description) | html %]</h2>
    </div>
    <div class="Content" id="TestParams">
        <fieldset class="TableLike">
            <label for="[% Data.Name | html %]">[% Data.Name | html %]</label>
            <div class="Field">
                <input type="text" value="[% Data.Value | html %]" name="[%␣
↪Data.Name | html %]" id="[% Data.Name | html %]" />
            </div>
        </fieldset>
    </div>
</div>
```

A sablon egy egyszerű szöveges beviteli elemet jelenít meg a hozzárendelt címkével.

FEJEZET 4

# Hogyan tehetők közzé az OTRS kiterjesztések

## 4.1 Csomagkezelés

Az OPM (OTRS csomagkezelő) egy mechanizmus az OTRS keretrendszerhez való szoftvercsomagok terjesztésére HTTP-n, FTP-n vagy fájlfeltöltésen keresztül.

Például az OTRS projekt OTRS modulokat kínál OTRS csomagokban az internetes tárolókon vagy az FTP-kiszolgálóinkon keresztül, mint például naptár, fájlkezelő vagy webes levelező. A csomagok az adminisztrátori felületen keresztül kezelhetők (telepítés, frissítés vagy eltávolítás).

### 4.1.1 Csomagterjesztés

Ha egy internetes OPM tárolót szeretne létrehozni, akkor egyszerűen mondja meg az OTRS keretrendszernek a `Package::RepositoryList` rendszerbeállítási lehetőség bekapcsolásával, hogy hol van annak a helye, és adja meg az új helyet itt. Ezután egy új választási lehetősége lesz a csomagkezelőben.

A tárolójában hozzon létre egy index fájlt az OPM csomagokhoz. Az OTRS egyszerűen beolvassa ezt az index fájlt, és tudni fogja, hogy mely csomagok érhetők el.

```
shell> bin/otrs.Console.pl Dev::Package::RepositoryIndex /path/to/repository/␣
↪> /path/to/repository/otrs.xml
```

### 4.1.2 Csomagparancsok

A következő OPM parancsokat használhatja az adminisztrátori felületen vagy a `bin/otrs.Console.pl` parancsfájllal az adminisztrátori feladatok kezeléséhez az OPM csomagoknál.

#### Telepítés

OPM csomagok telepítése.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor: `bin/otrs.Console.pl Admin::Package::Install /útvonal/ehhez/csomag.opm`

#### Eltávolítás

OPM csomagok eltávolítása.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor: `bin/otrs.Console.pl Admin::Package::Uninstall /útvonal/ehhez/csomag.opm`

#### Frissítés

OPM csomagok frissítése.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor: `bin/otrs.Console.pl Admin::Package::Upgrade /útvonal/ehhez/csomag.opm`

#### Lista

Az összes OPM csomag felsorolása.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor: `bin/otrs.Console.pl Admin::Package::List`

## 4.2 Csomagkészítés

Ha egy OPM csomagot (`.opm`) szeretne létrehozni, akkor létre kell hoznia egy specifikációs fájl (`.sopm`), amely a csomag tulajdonságait tartalmazza.

### 4.2.1 Csomagspecifikációs fájl

Az OPM csomag XML alapú. A `.sopm` fájlt egy szöveg- vagy egy XML-szerkesztővel hozhatja létre és szerkesztheti. Ez metaadatokat, egy fájllistát és adatbázis-beállításokat tartalmaz.

**<Name>** * A csomag neve.

```
<Name>Calendar</Name>
```

**<Version>** * A csomag verziója.

```
<Version>1.2.3</Version>
```

**<Framework>** * A megcélzott keretrendszer verziója (a 7.0.x jelentése például 7.0.1 vagy 7.0.2).

```
<Framework>7.0.x</Framework>
```

Több alkalommal is használható.

```
<Framework>5.0.x</Framework>
<Framework>6.0.x</Framework>
<Framework>7.0.x</Framework>
```

**<Vendor>** * A csomag gyártója.

```
<Vendor>OTRS AG</Vendor>
```

**<URL>** * A gyártó URL-e.

```
<URL>https://otrs.com/</URL>
```

**<License>** * A csomag licence.

```
<License>GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007</License>
```

**<ChangeLog>** A csomag változásnaplója.

```
<ChangeLog Version="1.1.2" Date="2018-11-15 18:45:21">Added some feature.
↪</ChangeLog>
<ChangeLog Version="1.1.1" Date="2018-11-15 16:17:51">New package.</
↪ChangeLog>
```

**<Description>** * A csomag leírása különböző nyelveken.

```
<Description Lang="en">A web calendar.</Description>
<Description Lang="de">Ein Web Kalender.</Description>
```

**Csomagműveletek** A csomag lehetséges műveletei a telepítés után. Ha ezen műveletek valamelyike nincs meghatározva a csomagnál, akkor lehetségesként fogja tekinteni.

```
<PackageIsVisible>1</PackageIsVisible>
<PackageIsDownloadable>0</PackageIsDownloadable>
<PackageIsRemovable>1</PackageIsRemovable>
```

Egy különleges csomagművelet a `PackageAllowDirectUpdate`. Egy csomag csak akkor frissíthető egy alacsonyabb főverzióról (a legutolsónál korábbiról) a legutolsó verzióra (például egy OTRS 5-ös csomag frissítése OTRS 7-re), ha ez meg van határozva a csomagban és igazra van állítva.

```
<PackageAllowDirectUpdate>1</PackageAllowDirectUpdate>
```

**<BuildHost>** Ezt az OPM automatikusan ki fogja tölteni.

```
<BuildHost>?</BuildHost>
```

**<BuildDate>** Ezt az OPM automatikusan ki fogja tölteni.

```
<BuildDate>?</BuildDate>
```

**<PackageRequired>** Csomagok, amelyeket előzetesen telepíteni kell. Ha a `PackageRequired` használatban van, akkor a szükséges csomag egy verzióját meg kell adni.

```
<PackageRequired Version="1.0.3">SomeOtherPackage</PackageRequired>
<PackageRequired Version="5.3.2">SomeotherPackage2</PackageRequired>
```

**<ModuleRequired>** Perl-modulok, amelyeket előzetesen telepíteni kell.

```
<ModuleRequired Version="1.03">Encode</ModuleRequired>
<ModuleRequired Version="5.32">MIME::Tools</ModuleRequired>
```

**<OS>** A szükséges operációs rendszer.

```
<OS>linux</OS>
<OS>darwin</OS>
<OS>mswin32</OS>
```

**<Filelist>** Ez a csomagban lévő fájlok listája.

```
<Filelist>
    <File Permission="644" Location="Kernel/Config/Files/Calendar.pm"/>
    <File Permission="644" Location="Kernel/System/CalendarEvent.pm"/>
    <File Permission="644" Location="Kernel/Modules/AgentCalendar.pm"/>
    <File Permission="644" Location="Kernel/Language/de_AgentCalendar.pm"/
 ↪>
</Filelist>
```

**<DatabaseInstall>** Adatbázis-bejegyzések, amelyeket létre kell hozni, amikor a csomagot telepítik.

```
<DatabaseInstall>
    <TableCreate Name="calendar_event">
    <Column Name="id" Required="true" PrimaryKey="true" AutoIncrement=
 ↪"true" Type="BIGINT"/>
    <Column Name="title" Required="true" Size="250" Type="VARCHAR"/>
    <Column Name="content" Required="false" Size="250" Type="VARCHAR"/>
    <Column Name="start_time" Required="true" Type="DATE"/>
    <Column Name="end_time" Required="true" Type="DATE"/>
    <Column Name="owner_id" Required="true" Type="INTEGER"/>
    <Column Name="event_status" Required="true" Size="50" Type="VARCHAR"/>
    </TableCreate>
</DatabaseInstall>
```

Választhat `<DatabaseInstall Type="post">` vagy `<DatabaseInstall Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<DatabaseUpgrade>** Információk arról, hogy mely műveleteket kell végrehajtani egy frissítés esetén.

Például ha egy korábban telepített csomag verziója 1.3.4 alatt van (mondjuk 1.2.6), akkor végre lesz hajtva a meghatározott művelet:

```
<DatabaseUpgrade>
    <TableCreate Name="calendar_event_involved" Version="1.3.4">
        <Column Name="event_id" Required="true" Type="BIGINT"/>
        <Column Name="user_id" Required="true" Type="INTEGER"/>
    </TableCreate>
</DatabaseUpgrade>
```

Választhat `<DatabaseUpgrade Type="post">` vagy `<DatabaseUpgrade Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<DatabaseReinstall>** Információk arról, hogy mely műveleteket kell végrehajtani, ha a csomagot újratelepítik.

```
<DatabaseReinstall></DatabaseReinstall>
```

Választhat `<DatabaseReinstall Type="post">` vagy `<DatabaseReinstall Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<DatabaseUninstall>** A végrehajtandó műveletek a csomag eltávolításakor.

```
<DatabaseUninstall>
    <TableDrop Name="calendar_event" />
</DatabaseUninstall>
```

Választhat `<DatabaseUninstall Type="post">` vagy `<DatabaseUninstall Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<IntroInstall>** Egy telepítés előtti vagy utáni bevezető megjelenítéséhez a telepítési párbeszédablakban.

```
<IntroInstall Type="post" Lang="en" Title="Some Title"><![CDATA[
    Some information formatted in HTML.
]]></IntroInstall>
```

Használhatja a `Format` attribútumot is annak meghatározásához, hogy `html` (amely alapértelmezett) vagy `plain` (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy `<pre></pre>` címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

**<IntroUninstall>** Egy eltávolítás előtti vagy utáni bevezető megjelenítéséhez az eltávolítási párbeszédablakban.

```
<IntroUninstall Type="post" Lang="en" Title="Some Title"><![CDATA[
    Some information formatted in HTML.
]]></IntroUninstall>
```

Használhatja a `Format` attribútumot is annak meghatározásához, hogy `html` (amely alapértelmezett) vagy `plain` (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy `<pre></pre>` címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

**<IntroReinstall>** Egy újratelepítés előtti vagy utáni bevezető megjelenítéséhez az újratelepítési párbeszédablakban.

```
<IntroReinstall Type="post" Lang="en" Title="Some Title"><![CDATA[
    Some information formatted in HTML.
]]></IntroReinstall>
```

Használhatja a `Format` attribútumot is annak meghatározásához, hogy `html` (amely alapértelmezett) vagy `plain` (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy `<pre></pre>` címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

**<IntroUpgrade>** Egy frissítés előtti vagy utáni bevezető megjelenítéséhez a frissítési párbeszédablakban.

```
<IntroUpgrade Type="post" Lang="en" Title="Some Title"><![CDATA[
    Some information formatted in HTML.
]]></IntroUpgrade>
```

Használhatja a `Format` attribútumot is annak meghatározásához, hogy `html` (amely alapértelmezett) vagy `plain` (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy `<pre></pre>` címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

**<CodeInstall>** A végrehajtandó Perl-kód, amikor a csomagot telepítik.

```
<CodeInstall><![CDATA[
# log example
$Kernel::OM->Get('Kernel::System::Log')->Log(
    Priority => 'notice',
    Message => "Some Message!",
);
# database example
$Kernel::OM->Get('Kernel::System::DB')->Do(SQL => "SOME SQL");
]]></CodeInstall>
```

Választhat `<CodeInstall Type="post">` vagy `<CodeInstall Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<CodeUninstall>** A végrehajtandó Perl-kód, amikor a csomagot eltávolítják. A csomag eltávolításának előtti vagy utáni idejében.

```
<CodeUninstall><![CDATA[
# Some Perl code.
]]></CodeUninstall>
```

Választhat `<CodeUninstall Type="post">` vagy `<CodeUninstall Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<CodeReinstall>** A végrehajtandó Perl-kód, amikor a csomagot újratelepítik.

```
<CodeReinstall><![CDATA[
# Some Perl code.
]]></CodeReinstall>
```

Választhat `<CodeReinstall Type="post">` vagy `<CodeReinstall Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<CodeUpgrade>** A végrehajtandó Perl-kód, amikor a csomagot frissítik (a `version` címkétől függően).

Például ha egy korábban telepített csomag verziója 1.3.4 alatt van (mondjuk 1.2.6), akkor végre lesz hajtva a meghatározott művelet:

```
<CodeUpgrade Version="1.3.4"><![CDATA[
# Some Perl code.
]]></CodeUpgrade>
```

Választhat `<CodeUpgrade Type="post">` vagy `<CodeUpgrade Type="pre">` típust is a végrehajtás idejének külön-külön történő meghatározásához (a `post` az alapértelmezett). További információkért nézze meg a *Csomagéletciklus* szakaszt.

**<PackageMerge>** Ez a címke jelzi, hogy egy csomag egyesítve lett egy másik csomaggal. Ebben az esetben az eredeti csomagot el kell távolítani a fájlrendszerről és a csomagok adatbázisából, de az összes adatot meg kell tartani.

Tegyük fel, hogy az `ElsoCsomag` egyesítve lett a `MasodikCsomag` nevű csomaggal. Ekkor a `MasodikCsomag.sopm` fájlnak ezt kell tartalmaznia:

```
<PackageMerge Name="MergeOne" TargetVersion="2.0.0"></PackageMerge>
```

Ha az `ElsoCsomag` is tartalmaz adatbázis-szerkezetet, akkor meg kell győződnünk arról, hogy az a csomag legfrissebb elérhető verziójánál volt, hogy következetes állapot legyen az adatbázisban a csomag egyesítése után. A `TargetVersion` attribútum csak ennyi csinál: jelzi az `ElsoCsomag` utolsó ismert verzióját abban az időpontban, amikor a `MasodikCsomag` létrejött. Ez főleg azért van, hogy leállítsa a frissítési folyamatot, ha a felhasználó rendszerén megtalálható az `ElsoCsomag` egy olyan verziója, amely *újabb* a `TargetVersion` attribútumban megadottnál, mivel ekkor ez problémákhoz vezethet.

Továbbá lehetőség van a szükséges adatbázis és kódfrissítési címkék hozzáadására az `ElsoCsomag` nevű csomagnál annak biztosításához, hogy az megfelelően kerül frissítésre a `TargetVersion` verzióra az egyesítés *előtt* - a következetlenségi problémák elkerüléséhez. Itt látható, hogy ennek hogyan kellene kinéznie:

```
<PackageMerge Name="MergeOne" TargetVersion="2.0.0">
    <DatabaseUpgrade Type="merge">
        <TableCreate Name="merge_package">
            <Column Name="id" Required="true" PrimaryKey="true"␣
↪AutoIncrement="true" Type="INTEGER"/>
            <Column Name="description" Required="true" Size="200" Type=
↪"VARCHAR"/>
        </TableCreate>
    </DatabaseUpgrade>
</PackageMerge>
```

Amint láthatja, ebben az esetben a `Type="merge"` attribútumot kell beállítani. Ezek a szakaszok csak akkor lesznek végrehajtva, ha lehetséges egy csomagegyesítés.

**Csomagfeltételek** Az `IfPackage` and `IfNotPackage``attribútumok hozzáadhatók a szabályos ``Database*` és `Code*` szakaszokhoz. Ha ezek jelen vannak, akkor a szakasz csak akkor lesz végrehajtva, ha egy másik csomag létezik vagy nem létezik a helyi csomagtárolóban.

