



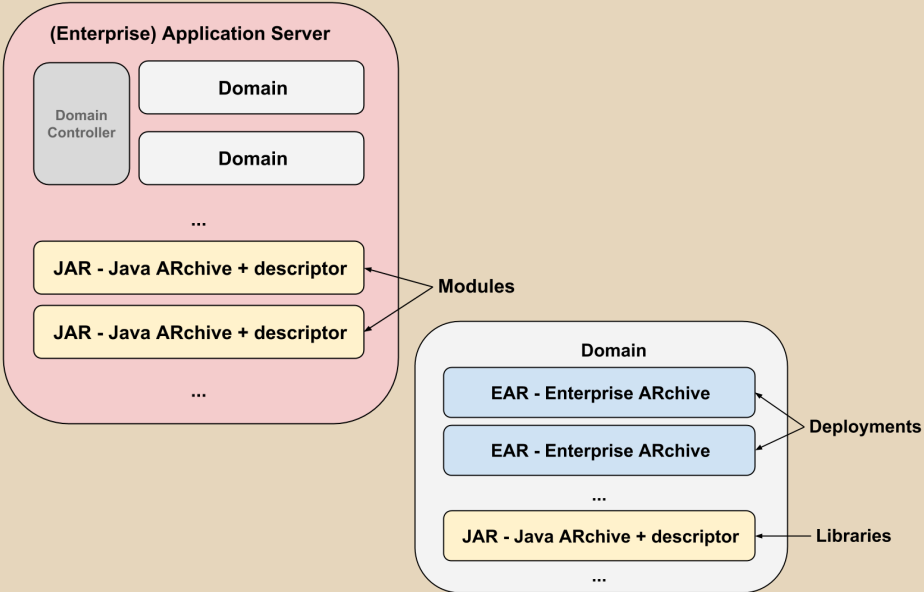
BookStore #maven

Enterprise Application, Git, EJB, EAP/EAS, Logging,
PostgreSQL/MySQL, JPA, Integration testing

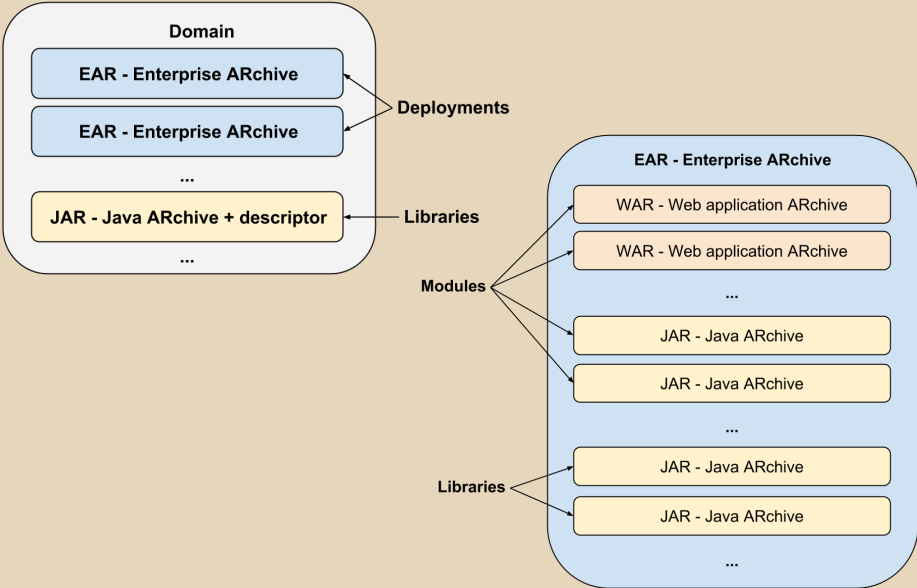
Óbudai Egyetem, Java Enterprise Edition
Műszaki Informatika szak
Labor 3

Bedők Dávid
2018-01-17
v1.1

EAS - Domain struktúra



EAS - Deployment struktúra



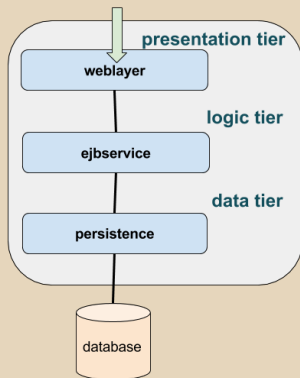


Feladat: hozzunk létre egy alkalmazást, mely egy könyvesbolt könyveinek lekérdezésére alkalmas.

A feladatot két részletben valósítjuk meg. Első körben a persistence réteget mock-olni/fake-elni fogjuk.

Alkalmazandó technikák:

- ▷ Java Enterprise Edition 6
 - EJB API 3.1
 - Servlet API 3.0.1
 - JPA 2.0
- ▷ JBoss 6.4 / WebLogic 12.1.3





```
1 > cd [GIT]\maven\jboss\booklight\  
2 > mvn clean package  
3 ...  
4 BUILD SUCCESS
```

Új EAR csomag:

[GIT]\maven\jboss\booklight\build\libs\booklight.ear

```
1 > cd [JBOSS-HOME]\bin  
2 > standalone.[bat|sh]
```

Másoljuk be az elkészült **booklight.ear** állományt a
[JBOSS-HOME]\standalone\deployments könyvtárba.

<http://localhost:8080/bl-weblayer/BookPing>

```
BookStub [isbn=978-0441172719, author=Frank Herbert, title=Dune,  
category=SCIFI, price=3500.0, numberOfPages=896]
```



- ▷ Projekt hierarchia létrehozása
- ▷ **Enterprise JavaBean** (ejb) alapok mint **business layer**
- ▷ Egyszerű servlet mint **weblayer**
- ▷ Mockolt adatok mint **persistence layer**

Enterprise Application Servers (EAS)

