Magazine
Java Authentication and Authorization Service

Óbudai Egyetem, Java Enterprise Edition
Műszaki Informatika szak
Labor 7

Bedők Dávid
2018-02-11
v1.0

Magazin

Bejelentkezéssel védett tartalmak kezelése

Feladat: a *BookStore* blueprint mintájára készítsünk el egy Magazinok CRUD műveletei köré épített Java Enterprise alkalmazást, mely bizonyos műveletek elvégzését sikeres bejelentkezéshez és jogosultság meglétéhez/birtoklátáshoz köti.

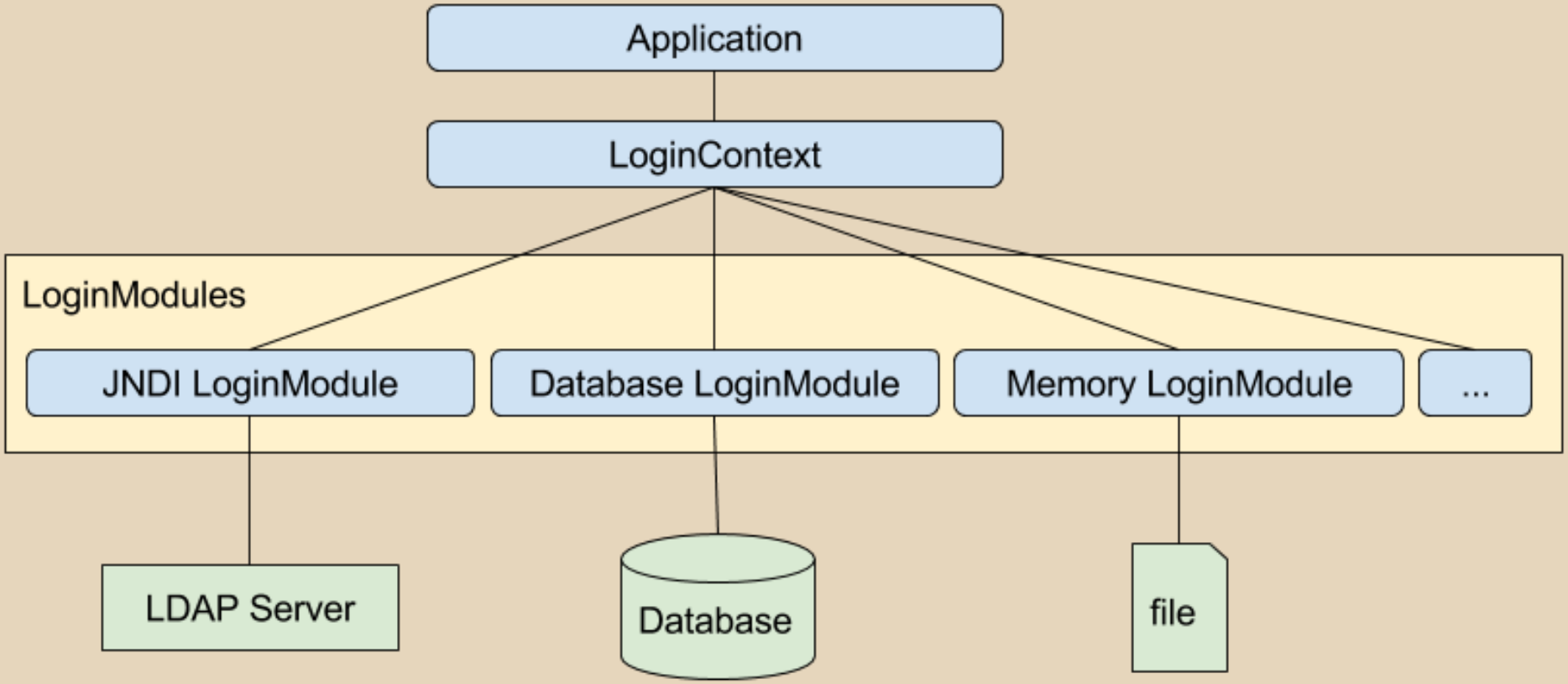
- A stílus lapok (`css`) elérhetőek bárki számára.
- A lekérdező műveletek csak sikeres bejelentkezés után érhetőek el (`maguser` szerepkör/jogosultság).
- A módosító műveletek (új magazin létrehozása, meglévő szerkesztése és törlése) kizárólag a `magadmin` szerepkörrel/jogosultsággal rendelkező felhasználók számára érhető el.

`[gradle|maven]\jboss\magazine`

Java Authentication and Authorization Service

- **Hitelesítés** / **Authentication**: a hitelesítés folyamata során a felhasználó identitása ellenőrzésre, megerősítésre kerül. Sokféle hitelesítési módszer létezik, mindegyiknek megvan a maga előnye és esetleges biztonság kockázati hátránya (jelszó begépelése, azonosító kártya lehúzása/érintése, biometrikus szkenner alkalmazása). A felhasználó lehet élő ember, de lehet akár egy másik számítógépes folyamat is.
- **Engedélyezés** / **Authorization**: az engedélyezési folyamat során egy már sikeresen hitelesített felhasználó jogosultsága kerül ellenőrzésre. Ez alapján a felhasználó adott erőforráshoz hozzáférhet (vagy nem férhet hozzá) avagy adott műveletet elvégezhet (vagy nem végezhet el). Egy rendszernek általában több modulja van, és nem minden modulhoz férhet hozzá minden felhasználó.
- **JaaS**: a `JaaS` egy szabványos keretrendszer és egy API a felhasználók hitelesítéséhez és engedélyezéséhez, legyenek a felhasználók emberek avagy gépek[1].

[1] A `JaaS` egy ún. **Pluggable Authentication Module** (`PAM`) keretrendszer.



A **LoginModule**-ok egy része ellenőrzi a felhasználó név-jelszó párt, mások hang vagy ujjlenyomat alapján dolgoznak, megint mások valamilyen kulcs birtoklásáról győződnek meg matematikailag megcáfolhatatlan módon.

Realm - Birodalom

A **Realm**-ek tekinthetőek olyan felhasználó név-jelszó párosok "adatbázisának", melyek azonosítják egy (vagy több) (web)alkalmazás aktív felhasználóit, illetve az általuk birtokolt szerepkörök összerendelését (illetve biztosítják az elérhető szerepkörök (*role*) listáját is). Egy felhasználó több szerepkörrel is rendelkezhet egyszerre.

A hitelesítéshez szükséges információkat...

- **JDBCRealm**: ...relációs adatbázisból `JDBC`-n keresztül éri el.
- **DataSourceRealm** ...relációs adatbázisból `JNDI`-on keresztül elért `JDBC` *DataSource*-on keresztül éri el.
- **JNDIRealm** ...`LDAP` alapú rendszerből `JNDI`-on keresztül éri el.
- **MemoryRealm** - memóriában tárolt, `XML` dokumentumból inicializált adatokként éri el.
- **JAASRealm** - egy bekonfigurált `JaaS` keretrendszeren keresztül éri el.