```
<DatabaseInstall IfPackage="AnyPackage">
    # ...
</DatabaseInstall>
```

vagy

```
<CodeUpgrade IfNotPackage="OtherPackage">
    # ...
</CodeUpgrade>
```

Ezek az attribútumok beállíthatók a `PackageMerge` címkéken belüli `Database*` és `Code*` szakaszokban is.

### 4.2.2 Példa .sopm

Ez egy példa specifikációs fájl kinézete a fenti címkék egy részével.

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_package version="1.0">
    <Name>Calendar</Name>
    <Version>0.0.1</Version>
    <Framework>7.0.x</Framework>
    <Vendor>OTRS AG</Vendor>
    <URL>https://otrs.com/</URL>
    <License>GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007</License>
    <ChangeLog Version="1.1.2" Date="2018-11-15 18:45:21">Added some feature.
↪</ChangeLog>
    <ChangeLog Version="1.1.1" Date="2018-11-15 16:17:51">New package.</
↪ChangeLog>
    <Description Lang="en">A web calendar.</Description>
    <Description Lang="de">Ein Web Kalender.</Description>
    <IntroInstall Type="post" Lang="en" Title="Thank you!">Thank you for␣
↪choosing the Calendar module.</IntroInstall>
    <IntroInstall Type="post" Lang="de" Title="Vielen Dank!">Vielen Dank fuer␣
↪die Auswahl des Kalender Modules.</IntroInstall>
    <BuildDate>?</BuildDate>
    <BuildHost>?</BuildHost>
    <Filelist>
        <File Permission="644" Location="Kernel/Config/Files/Calendar.pm"></
↪File>
        <File Permission="644" Location="Kernel/System/CalendarEvent.pm"></
↪File>
        <File Permission="644" Location="Kernel/Modules/AgentCalendar.pm"></
↪File>
        <File Permission="644" Location="Kernel/Language/de_AgentCalendar.pm">
↪</File>
        <File Permission="644" Location="Kernel/Output/HTML/Standard/
↪AgentCalendar.tt"></File>
        <File Permission="644" Location="Kernel/Output/HTML/
↪NotificationCalendar.pm"></File>
        <File Permission="644" Location="var/httpd/htdocs/images/Standard/
↪calendar.png"></File>
    </Filelist>
    <DatabaseInstall>
        <TableCreate Name="calendar_event">
            <Column Name="id" Required="true" PrimaryKey="true" AutoIncrement=
↪"true" Type="BIGINT"/>
            <Column Name="title" Required="true" Size="250" Type="VARCHAR"/>
            <Column Name="content" Required="false" Size="250" Type="VARCHAR"/
↪>
            <Column Name="start_time" Required="true" Type="DATE"/>
            <Column Name="end_time" Required="true" Type="DATE"/>
            <Column Name="owner_id" Required="true" Type="INTEGER"/>
            <Column Name="event_status" Required="true" Size="50" Type=
↪"VARCHAR"/>
        </TableCreate>
```

(continues on next page)

(folytatás az előző oldalról)

```
    </DatabaseInstall>
    <DatabaseUninstall>
        <TableDrop Name="calendar_event"/>
    </DatabaseUninstall>
</otrs_package>
```

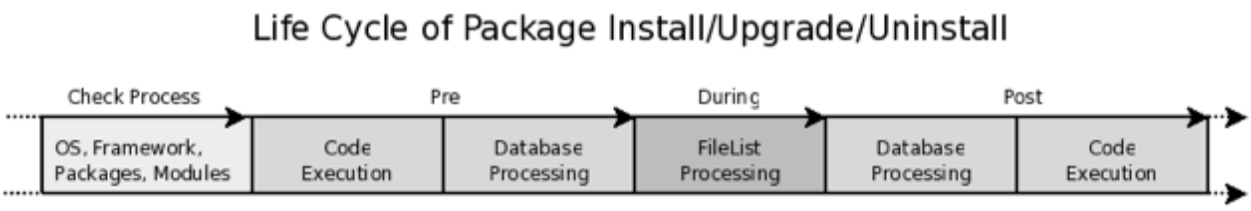
### 4.2.3 Csomagösszeállítás

Egy .opm csomag összeállításához a specifikációs opm fájlból.

```
shell> bin/otrs.Console.pl Dev::Package::Build /path/to/example.sopm /tmp
Building package...
Done.
shell>
```

### 4.2.4 Csomagéletciklus

A következő kép azt mutatja be lépésről lépésre, hogy egy csomag telepítési, frissítési vagy eltávolítási életciklusa hogyan működik a háttérprogramban.




1. ábra: Csomagéletciklus

## 4.3 Csomagátírás

Az OTRS minden új hibajavító vagy fő verziójával át kell írnia a csomagjait, és meg kell győződnie arról, hogy azok továbbra is működnek az OTRS API-val.

Ez a szakasz azokat a változtatásokat sorolja fel, amelyeket meg kell vizsgálnia, amikor átírja a csomagját az OTRS 6-ról 7-re.

### 4.3.1 A munkamenetek mindig igénylik a sütiket

Az OTRS 7-től kezdve a munkameneteknek mindig szükségük van az engedélyezett sütikre. Ennélfogva a `SessionIDCookie` értéke el lett távolítva a `LayoutObject`, *Sablonozó mechanizmus* és JavaScript kódokból. Többé nincs szükség a munkamenet-változók hozzáfűzésére az URL-eknél vagy a HTTP-kérések hasznos terhében.

### 4.3.2 A `groups` tábla átnevezésre került

A MySQL 8-ban végzett változtatás miatt a `groups` táblát át kellett nevezni `groups_table` nevűre. Ha használja ezt a táblát közvetlenül valamilyen SQL-lekérdezésekben, akkor azokat hozzá kell igazítani. További információkért nézze meg a 13866-os hibajegyet.

### 4.3.3 Új külső felület

A meglévő ügyfél (`customer.pl`) és nyilvános (`public.pl`) felületeket leváltotta egy új REST háttérprogram (`Kernel/WebApp`) és egy modern Vue.js alapú előtétprogram alkalmazás. Ez azt jelenti, hogy az összes ehhez kapcsolódó kódot át kell írni és/vagy újra kell írni.

Van egy különleges eset a nyilvános előtétprogram-moduloknál, amely nem szolgál ki HTML alkalmazást. Ezt viszonylag könnyen át lehet írni az új REST háttérprogramra (nézze meg a REST API dokumentációját). Nézze meg például a `Kernel/WebApp/Controller/API/Public/Package/Repository.pm` fájlt. Ez a példa azt is bemutatja, hogy a végpontok hogyan támogathatják mindkét új REST-szerű URL-t, de ugyanakkor az örökölt `/otrs/customer.pl?Action=MyAction` alapú URL útvonalakat is.

Az ügyfélfelületen lévő néhány fontos URL-hez, amelyek örökölt rendszerekből vannak hivatkozva, átirányítási vezérlőket kellene létrehozni annak biztosításához, hogy a régi URL-ek továbbra is működjenek.

### 4.3.4 Változtatások a folyamatkezelésben

A `scripts/DBUpdate-to-7.pl` költöztető parancsfájl frissíteni fogja az összes folyamatot, amelyek már az adatbázisban vannak. Kézi művelet csak akkor szükséges, ha használni szeretne valamilyen új funkciót, amelyet az OTRS 7 nyújt.

#### Új tevékenységtípusok

Mivel az OTRS 7 több tevékenységtípust képes kezelni, az összes meglévő tevékenység mostantól *felhasználói feladat* tevékenységgé vált. Egy feladatmeghatározás frissítéséhez a YAML-fájlban adja hozzá a `Type: UserTask` bejegyzést ugyanolyan behúzással, mint amilyen a `Name` vagy `ID` sornak van.

#### Átnevezett folyamatkezelés összetevők

**`Transition` átnevezve erre: `SequenceFlow`** Ezt a folyamatösszetevőt azért nevezték át, hogy jobban igazodjon a BPMN-hez. A fájlokat, osztályokat és metódusokat is eszerint nevezték át. A személyre szabott fájlokat frissíteni kell az új egyezmény követéséhez.

**`TransitionAction` átnevezve erre: `SequenceFlowAction`** Ezt a folyamatösszetevő nem létezik a BPMN-ben, de ezt is át kell nevezni, hogy következetes legyen az egyéb változtatásokkal. A fájlokat, osztályokat és metódusokat is eszerint nevezték át. A személyre szabott fájlokat frissíteni kell az új egyezmény követéséhez.

**`TransitionActionModules` átnevezve erre: `ProcessManagementModules`** Ezeket a folyamatösszetevőket nem csak a *szekvenciafolyam-műveletek* használják, hanem a *vezérelt feladatok* tevékenységei is, és át lettek helyezve a `Kernel/System/ProcessManagement/TransitionAction` mappából a `Kernel/System/ProcessManagement/Modules` mappába.

Az új folyamatkezelés-modulok több mezőtípust és lehetőséget nyújtanak a paraméterek megjelenítéshez a folyamattervező számára. Kövesse a *Folyamatkezelés* dokumentációban leírt utasításokat, hogy többet tudjon meg erről az új funkcióról, és hogy a meglévő modulokat hogyan lehet továbbfejleszteni.

Az összes szállított folyamatmeghatározást frissíteni kell az új névséma használatához.

- `Transition` cseréje erre: `SequenceFlow`.
- `Transitions` cseréje erre: `SequenceFlows`.
- `TransitionAction` cseréje erre: `SequenceFlowAction`.
- `TransitionActions` cseréje erre: `SequenceFlowActions`.
- A `Kernel::System::ProcessManagement::TransitionAction` eltávolítása a `Module:` szakaszból az összes `SequenceFlowAction` esetén. Például a `Module: Kernel::System::ProcessManagement::TransitionAction::TicketLockSet` a következő legyen: `Module: TicketLockSet`.

### 4.3.5 Változtatások a `LayoutObject` objektumban

Változtatások történtek a `Kernel/Output/HTML/Layout.pm` fájlban, amelyek ahhoz szükségesek, hogy a tartalom megfelelően jelenjen meg a Mojolicious valós idejű webes keretrendszer használatával.

#### Nem megjelenített vagy üres táblázatok a képernyőkön

Győződjön meg arról, hogy ellenőrzött-e minden olyan képernyőt, amely táblázatszerű kimenetet állít elő (például `Kernel/Modules/AgentTicketStatusView.pm`). Ha például a jegyek listája üres vagy akár egyáltalán nem jelenik meg, ellenőrizze, hogy az `Output => 1` paraméter használatban van-e az oldal kimenetének létrehozásakor.

#### Kódolási problémák az örökölt előtétprogram-modulokban

Ha problémája lenne a sérült karakterekkel, például az umlautokkal vagy ékezetekkel, akkor előfordulhat, hogy a megjelenítendő tartalmat a `Print()` metódus jeleníti meg. Ennek javításához váltson át a kódban a `Print()` metódus használatáról az előtétprogram-modulból érkező teljes válasz visszaadásának normál módszerére.

FEJEZET 5

# Dokumentáció

Az OTRS 7-től kezdve a dokumentációk *reStructuredText* formátumban érhetők el, amely a régi *DocBook* formátumot cseréli le. Különféle kimenetek érhetők el az OTRS dokumentációs oldalán, mint például HTML, EPUB és PDF.

A dokumentációt angol nyelven írták, és több nyelvre is le van fordítva.

## 5.1 Dokumentációs infrastruktúra

Az OTRS a Sphinx programot használja a kimenetek előállításához. A HTML kimenet a Read the Docs téma használatával kerül előállításra.

**Megjegyzés:** Az összes kimenet személyre van szabva az összeállító kiszolgálón. Ha a helyi gépén szeretne egy fejlesztői környezetet felállítani a teszteléshez és a dokumentáció írásához, akkor a kimenetek eltérőek lehetnek.

## 5.2 reStructuredText alapozó

A dokumentációformátum neve **reStructuredText** (egybe írva, ez a helyes írásmód). Ez egy könnyen olvasható dokumentumformátum, amely egyszerű szöveget és kisebb sorközi jelölőket használ.

Ez a rövid ismertető végigvezeti Önt a dokumentációk létrehozásán és frissítésén. A *reStructuredText* formátum használatáról szóló teljes körű ismertető biztosítása túlmutat ezen dokumentáció hatókörén, de számos ismertető (például a Sphinx reStructuredText alapozó és a reStructuredText felhasználói dokumentáció) és internetes szerkesztő érhető el az interneten.

A következő példák a leggyakrabban használt dokumentációs elemeket mutatják be.

### 5.2.1 Címsorok

Címsorok használatához a dokumentációban alá kell húznia a címeket különleges karakterekkel. Az aláhúzásnak a cím első karakterétől kell kezdődnie és a cím utolsó karakterénél kell végződnie. A különleges karakterek hierarchiája a következő: =, -, ~, ^, .

A következő példa a címsorok használatát mutatja be:

```
Chapter title
=============

This is the heading 1 title. It has numbering like 1.

Section title
-------------

This is the heading 2 title. It has numbering like 1.1.

Subsection title
~~~~~~~~~~~~~~~~

This is the heading 3 title. It has numbering like 1.1.1.

Subsubsection title
^^^^^^^^^^^^^^^^^^^

This is the heading 4 title. It has numbering like 1.1.1.1.

Subsubsubsection title
......................