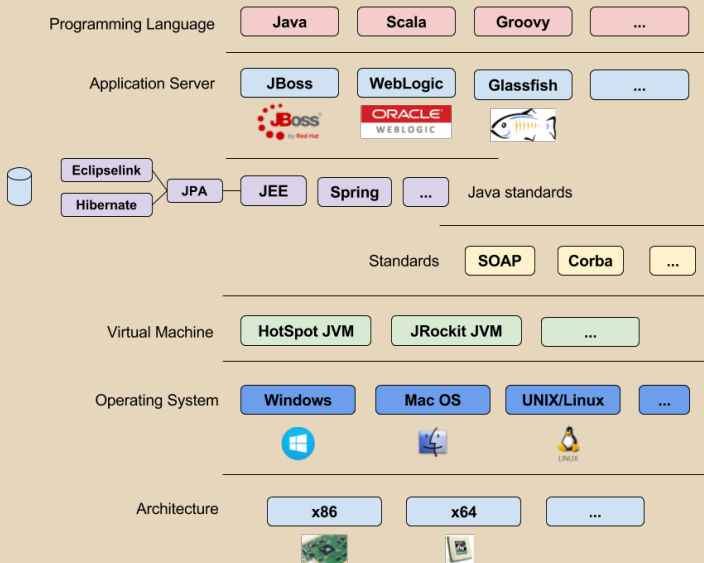
- ▷ RedHat JBoss 6.4
- ▷ Oracle WebLogic 12.1.3

Másik EAS választásának lehetősége

A JavaEE-ben megvalósított alkalmazások minden olyan Java alkalmazás szerveren futnak, melyek az adott verziójú JavaEE-t támogatják. Ez az átjárhatóság tovább erősíti a Java egyébként is **cross-platform** tulajdonságát (bár attól hogy egy nyelv cross-platform, még nem lenne természetes ekkora komplexitású alkalmazások mozgatása AS-ek között).

A feladatot két különböző EAS-ra fogjuk elkészíteni/deploy-olni.

Cross-Platform Java





```
[gradle|maven]\jboss\booklight
```

```
1 > mvn clean package
```

Megjegyzés: a maven az így végrehajtott package goal-t minden subproject-en, a függőségek figyelembevételével végre fogja hajtani, és mindegyik gyermek project külön-külön elkészíti saját kimenetét.



A blueprint repository frissítése (Git pull) szükséges, ha az origin/master (branch) változott. E művelet előtt a lokális változásainkat el kell dobni.

Working copy kiválasztása (root könyvtárban)

```
1 > git checkout .
```

Lokális módosítások eldobása

```
1 > git reset --hard
```

Új állományok/könyvtárak eldobása

```
1 > git clean -fd
```

Változások leszedése a **master**-ről

```
1 > git pull
```



<http://gitextensions.github.io/>
<https://github.com/gitextensions>

Verzió: **2.50.02**

OS: **Windows** (NIX esetén Mono 5.0 szükséges)

Grafikus felületet biztosít a Git parancsok végrehajtásához. Nem csak eszköz, tanítja is a kiadott parancsokat.

A Git használat nem lineáris, és ugyanaz a feladat többféleképpen is megoldható. Minden külső eszköz használata e végett magában foglal némi kockázatot (nem biztos hogy a tool azt hajtja végre, amit mi szeretnénk).

Alapfunkciók esetén szinte kockázatmentes.



<https://www.sourcetreeapp.com/>

Verzió: **2.6.3a**

OS: **Windows / Mac OS**

Elegáns, modern felület, néhol kicsit körülményes (pl. git log/blame esetén), felületét szokni kell, de azt követően hatékony eszköz. Windows rendszereken korábban lassú volt. Nem tanítja a git parancsokat, mint pl. a GitExtensions.



Állapot lekérdezése:

```
1 > git status
```

Stage

```
1 > git add [FILENAME]
```

Commit (-a/--all, -m/--message)

```
1 > git commit -a -m "[COMMIT_MESSAGE]"
```

Változások felküldése a master-re

```
1 > git push
```

Enterprise Maven project

Hierarchical Multi-project



Számos Maven **Archetype** létezik a témában (pl. `maven-archetype-j2ee-servlet`), mely mankónak illetve ötlet szerzésnek használható, azonban későbbiekben zavarhat és szemetelhet. Nem fogjuk ezeket alkalmazni a labor során.

Több project, hierarchiába szervezve

- ▷ Root Aggregator project
 - maven **pom** packaging → aggregator project
- ▷ WebLayer
 - maven **war** packaging → war
 - EAR **web module**
- ▷ EJBSERVICE
 - maven **ejb** packaging → jar
 - EAR **ejb module**
- ▷ EAR project
 - maven **ear** packaging → ear

A korábban megismert **jar** packaging mellett most további 4 típussal ismerkedünk meg.

Könyvtár struktúra

```
META-INF/  
  MANIFEST.MF  
WEB-INF/  
  classes/  
    hu/  
      qwaevisz/  
        demo/  
          Lorem.class  
          Ipsum.class  
  lib/  
    dummy.jar  
  web.xml  
index.html  
logo.jpg  
base.css
```

public resources (alkönyvtárakba elhelyezhetőek)

```
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 </web-app>
```

web application deployment descriptor

web.xml

Könyvtár struktúra

```
META-INF/  
  MANIFEST.MF  
  application.xml  
lib/  
  dummy.jar  
  xyz.jar  
  abc.war
```

3rd party library

```
1 <?xml version="1.0"?>  
2 <application xmlns="http://java.sun.com/xml/ns/javaee"  
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
   http://java.sun.com/xml/ns/javaee/application_6.xsd"  
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="6">  
3   <display-name>sample</display-name>  
4   <module>  
5     <ejb>xyz.jar</ejb>  
6   </module>  
7   <module>  
8     <web>  
9       <web-uri>abc.war</web-uri>  
10      <context-root>abcwebapp</context-root>  
11     </web>  
12   </module>  
13   <library-directory>lib</library-directory>  
14 </application>
```


Multi-project könyvtár struktúra létrehozása

Egyszerűsített könyvesbolt

Könyvtár struktúra

```
booklight/  
  bl-ear/  
    pom.xml  
  bl-ejb-service/  
    src/  
      main/  
        java/  
        resources/  
          META-INF/  
            ejb-jar.xml  
      pom.xml  
  bl-weblayer/  
    src/  
      main/  
        java/  
        webapp/  
          WEB-INF/  
            web.xml  
      pom.xml  
  pom.xml
```

EJB module - Deployment Descriptor

ejb-jar.xml

src | main | resources | META-INF | ejb-jar.xml

```
1 <ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
2  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
  http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd"  
3  version="3.0">  
4  
5 </ejb-jar>
```

ejb-jar.xml

Kötelező elem?