Mit kell tudnia egy ilyen framework-nek?

Webalkalmazás és authentikáció kontextusában

Az alkalmazásnak vannak **public** és **protected** erőforrásai (weboldalai). A nyilvános elemeket bejelentkezés nélkül el lehet érni (ilyen alapértelmezés szerint a *login* oldal, de más erőforrások is lehetnek (pl. képek, stílus lapok, vagy akár nem szenzitív adatot tartalmazó lapok), míg a védett elemeket csak sikeres authentikációt követően.

Ha egy védett weboldalt meglátogatunk, akkor a framework megnézi "hitelesítve vagyunk-e", és ha nem, átdob minket a bejelentkező képernyőre (automatikusan). Sikeres bejelentkezést követően az eredetileg meglátogatni kívánt oldalra irányít minket (ez az ún. *landing page*).

Mit kell tudnia egy ilyen framework-nek?

Webalkalmazás és authorization kontextusában

Bejelentkezést követően bizonyos műveletek csak bizonyos szerepkörű felhasználók számára érhető el. E védett **műveleteket szerepkörökhöz rendeljük**, és csak akkor jelennek meg/érhetőek el, ha a bejelentkezett felhasználó a megadott szerepkör birtokában van (*authorizáció*).

- Magukat a védett erőforrásokat is köthetjük egy konkrét szerepkörhöz (ez általában egy általános szerepkör, mellyel a legtöbb felhasználó rendelkezik, de pl. egy *Admin felületet* csak az adminisztrátorok számára teszünk elérhetővé).
- Egy elérhető erőforráson belül bizonyos műveleteket teszünk jogosultság/szerepkör függvényében elérhetővé.

Nem kell saját *LoginModule*-t készítenünk, választhatunk előre elkészítettek közül, melyeket paraméterek segítségével igényeinkre szabunk.

- PostgreSQL-ben tárolt felhasználók/jelszavak/szerepkörök és összerendeléseik használata
- Konfiguráció: lekérdezések beállítása JBoss számára, melyeket automatikusan használjon authentikáció és authorizáció során
- Saját webes bejelentkezés készítése (`XHTML` űrlap alapú)
- Védett erőforrások (bejelentkezéshez kötött) és védett műveletek (szerepkörhöz kötött) definiálása

PostgreSQL lekérdezések naplózása

Adatbázis oldalon

Mivel a lekérdezeket automatikusan a JBoss fogja végrehajtani, fejlesztési szempontból nagyon hasznos ha ezen automatizmus "másik oldalán" tudjuk naplózni az így végrehajtott lekérdezéseket.

Napló file: `[PG-HOME]/data/pg_log/postgresql-[YYYY-MM-DD]_[random].log` Konfigurációs file: `[PG-HOME]/data/postgresql.conf`

```
1 log_statement = 'all' # none, ddl, mod, all
```

postgresql.conf

PostgreSql újraindítása:

- Windows service-ek között megtalálható (Stop/Start)
-
```
1 [PG-HOME]/bin/pg_ctl -D [PG-HOME]/data/ restart
```

PG_HOME környezeti változó:

- WINOS: c:\ProgramFiles\PostgreSQL\[version]
- MACOS: /Library/PostgreSQL/[version]

Adatbázis

'A' minta

Más elnevezések is lehetségesek, de egyezzünk meg abban, hogy a **user**-ek számára lehetséges a hitelesítés (pl. jelszó vagy annak lenyomatának ellenőrzése alapján), és az alkalmazás az egyes engedélyeket (pl. erőforrások elérése, műveletek végrehajtása) **role**-okhoz köti (ez mindegyik bemutatott mintára igaz lesz).

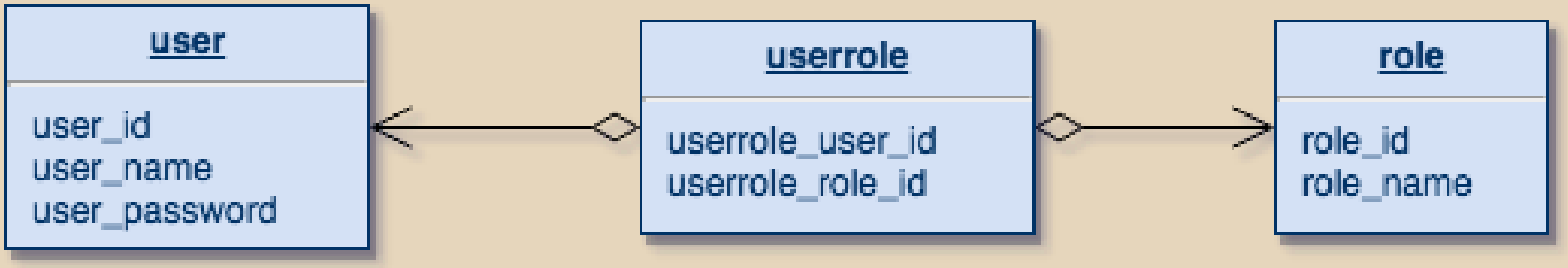
Az 'A' minta előnye az egyszerűsége. Összesen két tábla, de ha a `user_role_id` helyett egy `user_role`-t tárolunk (egy hozzá tartozó index-el), akkor egy táblával is megoldható. Komoly hátrányok is látszódnak azonban: minden felhasználó csak egy szerepkör tagja lehet, így ezen schema alkalmazása esetén a fejlesztett alkalmazásban a jogosultságkezelés várhatóan komplexebb lesz:

- ha egy művelet több szerepkör által is elvégezhető, ezen szerepkörök mindegyike felsorolásra kell hogy kerüljön az engedélyezés során
- az alkalmazás jogosultságkezelése függ a tárolásból származó korlátoktól (kerülendő)

Adatbázis

'B' minta

A felhasználók és a szerepkörök közé egy kapcsolótábla beillesztésével lehetőséget kapunk arra, hogy egy felhasználó több szerepkörrel rendelkezzen egyszerre, ezáltal az alkalmazásban a jogosultságok kezelése egyszerűbb lesz, hiszen könnyedén definiálhatunk szerepkört akár minden végrehajtható művelet vagy műveletcsoport számára.