This is the heading 5 title. It has numbering like 1.1.1.1.1. Please don't␣
↪use this level of heading.
```

### 5.2.2 Bekezdések

Bekezdések írásához a mondatokat a sor elején kell kezdenie. Új bekezdés létrehozásához egyszerűen hagyjon egy üres sort a bekezdések között. Példa:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed dictum imperdiet␣
↪enim. Curabitur
nisi diam, lobortis facilisis quam ut, porttitor consequat lectus. Nam␣
↪elementum, ipsum id
feugiat vestibulum, dolor ante dictum quam, ac bibendum ipsum felis in orci.

Vestibulum maximus egestas orci, eget consequat nibh imperdiet eget.␣
↪Suspendisse sagittis tempus
sapien, sit amet tincidunt tortor efficitur et. Etiam ac lacus sem. Sed ut␣
↪magna imperdiet,
viverra quam vitae, consequat mauris.
```

### 5.2.3 Sorközi jelölők

A szabványos sorközi jelölők viszonylag egyszerűek:

- Egy csillag: *szöveg* a *hangsúlyozáshoz* (döntött).
- Két csillag: **szöveg** az **erős hangsúlyozáshoz** (vastagított).
- Hangsúlyjelek: ‘‘szöveg‘‘ a `szó szerinti szövegekhez` (kódpéldák).

Ha csillagok vagy hangsúlyjelek fordulnak elő a futó szövegben, amelyek összekeverhetők a sorközi jelölők határolóival, akkor el kell fedni azokat egy visszafelé álló perjellel, mint például *.

### 5.2.4 Listák

Felsorolások létrehozásához kezdje a sort egy csillaggal (*) vagy kötőjellel (-). Számozott listák létrehozásához kezdje a sort számokkal vagy kettős kereszttel (#). Ha egymásba ágyazott listákra van szüksége, akkor hagyjon egy üres sort a listaelemek között, és használjon 3 szóköznyi behúzást. Példa:

```
* This is a bulleted list.
* It has two items, the second
  item uses two lines.

1. This is a numbered list.
2. It has two items too.

#. This is a numbered list.
#. It has two items too.
```

Példa egymásba ágyazott listára:

```
- this is
- a list

  - with a nested list
  - and some subitems

- and here the parent list continues
```

### 5.2.5 Szó szerinti blokkok

A szó szerinti blokkok olyan szövegek, amelyeket betűhelyesen kell megjeleníteni. Szó szerinti blokkok létrehozásához a következőt tegye:

1. Írjon be 2 kettőspontot (::) egy új sorba.
2. Hagyjon egy üres sort.
3. Írja be a szöveget 3 szóköznyi behúzással.

Használjon szó szerinti blokkokat kódrészletekhez, terminálkimenetekhez, beállítófájlokhoz, stb. Példa:

```
::

   $Self->{DatabaseHost} = '127.0.0.1';
```

(continues on next page)

(folytatás az előző oldalról)

```
   $Self->{Database} = 'otrs';
   $Self->{DatabaseUser} = 'otrs';
```

Ha ismert a kódrészlet nyelve, akkor megadhatja azt a szintaxis kiemeléséhez:

```
.. code-block:: perl

   $Self->{DatabaseHost} = '127.0.0.1';
   $Self->{Database} = 'otrs';
   $Self->{DatabaseUser} = 'otrs';
```

```
.. code-block:: xml

   <Setting Name="FAQ::Agent::StateTypes" Required="1" Valid="1">
       <Description Translatable="1">List of state types which can be used in␣
↪the agent interface.</Description>
       <Navigation>Core::FAQ</Navigation>
       <Value>
           <Array>
               <Item>internal</Item>
               <Item>external</Item>
               <Item>public</Item>
           </Array>
       </Value>
   </Setting>
```

### 5.2.6 Táblázatok

Rácsos táblázatok létrehozásához meg kell rajzolnia a táblázatot. Példa:

```
+------------------------+------------+----------+----------+
| Header row, column 1   | Header 2   | Header 3 | Header 4 |
| (header rows optional) |            |          |          |
+========================+============+==========+==========+
| body row 1, column 1   | column 2   | column 3 | column 4 |
+------------------------+------------+----------+----------+
| body row 2             | ...        | ...      |          |
+------------------------+------------+----------+----------+
```

### 5.2.7 Hiperhivatkozások

A hiperhivatkozások lehetnek sorköziek vagy hivatkozottak. Sorközi használathoz fogja közre a hivatkozás szövegét és az URL-t hangsúlyjelekkel, és zárja két aláhúzás karakterrel.

```
Visit `OTRS website <https://otrs.com>`__ for more information.
```

A fenti hivatkozás így lesz megjelenítve: OTRS website.

Hivatkozott hivatkozások létrehozásához külön kell választani a szöveget és a hivatkozást. Példa:

```
The documentations are available in the `OTRS documentation portal`_.

.. _OTRS documentation portal: https://doc.otrs.com/
```

### 5.2.8 Képek

A dokumentációba egy kép beszúráshoz:

1. Tegye a képet az `images` mappába.
2. Hozzon létre egy hivatkozást a képre ezzel:

```
.. figure:: images/admin-general-catalog-management-class.png
   :alt: Admin General Catalog

   Admin General Catalog
```

### 5.2.9 Színes dobozok

Ezeknek a dobozoknak különleges jelentésük van, és alapértelmezetten kiemelésre kerülnek.

Figyelmeztetés doboz:

```
.. warning::

   This is a warning box.
```

**Figyelem:** Ez egy figyelmeztetés doboz.

Megjegyzés doboz:

```
.. note::

   This is a note box.
```

**Megjegyzés:** Ez egy megjegyzés doboz.

Lásd még doboz:

```
.. seealso::

   This is a see also box.
```

**Lásd még:**

Ez egy lásd még doboz.

## 5.3 Stílus irányelvek

A dokumentáció ezen része csak a látható stílushoz és a fogalmazáshoz van.

### 5.3.1 Tartalom írása

Létezik egy **TL;DR** internetes szleng, amely azt jelenti, hogy *túl hosszú, nem olvastam el* (további információért nézze meg a Wikipédiát). Sok ember nem szeret hosszú szövegeket olvasni, ezért arra kérjük, hogy tartsa a dokumentációt olyan röviden, amennyire csak lehetséges. Használjon lépésről lépésre bemutató ismertetőt a tömör szöveg írása helyett.

Például ez egy **rossz** példa a tartalom írására:

```
The agents are able to change the interface language of OTRS. To change the
interface language, click on your avatar on the top left corner, then select
Personal Settings menu item. A new screen will be displayed. On this screen
click on the User Profile, and then find a widget named Language. Select the
desired language in the drop-down menu. Please make sure to click on the
Save button next to the language widget.
```

Ugyanez a tartalom az *ember által is olvasható* **ajánlott** formátumban:

```
To change the interface language of OTRS:

1. Click on your avatar on the top left corner.
2. Select *Personal Settings*.
3. Click the *User Profile* in the new screen.
4. Choose a language from the drop-down menu of the *Language* widget.
5. Click the *Save* button next to the widget.
```

Az utóbbit egyszerűbb lefordítani, mert 6 rövid mondat kerül felvételre a nyelvi fájlba. Ha a tartalom megváltozik a mondatok egyikében, akkor csak a megváltozott mondatot kell átnézni és újra lefordítani. Az első rossz példa csak egyetlen hatalmas karakterláncot tesz a nyelvi fájlba, és ha valamilyen változtatást végeznek a forrásszövegen, akkor a fordítónak a teljes szöveget át kell néznie és újrafordítania.

### 5.3.2 Képernyőképek

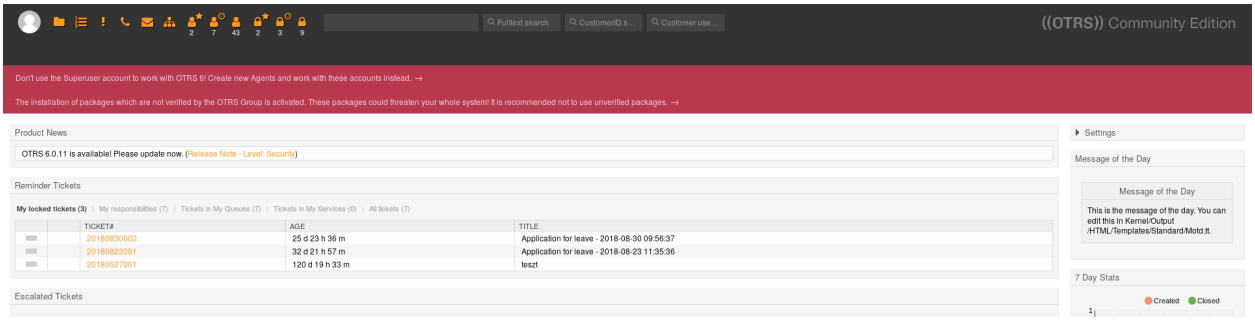
Ne használja a számítógépe natív felbontását. Általában az teljes HD vagy nagyobb, így az ezzel a felbontással készített képernyőkép olvashatatlanná válik egyes kimeneteken, mert egy PDF esetén az összes képet össze kell zsugorítani az A4-es papír szélességére. Az OTRS alkalmazkodó megjelenítést használ, így 1025 képpont a minimum, amelyet az OTRS nagy kijelzőnek feltételez. Használja ezt a képernyőképek szélességeként.

**Lásd még:**

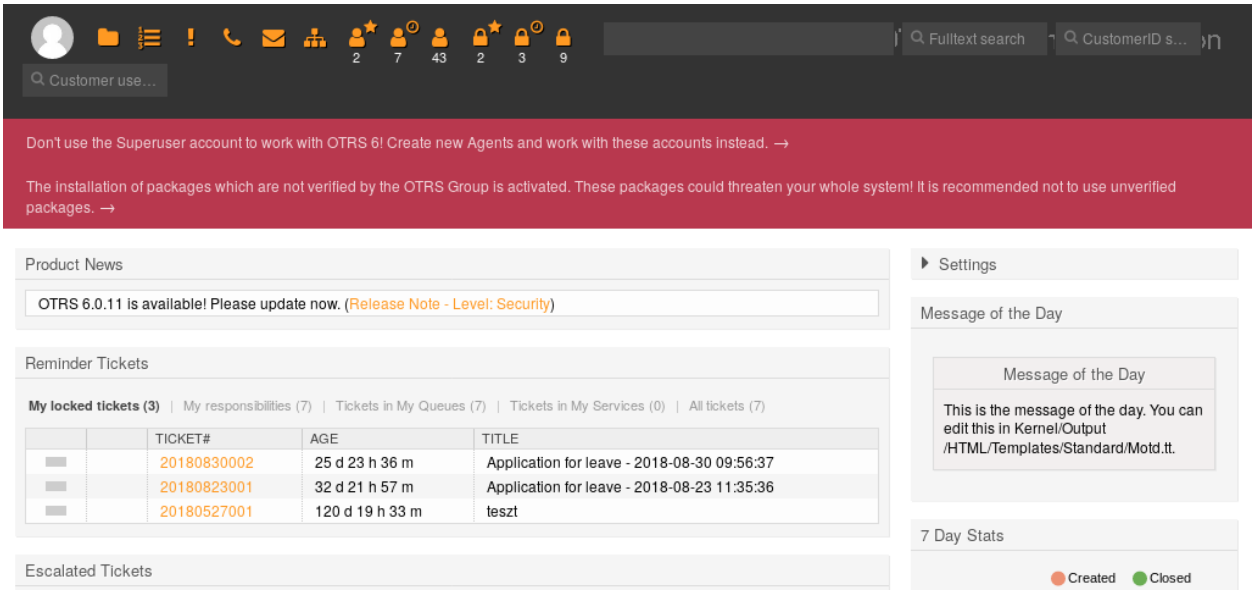
A felbontást az összes webböngészőben be lehet állítani egy úgynevezett *mobil mód* vagy *alkalmazkodó megjelenés* funkcióval. Nézze meg a böngészője felhasználói kézikönyvét a funkció használatához, és állítsa a képernyő szélességét 1025 képpontra.

Ez egy példa egy **rossz** képernyőképre, mivel az teljes HD felbontású. Az automatikus zsugorítás miatt a szövegeket nehéz elolvasni a képernyőképen:

Ugyanaz a képernyőkép az **ajánlott** felbontásban. A szövegeket sokkal könnyebb elolvasni:

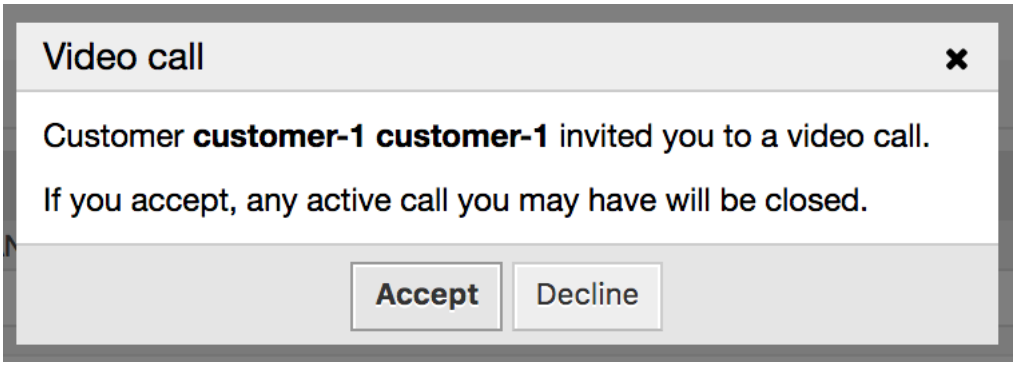



1. ábra: Ügyintézői vezérlőpult (1920 képpont szélességben)




2. ábra: Ügyintézői vezérlőpult (1025 képpont szélességben)

Az szintén rossz, ha a képernyőkép jó képpontfelbontással rendelkezik, de magas DPI értékkel. Például ez a képernyőkép **rossz**, mert a rajta lévő szöveg sokkal nagyobb a dokumentációban lévő szövegnél:

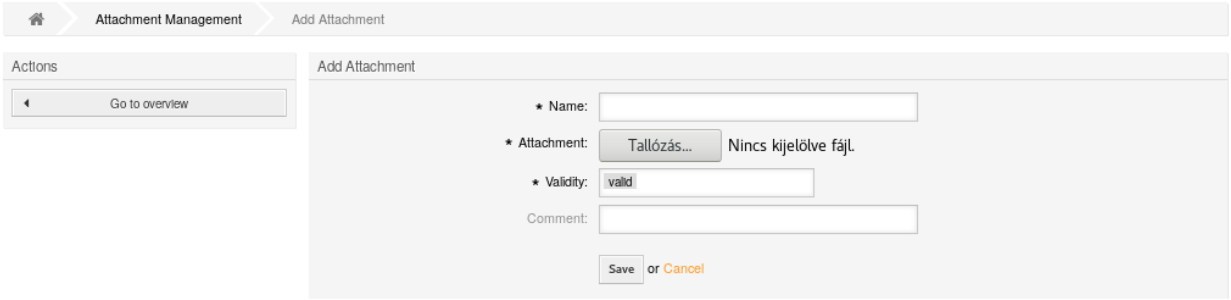



3. ábra: Videomeghívási párbeszédablak (756 képpont szélességben, de magas DPI értékkel)

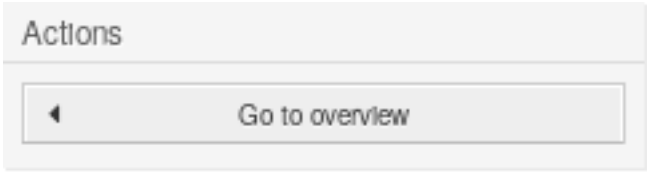
### Képernyőképek készítése Firefox használatával

Ha csak a képernyőkép egy része szükséges, akkor a képernyőképet meg kell vágni. Az OTRS adminisztrátori felülete egy bal oldalsávból és egy fő tartalmi oszlopból áll. A Firefox használatával történő képernyőkép készítéséhez:

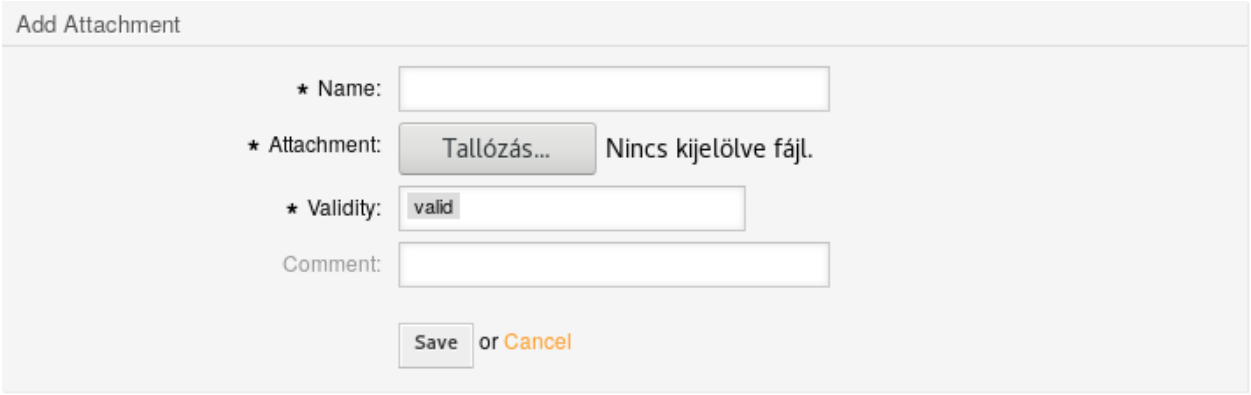
1. Kattintson a jobb egérgombbal egy elemre a böngészőben, és válassza az *Elem vizsgálata* lehetőséget.
2. Válassza ki az elemet a DOM-ban, ha nem lett kiválasztva.
3. Kattintson a jobb egérgombbal a csomópontra, és válassza a *Képernyőkép a csomópontról* lehetőséget.




4. ábra: Példa képernyőkép a fő tartalomról




5. ábra: Példa képernyőkép a bal oldalsávról




6. ábra: Példa képernyőkép a fő tartalom oszlopáról

### 5.3.3 Nagybetűs írás a dokumentációban

A címeknél mindig minden szót nagybetűvel kezdjen, ami azt jelenti, hogy a címekben mindig nagybetűvel kezdődnek a következők:

- Főnevek (ember, busz, könyv).
- Melléknevek (mérges, kedves, kicsi).
- Igék (fut, eszik, alszik).
- Határozószavak (lassan, gyorsan, csendben).
- Névmások (ő, mi, az).
- Alárendelt kötőszavak (mivel, mert, amely).

A címekben ne kezdje nagybetűvel a következőket:

- Névelők: a, az, egy.
- Mellérendelt kötőszavak: és, de, vagy, stb.
- Elöljárószavak (öt betűnél rövidebbek): rá, oda, tőle, stb.

A normál mondatokban ne kezdjen nagybetűvel semmilyen szót, csak a neveket és a címekre hivatkozásokat kell nagybetűvel kezdeni. Ez egy **rossz** példa:

```
An Agent is a user, who handles Tickets in the Ticket Zoom screen.
```

Az **ajánlott** mondat megfelelő nagybetűs írással. Kivéve a *Ticket Zoom* nevet, ami a képernyő neve, ezért ki kell hangsúlyozni:

```
An agent is a user, who handles tickets in the *Ticket Zoom* screen.
```

### 5.3.4 Gombok és képernyőnevek

A tartalom mondataiban az összes gombot és képernyőt ki kell hangsúlyozni, és nagy kezdőbetűkkel vagy minden szót nagy kezdőbetűvel kell írni. Ne használjon aposztrófot vagy idézőjelet a hangsúlyozáshoz.

Ez a mondat **rossz**, mert aposztrófokat használnak a hangsúlyozáshoz:

```
If you click the 'Save and Finish' button, you will be redirected to the
↪'Ticket Zoom' screen.
```

Az **ajánlott** mód a csillagok használata a hangsúlyozáshoz:

```
If you click the *Save and Finish* button, you will be redirected to the␣
↪*Ticket Zoom* screen.
```

### 5.3.5 Fogalmazás

Ne használjon változóneveket a mondatokban. Ez a mondat **rossz**, mert egy változónév néhány ember számára értelmetlen:

```
Add a new widget to AgentTicketZoom.
```

Ugyanaz a mondat változónév nélkül, ez az **ajánlott**:

```
Add a new widget to the *Ticket Zoom* screen of the agent interface.
```

### 5.3.6 Változónevek

A változóneveket mindig `szó szerinti` tartalomként kell megjelölni. Ez hasznos a fordítók számára, mivel így pontosan tudják, hogy a karakterláncot nem szabad lefordítani. Ha egy karakterlánc nem szó szerinti szövegként van megjelölve, akkor általában le kell fordítani. Például:

```
The ``ObjectManager`` object has an ``Init()`` function. Additional␣
↪configuration can be set in ``Kernel::Config::Config.pm`` file.
```

## 5.4 A dokumentáció fordítása

Az OTRS grafikus felhasználói felületének, a nyilvános kiterjesztőmoduloknak és a dokumentációknak az összes fordítását Weblate használatával kezelik.

Az OTRS 7-től kezdve a dokumentációk *reStructuredText* formátumban érhetők el, amely a régi *DocBook* formátumot cseréli le. Legyen óvatos, nehogy elrontsa a szerkezetet a dokumentum fordítása során.

Itt van néhány példa.

**Hangsúlyozás** A hangsúlyos szövegek két csillag közt találhatók. A szöveget le kell fordítani. Ezt általában képernyőneveknél, címeknél, gomboknál és címkéknél használják. Nézze meg a felhasználói felület fordításait, hogy megtalálja és ugyanazt a megfogalmazást használja a dokumentációban is.

Példa az eredeti szövegre:

```
Use the *Ticket Zoom* screen to see the ticket details.
```

Példa a magyarra történő fordításra:

```
Használja a *Jegynagyítás* képernyőt a jegy részleteinek megtekintéséhez.
```

**Hangsúlyozás** A hangsúlyos szövegek két dupla csillag között vannak. A szöveget le kell fordítani. Ezt általában fontos információknál használják.

Példa az eredeti szövegre:

```
**Don't continue** the update if you get an error message.
```

Példa a magyarra történő fordításra:

```
**Ne folytassa** a frissítést, ha hibaüzenetet kap.
```

**Szó szerinti szövegek** A szó szerinti szövegek két dupla hangsúlyjel között vannak. Ezt általában változóneveknél, beállítások neveinél és fájlok útvonalainál használják. **Tilos** lefordítani, különben tönkre fogja tenni a szerkezetet.

Példa az eredeti szövegre:

```
Activate ``group`` in system configuration ``ExamplePermission###100``.
```

Példa a magyarra történő fordításra:

```
Aktiválja a ``group`` értéket az ``ExamplePermission###100``␣
↪rendszerbeállításban.
```

**Belső hivatkozások** A belső hivatkozások más oldalakra vagy oldalak címsoraira mutatnak. A `:doc:`page-name`` használható egy oldalra történő hivatkozáshoz és a `:ref:`Heading Title`` használható egy címsorra való hivatkozáshoz. Létezik egy egyéni `:sysconfig:`System Configuration Name`` címke egy rendszerbeállításra való hivatkozáshoz. A *page-name*, *Heading Title* és *System Configuration Name* szövegeket **tilos** lefordítani, különben tönkre fogják tenni a szerkezetet.

Példa az eredeti szövegre:

```
See page :doc:`queues` to add a queue, especially section :ref:`Queue␣
↪Settings`.
```

Példa a magyarra történő fordításra:

```
Nézze meg a :doc:`queues` oldalt, különösen a :ref:`Queue Settings`␣
↪szakaszt.
```

**Külső hivatkozások** A külső hivatkozások egy látható szövegből és egy URL-ből állnak *'látható szöveg <https://example.com>'__* formában. A *látható szöveget* le kell fordítani.

Példa az eredeti szövegre:

```
See `OTRS website <https://otrs.com/>`__ for more information.
```

Példa a magyarra történő fordításra:

```
Nézze meg az `OTRS weboldalát <https://otrs.com/>`__ a további␣
↪információkért.
```

FEJEZET 6

# Közreműködés az OTRS-ben

Ez a fejezet azt fogja bemutatni, hogy hogyan működhet közre az OTRS keretrendszerben azért, hogy a többi felhasználó is képes legyen hasznot húzni a munkájából.

## 6.1 Hozzájárulások küldése

Az ((OTRS)) Community Edition és a további nyilvános modul forráskódja megtalálható a GitHubon. Innen eljuthat az összes elérhető tároló felsorolásához. Leírja a jelenleg aktív ágakat is, és hogy hova kell kerülniük a hozzájárulásoknak (stabil vagy fejlesztői ágak).

Erősen ajánlott a *Hasznos eszközök* fejezetben bemutatott OTRSCodePolicy OTRS kódminőség-ellenőrző használata, még mielőtt elküldené a hozzájárulásait. Ha a kódját nem érvényesíti ezzel az eszközzel, akkor valószínűleg nem fogják elfogadni.

A legegyszerűbb módja a hozzájárulások elküldésének az OTRS fejlesztőcsapata számára egy *pull request* létrehozása a GitHubon. Vessen egy pillantást a GitHubon lévő utasításokra, különösen a tároló elágaztatására és a beolvasztási kérések küldésére.

Az alap munkafolyamatnak így kellene kinéznie:

1. Regisztráljon a GitHubon, ha még nincs fiókja.
2. Ágaztassa el azt a tárolót, amelynél közre szeretne működni, és váltson át arra az ágra, amelybe a változtatásokat be kell tenni.
3. Hozzon létre egy új fejlesztői ágat a javításhoz/szolgáltatáshoz/hozzájáruláshoz az aktuális ág alapján.
4. A változtatások befejezése és a véglegesítésük után küldje be az ágat a GitHubra.
5. Hozzon létre egy beolvasztási kérést. Az OTRS fejlesztőcsapata értesülni fog erről, ellenőrizni fogják a beolvasztási kérését, és beolvasszák azt, vagy valamilyen visszajelzést adnak a lehetséges továbbfejlesztésekről.

Ez esetleg bonyolultan hangzik, de miután beállította ezt a munkafolyamatot, látni fogja, hogy a hozzájárulások elvégzése hihetetlenül egyszerű.

## 6.2 Fordítás

A fordításokat többnyire az OTRS felhasználói készítik el és tartják karban, ezért az *Ön* segítsége is szükséges.

Az OTRS grafikus felhasználói felületének, a nyilvános kiterjesztőmoduloknak és a dokumentációknak az összes fordítását Weblate használatával kezelik.

A fordításban való közreműködéshez:

1. Regisztráljon egy ingyenes fordítói fiókot a Weblate-en.
2. Válasszon egy fordítási összetevőt és egy nyelvet a fordításhoz.
3. Kezdje el frissíteni a fordítást. Nincs szükség további szoftverekre vagy fájlokra.

---

**Megjegyzés:** Ha a nyelve nincs felsorolva a vezérlőpulton, akkor kérelmezhet egy nyelvet. Az elfogadása után el is kezdheti a fordítást.

---

Az OTRS 7-től kezdve a dokumentációk *reStructuredText* formátumban érhetők el, amely a régi *DocBook* formátumot cseréli le. Legyen óvatos, nehogy elrontsa a szerkezetet a dokumentum fordítása során.

**Lásd még:**

A *Dokumentáció* fejezetben találhat néhány példát.

Az OTRS fejlesztői időről időre le fogják tölteni a fordításokat az OTRS forráskód tárolóiba, így nem kell semmit sem elküldenie sehova.

## 6.3 Kódolási stílus irányelvek

Az OTRS projekt következetes fejlesztésének megtartása érdekében irányelveket fektettünk le a stílusra vonatkozóan a különböző programnyelvekhez.

### 6.3.1 Perl

#### Üres karakterek

TABULÁTOR: 4 szóközt használunk. Példa a zárójelekre:

```
if ($Condition) {
    Foo();
}
else {
    Bar();
}

while ($Condition == 1) {
    Foo();
}
```

### A sorok hossza

A sorok általában nem lehetnek hosszabbak 120 karakternél, hacsak ez különleges okok miatt nem szükséges.

