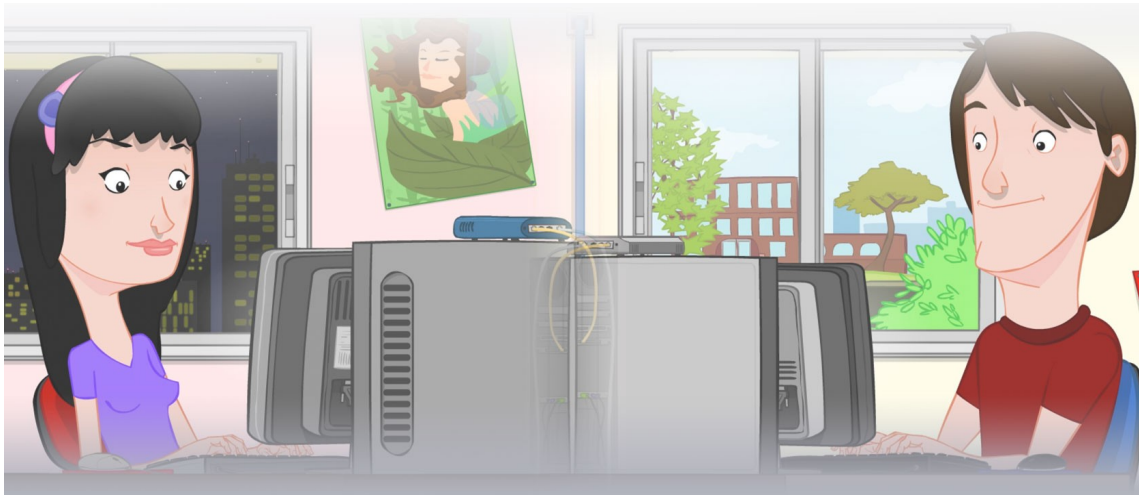
8. PRAKTIKA

JSF, EJB eta JPA (taula bakarra)



Universidad del País Vasco
Euskal Herriko Unibertsitatea

OCW 2015

UPV/EHU

ZERBITZU TELEMATIKO AURRERATUAK: 8. PRAKTIKA

AURKIBIDEA

8. PRAKTIKA: JSF, EJB eta JPA (taula bakarra)

1 Helburua

Praktika honetan aurkezpenerako Web Aplikazio bat sortuko dugu berriro ere, baina oraingoan JavaEE teknologien gaitasun osoa erabiliko duena. Horretarako, softwarearen 3 mailetako bakoitzean dagokion JavaEE teknologia erabiliko dugu, hau da, datuen mailan JPA, negozio mailan EJB eta Aurkezpen mailan JSF.

Hortaz, gure online dendako produktuak jadanik ez dira XML fitxategi batean mantenduko, MySQL datu-base batean baizik, JPA bidez atzituko duguna gure datuen mailan. Aurreko praktiketan bezala, administrariak produktuak ez ditu zuzenean sartuko datu-basera, memorian mantenduko den zerrenda batera baizik; uneren batean, memoriako zerrenda hori datu-basera pasatuko du. Erabiltzaileak ikusiko dituen produktuak berriz, beti izango dira datu-basekoak.

2 Proiektua

Aurreko praktikan bezala, JBoss zerbitzaria erabili beharko dugu, EJB eta JPArako JavaEE ingurune oso bat behar dugulako.

- Dynamic Web Project→Project name: nahi duguna, Target Runtime: JBoss 7.1 Runtime (aurreko praktikatik jadanik prestatuta)
- Next→Next→ Generate web.xml deployment descriptor
- Properties→Project Facets: JSF 2.2→Further configuration available...: egiaztatu aurreko proiektuan bezala dagoela
  - Type: Library Provided by Target Runtime, URL Mapping Patterns: Add→*.xhtml
- Datu-baseekin JPA bidez lan egin ahal izateko Project Facets horietan ere JPA-ren erabilera ahalbidetuko dugu. JPA-ren konfigurazioan, Hibernate plataforma erabiliko dela adierazi eta bere inplementazioa target runtime-arena dela.

3 Datu-basea

Dendako produktuak datu-base batean mantenduko dira.

- Horretarako proiektu honetarako datu-basea sortu behar dugu, MySQL Client programarekin edo MySQL-Workbench aplikazioarekin. Oraingoz, datu-baseak taula bakarra izango du, produktuen datuak gordetzeko balioko duena.
- Datu-baseen zerbitzarian, erabiltzaile berri bat sortuko dugu ere, datu-basearekiko baimen guztiak dituena.

Ondoren, datu-basea JBoss-etik erabiltzeko konfigurazioak egingo ditugu:

1. Gure makinan MySQL connector delakoa instalatu:

- JBoss zerbitzariaren karpetaren barruan, /modules/com/mysql/main karpeta sortu.
- Bertan MySQL connector delakoa kopiatu: `/usr/share/java/mysql-connector-java-bertsioa-bin.jar`
- Karpeta horretan ere, `module.xml` fitxategia sortu, eduki honekin:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
    <resources>
        <resource-root path="mysql-connector-java-bertsioa-bin.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
        <module name="javax.servlet.api" optional="true"/>
    </dependencies>
</module>
```

2. Datu-basearekiko konexioa konfiguratu behar da orain. Horretarako, JBoss-en `standalone.xml` fitxategian, dagokion lekuan (GOGORATU XML sintaxia), hau sartu behar da:

```
<datasource jndi-name="java:jboss/datasources/ZureDBa" pool-name="ZureDBa">
    <connection-url>jdbc:mysql://localhost:3306/ZureDBa</connection-url>
    <driver>com.mysql</driver>
    <security>
       <user-name>dbErabiltzailea</user-name>
       <password>pasahitza</password>
    </security>
</datasource>
<drivers>
    <driver name="com.mysql" module="com.mysql">
        <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-class>
    </driver>
</drivers>
```

Oraingoz ez dugu taularik sortuko eta ez dugu produkturik sartuko, hori JPArekin egingo dugulako, baina datu-basean gertatzen diren aldaketak ikusi ahalko ditugu beharrezko `SELECT` eskaerak eginda (MySQL Client edo Workbench-etik).

4 Model

4.1 Datuen Atzipena

Lehenik, `persistence.xml` fitxategia aldatuko dugu, proiektuan gure datu-basea erabili ahal izateko. Horretarako, elementu hauek gehitu behar ditugu `persistence-unit` barruan:

```
<jta-data-source>java:jboss/datasources/ZureDBa</jta-data-source>
<properties>
    <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>
    <property name="hibernate.hbm2ddl.auto" value="update"/>
</properties>
```

Ondoren, entitate klase bat sortuko dugu, datu-baseko produktuen taulan oinarrituta, bere objektuak iraunkortasun entitateak izan daitezen:

1. Entitate hau dendaren aurreko bertsioan erabilitako `Produktua` klasea izango da, baina JAXB

oharrak beharrean JPA oharrekin.

- JPA oharrekin `Produktua` klasea entitatea dela adieraziko dugu, erreferentzia atributua taulako `primary-key` dela eta beste atributu guztiak `null` ezin daitezkeela izan.
- `NamedQuery` eskaera bat programatuko dugu ere, taulako produktu guztien datu guztiak kontsultatzeko balioko diguna.

2. Dendaren aurreko bertsioetan ez bezala, datuen maila honetan ez dugu beste ezer programatu beharko
   - Ez dugu zerrendekin lan egiten duen klaserik behar, produktuak ez direlako zerrendetan gordetzen datu-basean
   - Ez dugu datu-basetik irakurri eta idazteko metodoak dituen klaserik idatzi behar, jadanik hori programatuta dagoelako JPA liburutegietan.

4.2 Negozioaren logika

Aurreko praktikako EJB klase berdina erabiliko dugu negozioaren logika osoa programatzeko, baina datu-basea eta JPArekin lan egin beharragatik, aldaketa batzuk egin beharko ditugu:

- `EntityManager` motako iraunkortasun testuinguruaren kudeatzaile bat deklaratu atributu modura eta `@PersistenceContext` oharra jarri.
- Datu-baseko produktuen zerrenda itzuliko duen metodoa: egin beharreko bakarra, entitate klasean idatzitako `NamedQuery` eskaera erabiltzea izango da, datu-basetik produktuen `List` motako zerrenda lortzeko; zerrenda hori izango da `return`-ekin itzuliko duena.
- Datu-basean produktuen zerrenda sartzen duen metodoa: parametro modura produktuen zerrenda bat den objektua jaso eta produktuok banan-banan iraunkortasun testuinguruan sartuko ditu (datu-basean gehituak izan daitezen).

5 View

5.1 ManagedBeans

Hemen ere aurreko praktikako ManagedBean berdinak erabiliko ditugu, baina EJBak jasandako aldaketak direla eta, moldatu beharko ditugu:

1. `ProduktuaMB` klasea: aldaketarik gabe, Datuek Atzipen mailan eta Negoazioaren Logikaren mailan egondako aldaketek ez diote baitiote aldatzeko moduan eragiten.
2. `ProduktuenZerrendaMB` klasea: aurreko praktikarekiko aldaketak hauek izango dira
   - Object klasearen deribatua izango da: Datuen Atzipen mailatik bere ama-klasea zena ezabatu dugunez, herentzia erlazio hori kendu behar da.
   - Atributu berri bat izango du, `Produktua` motako objektuen `List` zerrenda bat izango dena: berez, atributu hau ez da guztiz berria, aurreko praktiketan herentziagatik zuelako; oraingoan herentzia erlazioa aldatu denez, esplizituki deklaratu behar da. Zerrenda hau izango da administrariak aldatuko duena produktu berri bat sartzen duen bakoitzean.

- Eraikitzailea: `List` motako atributua sortu behar du.
- Atributu berriaren `getter` eta `setter` metodoak idatzi, beharrezkoak badira.
- `ProduktuaMB` motako objektu bat atribututzat dugun `List` objektuan sartzen duen metodoa: sinpleagoa izan daiteke, atributua klase honetan bertan deklaratua dagoelako.
- Dendako produktuen zerrenda datu-basetik irakurri eta itzultzen duen metodoa: EJB atributuaren metodo egokiak erabilita.
- Atribututzat dugun produktuen zerrenda datu-basean idatziko duen metodoa: EJB atributuaren metodo egokiak erabilita.
- Dendako produktuen zerrenda datu-basetik irakurri eta List motako atributua bere balioarekin hasieratzen duen metodoa: aurreko praktikan bezala, baina oraingoan produktuen zerrenda datu-basetik irakurrita. Eraikitzailearen ondoren exekutatu behar denez, `@PostConstruct` oharrarekin apaindu.

5.2 xhtml

Praktika honetan ere, View ataleko ManagedBeans klaseetako metodo eta atributuen izenik aldatu ez badugu, xhtml fitxategiak ez dira aldatu beharko; bestela, dagozkien izenak erabiltzea izango da aldaketa bakarra, kasuan-kasu.

6 Controller

Amaitzeko, Controller atala falta zaigu, aurreko praktikan bezala landu behar dena. `web.xml` fitxategian adierazi beharrekoa:

1. Proiektuari dagokion `welcome-file` elementua: aurreko praktikan bezala.
2. JSFekiko, `*.xhtml` fitxategiak `FacesServlet`-era mapeatzen dituen elementua: aurreko proiektuan bezala.