Bár a Gradle és a JBoss nem követeli meg az üres `ejb-jar.xml` jelenlétét a classpath-on, érdemes szerepeltetni az átjárhatóság végett (pl. Maven megköveteli a jelenlétét). A *deployment descriptor*-ban definiálható szabályok jelentős részét ma már annotációk segítségével a forráskódban helyezzük el. A leíró xml precedenciája mindig magasabb (felül lehet írni az annotációk által szabályozott elemeket).

Maven Aggregator project

Root project



```
1 <project [..]>
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>hu.qwaevisz.booklight</groupId>
4   <artifactId>booklight</artifactId>
5   <version>1.0</version>
6   <packaging>pom</packaging>
7   <name>BookStore-Light Aggregator</name>
8
9   <modules>
10     <module>bl-ear</module>
11     <module>bl-ejb-service</module>
12     <module>bl-weblayer</module>
13   </modules>
14
15   <properties>
16     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17     <maven.compiler.source>1.8</maven.compiler.source>
18     <maven.compiler.target>1.8</maven.compiler.target>
19   </properties>
20 </project>
```

A **pom** packaging module-okat fog össze. Az itt megadott module-ok alkönyvtárak az aggregator projekt szintjén, és mindegyik egy Maven projekt is egyben (tartalmaznak pom.xml-t).



```
1 <project [..]>
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>hu.qwaevisz.booklight</groupId>
4   <artifactId>bl-ejbservice</artifactId>
5   <version>1.0</version>
6   <packaging>ejb</packaging>
7   <name>BookStore-Light EJB Services</name>
8
9   <parent>
10    <groupId>hu.qwaevisz.booklight</groupId>
11    <artifactId>booklight</artifactId>
12    <version>1.0</version>
13  </parent>
14
15  <dependencies>
16    [..]
17  </dependencies>
18  <build>
19    [..]
20  </build>
21 </project>
```

Az **ejb** packaging JAR-t állít elő, hasonlóan a **jar** packaging-hez. Az `ejb-jar.xml` deployment descriptor megléte kötelező.

A sub Maven project hivatkozik szülőjére, annak pontos artifact URL-jének megadásával.

Maven EJB Service project

Továbbiak..



```
1 <project [..]>
2   [..]
3   <dependencies>
4     <dependency>
5       <groupId>javax</groupId>
6       <artifactId>javaee-api</artifactId>
7       <version>6.0</version>
8       <scope>provided</scope>
9     </dependency>
10  </dependencies>
11
12  <build>
13    <plugins>
14      <plugin>
15        <groupId>org.apache.maven.plugins</groupId>
16        <artifactId>maven-ejb-plugin</artifactId>
17        <version>2.5.1</version>
18      </plugin>
19    </plugins>
20  </build>
21 </project>
```

Az EJB service projekt fordításához szükségünk van a Java EE API 6.0-ra, futás közben azonban az EAP/EAS fogja ezeket biztosítani (és classpath-ra pakolni). A **provided** scope pontosan erre szolgál.



```
1 <project [..]>
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>hu.qwaevisz.booklight</groupId>
4   <artifactId>bl-weblayer</artifactId>
5   <version>1.0</version>
6   <packaging>war</packaging>
7   <name>BookStore-Light Webapplication</name>
8
9   <parent>
10    <groupId>hu.qwaevisz.booklight</groupId>
11    <artifactId>booklight</artifactId>
12    <version>1.0</version>
13  </parent>
14
15  <dependencies>
16    [..]
17  </dependencies>
18 </project>
```

A **war** packaging WAR-t állít elő. A web.xml deployment descriptor megléte kötelező.

pom.xml

Maven WebLayer project

Továbbiak..



```
1 <project [..]>
2   [..]
3   <dependencies>
4     <dependency>
5       <groupId>hu.qwaevisz.booklight</groupId>
6       <artifactId>bl-ejbsservice</artifactId>
7       <version>1.0</version>
8       <scope>provided</scope>
9     </dependency>
10
11    <dependency>
12      <groupId>javax</groupId>
13      <artifactId>javaee-api</artifactId>
14      <version>6.0</version>
15      <scope>provided</scope>
16    </dependency>
17  </dependencies>
18 </project>
```

A weblayer "szeretne" meghívni majd EJB service-eket, melyek az bl-ejbsservice projectben vannak, ezért szükséges a classpath-on szerepeltetni ezt a subprojektet annak teljes artifact URL-jének megadásával.

pom.xml



```
1 <project [..]>
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>hu.qwaevisz.booklight</groupId>
4   <artifactId>bl-ear</artifactId>
5   <version>1.0</version>
6   <packaging>ear</packaging>
7   <name>BookStore-Light EAR</name>
8
9   <parent>
10    <groupId>hu.qwaevisz.booklight</groupId>
11    <artifactId>booklight</artifactId>
12    <version>1.0</version>
13  </parent>
14
15  <dependencies>
16    [..]
17  </dependencies>
18  <build>
19    [..]
20  </build>
21 </project>
```

Az **ear** packaging egy EAR-t fog előállítani a felsorolt függőségek alapján. Elkészíti az EAR-hoz szükséges deployment descritort is (application.xml).

Maven EAR project

Továbbiak..



```
1 <project [..]>
2   [..]
3   <dependencies>
4     <dependency>
5       <groupId>hu.qwaevisz.booklight</groupId>
6       <artifactId>bl-ejbservice</artifactId>
7       <version>1.0</version>
8       <type>ejb</type>
9     </dependency>
10    <dependency>
11      <groupId>hu.qwaevisz.booklight</groupId>
12      <artifactId>bl-weblayer</artifactId>
13      <version>1.0</version>
14      <type>war</type>
15    </dependency>
16  </dependencies>
17  <build>
18    <plugins>
19      <plugin>
20        <groupId>org.apache.maven.plugins</groupId>
21        <artifactId>maven-ear-plugin</artifactId>
22        <version>2.10.1</version>
23      </plugin>
24    </plugins>
25  </build>
26 </project>
```

Az `application.xml` elemei az itt felsorolt függőségek lesznek a **type** értékének megfelelően (ejb-ből EJB module, war-ból web module lesz).