### Szóközök és zárójelek

A jobb olvashatóság érdekében szóközöket használunk a kulcsszavak és a nyitó zárójelek között.

```
if ()...
for ()...
```

Ha csak egy egyedülálló változó van, akkor a zárójelek belül szóközök nélkül veszik körbe a változót.

```
if ($Condition) { ... }

# instead of

if ( $Condition ) { ... }
```

Ha a feltétel nem csak egy egyedülálló változó, akkor szóközöket használunk a zárójelek és a feltétel között. És továbbra is szóköz van a kulcsszó (például `if`) és a nyitó zárójel között.

```
if ( $Condition && $ABC ) { ... }
```

Ne feledje, hogy a Perl beépített függvényeinél nem használunk zárójeleket:

```
chomp $Variable;
```

### Forráskód fejléc és karakterkódolás

Csatolja hozzá a következő fejlécet minden egyes forrásfájlhoz. A forrásfájlok UTF-8 karakterkódolással vannak elmentve.

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

A végrehajtható fájloknak (`*.pl`) különleges fejlécük van.

```
#!/usr/bin/perl
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
```

(continues on next page)

(folytatás az előző oldalról)

```
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see https://www.gnu.org/licenses/gpl-3.0.
↪txt.
# --
```

### Feltételek

A feltételek meglehetősen összetettek lehetnek, és lehetnek *láncolt* feltételek is (logikai *vagy* vagy *és* operátorral összekapcsolva). Az OTRS kódolásakor tisztában kell lennie számos helyzettel.

A bevált Perl gyakorlatok azt mondják, hogy a magas precedenciájú operátorokat (`&&` és `||`) nem kellene keverni az alacsony precedenciájú operátorokkal (`and` és `or`). A zűrzavar elkerülése érdekében mindig a magas precedenciájú operátorokat használjuk.

```
if ( $Condition1 && $Condition2 ) { ... }

# instead of

if ( $Condition and $Condition2 ) { ... }
```

Ez azt jelenti, hogy tisztában kell lennie a buktatókkal. Néha zárójeleket kell használnia, hogy világossá tegye, mit szeretne.

Ha hosszú feltételei vannak (a sor 120 karakternél hosszabb), akkor több sorra kell tördelnie azt. Továbbá a feltételek kezdete egy új sorban van (nem az `if` sorában).

```
if (
    $Condition1
    && $Condition2
    )
{ ... }

# instead of

if ( $Condition1
    && $Condition2
    )
{ ... }
```

Jegyezze meg azt is, hogy a jobb oldali zárójel egyedül áll a sorban, valamint a bal oldali kapcsos zárójel szintén új sorban van, és ugyanolyan behúzással rendelkezik mint az `if`. Az operátorok egy új sor elején vannak! A következő példák bemutatják, hogyan kell ezt csinálni.

```
if (
    $XMLHash[0]->{otrs_stats}[1]{StatType}[1]{Content}
    && $XMLHash[0]->{otrs_stats}[1]{StatType}[1]{Content} eq 'static'
```

(continues on next page)

(folytatás az előző oldalról)

```
    )
{ ... }

if ( $TemplateName eq 'AgentTicketCustomer' ) {
    ...
}

if (
    ( $Param{Section} eq 'Xaxis' || $Param{Section} eq 'All' )
    && $StatData{StatType} eq 'dynamic'
    )
{ ... }

if (
    $Self->{TimeObject}->TimeStamp2SystemTime( String => $Cell->{TimeStop} )
    > $Self->{TimeObject}->TimeStamp2SystemTime(
        String => $ValueSeries{$Row}{$TimeStop}
    )
    || $Self->{TimeObject}->TimeStamp2SystemTime( String => $Cell->{TimeStart}
↪ )
    < $Self->{TimeObject}->TimeStamp2SystemTime(
        String => $ValueSeries{$Row}{$TimeStart}
    )
    )
{ ... }
```

#### Hátul álló `if`

Általánosan azért használunk *hátul álló "if"* utasításokat, hogy csökkentsük a szintek számát. De ne használjuk többsoros utasításoknál, és csak akkor megengedett, amikor visszatérési utasításokat hoz magával a függvény, vagy egy ciklus befejezéséhez, illetve a következő iterációra való ugráshoz.

Ez helyes:

```
next ITEM if !$ItemId;
```

Ez hibás:

```
return $Self->{LogObject}->Log(
    Priority => 'error',
    Message  => 'ItemID needed!',
) if !$ItemId;
```

Ez kevésbé karbantartható ennél:

```
if( !$ItemId ) {
    $Self->{LogObject}->Log( ... );
    return;
}
```

Ez helyes:

```
for my $Needed ( 1 .. 10 ) {
    next if $Needed == 5;
    last  if $Needed == 9;
}
```

Ez hibás:

```
my $Var = 1 if $Something == 'Yes';
```

### Néhány beépített Perl szubrutin használatának korlátozása

A Perl néhány beépített szubrutinját nem lehet használni semmilyen helyen:

- Ne használja a `die` és `exit` szubrutinokat a `.pm` fájlokban.
- Ne használja a `Dumper` függvényt a kiadott fájlokban.
- Ne használja a `print` utasítást a `.pm` fájlokban.
- Ne használja a `require` kulcsszót, inkább használja a `Main::Require()` metódust.
- Használja a `DateTimeObject` függvényeit az olyan beépített függvények helyett, mint például a `time()`, `localtime()`, stb.

### Reguláris kifejezések

A reguláris kifejezéseknél *a forráskódban* mindig kapcsos zárójelekkel használjuk az `m//` operátort elválasztóként. Használjuk az `x`, `m` és `s` módosítókat is alapértelmezetten. Az `x` módosító lehetővé teszi a reguláris kifejezések megjegyzéssel történő ellátását, és szóközök használatát a logikai csoportok megjelenésbeli elkülönítéséhez.

```
$Date =~ m{ \A \d{4} - \d{2} - \d{2} \z }xms
$Date =~ m{
    \A      # beginning of the string
    \d{4} - # year
    \d{2} - # month
    [^\n]   # everything but newline
    #..
}xms;
```

Mivel a szóköznek többé nincs különleges jelentése, ezért egy egyedüli karakterosztályt kell használnia egy egyedülálló szóköz illesztéséhez (`[ ]`). Ha akármennyi szóközre szeretne illeszteni, akkor azt a `\s` használatával teheti meg.

A reguláris kifejezésben a pont (`.`) tartalmazza az új sort (minthogy az `s` módosító nélküli reguláris kifejezésben a pont azt jelenti, hogy „minden, kivéve az új sor” ). Ha bármire szeretne illeszteni az új sort kivéve, akkor a tagadott egyedüli karakterosztályt kell használnia (`[^\n]`).

```
$Text =~ m{
    Test
    [ ]    # there must be a space between 'Test' and 'Regex'
    Regex
}xms;
```

A fenti megegyezésre vonatkozó kivétel minden olyan esetre vonatkozik, amikor a reguláris kifejezéseket nem írják statikusan a kódba, hanem a *felhasználók adják meg* egy űrlapon vagy máshol (például a rendszerbeállításokon vagy egy levelezési szűrő beállításán keresztül). Egy ilyen reguláris kifejezés bármely kiértékelését mindenféle módosító nélkül kell elvégezni (például `$Variable =~ m{$Regex}`) annak érdekében, hogy megfeleljen a (többnyire tapasztalatlan) felhasználók elvárásainak és visszafelé kompatibilis legyen.

Ha a módosítók pontosan szükségesek a felhasználó által megadott reguláris kifejezésekhez, akkor mindig lehetséges beágyazott módosítók használata (például `(?:(?i)KiCsI vAgY nAgY)`). A részletekért nézze meg a perlretut dokumentációját.

Az `r` módosító használata erősen javasolt, például ha ki kell nyernie egy szöveg egy részét egy másik változóba. Ez a módosító érintetlenül hagyja az illesztett változót, ahelyett hogy a helyettesítés eredményt nyújtaná visszatérési értékként.

Használja ezt:

```
my $NewText = $Text =~ s{
    \A
    Prefix
    (
        Text
    )
}
{NewPrefix$1Postfix}xmsr;
```

Ehelyett:

```
my $NewText = $Text;
$NewText =~ s{
    \A
    Prefix
    (
        Text
    )
}
{NewPrefix$1Postfix}xms;
```

Ha egy **szöveg** elejénél vagy végénél szeretne illeszteni, akkor általában az `\A` és a `\z` módosítót kell használnia az általánosabb `^` és `$` helyett, kivéve hogyha valóban a **sorok** elejére vagy végére szeretne illeszteni egy többsoros szövegen belül.

```
$Text =~ m{
    \A      # beginning of the string
    Content # some string
    \z      # end of the string
}xms;

$MultilineText =~ m{
    \A                      # beginning of the string
    .*
    (?: \n Content $ )+      # one or more lines containing the same string
    .*
    \z                      # end of the string
}xms;
```

Az elnevezett elfogási csoportok használata szintén erősen javasolt, különösen többszörös illesztéseknél. Az elnevezett elfogási csoportokat egyszerűbb olvasni és megérteni, megakadályozzák az összekeverést, amikor egynél több elfogási csoportot illeszt, és lehetővé teszik a kiterjesztést, anélkül hogy véletlenül programhibákat vinne a rendszerbe.

Használja ezt:

```
$Contact =~ s{
    \A
    [ ]*
    (?'TrimmedContact'
        (?'FirstName' \w+ )
        [ ]+
        (?'LastName' \w+ )
    )
    [ ]+
    (?'Email' [^ ]+ )
    [ ]*
    \z
}
{$+{TrimmedContact}}xms;
my $FormattedContact = "$+{LastName}, $+{FirstName} ($+{Email})";
```

Ehelyett:

```
$Contact =~ s{
    \A
    [ ]*
    (
        ( \w+ )
        [ ]+
        ( \w+ )
    )
    [ ]+
    ( [^ ]+ )
    [ ]*
    \z
}
{$1}xms;
my $FormattedContact = "$3, $2 ($4)";
```

### Elnevezés

A neveket és a megjegyzéseket angolul kell írni. A változókat, objektumokat és metódusokat leíró főnevekkel vagy főnévi igenevekkel írjuk úgy, hogy az első betű nagybetűs legyen (CamelCase).

A neveknek annyira leírónak kell lenniük, amennyire csak lehetséges. Az olvasónak egy név alapján meg kell tudni mondania, hogy az mit jelent anélkül, hogy túl mélyre ásná magát a kódban. Például használja a `$ConfigItemID` nevet az `$ID` helyett. Példák: `@TicketIDs`, `$Output`, `StateSet()`, stb.

### Változódeklaráció

Ha több változója van, akkor deklarálhatja azokat egyetlen sorban, ha azok *összetartoznak*:

```
my ( $Minute, $Hour, $Year );
```

Egyébként tördelje azokat külön sorokba:

```
my $Minute;
my $ID;
```

Ne állítson be `undef` vagy `''` kezdeti értéket a deklarációban, ugyanis ez elrejtheti a hibákat a kódban.

```
my $Variable = undef;

# is the same as

my $Variable;
```

Akkor állíthat be egy változót `''` értékre, ha szövegeket szeretne összefűzni:

```
my $SqlStatement = '';
for my $Part (@Parts) {
    $SqlStatement .= $Part;
}
```

Egyébként *előkészítetlen* figyelmeztetést kaphat.

#### Paraméterek kezelése

A szubrutinoknak átadott paraméterek lekéréséhez az OTRS normális esetben a `%Param` kivonatot használja (nem a `%Params` kivonatot). Ez jobban olvasható kódot eredményez, mivel minden esetben tudjuk, hogy amikor `%Param` kivonatot használjuk a szubrutin kódokban, akkor paraméterkivonat került átadásra a szubrutinnak.

Csak néhány kivételnél kell a paraméterek szabályos listáját használni. Így el szeretnénk kerülni az ehhez hasonlókat:

```
sub TestSub {
    my ( $Self, $Param1, $Param2 ) = @_;
}
```

Inkább ezt szeretnénk használni:

```
sub TestSub {
    my ( $Self, %Param ) = @_;
}
```

Ennek számos előnye van:

- Nem kell megváltoztatnunk a kódot a szubrutinban, amikor egy új paramétert kell átadni.
- Elnevezett paraméterekkel rendelkező függvény hívása sokkal olvashatóbb.

#### Több elnevezett paraméter

Ha egy függvényhívás egynél több elnevezett paramétert igényel, akkor tördelje azokat több sorba.

Használja ezt:

```
$Self->{LogObject}->Log(
    Priority => 'error',
    Message  => "Need $Needed!",
);
```

Ehelyett:

```
$Self->{LogObject}->Log( Priority => 'error', Message  => "Need $Needed!", );
```

#### `return` utasítások

A szubrutinoknak rendelkezniük kell egy `return` utasítással. Az explicit `return` utasítás előnyben részesített az implicit módszernél (az utolsó utasítás eredménye a szubrutinban), mivel ez tisztázza, hogy mit ad vissza a szubrutin.

```
sub TestSub {
    ...
    return; # return undef, but not the result of the last statement
}
```

#### Explicit visszatérési értékek

Az explicit visszatérési értékek azt jelentik, hogy nem kell egy `return` utasítást tenni egy szubrutinhívást követően.

```
return $Self->{DBObject}->Do( ... );
```

A következő példa jobb, mivel ez explicit módon megmondja, hogy mi kerül visszaadásra. A fenti példával az olvasó nem tudja, hogy mi a visszatérési érték, mivel nem tudhatja, hogy a `Do()` mit ad vissza.

```
return if !$Self->{DBObject}->Do( ... );
return 1;
```

Ha egy szubrutin eredményét hozzárendeli egy változóhoz, akkor egy *jó* változónév jelzi, hogy mi lett visszaadva:

```
my $SuccessfulInsert = $Self->{DBObject}->Do( ... );
return $SuccessfulInsert;
```

#### `use` utasítások

A `use strict` és `use warnings` utasításoknak kell az első két *use*-nak lennie egy modulban.

Ez helyes:

```
package Kernel::System::ITSMConfigItem::History;

use strict;
use warnings;
```

(continues on next page)

(folytatás az előző oldalról)

```
use Kernel::System::User;
use Kernel::System::DateTime;
```

Ez hibás:

```
package Kernel::System::ITSMConfigItem::History;

use Kernel::System::User;
use Kernel::System::DateTime;

use strict;
use warnings;
```

### Objektumok és azok lefoglalása

Az OTRS-ben sok objektum érhető el. De nem kell minden egyes objektumot használnia minden fájlban az előtétprogram/háttérprogram elválasztásának megtartásához.