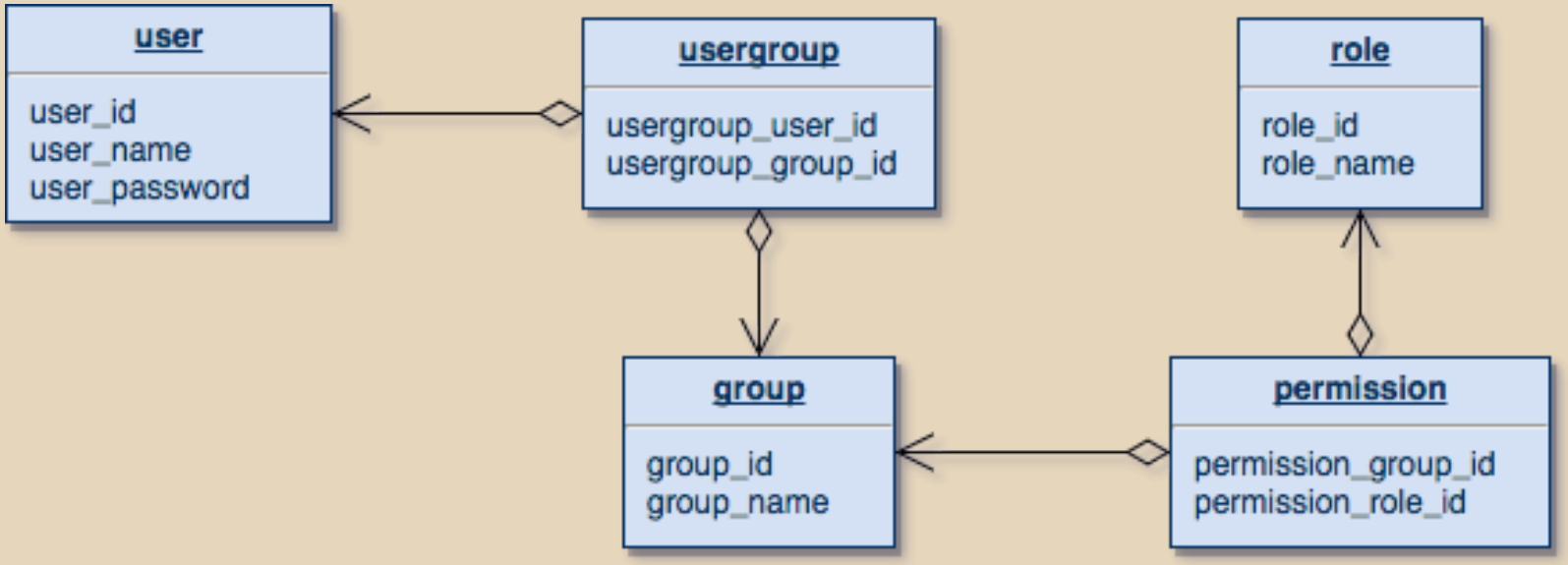


Idővel kialakul, hogy ha egy *manager* érkezik a vállalathoz, akkor számára mely N darab szerepkör hozzárendelése lesz szükséges, és melyek kellenek akkor ha az új munkavállaló egy *salesman*.
A módszer hátránya hogy lehetőséget ad az adminisztrációs hibák elkövetésére. Mi történik akkor ha egy új rendszer vagy komponens is üzembe áll? Az új munkavállalók megkapják az ehhez szükséges szerepköröket, a régiek viszont csak ad-hoc jelleggel, amikor épp jelzik hogy nem érnek el valamilyen műveletet. Ez könnyen káoszhoz vezethet.

Adatbázis

'C' minta

A 'B' mintánál felvetett probléma megoldására érdemes a szerepköröket csoportosítani, és adatbázisban tárolni hogy egy adott munkakörhöz milyen jogosultságok tartoznak. A szerepkör jogosultságainak változása nem okoz anomáliát a rendszerben. Ennek a mintának is van azonban hátránya: egyedi konfiguráció kialakításához saját *csoportot* kell létrehozni.



A minták összehasonlítása kedvéért az eredeti táblák neveit nem változtattam meg (illetve fontos megjegyezni hogy az alkalmazásban minden esetben a `role` tábla elemeit használjuk), azonban a köznapi használatban a `group`-ot nevezzük *szerepkörnek*, míg a `role` táblának jobb név volna `jogosultság`.

Adatbázis

Schema létrehozása


```
1  CREATE TABLE magazinecategory ( [..] );
2  CREATE TABLE magazine ( [..] );
3
4  CREATE TABLE appuser (
5    appuser_id SERIAL NOT NULL,
6    appuser_name CHARACTER VARYING(100) NOT NULL,
7    appuser_password CHARACTER VARYING(100) NOT NULL,
8    CONSTRAINT PK_APPUSER_ID PRIMARY KEY (appuser_id)
9  );
10 CREATE UNIQUE INDEX UI_APPUSER_NAME ON appuser USING
11
12 CREATE TABLE role (
13   role_id SERIAL NOT NULL,
14   role_name CHARACTER VARYING(100) NOT NULL,
15   CONSTRAINT PK_ROLE_ID PRIMARY KEY (role_id)
16 );
17
18 CREATE TABLE userrole (
19   userrole_id SERIAL NOT NULL,
20   userrole_appuser_id INTEGER NOT NULL,
21   userrole_role_id INTEGER NOT NULL,
22   CONSTRAINT PK_USERROLE_ID PRIMARY KEY (userrole_id),
23   CONSTRAINT FK_USERROLE_USER FOREIGN KEY (userrole_appuser_id)
24     REFERENCES appuser (appuser_id) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT,
25   CONSTRAINT FK_USERROLE_ROLE FOREIGN KEY (userrole_role_id)
26     REFERENCES role (role_id) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT
27 );
```

A Magazin project megoldása során a **B mintát** valósítjuk meg, mely általános célra már elégséges, és a komplexitása nem veszi el a figyelmet a lényegi, konfigurációs részektől. A user tábla név a keyword jelleg miatt kerülendő, így a felhasználókat az appuser táblában fogjuk tárolni.
A jelszót *plaintext* formában fogjuk tárolni első körben.

Adatbázis

Kezdeti adatok létrehozása


```
1  INSERT INTO role (role_id, role_name) VALUES (0, 'maguser');
2  INSERT INTO role (role_id, role_name) VALUES (1, 'magadmin');
3
4  SELECT SETVAL('role_role_id_seq', COALESCE(MAX(role_id), 1) ) FROM role;
5
6  INSERT INTO appuser (appuser_id, appuser_name, appuser_password) VALUES (0, 'alice',
       'a123');
7  INSERT INTO appuser (appuser_id, appuser_name, appuser_password) VALUES (1, 'bob',
       'b123');
8  INSERT INTO appuser (appuser_id, appuser_name, appuser_password) VALUES (2, 'charlie',
       'c123');
9
10 SELECT SETVAL('appuser_appuser_id_seq', COALESCE(MAX(appuser_id), 1) ) FROM appuser;
11
12 INSERT INTO userrole (userrole_appuser_id, userrole_role_id) VALUES (0, 0);
13 INSERT INTO userrole (userrole_appuser_id, userrole_role_id) VALUES (0, 1);
14 INSERT INTO userrole (userrole_appuser_id, userrole_role_id) VALUES (1, 0);
15 INSERT INTO userrole (userrole_appuser_id, userrole_role_id) VALUES (2, 0);
16
17 [..]
```

Mind a három létrehozott felhasználó rendelkezik a **maguser** szerepkörrel, *alice* azonban e mellett még **megadmin** is.