pom.xml

- ▷ Szerveroldali üzleti komponensek (back-end services)
- ▷ JSR19, JSR152, JSR220, JSR318, JSR345
- ▷ IBM (1997), majd Sun Microsystems (1999)
- ▷ Egyfajta "best-practice" annak érdekében, hogy az üzleti értéket "kelljen csak" lefejlesztetni, ne a "tipikus" keret dolgokat (boilerplate elemeket).
 - Tranzakciókezelés
 - Konkurenciakezelés
 - Aszinkron metódushívás
 - Eseménykezelés
 - Java Message Service integráció (Message Driven Beans)
 - Perzisztencia integráció támogatása (de már nem része a persistence az EJB specifikációnak)
- ▷ Verziók
 - EJB1.x: minden "remote"
 - EJB2.x: nagyon "kényelmetlen", túlbonyolított
 - EJB3.x: a Spring Framework és a Hibernate tapasztalatai alapján újraalkotott elképzelés (POJO-kal dolgozik mint a Spring FW és szakít az entity bean-ekkel (inkább támogatja a perzisztens réteget, nem megvalósítja))

▷ Session Beans

- **Stateless SB** (SLSB)
 - Állapotmentes
- **Stateful SB** (SFSB)
 - Állapottartó
 - Aktiválás/Passzíválás
 - Fontos a szerializálhatóság
- **Singleton SB** (SSB)
 - "Egyke"

▷ Message Driven Beans (MDB)

- Üzenetvezérelt
- Elsősorban aszinkron feldolgozás
- Nincs kliens interface

▷ Entity Beans

- Deprecated, EJB 3.x-től a JPA helyettesíti

Definiáljunk néhány üzleti metódust:

- ▷ ISBN¹ alapján egy könyv adatainak biztosítása
 - `BookStub` `getBook (String isbn)`
- ▷ Keresési feltételek alapján a feltételnek megfelelő könyvek adatainak kinyerése
 - `List<BookStub> getBooks (BookCriteria criteria)`
- ▷ `FacadeException`: ha az üzleti kérés nem teljesíthető

Definiáljunk az ezekhez szükséges DTO²-kat:

- ▷ `BookStub`: `isbn (String)`, `author (String)`, `title (String)`, `category (BookCategoryStub)`, `price (double)`, `numberOfPages (int)`
- ▷ `BookCategoryStub`: SCIFI, LITERATURE, HISTORICAL, PHILOSOPHY
- ▷ `BookCriteria`: `author (String)`, `title (String)`, `minPrice (int)`, `maxPrice (int)`

¹ International Standard Book Number

² Data Transfer Object

Domain osztályok - Könyv kategóriák

```
1 package hu.qwaevisz.booklight.ejbsservice.domain;
2 public enum BookCategoryStub {
3
4     SCIFI("Sci-Fi"),
5     LITERATURE("Literature"),
6     HISTORICAL("Historical"),
7     PHILOSOPHY("Philosophy");
8
9     private final String label;
10
11     private BookCategoryStub(String label) {
12         this.label = label;
13     }
14
15     public String getLabel() {
16         return this.label;
17     }
18     public String getName() {
19         return this.name();
20     }
21 }
```

BookCategoryStub.java



Domain osztályok - Könyv

```
1 package hu.qwaevisz.booklight.ejbsservice.domain;
2 public class BookStub {
3     private String isbn;
4     private String author;
5     private String title;
6     private BookCategoryStub category;
7     private double price;
8     private int numberOfPages;
9
10    public BookStub(String isbn, String author, String title,
11                   BookCategoryStub category, double price, int numberOfPages)
12    {
13        this.isbn = isbn;
14        this.author = author;
15        this.title = title;
16        this.category = category;
17        this.price = price;
18        this.numberOfPages = numberOfPages;
19    }
20    [...]
21 }
```

BookStub.java



Domain osztályok - Keresési feltételek

```
1 package hu.qwaevisz.booklight.ejbservice.domain;
2
3 public class BookCriteria {
4
5     private String author;
6     private String title;
7     private BookCategoryStub category;
8     private int minimumPrice;
9     private int maximumPrice;
10
11     public BookCriteria() {
12     }
13
14     [...]
15 }
```

BookCriteria.java

Domain osztályok - Dobható üzleti kivétel

```
1 package hu.qwaevisz.booklight.ejbservice.exception;
2
3 public class FacadeException extends Exception {
4
5     private static final long serialVersionUID = 1L;
6
7     public FacadeException(String message) {
8         super(message);
9     }
10
11    public FacadeException(String message, Throwable cause) {
12        super(message, cause);
13    }
14
15 }
```

FacadeException.java

Stateless Session Bean

Local interface

```
1 package hu.qwaevisz.booklight.ejbservice.facade;
2
3 import java.util.List;
4 import javax.ejb.Local;
5 import hu.qwaevisz.booklight.ejbservice.domain.BookCriteria;
6 import hu.qwaevisz.booklight.ejbservice.domain.BookStub;
7 import hu.qwaevisz.booklight.ejbservice.exception.FacadeException;
8
9 @Local
10 public interface BookFacade {
11     BookStub getBook(String isbn) throws FacadeException;
12
13     List<BookStub> getBooks(BookCriteria criteria) throws FacadeException;
14 }
15 }
```

BookFacade.java

@Remote: költséges és lassú (hálózati hívás, serializáció), RMI, eredeti koncepció csak ezt támogatta, különböző JVM-ek között használjuk

@Local: olcsó és gyors, azonos JVM-en belül (elsősorban adott EAR-on belül)

Stateless Session Bean

Implementation

```
1 package hu.qvaevisz.booklight.ejbservice.facade;
2
3 [...]
4
5 @Stateless(mappedName = "ejb/bookFacade")
6 public class BookFacadeImpl implements BookFacade {
7
8     @Override
9     public BookStub getBook(String isbn) throws FacadeException {
10         return new BookStub(isbn, "Frank Herbert", "Dune", BookCategoryStub.SCIFI, 3500,
11             896);
12     }
13
14     @Override
15     public List<BookStub> getBooks(BookCriteria criteria) throws FacadeException {
16         List<BookStub> stubs = new ArrayList<BookStub>();
17         stubs.add(new BookStub("978-0441172719", "Frank Herbert", "Dune",
18             BookCategoryStub.SCIFI, 3500, 896));
19         stubs.add(new BookStub("978-0684824710", "Daniel C. Dennett", "Darwin's dangerous
20             idea", BookCategoryStub.PHILOSOPHY, 4500, 586));
21     }
22 }
```