- Ne használja a `LayoutObject` objektumot az alapmodulokban (`Kernel/System`).
- Ne használja a `ParamObject` objektumot az alapmodulokban (`Kernel/System`).
- Ne használja a `DBObject` objektumot az előtétprogram-modulokban (`Kernel/Modules`).

### Háttérprogram-modulok dokumentálása

**A `NAME` szakasz** Ennek a szakasznak kell tartalmaznia a modul nevét, a ‘‘ - ‘‘ karaktert elválasztóként és a modul céljának rövid leírását.

```
=head1 NAME

Kernel::System::MyModule - Functions to read from and write to files
```

**A `SYNOPSIS` szakasz** Ennek a szakasznak a gyakran használt modulfüggvények rövid használati példáját kell adnia.

A szakasz használata elhagyható.

```
=head1 SYNOPSIS

my $Object = $Kernel::OM->Get('Kernel::System::MyModule');

Read data

    my $FileContent = $Object->Read(
        File => '/tmp/testfile',
    );

Write data

    $Object->Write(
        Content => 'my file content',
```

(continues on next page)

(folytatás az előző oldalról)

```
        File    => '/tmp/testfile',
    );
```

**A `DESCRIPTION` szakasz** Ennek a szakasznak mélyebb információkat kell adnia a modullal kapcsolatban, ha szükségesnek tekintik (egy hosszú `NAME` szakasz megléte helyett).

A szakasz használata elhagyható.

```
=head1 DESCRIPTION

This module does not only handle files.

It is also able to:
- brew coffee
- turn lead into gold
- bring world peace
```

**A `PUBLIC INTERFACE` szakasz** Ez a szakasz jelöli az összes olyan függvény kezdetét, amely az API része, és ennélfogva egyéb modulok használni kívánják.

```
=head1 PUBLIC INTERFACE
```

**A `PRIVATE FUNCTIONS` szakasz** Ez a szakasz jelzi a privát függvények kezdetét.

A lenti függvények nem részei az API-nak, csak a modulon belül használhatók, és ennélfogva nem tekinthetők stabilnak.

Javasolt ennek a szakasznak a használata, amikor egy vagy több privát függvény létezik.

```
=head1 PRIVATE FUNCTIONS
```

### Szubrutinok dokumentálása

A szubrutinokat mindig dokumentálni kell. A dokumentum tartalmaz egy általános leírást arról, hogy mit csinál a szubrutin, egy minta szubrutinhívást, és hogy mit ad vissza a szubrutin. Ezeknek ebben a sorrendben kell lenniük. Egy minta dokumentáció így néz ki:

```
=head2 LastTimeObjectChanged()

Calculates the last time the object was changed. It returns a hash reference␣
↪with
    information about the object and the time.

    my $Info = $Object->LastTimeObjectChanged(
        Param => 'Value',
    );

This returns something like:

    my $Info = {
        ConfigItemID    => 1234,
        HistoryType     => 'foo',
        LastTimeChanged => '08.10.2009',
```

(continues on next page)

(folytatás az előző oldalról)

```
    };

=cut
```

Lemásolhat és beilleszthet egy `Data::Dumper` kimenetet a visszatérési értékekhez.

### Kódmagyarázatok Perlben

Általánosságban meg kell próbálni olvashatóan és önmagát magyarázóan írni a kódot, amennyire csak lehetséges. Ne írjon megjegyzést annak magyarázásához, hogy a nyilvánvaló kód mit csinál, mert az szükségtelen megkettőzés. A jó megjegyzéseknek azt kell elmagyarázniuk, hogy **miért** van valami a kódban, mik a lehetséges mellékhatások és bármi egyéb, amely különleges lehet vagy szokatlanul bonyolult a kóddal kapcsolatban.

Ragaszkodjon a következő irányelvekhez:

**Tegye a kódot annyira olvashatóvá, hogy ne legyen szükség magyarázatra, ha ez lehetséges.** Mindig vonzóbb a kódot úgy írni, hogy nagyon olvasható és önmagát magyarázó legyen, például pontos változónevekkel és függvénynevekkel.

**Ne mondja el, amit a kód is elmond (DRY: Don't Repeat Yourself – Ne ismételd önmagad).** Ne ismételjen (nyilvánvaló) kódot a magyarázatokban.

```
# WRONG:

# get config object
my $ConfigObject = $Kernel::OM->Get('Kernel::Config');
```

**Azt dokumentálja, hogy a kód miért van ott, és ne azt, hogy hogyan működik.** Általában a kódmagyarázatoknak a kód *célját* kellene elmagyarázniuk, és nem azt, hogy részletesen hogyan működik. Lehetnek kivételek különösen bonyolult kódnál, de ebben az esetben egy átszerkesztés lenne dicséretes, hogy olvashatóbb legyen.

**Dokumentálja a buktatókat.** Mindent dokumentálni kell, ami nem világos, furfangos vagy amit összerakott a fejlesztés során.

**Használjon teljes soros mondatszerű magyarázatokat az algoritmus bekezdéseinek dokumentálásához.** Mindig teljes mondatokat használjon (első betűt nagybetűvel írva és központozással). Egy mondat következő sorait be kell húzni.

```
# Check if object name is provided.
if ( !$_[1] ) {
    $_[0]->_DieWithError(
        Error => "Error: Missing parameter (object name)",
    );
}

# Record the object we are about to retrieve to potentially build better␣
↪error messages.
# Needs to be a statement-modifying 'if', otherwise 'local' is local
#   to the scope of the 'if'-block.
local $CurrentObject = $_[1] if !$CurrentObject;
```

**Használjon rövid sorvégi magyarázatokat a részletes információk hozzáadásához.** Ez lehet vagy teljes mondat (nagy kezdőbetű és központozás), vagy csak egy kifejezés (kis kezdőbetű és nincs központozás).

```
$BuildMode = oct $Param{Mode};    # *from* octal, not *to* octal

# or

$BuildMode = oct $Param{Mode};    # Convert *from* octal, not *to* octal.
```

### SQL-utasítások deklarációja

Ha nincs esély az SQL-utasítás megváltoztatására, akkor azt a `Prepare` függvényben kell használni. Ennek az az oka, hogy az SQL-utasítás és a kötési paraméterek közelebb vannak egymáshoz.

Az SQL-utasítást egy összefűzések nélküli, pontosan behúzott szövegként kell megírni úgy, mint például ezt:

```
return if !$Self->{DBObject}->Prepare(
    SQL => '
        SELECT art.id
        FROM article art, article_sender_type ast
        WHERE art.ticket_id = ?
            AND art.article_sender_type_id = ast.id
            AND ast.name = ?
        ORDER BY art.id',
    Bind => [ \$Param{TicketID}, \$Param{SenderType} ],
);
```

Ezt könnyű olvasni és módosítani, és az üres karaktereket jól tudják kezelni a támogatott DBMS-ek. Az automatikusan előállított SQL-kódnál (mint a `TicketSearch` modulban) ez a behúzás nem szükséges.

### Visszatérés hibák esetén

Valahányszor adatbázis-függvényeket használ, kezelnie kell a hibákat. Ha valami elromlik, az visszakerül a szubrutinból:

```
return if !$Self->{DBObject}->Prepare( ... );
```

### Korlát használata

Használja a `Limit => 1` korlátozást, ha csak egyetlen sort vár visszatérésként.

```
$Self->{DBObject}->Prepare(
    SQL   => 'SELECT id FROM users WHERE username = ?',
    Bind  => [ \$Username ],
    Limit => 1,
);
```

#### A `while` ciklus használata

Mindig használja a `while` ciklust még akkor is, ha csak egyetlen sort vár visszatérésként, mivel néhány adatbázis nem szabadítja fel az utasításleírót, és ez furcsa hibákhoz vezethet.

### 6.3.2 JavaScript

Az összes JavaScript betöltődik minden böngészőben (nincsenek böngésző trükközések a sablonfájlokban). A kód felelős annak eldöntéséért, hogy ki kell hagynia vagy végre kell hajtania saját magának bizonyos részeit az egyes böngészőkben.

#### Könyvtárszerkezet

Könyvtárszerkezet a `var/httpd/htdocs/js/` mappán belül:

```
* js
    * thirdparty              # thirdparty libs always have the version␣
↪number inside the directory
        * ckeditor-3.0.1
        * jquery-1.3.2
    * Core.Agent.*            # stuff specific to the agent interface
    * Core.Customer.*         # customer interface
    * Core.*                  # common API
```

#### Harmadik féltől származó kód

Minden harmadik féltől származó modul saját alkönyvtárat kap: *modulnév-verziószám* (például ckeditor-4.7.0, jquery-3.2.1). Ezen belül a fájlneveknek nem kell verziószámot vagy előtagot tartalmaznia (hibás: `jquery/jquery-3.2.1.min.js`, helyes: `jquery-3.2.1/jquery.js`).

#### JavaScript változók

A változóneveket CamelCase jelölésrendszerben kell írni, akárcsak a Perlben.

A jQuery objektumot tartalmazó változókat `$` karakterrel kell kezdeni, például: `$Tooltip`.

#### Függvények

A függvényneveket CamelCase jelölésrendszerben kell írni, akárcsak a Perlben.

#### Névterek

#### Kódmagyarázatok JavaScriptben

A *Kódmagyarázatok Perlben* szakaszban leírtak a JavaScriptre is vonatkoznak.

- Egysoros megjegyzéseket `//` karakterekkel kell készíteni.
- Hosszabb megjegyzéseket `/* ... */` karakterekkel kell készíteni.

- Ha megjegyzésre állítja a JavaScript kód egyes részeit, akkor csak a `//` karaktereket használja, ugyanis a `/* ... */` használata problémákat okozhat a reguláris kifejezéseknél a kódban.

#### Eseménykezelés

Mindig az `$.on()` függvényt használja a jQuery rövidített nevű eseménymetódusai helyett a jobb olvashatóságért (hibás: `$SomeObject.click(...)`, helyes: `$SomeObject.on('click', ...)`).

Ha eseményeket köt az `$.on()` függvénnyel, akkor győződjön meg arról, hogy korábban leválasztotta azokat az `$.off()` használatával annak biztosításához, hogy az események ne legyenek kétszer kötve, ne kelljen a kódot még egy alkalommal végrehajtani.

Győződjön meg arról, hogy az `$.on()` függvényt névtérrel használja-e, mint például `$.on('click.<Name>')`.

### 6.3.3 HTML

Használjon HTML 5 jelölést. Ne használjon önmagát lezáró címkéket nem üres elemeknél (mint például `div`, `span`, stb.).

Használjon megfelelő behúzást. Azok az elemek, amelyek egyéb, nem üres gyermekelemet tartalmaznak, nem lehetnek ugyanazon a szinten, mint a gyermekeik.

Ne használjon HTML elemeket elrendezési okokból (például `br` elemek használata más elemek fölé vagy alá történő további térköz adásához). Használja helyette a megfelelő CSS osztályokat.

Ne használjon beágyazott CSS-t. Az összes CSS-t vagy az előre meghatározott osztályokkal, vagy (ha szükséges) JavaScript használatával kell hozzáadni (például elemek megjelenítéséhez vagy elrejtéséhez).

Ne használjon JavaScriptet a sablonokban. Az összes szükséges JavaScriptnek egy bizonyos előtétprogram-modul megfelelő programkönyvtára részének, vagy egy megfelelő globális programkönyvtár részének kell lennie. Ha JavaScript adatokat kell átadnia az előtétprogramnak, akkor használja a `$LayoutObject->AddJSData()` metódust.

### 6.3.4 CSS

A legkisebb felbontás 1024×768 képpont.

Az elrendezés folyékony, amely azt jelenti, hogy ha a képernyő szélesebb, akkor a helyet fel fogja használni.

Az abszolút méretmeghatározásokat képpontban (px) kell megadni, hogy következetes kinézetet kapjon a legtöbb platformon és böngészőben.

A dokumentáció CSSDOC használatával készül (nézze meg a CSS-fájlokat példaként). Az összes logikai blokknak rendelkeznie kell egy CSSDOC megjegyzéssel.

#### CSS szerkezet

Az objektumorientált CSS megközelítést követjük. Lényegében ez azt jelenti, hogy az elrendezés különböző általános építőkockák egyesítésével érhető el egy bizonyos látványterv megvalósításához.

Ahol csak lehetséges, nem szabad a modulra jellemző látványtervet használni. Például ezért nem dolgozunk azonosítókkal a `body` elemen sem, ha az elkerülhető.

#### CSS stílus

Az összes meghatározásnak ugyanabban a sorában van a { karakter mint a kiválasztó, az összes szabály szabályonként egy sorban van meghatározva, a meghatározások egyetlen } karaktert tartalmazó sorral végződnek.

Nézze meg a következő példát:

```
#Selector {
    width: 10px;
    height: 20px;
    padding: 4px;
}
```

- A : és szabály értéke között van egy szóköz.
- Minden szabály 4 szóközzel van behúzva.
- Ha több kiválasztó van megadva, akkor vesszővel válassza el azokat, és mindegyiket tegye külön sorba:

```
#Selector1,
#Selector2,
#Selector3 {
    width: 10px;
}
```

- Ha a szabályok egyesíthetők, akkor egyesítse azokat (például egyesítse a `background-position`, `background-image`, stb. szabályokat a `background` szabályba).
- A szabályoknak logikai sorrendben kell lenniük egy meghatározáson belül (az összes színre jellemző szabály együtt, az összes pozicionáló szabály együtt, stb.).
- Az összes azonosító és név CamelCase jelölésrendszerben van írva:

```
<div class="NavigationBar" id="AdminMenu"></div>
```

## 6.4 Felhasználó felület tervezése

### 6.4.1 Nagybetűs írás

Ez a szakasz azt mutatja be, hogy az angol felhasználói felület különböző részeit hogyan kell nagybetűsen írni. További információkért érdemes átnézni ezt a hasznos oldalt.

A címsorok (h1-h6) és a címek (nevek, mint például *Queue View*) *címstílusú* nagy kezdőbetűs írásmódban vannak, amely azt jelenti, hogy az összes első betű nagybetűvel lesz írva (néhány kivétellel, mint például *this*, *and*, *or*, stb.).