Datasource konfigurálása

JBoss

```
<subsystem xmlns="urn:jboss:domain:datasources:1.2">
    <datasources>
        <datasource jndi-name="java:jboss/datasources/magazineds"
             pool-name="MagazineDSPool" enabled="true" use-java-context="true">
            <connection-url>jdbc:postgresql://localhost:5432/magazinedb</connection-url>
            <driver>postgresql</driver>
            <security>
                <user-name>magazine_user</user-name>
                <password>123topSECret321</password>
            </security>
            <validation>
                <check-valid-connection-sql>SELECT 1</check-valid-connection-sql>
                <validate-on-match>true</validate-on-match>
                <background-validation>false</background-validation>
            </validation>
            <statement>
                <share-prepared-statements>false</share-prepared-statements>
            </statement>
        </datasource>
    </datasources>
</subsystem>
```

standalone.xml

A *datasource* JNDI neve `java:jboss/datasources/magazineds` lesz, melyre nem csupán az alkalmazás perzisztens rétegében, de a *Security Domain* konfiguráció során is hivatkozni fogunk.

Security Domain és Login Module konfigurálása

JBoss

```
1  <subsystem xmlns="urn:jboss:domain:security:1.2">
2      <security-domains>
3          <security-domain name="mag-security-db-domain">
4              <authentication>
5                  <login-module code="Database" flag="required">
6                      <module-option name="dsJndiName"
                           value="java:jboss/datasources/magazineds"/>
7                      <module-option name="principalsQuery" value="SELECT
                           appuser_password FROM appuser WHERE appuser_name = ?"/>
8                      <module-option name="rolesQuery" value="SELECT role_name, 'Roles'
                           FROM userrole INNER JOIN appuser ON (appuser_id =
                           userrole_appuser_id) INNER JOIN role ON (role_id =
                           userrole_role_id) WHERE appuser_name = ?"/>
9                  </login-module>
10             </authentication>
11         </security-domain>
12     </security-domains>
13 </subsystem>
```

standalone.xml

A *Security Domain* JNDI neve java:/jaas/**mag-security-db-domain** lesz.
A *LoginModule* code mezőnek konstansa mögött (Database) egy osztály van, mely megadása ugyanolyan jó (pl. UserRoles esetén org.jboss.security.auth.spi.UsersRolesLoginModule, Database esetén pedig org.jboss.security.auth.spi.DatabaseServerLoginModule).

SQL konfiguráció

Megbízók és *Szerepkörök* lekérdezések

Pricipals query: A lekérdezésnek vissza kell adnia a jelszó összehasonlításához szükséges értéket a felhasználó egyedi azonosítója függvényében. Ha nem definiálják, értéke a `select Password from Principals where PrincipalID=?` lesz.

```
1 SELECT appuser_password
2 FROM appuser
3 WHERE appuser_name = ?
```

Roles query: A lekérdezésnek vissza kell adnia az összes szerepkört a már hitelesített felhasználó egyedi azonosítójának függvényében. Ha nem definiálják, értéke a `select Role, RoleGroup from Roles where PrincipalID=?` lesz. A `RoleGroup` oszlopnak minden esetben `Roles`-t kell visszaadnia (ez JBoss specifikus, figyeljünk a nagy 'R' betűre is!).

```
1 SELECT
2   role_name,
3   'Roles'
4 FROM userrole
5   INNER JOIN appuser ON
6     (appuser_id = userrole_appuser_id)
7   INNER JOIN role ON
8     (role_id = userrole_role_id)
9 WHERE appuser_name = ?
```

Egy *Security Domain* több *Login Module*-t is egységbe zárhat. A modulok közötti függőséget a **Control Flag** definiálja.

```
1 <login-module code="..." flag="required">...</login-module>
```

- `required`: A modul sikeressége **kötelező**. Ha a modul hitelesség vizsgálata elbukik, a teljes domain hitelesség vizsgálata is el fog bukni. A domain-ben található további modulok vizsgálata meg fog történni, tekintet nélkül ezen modul sikerességére (ez magasabb fokú biztonságot jelent, mivel nem lehet pl. processzor terhelés vagy válaszidő átlagból következtetni arra, hogy melyik modul hitelesség vizsgálata bukott el).
- `requisite`: A module sikeressége **szükséges**. Ha a modul hitelesség vizsgálta elbukik, a teljes domain hitelesség vizsgálta is azonnal el fog bukni (a további modulok már nem lesznek megszólítva). Ott érdemes használni, ahol egy-egy modul érzékeny pl. a DOS támadásra, ezért egy előtte lévő (jobban terhelhető) modul megvédi ettől.
- `sufficient`: A modul sikeressége **elegendő** lehet, de nem kötelező! Ha a modul hitelesség vizsgálata sikeres, további modulok vizsgálata már nem fog megtörténni, viszont ha elbukik, akkor a *login stack* további eleme(i) (következő eleme) még lefut(nak).
- `optional`: A modul sikeressége **opcionális**, nem kötelező. A modul hitelesség vizsgálatának eredményétől függetlenül a domain megvizsgálja a *login stack* következő elemét.

Jelszó hash tárolása és ellenőrzése

JBoss

Egy alkalmazásnak legtöbbször nem megengedett/illendő a felhasználó által használt jelszavakat tárolni. Ez természetesen nem csupán etikai, hanem komoly biztonságtechnikai kérdés is. Gyakori megoldás a jelszavak helyett azok lenyomatának tárolása. A hash-eket hashfüggvények állítják elő (pl. SHA család, MD5, stb.), jellemzőjük hogy a lenyomat a gyakorlatban tökéletes validálásra, de elvben visszafordíthatatlan a konverzió (a hash-hez megfelelő jelszót találni jó esetben mai körülmények között lehetetlen). JBoss esetén a `hashUserPassword` kapcsolóval állíthatjuk be a *Login Module*-t a hash jelszavak kezelésére. Az adatbázis oldalon az `initial-data.sql`-ben át kell írni a *plaintext* jelszavakat (pl. az 123 MD5-hex formátumban 202cb962ac59075b964b07152d234b70).

```
1 <login-module code="Database" flag="required">
2     <module-option name="dsJndiName"
          value="java:jboss/datasources/magazineds"/>
3     <module-option name="principalsQuery" value="[..]"/>
4     <module-option name="rolesQuery" value="[..]"/>
5     <module-option name="hashAlgorithm" value="MD5"/>
6     <module-option name="hashEncoding" value="hex"/>
7     <module-option name="hashUserPassword" value="true"/>
8 </login-module>
```

standalone.xml

```
1  <management>
2      <security-realms>
3          <security-realm name="mag-realm">
4              <authentication>
5                  <jaas name="mag-security-db-domain"/>
6              </authentication>
7          </security-realm>
8      </security-realms>
9  </management>
10 [..]
11
12 <profile>
13     [..]
14     <subsystem xmlns="urn:jboss:domain:logging:1.5">
15         [..]
16         <logger category="org.jboss.security">
17             <level name="TRACE"/>
18         </logger>
19         [..]
20     </subsystem>
21 </profile>
```

Realm-re az alkalmazás konfigurációjának szintjén elsősorban a BASIC authentikácó esetén van szükség (megvizsgáljuk, de nem alkalmazzuk majd). A *Realm*-ek is hasonlóan konfigurálhatóak mint a *Login Module*-ok (de saját XSD-vel rendelkezik), azonban van lehetőség arra hogy egy birodalom hivatkozzon egy modulra, illetve fordítva is igaz (természetesen nem egyszerre..).