Mock implementáció
(literálokat ad vissza)

BookFacadeImpl.java

Stateless Session Bean

@Stateless annotáció

@Stateless: Stateless Session Bean EJB típus definiálása

- ▷ `name`: ejb neve
 - JNDI lookup során használatos
 - alapértelmezetten az *unqualified name* (pl.: `BookFacadeImpl`)
- ▷ `mappedName`: global JNDI név
 - gyártó specifikus (vendor specific)
- ▷ `description`: leírás

Servlet

Egyelőre csak tesztelési célból

```
1 package hu.qwaevisz.booklight.weblayer.servlet;
2 [...]
3 @WebServlet("/BookPing")
4 public class BookPingServlet extends HttpServlet {
5
6     @EJB
7     private BookFacade facade;
8
9     @Override
10    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
11        response.setCharacterEncoding("UTF-8");
12        final PrintWriter out = response.getWriter();
13        try {
14            final BookStub book = this.facade.getBook("978-0441172719");
15            out.println(book.toString());
16        } catch (final FacadeException e) {
17            out.println(e.getLocalizedMessage());
18        }
19        out.close();
20    }
21 }
```

BookPingServlet.java

@EJB: Session Bean proxy beszúrása

- ▷ Csak az EJB "érdekszférájában" használható
 - Minden EJB ide tartozik
 - Minden Servlet ide tartozik (@WebServlet)
 - stb..
- ▷ `beanName` vagy `lookup`: az EJB feloldásához használható attribútumok
 - A kettő közül egy időben csak az egyik használható
 - Ha csak egy implementáció van, akkor alkalmazáson belül egyik attribútumot sem kell használni (típus alapján megtalálja a container a megfelelőt).
 - `beanName`: A Session bean name értékére refereál
 - `lookup`: A Session bean JNDI nevére refereál

Webalkalmazás fejlesztés alapjai Java nyelven

Servletek és JPS-k

Az Óbudai Egyetem "Modern programozási nyelvek" laborjának ide vonatkozó részei szükséges és elégséges feltétele a jelen labor során bemutatott frontend fejlesztésnek.

A tárgy nem hivatalos oldala:

<http://users.nik.uni-obuda.hu/bedok.david/jse.html>

A tárgy hivatalos oldala:

<http://users.nik.uni-obuda.hu/java/>

<http://www.jboss.org/>

Install: unzip

Enterprise JavaBeans Open Source Software → EJBoss

→ JBoss (az EJB® Sun trademark volt)

JEE 7 compliant verzió: **v7.0.0** (2016.05.10.)

JEE 6 compliant verzió: **v6.4.0** (2015.04.15.)

Kereskedelmi termék!

Standalone / Domain mode

Tartalma: <https://access.redhat.com/articles/112673>

- ▷ RESTEasy - 2.3.10.Final
- ▷ Hibernate Core - 4.2.18.Final
- ▷ Hibernate JPA 2.0 API - 1.0.1.Final
- ▷ JSF2 - 2.1.28.Final
- ▷ HornetQ - 2.3.25.Final (EAP 7.x → JBoss A-MQ)
- ▷ JBoss Logging - 3.1.4.GA
- ▷ JAXB - 2.2.5-redhat-9
- ▷ Apache Web Server - 2.2.26
- ▷ ...



<http://wildfly.org/>

Install: unzip

JEE 7 compliant verzió: **v11.0.0.Final** (2017.10.23.)

JEE 7 compliant verzió: **v10.1.0.Final** (2016.08.19.)

JEE 7 compliant verzió: **v8.0.0.Final** (2014.02.11.)

A JBoss **ingyenes** változata.

Standalone / Domain mode

JBoss vs. WildFly

A két alkalmazáserver gyakorlati használata közel megegyezik, ha valaki az egyikkel dolgozott már, a másik vonatkozásában nem fog váratlan nehézségekbe belefutni.

Standalone mód ([JBASS-HOME]\standalone):

- ▷ \configuration\standalone.xml
- ▷ \deployments
- ▷ logs\server.log

Standalone mód elindítása (standalone.xml):

```
1 > cd [JBASS-HOME]\bin
2 > standalone.[bat|sh]
```

Standalone mód elindítása (custom.xml használata):

```
1 > [JBASS-HOME]\bin\standalone.[bat|sh] custom.xml
```

Standalone server instance leállítása:

```
1 > [JBASS-HOME]/bin/jboss-cli.[bat|sh] --connect command=:shutdown
```

<http://www.oracle.com/technetwork/middleware/weblogic/>

JEE 7 compliant verzió: **v12.2.1.3**

JEE 6 compliant verzió: **v12.1.3**

Kereskedelmi termék!

Az Oracle felvásárlását megelőzően a **BEA** terméke volt.

A JavaEE mellett a Spring Framework-öt is támogatja (a SF számára egy *Apache Tomcat*® is legtöbbször elégséges).

- ▷ Unzip (wls1213_dev_update3.zip)
 - pl.: c:\work\wls12130\
- ▷ Környezeti változó:
 - **MW_HOME** → c:\work\wls12130\
- ▷ Telepítő futtatása (*Run as Administrator*)
 - [MW_HOME]\configure.cmd
 - Do you want to configure a new Domain? Yes (Y)
 - username: **weblogic**
 - password: **AlmafA1#**
- ▷ WebLogic Server Administration Console
 - <http://localhost:7001/console>

Domain elindítása:

```
1 > [MW_HOME]\user_projects\domains\mydomain\startWebLogic.cmd
```

Domain leállítása:

- ▷ WebLogic Server Administration Console-ról illetve a console ablak terminálása (Ctrl+C)
 - Environment | Servers | Control tab
 - select “myserver” → Shutdown
 - * When work completes vagy
 - * Force Shutdown now

- ▷ WebLogic Server Administration Console
 - Deployments | Install..
 - Browse EAR
 - Ezt követően lehet ugyanezen path-ról egy-két kattintással frissíteni az alkalmazást (redeploy)

A WebLogic gazdag statisztikákat kezel, monitorozza a futó komponenseket. Érdeemes az Admin Console-on megvizsgálni a lehetőségeket.