Példák:

- *Műveletlista*
- *Ügyfél–Csoport kapcsolatok kezelése*

Az egyéb szerkezeti elemek (mint például gombok, címkék, lapok, menüpontok) *mondatstílusú* nagy kezdőbetűs írásmódban vannak (csak a kifejezés első betűje van nagybetűvel írva), de nincs lezáró pont hozzáadva a kifejezés mondatként való befejezéséhez.

Példák:

- *Utónév*
- *Várólista frissítési idejének kiválasztása*
- *Jegy nyomtatása*

A leíró szövegek és a buboréksúgó tartalmak teljes mondatként vannak írva.

Példa:

- *Ez a mező kötelező.*

A fordításoknál ellenőrizni kell, hogy címstílusú nagybetűs írás megfelelő-e a célnyelven is. Lehet, hogy meg kell változtatni mondatstílusú nagybetűs írásra vagy valami másra.

## 6.5 Akadálymentesítési útmutató

Ez a dokumentum hivatott elmagyarázni az akadálymentesítési problémákkal kapcsolatos alapokat, és irányelveket ad az OTRS-ben való közreműködéshez.

### 6.5.1 Akadálymentesítési alapok

#### Mi az akadálymentesítés?

Az akadálymentesítés egy általános kifejezés annak leírásához, hogy egy termék, eszköz, szolgáltatás vagy környezet milyen mértékben érhető el a lehető legtöbb ember által. Az akadálymentesítés tekinthető úgyis mint *képesség a hozzáféréshez*, és néhány rendszer vagy dolog lehetséges előnyeként. Az akadálymentesítés gyakran a fogyatékkal rendelkező emberekre és a dolgokhoz való hozzáférésük jogára összpontosít, gyakran kisegítő technológiák használatán keresztül.

A webfejlesztés környezetében akadálymentesítéskor a sérült embereknek a webes felületekhez való teljes hozzáférés engedélyezésén van a hangsúly. Például az emberek ezen csoportja részlegesen látássérült vagy teljesen vak embereket tartalmazhat. Míg az előbbiek továbbra is képesek részlegesen használni a grafikus felhasználói felületet, addig az utóbbiaknak teljes mértékben a kisegítő technológiákra kell támaszkodniuk, az olyan szoftverekre, amelyek felolvassák számukra a képernyőt (képernyőolvasók).

#### Miért fontos ez az OTRS-nél?

A sérült felhasználóknak történő hozzáférés engedélyezése az OTRS rendszerekhez önmagában indokolt cél. Tiszteletet tanúsít.

Továbbá az akadálymentesítési szabványok teljesítése egyre inkább fontossá válik az állami szektorban (kormányzati intézményeknél) és a nagyvállalatoknál, amely mindkettő az OTRS célcsoportjához tartozik.

#### Hogyan tudok sikeresen dolgozni az akadálymentesítési problémákon akkor is, ha nem vagyok fogyatékos?

Ez nagyon egyszerű. Tegyen úgy, mintha vak lenne.

- Ne használja az egeret.
- Ne nézzen a képernyőre.

Ezután próbálja meg az OTRS-t csak egy képernyőolvasó és a billentyűzet segítségével használni. Ez ötletet adhat arra, hogy mit fog érezni egy vak ember.

#### Rendben, de nincs képernyőolvasóm!

Amíg a kereskedelmi képernyőolvasók - mint például a JAWS (talán a legismertebb) - nagyon drágák lehetnek, léteznek nyílt forrású képernyőolvasók, amelyeket telepíthet és használhat:

- NVDA, egy képernyőolvasó Windows alá.
- ORCA, egy képernyőolvasó Gnome/Linux alá.

Most már nincs többé mentsége. ;)

### 6.5.2 Akadálymentesítési szabványok

Ez a szakasz csak hivatkozásként szolgál, nem kell magát a szabványokat áttanulmányoznia ahhoz, hogy képes legyen akadálymentesítési problémákon dolgozni az OTRS-ben. Megpróbáljuk kigyűjteni a fontos irányelveket ebben a dokumentumban.

#### Webtartalom akadálymentesítési irányelvek (WCAG)

Ez a W3C szabvány általános irányelveket ad ahhoz, hogyan hozhatók létre akadálymentes weboldalak.

- WCAG 2.0
- Hogyan tegyen eleget a WCAG 2.0 szabványnak
- A WCAG 2.0 megértése

A WCAG az akadálymentesítési támogatás különböző szintjeivel rendelkezik. Jelenleg az A szint támogatását tervezzük, ugyanis az AA és az AAA olyan kérdésekkel foglalkozik, amely nem tűnik fontosnak az OTRS-nél.

#### Akadálymentes gazdag internetes alkalmazások (WAI-ARIA) 1.0

Ez a szabvány foglalkozik a statikus tartalom eltolásából eredő különleges problémákkal a dinamikus webalkalmazásoknál. Olyan kérdésekkel foglalkozik, mint például hogyan értesülhet a felhasználó az AJAX kérésekből eredményezett felhasználói felület változásairól.

- WAI-ARIA 1.0

### 6.5.3 Megvalósítási irányelvek

#### Alternatívák biztosítása a nem szöveges tartalomhoz

Cél: *a felhasználónak megjelenő összes nem szöveges tartalomnak legyen szöveges alternatívája, amely ugyanazt a célt szolgálja.* (WCAG 1.1.1)

Nagyon fontos megérteni, hogy a képernyőolvasók csak a szöveges információkat és az elérhető metaadatokat tudják megjeleníteni a felhasználónak. Hogy egy példát mondjunk, amikor egy képernyőolvasó a `<a href="#" class="CloseLink"></a>` hivatkozást látja, akkor csak a *hivatkozást* tudja felolvasni a felhasználónak, nem a hivatkozás célját. Egy apró fejlesztéssel akadálymentessé válhat: `<a href="#"`

`class="CloseLink" title="Close this widget"></a>`. Ebben az esetben a felhasználó a *hivatkozás felületi elem bezárása* szöveget hallhatja, íme!

Fontos, hogy a szöveget mindig a leginkább *beszédes* módon fogalmazza meg. Egyszerűen képzelje el, hogy csak ennyi információval rendelkezik. Segíteni fog önnek? Képes megérteni a célját pusztán hallás után?

Kövesse ezeket a szabályokat, amikor az OTRS-en dolgozik:

- *Szabály*: Ahol csak lehetséges, használjon beszédes szövegeket, és fogalmazzon meg valódi, érthető és pontos mondatokat. A *Felületi elem bezárása* sokkal jobb mint a *Bezárás*, mert az utóbbi fölösleges.
- *Szabály*: A hivatkozásoknak mindig legyen vagy olyan szöveges tartalma, amelyet a képernyőolvasók kimondanak (`<a href="#" >Delete this entry</a>`), vagy `title` attribútuma (`<a href="#" title="Close this widget"></a>`).
- *Szabály*: A képeknek mindig legyen alternatív szövege, amely felolvasható a felhasználónak (`<img src="house.png" alt="Image of a house" />`).

### A navigáció könnyűvé tétele

Cél: *lehetővé tenni a felhasználónak, hogy könnyen navigáljon az aktuális oldalon és az egész alkalmazásban.*

A `title` címke az első dolog, amit a felhasználó hall a képernyőolvasótól, amikor megnyit egy weboldalt. Az OTRS-nél mindig csak egyetlen `h1` elem van az oldalon az aktuális oldalt jelezve (a `title` elemből vett információ egy részét tartalmazza). Ez a navigációs információ segít megérteni a felhasználónak, hogy éppen hol van, és mi az aktuális oldal célja.

- *Szabály*: Mindig pontos címet adjon az oldalnak, amely lehetővé teszi a felhasználónak annak megértését, hogy jelenleg éppen hol van.

A képernyőolvasók képesek a HTML beépített dokumentumszerkezetét használni (címsorok `h1` és `h6` között) egy dokumentum szerkezetének meghatározásához, és lehetővé tenni a felhasználónak, hogy egyik szakaszról a másik szakaszra ugorjon. Azonban ez nem elegendő egy dinamikus webalkalmazás szerkezetének kifejezéséhez. Ez az, amiért az ARIA számos olyan *iránypont* szerepet határoz meg, amelyek megadhatók az elemeknek, hogy jelezzék a navigációs jelentésüket.

A HTML dokumentumok érvényességének megtartásához a `role` attribútumok (ARIA iránypont szerepek) nincsenek közvetlenül a forráskódba beszúrva, hanem olyan osztályok által kerülnek beszúrásra, amelyeket később az `OTRS.UI.Accessibility` osztályban lévő JavaScript függvények fognak használni a megfelelő `role` attribútumok beszúrásához a csomópontba.

- *Szabály*: Használjon WAI-ARIA iránypont szabályokat a tartalom szerkezetbe foglalásához a képernyőolvasók számára.
  - Reklámcsík: `<div class="ARIARoleBanner"></div>` helyett `<div class="ARIARoleBanner" role="banner"></div>` lesz
  - Navigáció: `<div class="ARIARoleNavigation"></div>` helyett `<div class="ARIARoleNavigation" role="navigation"></div>` lesz
  - Keresőfunkció: `<div class="ARIARoleSearch"></div>` helyett `<div class="ARIARoleSearch" role="search"></div>` lesz
  - Fő alkalmazásterület: `<div class="ARIARoleMain"></div>` helyett `<div class="ARIARoleMain" role="main"></div>` lesz
  - Lábléc: `<div class="ARIARoleContentinfo"></div>` helyett `<div class="ARIARoleContentinfo" role="contentinfo"></div>` lesz

A `<form>` elemeken belüli navigációnál szükséges a fogyatékos felhasználónak tudnia, hogy mi az egyes beviteli elemek célja. Ezt el lehet érni a szabványos HTML `<label>` elemek használatával, amelyek kapcsolatot hoznak létre a címke és az űrlap elem között.

Amikor egy beviteli elem megkapja a fókuszt, akkor a képernyőolvasó általában fel fogja olvasni a hozzá kapcsolt címkét, így a felhasználó hallhatja annak pontos célját. További előnye a látó felhasználóknak, hogy rákattinthatnak a címkére, és a beviteli elem meg fogja kapni a fókuszt (különösen jelölőnégyzeteknél hasznos például).

- *Szabály*: Adjon meg `<label>` elemeket az *összes* űrlapelem (`input`, `select`, `textarea`) mezőhöz.

  Példa: `<label for="date">Dátum:</label><input type="text" name="date" id="date"/>`

#### A kölcsönhatás lehetővé tétele

Cél: *lehetővé tenni a felhasználónak, hogy az összes kölcsönhatást végrehajtsa pusztán a billentyűzet használatával.*

Miközben technikailag lehetséges kölcsönhatásokat létrehozni JavaScript segítségével tetszőleges HTML elemen, ezt korlátozni kell azon elemekre, amelyekkel a felhasználó kölcsönhatásba léphet a billentyűzet használatával. Különösen azt kell lehetővé tenni számukra, hogy fókuszt adjanak az elemnek, és kölcsönhatásba lépjenek vele. Például egy felületi elemet ki-be kapcsoló nyomógombot nem egy JavaScript `onclick` eseményfigyelővel összekötött `span` elem használatával kellene megoldani, hanem egy `a` címkének kellene lennie (vagy tartalmaznia kellene), hogy világossá tegye a képernyőolvasónak, hogy ez az elem kölcsönhatást okozhat.

- *Szabály*: A kölcsönhatásoknál mindig olyan elemeket használjon, amelyek megkaphatják a fókuszt, mint például `a`, `input`, `select` és `button`.
- *Rule*: Győződjön meg arról, hogy a felhasználó mindig képes-e azonosítani a kölcsönhatás természetét (nézze meg a szabályokat a nem szöveges tartalommal és az űrlapelemek címkézésével kapcsolatban).

Cél: *Tudatni a felhasználóval a dinamikus változtatásokat.*

Az akadálymentesítési problémák egy különleges területe a dinamikus változtatások a felhasználói felületen, amelyeket JavaScript vagy AJAX hívások okoznak. A képernyőolvasó nem fog beszélni a felhasználónak a változtatásokról különleges óvintézkedések nélkül. Ez egy bonyolult téma, és még nem lehet itt teljesen elmagyarázni.

- *Szabály*: Mindig használja az `OTRS.Validate` érvényesítő keretrendszert az űrlap érvényesítéséhez.

  Ez biztosítani fogja, hogy a hiba buboréksúgókat felolvassa a képernyőolvasó. Ily módon a vak felhasználó a) megtudja azt az elemet, amelynek hibája van, és b) kap egy szöveget, amely leírja a hibát.

- *Szabály*: Használja az `OTRS.UI.Accessibility.AudibleAlert()` függvényt, hogy értesítse a felhasználót az egyéb fontos felhasználói felületet érintő változtatásokról.
- *Szabály*: Használja az `OTRS.UI.Dialog` keretrendszert a kizárólagos párbeszédablakok létrehozásához. Ezek már optimalizálva vannak az akadálymentesítéshez.

#### Általános képernyőolvasó optimalizálások

Cél: *segíteni a képernyőolvasókat a munkájukban.*

- *Szabály*: Minden egyes oldalnak azonosítania kell a saját fő nyelvét azért, hogy a képernyőolvasók ki tudják választani a megfelelő beszédszintetizátor motort.

  Példa: `<html lang="hu">...</html>`

## 6.6 Egységtesztek

Az OTRS biztosít egy tesztelési alkalmazáscsomagot, amely egységtesztek fejlesztéséhez és futtatásához használható az összes rendszerrel kapcsolatos kódnál.

### 6.6.1 Egy tesztfájl létrehozása

A tesztfájlok `.t` fájlokban vannak tárolva a `scripts/test/*.t` helyen. Például vessünk egy pillantást a `scripts/test/Calendar.t` fájlra a naptár osztálynál.