Ha szeretnénk részletes információt szerezni az authentikáció és authorizáció lépéseiről, érdemes az alábbi csomag bejegyzéseit részletesebb szinten naplózni (hibakeresés során fontos, *production* rendszeren ez a naplózási szint nem javasolt).

standalone.xml

Webalkalmazás felkészítése JaaS számára

- **Bejelentkező oldal** készítése
  - Űrlap egy név és egy jelszó mezővel.
- **Bejelentkezési hiba oldal** készítése
  - Sikertelen bejelentkezést követően ez töltődik be.
  - Általában visszadobjuk a Bejelentkező oldalra egy hibaüzenet kíséretében (vagy a megjelenő oldalról link segítségével lehet a Bejelentkező oldalt ismét betölteni).
- **Kijelentkező oldal/servlet** készítése
  - A bejelentkezett felhasználó kiléptetése a rendszerből. Általában cél a Bejelentkező oldalra való visszadobás sikeres kijelentkeztetés esetén, de van ahol egy elköszönő oldal jelenik meg.
- **Hiba oldal(ak)** készítése
  - Minden egyéb *Security* hiba során betöltődő oldal(ak). HTTP hibakódhoz köthetőek.

Bejelentkezési servlet

mag-weblayer project

```
1  package hu.qwaevisz.magazine.weblayer.servlet;
2  [..]
3  @WebServlet("/Login")
4  public class LoginServlet extends HttpServlet implements
       LoginAttribute {
5
6    @Override
7    protected void doGet(HttpServletRequest request,
         HttpServletResponse response) throws ServletException,
         IOException {
8      request.setAttribute(ATTR_USERNAME, "");
9      request.setAttribute(ATTR_ERROR, "");
10     RequestDispatcher view =
           request.getRequestDispatcher("login.jsp");
11     view.forward(request, response);
12   }
13
14
15 }
```

Az `ATTR_USERNAME` és az `ATTR_ERROR` attribútumokban a bejelentkezési oldalon beírt felhasználó név, és az esetleges hibaüzenet szövegét tároljuk. A `login.jsp` dolgozza fel őket, hibás authentikáció esetén fogjuk tölteni az értéküket.

LoginServlet.java

Bejelentkezési oldal

mag-weblayer project

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
2  <%@ page import="hu.qwaevisz.magazine.weblayer.common.LoginAttribute" %>
3  [..]
4  <link rel="stylesheet" type="text/css" href="style/page.css" />
5  [..]
6  <body>
7    <%
8      String userName = (String) request.getAttribute(LoginAttribute.ATTR_USERNAME);
9      String errorMessage = (String) request.getAttribute(LoginAttribute.ATTR_ERROR);
10   %>
11   <form action="j_security_check" method="post">
12     <fieldset>
13       <legend>Login</legend>
14       <div class="error"><%= errorMessage %></div>
15       <div>
16         <label for="username">Username: </label>
17         <input type="text" name="j_username" id="username" value="<%= userName %>" />
18       </div>
19       <div>
20         <label for="password">Password: </label>
21         <input type="password" name="j_password" id="password" />
22       </div>
23       <in
24     </fie
25   </form>
26 </body>
27 </html>
```

Public resource!

Ha van hibaüzenet (errorMessage), illetve már korábban beírt felhasználónév (userName), akkor ezeket megjelenítjük a bejelentkezési oldalon (hibás authentikáció utáni újratöltés esete lesz). A `j_*` elnevezések (servlet url/action és az űrlap felhasználónév és jelszó mezőjének nevei) az API által előírtak, ebben a formában kell őket alkalmazni.

login.jsp

Bejelentkezési hiba servlet

mag-weblayer project

```
1  package hu.qwaevisz.magazine.weblayer.servlet;
2  [..]
3  @WebServlet("/LoginError")
4  public class LoginErrorServlet extends HttpServlet implements
       LoginAttribute {
5    @Override
6    protected void doPost(HttpServletRequest request,
         HttpServletResponse response) throws ServletException,
         IOException {
7      final String userName = request.getParameter("j_username");
8
9      request.setAttribute(ATTR_USERNAME, userName);
10     request.setAttribute(ATTR_ERROR, "Login failed");
11
12     RequestDispatcher view =
           request.getRequestDispatcher(Page.LOGIN.getJspName());
13     view.forward(request, response);
14   }
15
16 }
```

Hiba esetén a `j_security_check` servlet átdobja a kérést a bejelentkezési hiba oldalra (/LoginError jelen esetben). A kérésből kikérhetjük az űrlap mezőit (jelen esetben a nevet kikérjük, hogy át tudjuk adni a `login.jsp`-nek).

LoginErrorServlet.java

Kijelentkezési servlet

mag-weblayer project

```
1  @WebServlet("/Logout")
2  public class LogoutServlet extends HttpServlet
3    @Override
4    protected void doGet(HttpServletRequest request,
         HttpServletResponse response) throws ServletException,
         IOException {
5      response.setHeader("Cache-Control", "no-cache, no-store");
6      response.setHeader("Pragma", "no-cache");
7      response.setHeader("Expires", new
           java.util.Date().toString());
8      if (request.getSession(false) != null) {
9        request.getSession(false).invalidate();
10     }
11     if (request.getSession() != null) {
12       request.getSession().invalidate();
13     }
14     request.logout();
15     response.sendRedirect("list.jsp");
16   }
17 }
```

Ez egyfajta furcsa *best practice*, mely figyelembe veszi néhány alkalmazás szerver múltbéli hibáit is.

A **redirect** a servlet végén bármely védett oldalra mutathat. Ha sikeres volt a kijelentkeztetése a felhasználónak, a védett URL betöltése ki fogja váltani a bejelentkezési oldal betöltését.

LogoutServlet.java

Hiba oldal(ak)

mag-weblayer project

```java
@WebServlet("/Error")
public class ErrorServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest request,
      HttpServletResponse response) throws ServletException,
      IOException {
    final RequestDispatcher view =
        request.getRequestDispatcher("error.jsp");
    view.forward(request, response);
  }
}
```

ErrorServlet.java

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="style/page.css" />
<title>:: Error ::</title>
</head>
<body>Error</body>
</html>
```

Security-vel kapcsolatos hibák esetén definiálhatóak hiba oldalak (akár több is), melyek HTTP hibakódhoz köthetőek.

error.jsp

Webalkalmazás konfiguráció

Védett erőforrások

```
<web-app [..]>

  <security-constraint>
    <display-name>Magazine protected security constraint</display-name>
    <web-resource-collection>
      <web-resource-name>Protected Area</web-resource-name>
      <url-pattern>/*</url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
      <http-method>PUT</http-method>
      <http-method>DELETE</http-method>
      <http-method>HEAD</http-method>
      <http-method>OPTIONS</http-method>
      <http-method>TRACE</http-method>
    </web-resource-collection>
    <auth-constraint>
      <role-name>maguser</role-name>
      <role-name>magadmin</role-name>
    </auth-constraint>
    <user-data-constraint>
      <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
  </security-constraint>
  [..]
</web-app>
```

Minden oldal (minden `HTTP` method esetén) védett területen van (kivéve majd az, melyet nyitottá teszünk). Az oldalak `maguser` vagy `magadmin` szerepkör birtokában érhetőek el. Az adatbázisban is ezek a szerepkörök szerepelnek.