Autodeploy könyvtár:

```
[MW_HOME]\user_projects\domains\mydomain\autodeploy\
```

Megjegyzés: az auto deploy-olt alkalmazások nem törölhetőek/frissíthetőek
A Deployments alól (a autodeploy könyvtárból kell kezelnünk ezt is).



```
[gradle|maven]\jboss\booklight\build\libs\booklight.ear
```

Másolás ide: [JBASS-HOME]\standalone\deployments\
[

```
09:14:13,505 INFO [org.jboss.as.server.deployment] (MSC service thread 1-6)
JBAS015876: Starting deployment of "booklight.ear" (runtime-name: "booklight.ear")
09:14:13,533 INFO [org.jboss.as.server.deployment] (MSC service thread 1-6)
JBAS015973: Starting subdeployment (runtime-name: "bl-ejb-service.jar")
09:14:13,534 INFO [org.jboss.as.server.deployment] (MSC service thread 1-5)
JBAS015973: Starting subdeployment (runtime-name: "bl-weblayer.war")
09:14:13,577 INFO
[org.jboss.as.ejb3.deployment.processors.EjbJndiBindingsDeploymentUnitProcessor] (MSC
service thread 1-1) JNDI bindings for session bean named BookFacadeImpl in deployment
unit subdeployment "bl-ejb-service.jar" of deployment "booklight.ear" are as follows:
  java:global/booklight/bl-ejb-service/BookFacadeImpl!hu.qwaevisz.booklight.ejb-service.facade
  java:app/bl-ejb-service/BookFacadeImpl!hu.qwaevisz.booklight.ejb-service.facade.BookFacade
  java:module/BookFacadeImpl!hu.qwaevisz.booklight.ejb-service.facade.BookFacade
  java:global/booklight/bl-ejb-service/BookFacadeImpl
  java:app/bl-ejb-service/BookFacadeImpl
  java:module/BookFacadeImpl
09:14:13,646 INFO [org.jboss.web] (ServerService Thread Pool -- 11) JBAS018210:
Register web context: /bl-weblayer
09:14:13,739 INFO [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS015859:
Deployed "booklight.ear" (runtime-name : "booklight.ear")
```

server.log



<http://localhost:8080/bl-weblayer/BookPing>

```
BookStub [isbn=978-0441172719, author=Frank Herbert, title=Dune,  
category=SCIFI, price=3500.0, numberOfPages=896]
```



A **BookStore** project teljes egészében tartalmazza a **BookLight** funkcióit, azonban az egyértelműség végett a projekt és a csomagok neve különböző.

```
[gradle|maven]\jboss\bookstore
```

Innentől kezdve a **BookStore** gyökere elkészíthető a **BookLight** alapján, és a fejlesztés a **BookStore** projektben folytatható.

A **BookStore** már tartalmazni fogja a *data tier* réteget (persistence layer).



Számos ponton redundáns - illetve redundáns lehet - a több module-ból álló Maven project hierarchia `pom.xml` állományai. Az XML alapból redundáns formátum, ez ellen sokat tenni nem lehet, azonban némi átalakítást mindenképpen érdemes megtenni.

Fontos!

A build rendszernek egyértelműnek és áttekinthetőnek kell maradnia! Ne eszközöljünk olyan módosítást, mely ezt a szabályt megtöri.



Saját tulajdonságok definiálhatók, melyek láthatóak lesznek a module-ok számára is (nem használhatóak minden 'element' esetében, ennek a Maven *lifecycle management*-ben keresendő az oka).

```
1 <project [..]>
2   [..]
3   <properties>
4     [..]
5
6     <version.javaee>6.0</version.javaee>
7     <version.plugin.ejb>2.5.1</version.plugin.ejb>
8     <version.plugin.ear>2.10.1</version.plugin.ear>
9
10  </properties>
11  [..]
12 </project>
```

pom.xml



```
1 <project [..]>
2   <modelVersion>4.0.0</modelVersion>
3   <artifactId>bs-ejbservice</artifactId>
4   <packaging>ejb</packaging>
5   <name>BookStore EJB Services</name>
6   <parent>[..]</parent>
7
8   <dependencies>
9     <dependency>
10      <groupId>javax</groupId>
11      <artifactId>javaee-api</artifactId>
12      <version>${version.javaee}</version>
13      <scope>provided</scope>
14    </dependency>
15  </dependencies>
16
17  <build>
18    [..]
19  </build>
20 </project>
```

A groupId és a version információ elhagyható, ha azonosak a *parent* project azonos nevű metaadataival!

Használjuk fel a *parent* project-ben definiált `${version.javaee}` tulajdonságot!

Maven EJB Service project

Továbbiak..



```
1 <project [..]>
2   [..]
3   <build>
4     <sourceDirectory>src/main/java</sourceDirectory>
5     <testSourceDirectory>src/test/java</testSourceDirectory>
6     <resources>
7       <resource>
8         <directory>src/main/resources</directory>
9       </resource>
10    </resources>
11    <testResources>
12      <testResource>
13        <directory>src/test/resources</directory>
14      </testResource>
15    </testResources>
16    <plugins>
17      <plugin>
18        <groupId>org.apache.maven.plugins</groupId>
19        <artifactId>maven-ejb-plugin</artifactId>
20        <version>${version.plugin.ejb}</version>
21      </plugin>
22    </plugins>
23  </build>
24 </project>
```

Ezen **source** könyvtár beállítások az alapértelmezettek, de ha ettől szeretnénk eltérni, akkor itt tudjuk konfigurálni.

pom.xml

Maven WebLayer project

Módosítás



Ha a module version információját elhagyjuk, és az örökli a szülőtől, akkor ez a tulajdonság elérhető a megfelelő tulajdonság címezésével.

```
1 <project [...]>
2   [...]
3   <dependencies>
4     <dependency>
5       <groupId>hu.qwaevisz.bookstore</groupId>
6       <artifactId>bs-ejbsservice</artifactId>
7       <version>${project.parent.version}</version>
8       <scope>provided</scope>
9     </dependency>
10
11    <dependency>
12      <groupId>javax</groupId>
13      <artifactId>javaee-api</artifa
14      <version>${version.javaee}</ve
15      <scope>provided</scope>
16    </dependency>
17  </dependencies>
18 </project>
```