Minden tesztfájlnak ideális esetben példányosítania kell az egységteszt segítő objektumot az elején, így részesülhet néhány beépített metódusból, amelyet az biztosít:

```
# --
# Copyright (C) 2001-2020 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

use strict;
use warnings;
use utf8;

use vars (qw($Self));

$Kernel::OM->ObjectParamAdd(
    'Kernel::System::UnitTest::Helper' => {
        RestoreDatabase => 1,
    },
);
my $Helper = $Kernel::OM->Get('Kernel::System::UnitTest::Helper');
```

A `RestoreDatabase` paraméternek a segítő konstruktor számára történő átadásával az egységteszt közben végrehajtott bármely adatbázis-utasítás vissza lesz állítva a végén, ezáltal azt biztosítva, hogy ne történjen végleges változtatás.

Mint bármely egyéb tesztelési alkalmazáscsomag, az OTRS is biztosít állítás metódusokat, amelyek a feltételek teszteléséhez használhatók. Például it az látható, hogy hogyan hozunk létre egy teszt felhasználót, és hogyan teszteljük le, hogy valóban létrejött:

```
my $UserLogin = $Helper->TestUserCreate();
my $UserID = $UserObject->UserLookup( UserLogin => $UserLogin );

$Self->True(
    $UserID,
```

(continues on next page)

(folytatás az előző oldalról)

```
    "Test user $UserID created"
);
```

Nézze meg a lenti API szakaszt az állítás metódusok teljes listájáért.

Mindig jó gyakorlat véletlenszerű adatokat létrehozni az egységtesztekben, ami segíthet megkülönböztetni az előzőleg hozzáadott adatoktól. Használja a véletlen metódusokat az API-ból, hogy megkapja a karakterláncokat, és beágyazza azokat a paraméterekbe:

```
my $RandomID = $Helper->GetRandomID();

# create test group
my $GroupName = 'test-calendar-group-' . $RandomID;
my $GroupID   = $GroupObject->GroupAdd(
    Name    => $GroupName,
    ValidID => 1,
    UserID  => 1,
);

$Self->True(
    $GroupID,
    "Test group $GroupID created"
);
```

A jó fejlesztők az egységtesztjeiket könnyen karbantarthatóvá teszik. Fontolja meg az összes teszteset egy tömbbe történő elhelyezését, majd lépkedjen végig azokon valamilyen kóddal. Ez egyszerű módot fog biztosítani a teszt későbbi bővítéséhez:

```
#
# Tests for CalendarCreate()
#
my @Tests = (
    {
        Name    => 'CalendarCreate - No params',
        Config  => {},
        Success => 0,
    },
    {
        Name   => 'CalendarCreate - All required parameters',
        Config => {
            CalendarName => "Calendar-$RandomID",
            Color        => '#3A87AD',
            GroupID      => $GroupID,
            UserID       => $UserID,
        },
        Success => 1,
    },
    {
        Name   => 'CalendarCreate - Same name',
        Config => {
            CalendarName => "Calendar-$RandomID",
            Color        => '#3A87AD',
            GroupID      => $GroupID,
```

(continues on next page)

(folytatás az előző oldalról)

```
            UserID       => $UserID,
        },
        Success => 0,
    },
);

for my $Test (@Tests) {

    # make the call
    my %Calendar = $CalendarObject->CalendarCreate(
        %{ $Test->{Config} },
    );

    # check data
    if ( $Test->{Success} ) {
        for my $Key (qw(CalendarID GroupID CalendarName Color CreateTime␣
↪CreateBy ChangeTime ChangeBy ValidID)) {
            $Self->True(
                $Calendar{$Key},
                "$Test->{Name} - $Key exists",
            );
        }

        KEY:
        for my $Key ( sort keys %{ $Test->{Config} } ) {
            next KEY if $Key eq 'UserID';

            $Self->IsDeeply(
                $Test->{Config}->{$Key},
                $Calendar{$Key},
                "$Test->{Name} - Data for $Key",
            );
        }
    }
    else {
        $Self->False(
            $Calendar{CalendarID},
            "$Test->{Name} - No success",
        );
    }
}
```

### 6.6.2 Előfeltételek a teszteléshez

Hogy képes legyen lefuttatni az egységteszteket, arra van szüksége, hogy az összes választható környezeti függőség (Perl-modulok és egyéb modulok) telepítve legyen, kivéve azokat, amelyek az Ön által használt adatbázis háttérprogramoktól eltérőkhöz valók. Futtassa a `bin/otrs.CheckEnvironment.pl` parancsfájlt a modultelepítés ellenőrzéséhez.

Szüksége van egy teljes képzésű tartományneven (FQDN) futó OTRS webes előtétprogram egy példányára, amely az OTRS helyi `Config.pm` fájljában be van állítva. Ennek az OTRS példánynak ugyanazt az adatbázist kell használnia, amelyek az egységtesztekhez vannak beállítva.

### 6.6.3 Tesztelés

A tesztek futtatásához egyszerűen használja a `bin/otrs.Console.pl Dev::UnitTest::Run --test Calendar` parancsot a `scripts/test/Calendar.t` fájl használatához.

```
shell:/opt/otrs> bin/otrs.Console.pl Dev::UnitTest::Run --test Calendar
+-------------------------------------------------------------------+
/opt/otrs/scripts/test/Calendar.t:
+-------------------------------------------------------------------+
.....................................................................
 ↪...................
=====================================================================
yourhost ran tests in 2s for OTRS 6.0.x git
All 97 tests passed.
shell:/opt/otrs>
```

Akár több tesztet is lefuttathat egyszerre, csak adjon meg további `--test` argumentumokat a parancsnak:

```
shell:/opt/otrs> bin/otrs.Console.pl Dev::UnitTest::Run --test Calendar --
 ↪test Appointment
+-------------------------------------------------------------------+
/opt/otrs/scripts/test/Calendar.t:
+-------------------------------------------------------------------+
.....................................................................
 ↪...................
+-------------------------------------------------------------------+
/opt/otrs/scripts/test/Calendar/Appointment.t:
+-------------------------------------------------------------------+
.....................................................................
 ↪.....................................
=====================================================================
yourhost ran tests in 5s for OTRS 6.0.x git
All 212 tests passed.
shell:/opt/otrs>
```

Ha argumentumok nélkül hajtja végre a `bin/otrs.Console.pl Dev::UnitTest::Run` parancsot, akkor le fogja futtatni a rendszeren található összes tesztet. Ne feledje, hogy ez eltarthat egy ideig, amíg befejeződik.

Adja meg a `--verbose` argumentumot annak érdekében, hogy láthassa a sikeres tesztekkel kapcsolatos üzeneteket is. A teszt során előforduló bármilyen hiba megjelenítésre kerül függetlenül ettől a kapcsolótól, és biztosítva lesz, hogy azok ténylegesen felvételre kerülnek a tesztbe.

### 6.6.4 Egységteszt API

Az OTRS API-t biztosít az egységteszteléshez, amely az előző példában volt használva. Itt fel fogjuk sorolni a legfontosabb függvényeket, de nézze meg a `Kernel::System::UnitTest <https://otrs.github.io/doc/api/otrs/7.0/Perl/Kernel/System/UnitTest.pm.html>`__ internetes API hivatkozását is.

**`True()`** Ez a függvény azt teszteli, hogy a megadott skalár érték igaz érték-e a Perlben.

```
$Self->True(
    1,
```

(continues on next page)

(folytatás az előző oldalról)

```
    'Scalar 1 is always evaluated as true'
);
```

**False()** Ez a függvény azt teszteli, hogy a megadott skalár érték hamis érték-e a Perlben.

```
$Self->False(
    0,
    'Scalar 0 is always evaluated as false'
);
```

**Is()** Ez a függvény azt teszteli, hogy a megadott skalár változók egyenlők-e.

```
$Self->Is(
    $A,
    $B,
    'Test Name',
);
```

**IsNot()** Ez a függvény azt teszteli, hogy a megadott skalár változók nem egyenlők-e.

```
$Self->IsNot(
    $A,
    $B,
    'Test Name'
);
```

**IsDeeply()** Ez a függvény összetett adatszerkezeteket hasonlít össze az egyenlőséghez. `$A` és `$B` hivatkozás kell legyen.

```
$Self->IsDeeply(
    $A,
    $B,
    'Test Name'
);
```

**IsNotDeeply()** Ez a függvény összetett adatszerkezeteket hasonlít össze a nem egyenlőséghez. `$A` és `$B` hivatkozás kell legyen.

```
$Self->IsNotDeeply(
    $A,
    $B,
    'Test Name'
);
```

Emellett az egységteszt segítő objektum biztosít néhány hasznos metódust is a gyakori tesztelési feltételekhez. A teljes hivatkozásért nézze meg a `Kernel::System::UnitTest::Helper <https://doc.otrs.com/doc/api/otrs/7.0/Perl/Kernel/System/UnitTest/Helper.pm.html>‘__ internetes API hivatkozását.

**GetRandomID()** Ez a függvény előállít egy véletlenszerű azonosítót, amely egyedi azonosítóként használható a tesztekben. Garantált, hogy egy teszten belül ez a függvény soha nem ad vissza kettőzött értéket.

---

**Megjegyzés:** Ne feledje, hogy ezek a számok nem valódi véletlen számok, és csak tesztadatok

előállításához szabad használni.

```
my $RandomID = $Helper->GetRandomID();
# $RandomID = 'test6326004144100003';
```

**TestUserCreate()** Ez a függvény létrehoz egy teszt felhasználót, amely használható a tesztekben. Automatikusan érvénytelenre lesz állítva a destruktor közben. Visszaadja az új felhasználó bejelentkezési nevét, a jelszó pedig ugyanaz lesz.

```
my $TestUserLogin = $Helper->TestUserCreate(
    Groups   => ['admin', 'users'],          # optional, list of groups␣
↪to add this user to (rw rights)
    Language => 'de',                        # optional, defaults to 'en'␣
↪if not set
);
```

**FixedTimeSet()** Ez a függvény lehetővé teszi a rendszeridő felülbírálását egészen addig, amíg az objektum él. Átadhat egy elhagyható időparamétert, amelyet használni kell. Ha nincs átadva, akkor az aktuális rendszeridő lesz használva.

**Megjegyzés:** A `Kernel::System::Time` és a `Kernel::System::DateTime` metódusainak összes meghívása a megadott időt fogja használni ezután.

```
$HelperObject->FixedTimeSet(366475757);          # with Timestamp
$HelperObject->FixedTimeSet($DateTimeObject);    # with previously created␣
↪DateTime object
$HelperObject->FixedTimeSet();                   # set to current date and␣
↪time
```

**FixedTimeUnset()** Ez a függvény visszaállítja a rendes rendszeridő viselkedést.

**FixedTimeAddSeconds()** Ez a függvény valamennyi másodpercet ad ahhoz a rögzített rendszeridőhöz, amelyet a `FixedTimeSet()` korábban beállított. Átadhat negatív értéket, hogy visszamenjen az időben.

**ConfigSettingChange()** Ez a függvény átmenetileg rendszerszinten megváltoztat egy rendszerbeállítást egy másik értékre mind a `ConfigObject` aktuális példányában, mind a rendszerbeállításokban is a lemezen. Akkor lesz visszaállítva, amikor a `Helper` objektum megsemmisül.

**Megjegyzés:** Ne feledje, hogy ez jelenleg nem működik fürtözött környezetekben.

```
$Helper->ConfigSettingChange(
    Valid => 1,             # (optional) enable or disable setting
    Key   => 'MySetting',   # setting name
    Value => { ... } ,      # setting value
);
```

**CustomCodeActivate()** Ez a függvény átmenetileg egyéni kódot fog felvenni a rendszerbe. Például ezt használhatja egy másik osztályból származó szubrutin felüldefiniálásához. Ez a változtatás megmarad a teszt emlékeztetőjénél is. Az összes kód eltávolításra kerül, amikor a `Helper` objektum megsemmisül.

**Megjegyzés:** Ne feledje, hogy ez jelenleg nem működik fürtözött környezetekben.

```
$Helper->CustomCodeActivate(
    Code => q^
use Kernel::System::WebUserAgent;
package Kernel::System::WebUserAgent;
use strict;
use warnings;
{
    no warnings 'redefine';
    sub Request {
        my $JSONString = '{"Results":{},"ErrorMessage":"","Success":1}';
        return (
            Content => \$JSONString,
            Status  => '200 OK',
        );
    }
}
1;^,
    Identifier => 'News',    # (optional) Code identifier to include in file␣
↪name
);
```

**`ProvideTestDatabase()`** Ez a függvény egy átmeneti adatbázist fog biztosítani a teszthez. Először határozza meg a teszt adatbázis beállításait a `Kernel/Config.pm` fájlban, azaz:

```
$Self->{TestDatabase} = {
    DatabaseDSN  => 'DBI:mysql:database=otrs_test;host=127.0.0.1;',
    DatabaseUser => 'otrs_test',
    DatabasePw   => 'otrs_test',
};
```

A metódushívás felül fogja bírálni a globális adatbázis-beállítást a teszt időtartama alatt, azaz az átmeneti adatbázis fogja fogadni az összes olyan hívást, amelyet a `DBObject` rendszer küldött át.

Az összes adatbázis-tartalom automatikusan eldobásra kerül, amikor a `Helper` objektum megsemmisül.

Ez a metódus `undef` értéket ad vissza abban az esetben, ha a teszt adatbázis nincs beállítva. Ha be van állítva, de a mellékelt XML nem olvasható vagy nem hajtható végre, akkor a metódus `die()` függvénye lesz meghívva a teszt hibával történő megszakításához.

```
$Helper->ProvideTestDatabase(
    DatabaseXMLString => $XML,       # (optional) OTRS database XML schema␣
↪to execute
                                     # or
    DatabaseXMLFiles => [            # (optional) List of XML files to load␣
↪and execute
        '/opt/otrs/scripts/database/otrs-schema.xml',
        '/opt/otrs/scripts/database/otrs-initial_insert.xml',
    ],
);
```

FEJEZET 7

# További erőforrások

**otrs.com** Az OTRS weboldala a forráskóddal, dokumentációval és hírekkel a www.otrs.com címen érhető el. Itt a hivatalos szakmai szolgáltatásokkal és az OTRS adminisztrátorképzési szemináriumokkal kapcsolatos információkat is megtalálja az OTRS csoporttól, az OTRS készítőjétől.

**Internetes API könyvtár** Az OTRS fejlesztői API dokumentáció elérhető a Perl és a JavaScript programnyelvekhez.

**Fejlesztői levelezőlista** Az OTRS fejlesztői levelezőlista a https://lists.otrs.org/ címen érhető el.