`web.xml`

Webalkalmazás konfiguráció

Nyilvános erőforrások

```
<web-app [..]>
  [..]
  <security-constraint>
    <display-name>Magazine public security
        constraint</display-name>
    <web-resource-collection>
      <web-resource-name>Public</web-resource-name>
      <url-pattern>/style/*</url-pattern>
      <url-pattern>/Logout</url-pattern>
      <http-method>GET</http-method>
    </web-resource-collection>
  </security-constraint>
  [..]
</web-app>
```

web.xml

A stílus lapok lekérése és a kijelentkező servlet elérhető bejelentkezés nélkül is. A *Login page* (később konfiguráljuk) alapértelmezés szerint elérhető, viszont az általa hivatkozott erőforrásokról (pl. stílus lapok) már gondoskodnunk kell.

Webalkalmazás konfiguráció

FORM auth-method

```
1  <web-app [..]>
2    [..]
3    <login-config>
4      <auth-method>FORM</auth-method>
5      <form-login-config>
6        <form-login-page>/Login</form-login-page>
7        <form-error-page>/LoginError</form-error-page
8      </form-login-config>
9    </login-config>
10
11   <security-role>
12     <description>Generic user</description>
13     <role-name>maguser</role-name>
14   </security-role>
15
16   <security-role>
17     <description>Administrator</description>
18     <role-name>magadmin</role-name>
19   </security-role>
20
21   <error-page>
22     <error-code>403</error-code>
23     <location>/Error</location>
24   </error-page>
25
26 </web-app>
```

Az authentikáció formája többféle lehet. Saját űrlap készítése esetén (leggyakoribb) a FORM a helyes érték, de lehet még BASIC, DIGEST és CLIENT-CERT is.

Az űrlap authentikáció tulajdonsága a bejelentkezési (form-login-page) és a bejelentkezési hiba (form-error-page) oldal. Ide a servlet *url pattern*-jét kell megadni (JSP esetén ez a JSP neve).

Az elérhető szerepköröket is fel kell sorolni a security-role *element* segítségével.

HTTP 403 (*Forbidden*) *status code* esetén megadott hiba oldal.

web.xml

BASIC authentikáció

Böngésző beépített bejelentkező ablakot dob fel (javascript), melyen a felhasználónév és a jelszó megadható. BASIC *auth method* esetén megadható a **realm** neve a web.xml-ben, azonban JBoss esetén ekkor is használható a jboss-web.xml, ahol a *Security Domain* JNDI nevének a megadása szükséges.

```
1 <web-app [..]>
2   [..]
3   <login-config>
4     <auth-method>BASIC</auth-method>
5     <realm-name>mag-realm</realm-name>
6   </login-config>
7   [..]
8 </web-app>
```

web.xml

Figyelem! Ez a web.xml minta **NEM** a *Magazine* project-hez tartozik.

A JBoss `standalone.xml`-ben beállított *Security Domain* `JNDI` neve.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <jboss-web>
3     <security-domain>java:/jaas/mag-security-db-domain</security-doma
4 </jboss-web>
```

`jboss-web.xml`

- Webalkalmazás leírója (`web.xml`) hivatkozhat a JBoss `standalone.xml` `mag-realm` nevű `security-realm`-re (`BASIC` auth esetén)
- A `security-realm` hivatkozik a `mag-security-db-domain` nevű `security-domain`-re (`standalone.xml`-en belül).
- A webalkalmazás JBoss specifikus leírója (`jboss-web.xml`) közvetlenül is hivatkozik a JBoss `standalone.xml` `mag-security-db-domain` nevű `security-domain` nevére (`JNDI` név alapján, `FORM` auth esetén)
- A `mag-security-db-domain` tartalmazza a `Database` *login module* konfigurációját, melyen keresztül a fizikai adatbázis (*DataSource* `JNDI` neve) és az azon végrehajtandó natív lekérdezések is elérhetőek.

1. Erőforrások korlátozása

A webalkalmazás *deployment descriptorán* keresztül (`web.xml`) meghatározzuk a képek, (X)HTML és JSP lapok (stb.) elérhetőségét. Mely HTTP műveleteken keresztül milyen szerepkörrel rendelkező felhasználók érhetnek ezekhez hozzá.

2. Felületi elemek korlátozása

A webalkalmazás JSP/JSF lapjain bizonyos felületi elemek szerepkörtől függően jelenhetnek meg, de gyakori a felületi elemek *enabled/disabled* állapotának változtatása is. Ezen a szinten az authorizáció UX kérdésnek tekinthető, **security szempontjából nincs jelentősége** (a böngésző számára leküldött (X)HTML lapokat a kliens bármikor átszerkesztheti)!

3. Felhasználói akciók authorizációja servlet oldalon

A webalkalmazás által szerepkörhöz kötött műveleteket ellenőrizni szükséges a *servlet* oldalon, még azelőtt hogy a kérések az EJB réteghez kerülnének. Ennek oka az alatta lévő rétegek védelme, az EJB contextus, tranzakció kezelés elindításának megakadályozása, felesleges körök futása (erőforrás pazarlás, rendszer robosztusságának növelése, stb.).

4. EJB műveletek authorizációja

Egy több rétegű monolitikus alkalmazásban az EJB műveletek nem csupán a *frontend* oldaláról lehet megközelíteni, így nem bízhatunk meg vakon abban, hogy minden bejövő kérés a GUI-ról fog indulni. Ez a legalsó szint, melyhez már csak ritkán, többnyire programozói hiba miatt szabad hogy olyan kérés kerüljön, melynek megbukik az authorizációja (kivételes esetek azért vannak, pl. session timeout is lejárhat).

Webalkalmazás authorizáció

Role-Based Access Control

Ha egy oldal/servlet elérhető a már sikeresen authentikált felhasználó számára, avagy az erőforrás nyilvános (vagyis ha már a `web.xml` konfigurálásán túl vagyunk), a következő lépés a servlet-ek és a weblapok bizonyos részeinek, műveleteinek szerepkörökhöz kötése. Ha megjelenítéssel korlátozzuk egy művelet végrehajtását, minden esetben ellenőrizzük le a szerepkör meglétét a servlet oldalon is még egyszer!