A Maven-ben szokás a tulajdonság nevekben (XML element nevekben) pont karaktert használni (pl. `version.javaee`). Nem a legszerecsébb, figyelembe véve hogy a *dot notation* elválasztó karakter is (pl. `project.parant.version`).

pom.xml



Definiálhatjuk a kimeneti könyvtár nevét (Maven esetében a default a `target`), illetve a kimeneti állomány nevét. Az itt megadott eltér az alapértelmezettől, lévén a `bs-ear-1.0.ear` helyett előnyösebb ha `bookstore-1.0.ear` lesz a deployment neve.

```
1 <project [..]>
2   [..]
3   <build>
4     <directory>build</directory>
5     <finalName>${project.parent.artifactId}-${project.version}</finalName>
6     <plugins>
7       <plugin>
8         <groupId>org.apache.maven.plugins</groupId>
9         <artifactId>maven-ear-plugin</artifactId>
10        <version>${version.plugin.ear}</version>
11      </plugin>
12    </plugins>
13  </build>
14 </project>
```

pom.xml

<http://logging.apache.org/log4j/2.x/>

<http://logging.apache.org/log4j/1.2/>

Log4j2 verzió: **v2.9.1**

Log4j verzió: **v1.2.17**

JBoss 6.4 egyelőre a Log4j 1.2.x-et támogatja, a WebLogic alapértelmezetten nem támogatja.

Standard?

A JavaEE-nek nem része a Log4J, a standard a **JDK Logging**-ot tartalmazza (ezt a JBoss és a WebLogic is támogatja természetesen).

Simple Logging Facade for Java (SLF4J): képes elfedni a különböző logging megoldásokat, és mindezeket dinamikusan kezelni (java.util.logging, logback, log4j).

▷ Loggers

- Összerendeli a Java csomagot, a log level-t és az appendereket (1 logger N appender)
- Egy ún. Root Logger mindig létezik.

▷ Appenders

- Definiálja a log üzenet “pattern”-jét és a logolás típusát (pl. file, rolling file, e-mail, jms message, stb.).

Konfiguráció

Standalone alkalmazás esetében **log4j(2).xml** vagy **log4j(2).properties** állományban konfiguráljuk. A Jboss esetén a **Logging subsystem** csomagolja, és a **standalone.xml**-en keresztül konfiguráljuk.

▷ TRACE

- Nagyon részletes adatok, csak kivételes esetben érdemes bekapcsolni (pl. ciklusmagok belsejének logjai).

▷ DEBUG

- Hibakeresési/fejlesztési szint. Végigkövetik a log üzenetek az üzleti folyamatokat komponens szinten. A Debug logok segítségével reprodukálható pl. a felhasználó tevékenysége.

▷ INFO

- Azok az üzenetek, melyek minden esetben jelenjenek meg a logban. Különösen fontos és semmiképpen sem gyakori események írhatóak ki ezen a szinten. Az info szintű üzenetek kb. a "hahó, itt vagyok" típusúak, belőlük pl. hibát keresni, reprodukálni nem lehet.

▷ WARN

- Figyelmeztetés. Általában olyan esemény, mely azért nem hiba, mert az alkalmazás lekezeli valamilyen módon, vagy pl. a tranzakció végrehajtódik, de bizonyos korlátozásokkal.

▷ ERROR

- Hiba. Olyan részletes leírást tartalmazzon, amennyire lehetséges. Ez lesz az az információ, amit a hibajegy javításakor birtokolni fogunk. Ez alapján kérni kell tudni a reprodukciót (mely során már kérhetünk debug logokat is).

▷ FATAL

- Ritkán alkalmazzuk. Végzetes hiba. Az ügyfelek már az Error esetén is üvöltenek, nem érdemes fatális hibákkal ijesztgetni őket.

```
1 public class BookPingServlet extends HttpServlet {
2
3     private static final Logger LOGGER =
4         Logger.getLogger(BookPingServlet.class);
5
6     @Override
7     protected void doGet(HttpServletRequest request,
8         HttpServletResponse response) throws ServletException,
9         IOException {
10        LOGGER.info("Get Book by user");
11        [...]
12    }
13 }
```

BookPingServlet.java

```
1 public class BookFacadeImpl implements BookFacade {
2
3     private static final Logger LOGGER =
4         Logger.getLogger(BookFacadeImpl.class);
5
6     @Override
7     public BookStub getBook(String isbn) {
8         if (LOGGER.isDebugEnabled()) {
9             LOGGER.debug("Get Book (isbn: " + isbn + ")");
10        }
11        [..] LOGGER.debug(String.format("Get Book (isbn: %s)", isbn);
12    }
```

alternatív megoldás (kevesebb kódsor):
[..] LOGGER.debug(String.format("Get Book (isbn: %s)", isbn);

BookFacadeImpl.java

Fontos! Production kódban a DEBUG üzenetek se a teljesítményt, se a memóriát ne terheljék, mivel valós futás közben általában a DEBUG szintet nem naplózzuk. A bemutatott megoldás egy logikai ellenőrzés költségű, míg az alternatív megoldás egy String literált elhelyez a permspace-en. A String összefűzés futását mindenképpen kerüljük el ilyen esetekben (ebből StringBuilder-t készít a fordító, de feleslegesen optimalizálja a sort ha nem naplózzuk).

```
1 <server xmlns="urn:jboss:domain:1.7">
2   <extensions>
3     <extension module="org.jboss.as.logging"/>
4     [..]
5   </extensions>
6   <management>[..]</management>
7   <profile>
8     <subsystem xmlns="urn:jboss:domain:logging:1.5">
9       [APPENDERS DETAILS]
10      [LOGGER DETAILS]
11    </subsystem>
12    [..]
13  </profile>
14  <interfaces>[..]</interfaces>
15  <socket-binding-group>[..]</socket-binding-group>
16 </server>
```

standalone.xml

```
1 <console-handler name="CONSOLE">[..]</console-handler>
2 <periodic-rotating-file-handler
   name="FILE">[..]</periodic-rotating-file-handler>
3
4 <logger category="hu.qwaevisz">
5   <level name="DEBUG"/>
6 </logger>
7
8 <root-logger>
9   <level name="INFO"/>
10  <handlers>
11    <handler name="CONSOLE"/>
12    <handler name="FILE"/>
13  </handlers>
14 </root-logger>
15 <formatter name="PATTERN">[..]</formatter>
```

standalone.xml

A *Log4J* beállításához külön lépések szükségesek (alapértelmezetten nem támogatja a WebLogic, bár szállít hozzá wrapper-t).

