Security, Realms

- Zabezpečení webové vrstvy a EJB projektu
- Část nastavení specifická pro Glassfish, část dána Java EE
  - '*web.xml*'
  - '*glassfish-web.xml*'
  - dále nutno nastavit realm v admin. konzoli GF

Rozšíření DB modelu

- V DB modelu je třeba mít informace o uživatelích, které mohou přistupovat k aplikaci: přihlašovací jméno, heslo, skupinu
- Modře vždy ukázka na příkladu ze cvičení (navazuje na projekt `x33eja-jsf`)
- Přidat proměnné v entitě Person: `String login, String password` (+gettery,settery)
- Jako skupinu použijeme diskriminátor v inheritanci entity Person (student,teacher) – jméno diskriminátoru specifikujeme anotací třídy Person:

```
@DiscriminatorColumn(discriminatorType=DiscriminatorType.STRING, name="groupname")
```

Naplnění DB daty

- K příkladu je přiložen skript na vložení dat, ten je kvůli změně modelu potřeba upravit. Vložení dat je nutné, abychom se později mohli přihlásit.
- Stačí upravit data do tabulky person:

```
INSERT INTO person (birthnumber,groupname,surname,firstname,login,password)
VALUES (1, 'student', 'Novak', 'Frantisek', 'novak', 'n');
INSERT INTO person (birthnumber,groupname,surname,firstname,login,password)
VALUES (2, 'teacher', 'Kremen', 'Petr', 'kremen', 'k');
INSERT INTO person (birthnumber,groupname,surname,firstname,login,password)
VALUES (3, 'student', 'Blazkova', 'Hana', 'blazkova', 'b');
INSERT INTO person (birthnumber,groupname,surname,firstname,login,password)
VALUES (4, 'teacher', 'Kouba', 'Zdenek', 'kouba', 'k');
INSERT INTO person (birthnumber,groupname,surname,firstname,login,password)
VALUES (5, 'teacher', 'Valek', 'Frantisek', 'valek', 'v');
INSERT INTO person (birthnumber,groupname,surname,firstname,login,password)
VALUES (6, 'teacher', 'Smid', 'Marek', 'smid', 's');
INSERT INTO person (birthnumber,groupname,surname,firstname,login,password)
VALUES (7, 'student', 'Kozina', 'Jan', 'kozina', 'k');
```

Konfigurace Glassfish

- Je třeba vytvořit security realm v Glassfishi, což je serverově závislá operace, a ani Netbeans na to nemají podporu
- V administrační konzoli Glassfish (std. na portě 4848) vybereme v levém stromečku Configuration / Security / Realms / tlačítko New
- Realm si zde pojmenujeme, jako třídu vybereme JDBCRealm (čtení uživatelů z DB), JAAS context nastavíme na “jdbcRealm”, JDNI nastavíme na spojení s DB (JDBC resource), vyplníme jména a sloupce tabulek s informacemi o uživatelích, zbytek necháme prázdný. Jen ještě Digest algorithm je nutno vyplnit hashovací funkcí pro hesla (none, MD5...)
- Realm si pojmenujeme `eja_security`, vybereme `JDBCRealm`, a vyplníme:
  - `JAAS Context = jdbcRealm`
  - `JNDI = jdbc/jpa-example`
  - `User table = person`
  - `User name column = login`
  - `Password column = password`
  - `Group table = person`
  - `Group name column = groupname`
  - `Digest algorithm = none`

Konfigurace webového projektu

- Dále nastavíme security ve web.xml ve web. Projektu. Otevřeme web.xml a v grafickém editoru přepneme na záložku Security.
- V sekci Login configuration vybereme Form (přihlašovaní pomocí stránky) a nastavíme Form Login a Error Page, kde je formulář na přihlášení (viz příklad). Realm name musí odkazovat na vytvořený Realm v Glassfish konzoli.
- V sekci Security roles vyjmenujeme role, které chceme v aplikaci pro bezpečnost využívat (nemusejí se shodovat se skupinami v DB, mapování viz dále)
- Přidáme Security constraint, která řekne, pro které stránky musí být uživatel přihlášen. Vyplníme jméno, přidáme Web resource collection (např. URL pattern = `/*` pro ochranu všech stránek) a zapneme Authentication constraint, kde vyjmenujeme povolené role. Je zde také možno zapnout User data constraint, která v případě Transport guarantee = confidential vynutí použití SSL.

Konfigurace webového projektu

- web.xml / security:
- Login configuration: Form,
  Form Login Page = Form Error Page = `/login.xhtml`
  Realm name = `eja_security`
- Security roles: student, teacher
- New security constraint: name=”secured”,
  resource collection: “web sites”, URL pattern: “*”, all HTTP methods
  Enable authentication constraint: role names = student, teacher

Konfigurace webového projektu

- Dále nutno nastavit konfiguraci specifickou pro Glassfish v glassfish-web.xml. Opět v grafickém editoru v záložce Security:
- Zde nastavíme mapování
  skupiny (groups, v DB) → role (roles, zabezpečení aplikace)
- Do role `teacher` přidáme pouze skupinu `teacher`,
  do role `student` skupinu `student`

Vytvoření přihlašovací stránky

- Je nutno vytvořit stránku, na kterou bude uživatel přesměrován pokud není přihlášen, a snaží se přistoupit na zabezpečenou stránku. Přihlašovací stránka pošle vyplněné údaje Glassfishi, který je ověří proti nastavenému Realmu
- Ve stránce musí být formulář:

```
<form action="j_security_check" method="post">
  Login:    <input type="text" name="j_username" />
  Password: <input type="password" name="j_password" />
  <input type="submit" value="login" />
</form>
```

- V příkladu založíme JSF stránku login.xhtml, do které vložíme uvedený formulář

Použití security v JSF

- Ve webové vrstvě nejde příliš o zabezpečení, spíš o měnění vzhledu pro různé skupiny uživatelů – to lze zajistit XML atributem `rendered` u JSF komponent, který odkážeme na boolean metodu v backing beaně
- V této metodě můžeme použít
```
FacesContext.getCurrentInstance().getExternalContext()
.isUserInRole("teacher");
```
- Pro zjištění jména přihlášeného uživatele:
```
FacesContext.getCurrentInstance().getExternalContext()
.getUserPrincipal().getName();
```
- Pro odhlášení lze použít:
```
HttpSession sess = (HttpSession) FacesContext.getCurrentInstance()
  .getExternalContext().getSession(false);
sess.invalidate();
```

Zabezbečení EJB

- Jednotlivé metody v EJB session beanách lze jednoduše ochránit vymezením rolí, které mohou metodu volat – pomocí anotace:

```
@RolesAllowed({“teacher”, “...”})
public void ejbmetoda(...) { }
```

- Pro další používání security lze injekcí získat SessionContext:

```
@Resource
SessionContext ctx;
…
ctx.getCallerPrincipal().getName();
ctx.isCallerInRole(“teacher”);
```