```
1 protected void doGet(HttpServletRequest request,
      HttpServletResponse response) throws ServletException,
      IOException {
2   if ( request.isUserInRole("magadmin") ) {
3     [..]
4   }
5   [..]
6 }
```

```
1 <% if (request.isUserInRole("magadmin")) { %>
2   <div>
3     <a href="Magazine?reference=-1&edit=1">Create</a> a brand new
          magazine.
4   </div>
5 <% } %>
```

Hasonlóan a webalkalmazás `jboss-web.xml` leírójához, ha az **EJB module** projektekben is szeretnék `Role-Based Access Control`-t (`RBAC`) használni, akkor meg kell adnunk a *Security Domain* nevét a `jboss-ejb3.xml` leíróban (ebben a file-ban nem a `JNDI` név kell).

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <jboss:ejb-jar [..]>
3     <assembly-descriptor>
4         <s:security>
5             <ejb-name>*</ejb-name>
6             <s:security-domain>mag-security-db-domain</s:security-do
7         </s:security>
8     </assembly-descriptor>
9 </jboss:ejb-jar>
```

jboss-ejb3.xml

Megjegyzés: Ha minden jogosultság ellenőrzés nélkül minden EJB metódus hívható, akkor nincs szükség a konfigurációra! Az itt bemutatott konfiguráció annyit mond, hogy minden EJB legyen védett, ezáltal lehetőségünk lesz annotációkon keresztül a forráskódban ezt finomhangolni. Fordított megoldás is lehetséges.

Magazin törlése EJB rétegben

mag-ejblayer project

```
1  @PermitAll
2  @Stateless(mappedName = "ejb/magazineFacade")
3  public class MagazineFacadeImpl implements MagazineFacade {
4
5    [..]
6    @Override
7    @RolesAllowed("magadmin")
8    public void removeMagazine(String reference) throws
         FacadeException {
9      try {
10       this.service.delete(reference);
11     } catch (final PersistenceServiceException e) {
12       LOGGER.error(e, e);
13       throw new FacadeException(e.getLocalizedMessage());
14     }
15   }
16
17
18
19 }
```

@DenyAll: egyik szerepkör számára sem érhető el
@PermitAll: minden szerepkör számára elérhető
@RolesAllowed("magadmin") vagy @RolesAllowed("magadmin", "maguser"): a megadott vagy felsorolt szerepkörök számára elérhető
Osztály szinten: minden metódusra érvényes

MagazineFacadeImpl.java

RedHat JBoss vs. Oracle WebLogic

Azonos koncepció, apró különbségek

	JBoss 6.4	**WebLogic 12.2.1**
web module container specifikus dd[2]	`weblogic-ejb-jar.xml`	`weblogic.xml`
ejb module container specifikus dd	`jboss-ejb3.xml`	`jboss-web.xml`
security role nevére nézett ismert limitációk		Kötőjelek (hyphens) használata esetén az Admin Console-ról nem kezelhető
security role	az **application role** megegyezik a **security role**-lal	az **application role** logikailag szétválik a **security role**-tól
security realm-ekre nézett ismert limitációk	Tetszőleges számú security realm is létezhet egyidőben.	Több lehet, de kizárólag egy lehet aktív (az inaktív semmire sem használható!
elnevezésbeli különbségek	**Login Module** (*Control Flag*-ekkel)	**Authentication Provider** (*Control Flag*-ekkel)
hivatkozás alkalmazásból	a Security Domain JNDI vagy rövid nevére	Nincs hivatkozás Security Realm-re (kizárólag az egyetlen aktív használható).

[2] deployment descriptor

Az **Application Role** logikailag szétválik a JBossnál megismert **Security Role**-tól. Egy *Application Role*-hoz egy vagy több *User*, vagy egy vagy több *Security Role* tartozhat. Az összerendelés megadása azonban kötelező a container specifikus *deployment descriptor*ban (különben nem lesz kapcsolat a *Security Realm*-el).

```
1 <security-role-assignment>
2   <role-name>ROLE_A</role-name>
3   <principal-name>PRINCIPAL_1</principal-name>
4   <principal-name>PRINCIPAL_2</principal-name>
5 </security-role-assignment>
```

- `ROLE_A`: **Application Role**, `web.xml`-ben és Java kódban használt szerepkör
- `PRINCIPAL_1` és `PRINCIPAL_2`: **Security Role** (ajánlott) vagy **Security User** neve, mely a *Security Realm*-ben van konfigurálva
- A konfiguráció WEB modul esetén a `weblogic.xml`-ben, míg EJB modul esetén a `weblogic-ejb-jar.xml`-ben kap helyet.

Login Module vs. Authentication Provider

JBoss vs. WebLogic

Login Module

- RedHat JBoss
- Több *Login Module* létezhet egy **Security Domain** alatt.
- A **Security Realm** kötődhet egy **Security Domain**-hez (és visszafelé is igaz ez).
- A **Security Realm** leginkább remote elérés esetén használatos.

Authentication Provider

- Oracle WebLogic
- Több *Authentication Provider* létezhet egy **Security Realm** alatt.
- A *Security Domain* elnevezés nem jelenik meg WebLogic esetén.

Az alapértelmezett `myrealm` *Security Realm*-hez fogunk tudni felvenni egy olyan *Authentication Provider*-t, mely saját adatbázis alapján hitelesít. Ezt, és az eredetileg is itt lévő `DefaultAuthenticator` *Control Flag*-jét átállítjuk `SUFFICIENT`-re (eredetileg `REQUIRED`).

WebLogic DataSource konfigurálása

Server Administration Console-on keresztül

- PostgreSQL JDBC library hozzáadása a domain classpath-ához
  - JDBC driver (pl. `postgresql-9.4-1201.jdbc41.jar`) másolása ide: `[MW\_HOME]\user\_projects\domains\mydomain\lib\`
  - Server újraindítása
- WebLogic Server Administration Console
  - Services | Data Sources | Configuration tab
    - New → Generic Data Source
      * Name: **magazineds**
      * JNDI name: `jdbc/datasource/magazineds`
      * Target: "myserver"
      * Database type: PostgreSQL
    - Next, Next..
      * Database name: **magazinedb**
      * Host name: **localhost**
      * Port: **5432** (default)
      * Database user name: **magazine_user**
      * Password: **123topSECret321**
    - Érdemes tesztelni a kapcsolatot a felületen keresztül

Ismétlés..

- Security Realms → `myrealm` kiválasztása
  - Providers tab → New
    - Name: **magazineauthenticator**
    - Type: **SQLAuthenticator**
  - `magazineauthenticator` kiválasztása
    - Configuration tab | Common tab
      - Control Flag: **SUFFICIENT**
      - Save
    - Configuration tab | Provider Specific tab
      - Plaintext Passwords Enabled (checked)
      - Data Source Name: **magazineds**
      - Password Style Retained (checked)
      - Descriptions Supported (unchecked)
      - SQL Queries... → következő oldal

SQL lekérdezések I

Configuration tab | Provider Specific tab

SQL Get Users Password (JBoss: **principalsQuery**)	`SELECT appuser_password FROM appuser WHERE appuser_name =?`
SQL Set User Password	`UPDATE appuser SET appuser_password =? WHERE appuser_name =?`
SQL User Exists	`SELECT appuser_name FROM appuser WHERE appuser_name =?`
SQL List Users	`SELECT appuser_name FROM appuser WHERE appuser_name LIKE?`
SQL Create User	`INSERT INTO appuser (appuser_id, appuser_name, appuser_password) VALUES (?,?,?)`
SQL Remove User	`DELETE FROM appuser WHERE appuser_name =?`
SQL List Groups	`SELECT role_name FROM role WHERE role_name LIKE?`
SQL Group Exists	`SELECT role_name FROM role WHERE role_name =?`
SQL Create Group	`INSERT INTO role (role_id, role_name) VALUES (?,?)`
SQL Remove Group	`DELETE FROM role WHERE role_name =?`
SQL Is Member	`SELECT appuser_name FROM userrole INNER JOIN appuser ON ( appuser_id = userrole_appuser_id ) INNER JOIN role ON ( role_id = userrole_role_id ) WHERE appuser_name =? AND role_name =?`

SQL lekérdezések II

Configuration tab | Provider Specific tab

SQLSQL List Member Groups (JBoss: **rolesQuery**, `RoleGroup` nélkül)	`SELECT role_name FROM userrole INNER JOIN appuser ON ( appuser_id = userrole_appuser_id ) INNER JOIN role ON ( role_id = userrole_role_id ) WHERE appuser_name =?`
SQL List Group Members	`SELECT appuser_name FROM userrole INNER JOIN appuser ON ( appuser_id = userrole_appuser_id ) INNER JOIN role ON ( role_id = userrole_role_id ) WHERE appuser_name LIKE? AND role_name =?`
SQL Remove Group Memberships	`DELETE FROM userrole WHERE userrole_role_id = ( SELECT role_id FROM role WHERE role_name =? ) AND userrole_appuser_id = ( SELECT appuser_id FROM appuser WHERE appuser_name =? )`
SQL Add Member To Group	`INSERT INTO userrole (userrole_appuser_id, userrole_role_id) VALUES (?,?)`
SQL Remove Member From Group	`DELETE FROM userrole WHERE userrole_role_id = ( SELECT role_id FROM role WHERE role_name =? ) AND userrole_appuser_id = ( SELECT appuser_id FROM appuser WHERE appuser_name =? )`
SQL Remove Group Member	`DELETE FROM userrole WHERE userrole_role_id = ( SELECT role_id FROM role WHERE role_name =? )`
Description specifikus	Kikapcsoltuk a *description* használatát.

Alice sikeres bejelentkezése

SQL log

```
1  SELECT appuser_password
2  FROM appuser
3  WHERE appuser_name = 'alice';
4
5  SELECT role_name
6  FROM userrole
7    INNER JOIN appuser ON ( appuser_id = userrole_appuser_id )
8    INNER JOIN role ON ( role_id = userrole_role_id )
9  WHERE appuser_name = 'alice';
10
11 SELECT role_name
12 FROM userrole
13   INNER JOIN appuser ON ( appuser_id = userrole_appuser_id )
14   INNER JOIN role ON ( role_id = userrole_role_id )
15 WHERE appuser_name = 'maguser';
16
17 SELECT role_name
18 FROM userrole
19   INNER JOIN appuser ON ( appuser_id = userrole_appuser_id )
20   INNER JOIN role ON ( role_id = userrole_role_id )
21 WHERE appuser_name = 'magadmin';
```

Lefut a **principal name**-ekre nézve is (`weblogic.xml`), lévén ezek lehetnek *Security Role*-ok és *Security User*-ek is!

WebLogic Security Realm ellenőrzése

Server Administration Console-on keresztül

Server restart

- ▷ Security Realms → `myrealm` kiválasztása
  - Users and Groups tab
    - Users tab

Name	Provider
alice	`magazineauthenticator`
bob	`magazineauthenticator`
charlie	`magazineauthenticator`
weblogic	`DefaultAuthenticator`

    - Groups tab (alice kiválasztása után)

Name	Provider
magadmin	`magazineauthenticator`
maguser	`magazineauthenticator`
...	`DefaultAuthenticator`

Magazin projekt

Forráskód módosításai WebLogic esetén

- Log4j lecserélése **JDK Logging**-ra
- Hibernate dependency lecserélése **Eclipselink**-re
- `LogoutServlet`-ben a `request.logout()` kitörlése
  - `NullPointerException`-t dob (ismert hiba)
- `mag-user` és `mag-admin` *Application Role*-ok használata `maguser` és `magadmin` helyett
  - pl.: `list.jsp`
    ```
    <% if (request.isUserInRole("mag-admin")) { %>
    ```
  - pl.: `MagazineFacadeImpl`
    ```
    @RolesAllowed("mag-admin")
    ```
- `jboss-web.xml` helyett `weblogic.xml`
- `jboss-ejb3.xml` helyett `weblogic-ejb-jar.xml`

Web alkalmazás konfiguráció

mag-weblayer project

```
1  <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
2    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
4    [..]
5    <security-constraint>
6      <display-name>Magazine protected security constraint</display-name>
7      [..]
8      <auth-constraint>
9        <role-name>mag-user</role-name>
10       <role-name>mag-admin</role-name>
11     </auth-constraint>
12     [..]
13   </security-constraint>
14   [..]
15   <security-role>
16     <description>Generic user</description>
17     <role-name>mag-user</role-name>
18   </security-role>
19   <security-role>
20     <description>Administrator</description>
21     <role-name>mag-admin</role-name>
22   </security-role>
23   [..]
24 </web-app>
```

Application Role

web.xml

WEB modul konfiguráció WebLogic esetén

mag-weblayer project

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90"
3    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4
5    <security-role-assignment>
6      <role-name>mag-user</role-name>
7      <principal-name>maguser</principal-name>
8    </security-role-assignment>
9    <security-role-assignment>
10     <role-name>mag-admin</role-name>
11     <principal-name>magadmin</principal-name>
12   </security-role-assignment>
13
14 </weblogic-web-app>
```

weblogic.xml

Application Role: mag-user és mag-admin
Security Role: maguser és magadmin (ezek vannak adatbázisban)

EJB module konfiguráció WebLogic esetén

mag-ejblayer project

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <weblogic-ejb-jar xmlns="http://www.bea.com/ns/weblogic/90"
       xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://www.bea.com/ns/weblogic/90
         http://www.bea.com/ns/weblogic/90/weblogic-ejb-jar.xsd">
4
5    <security-role-assignment>
6      <role-name>mag-user</role-name>
7      <principal-name>maguser</principal-name>
8    </security-role-assignment>
9    <security-role-assignment>
10     <role-name>mag-admin</role-name>
11     <principal-name>magadmin</principal-name>
12   </security-role-assignment>
13
14 </weblogic-ejb-jar>
```

Kizárólag a root tag-ben tér el a weblogic.xml-től.

weblogic-ejb-jar.xml