Helyette **JDK Logging**-ot használjunk (JavaEE része, `java.util.logging.*`).

```
1 import java.util.logging.Level;
2 import java.util.logging.Logger;
3 [...]
4 private static final Logger LOGGER =
    Logger.getLogger(BookListController.class.getName());
5 [...]
6 LOGGER.info("Get All Books");
```

BookListController.java

JDK Logging vs Log4J

JDK Logging Level	Log4J Level	JDK Logging sample
FINEST	-	LOGGER.finest();
FINER	TRACE	LOGGER.finer();
FINE	DEBUG	LOGGER.fine();
CONFIG	-	LOGGER.config();
INFO	INFO	LOGGER.info();
WARNING	WARN	LOGGER.warning();
SEVERE	ERROR	LOGGER.severe();
-	ERROR	-

Ha a hiba mellé a kivételt is szeretnénk továbbadni (ez Log4J esetén egyszerű overload metódus):

```
1 LOGGER.log(Level.SEVERE, e.getMessage(), e);
```

```
1 <dependency>
2   <groupId>log4j</groupId>
3   <artifactId>log4j</artifactId>
4   <version>1.2.17</version>
5   <scope>provided</scope>
6 </dependency>
```

A fenti függőséget minden module számára elérhetővé kell tennünk. Ehhez minden module `pom.xml`-jének a módosítása szükséges. A verziószám kitehető egy változóba, ami sokat segít, de 6 sor még ettől 6 sort minden állományban.. a parent `pom.xml` állomány apró módosításával a 6 sort redukálhatjuk 4-re...



```
1 <project [...]>
2   [...]
3   <properties>
4     [...]
5     <version.log4j>1.2.17</version.log4j>
6 </properties>
7
8 <dependencyManagement>
9   <dependencies>
10    <dependency>
11      <groupId>log4j</groupId>
12      <artifactId>log4j</artifactId>
13      <version>${version.log4j}</version>
14      <scope>provided</scope>
15    </dependency>
16  </dependencies>
17 </dependencyManagement>
18 </project>
```

A `dependencyManagement` gyermekei olyan konfigurációs elemeket tartalmaznak, melyek elérhetővé válnak a *module*-ok számára is. Itt komplexebb konfiguráció is elképzelhető 1-1 függőség kapcsán, így nagyobb mennyiségű sor redundancia csökkentése is elképzelhető, nem csupán 1-2 sor mint a `log4j` esetében.

pom.xml



```
1 <project [..]>
2   [..]
3   <dependencies>
4     [..]
5     <dependency>
6       <groupId>log4j</groupId>
7       <artifactId>log4j</artifactId>
8     </dependency>
9     [..]
10  </dependencies>
11 </project>
```

A groupId és az artifactId megadása egyértelműsíti (jelen esetben) a parent project mely függőségi konfigurációjára szükséges itt hivatkozni! Ugyanez az átalakítás megtehető a javax:javaee-api függőségre is!



```
1 <project [..]>
2   [..]
3   <build>
4     <pluginManagement>
5       <plugins>
6         <plugin>
7           <groupId>org.apache.maven.plugins</groupId>
8           <artifactId>maven-ejb-plugin</artifactId>
9           <version>${version.plugin.ejb}</version>
10        </plugin>
11      </plugins>
12    </pluginManagement>
13  </build>
14
15 </project>
```

A `dependencyManagement` hasonlóan létezik egy `pluginManagement` element is (a `build` element-en belül), mely a `plugin` függőségek közös helyen való konfigurációját teszi lehetővé. Teljesen analog a használatuk.



A jelenlegi gyakorlatban itt most a verzió információ sora tűnt el, de minden további konfigurációs elem is ugyanezzel a módszerrel kezelhető, vagyis kizárólag a parent projekt-ben kell módosítást végezni ezt követően.

```
1 <project [...]>
2   [...]
3   <build>
4     <plugins>
5       <plugin>
6         <groupId>org.apache.maven.plugins</groupId>
7         <artifactId>maven-ejb-plugin</artifactId>
8       </plugin>
9     </plugins>
10  </build>
11 </project>
```

pom.xml



<http://www.postgresql.org/>

Verzió: **v10.0** (2017.10.05.)

Verzió: **v9.6.5** (2017.09.31.)

Telepítés: installer

Alapértelmezett adatok: **5432** (port) és **postgres** (database)

Alapértelmezett felhasználó: **postgres** (user) és **root** (password)

Szerkesztő: **pgAdmin III** vagy **pgAdmin IV**

Környezeti változók:

- ▷ **PSQL_HOME** → c:\ProgramFiles\PostgreSQL\9.4
- ▷ **Path** módosítása → %Path%;%PSQL_HOME%\bin

Professionális eszköz

A mellett, hogy a PostgreSQL hasonlóan professzionális management lehetőséggel rendelkezik, mint pl. az Oracle DB vagy a SAP Adaptive Server Enterprise (korábban Sybase ASE), létezik production környezetben rendszer melyben 4 TB adatot kezel.



Verzió lekérdezése

```
1 > psql --version
2 psql (PostgreSQL) 9.4.1
```

Csatlakozás (CLI)

```
1 > psql -d postgres -h localhost -p 5432 -U postgres
```

Csatlakozás és utasítások végrehajtása állományból

```
1 > psql -d postgres -h localhost -p 5432 -U postgres -f [FILENAME]
```

Oracle MySQL

A legnépszerűbb nyílt forráskódú adatbáziskezelő



<https://www.mysql.com/>

Letöltés: <https://dev.mysql.com/downloads/>

A Community verzió ingyenes (GPL).

Verzió: **v5.7.20**

Alapértelmezett adatok: **3306** (port)

Alapértelmezett felhasználó: **root** (user) és - (password)

Szerkesztő: **MySQL Workbench**

Környezeti változók:

- ▷ **MYSQL_HOME** → c:\ProgramFiles\MySQL\5.7\bin
- ▷ **Path** módosítása → %Path%;%MYSQL_HOME%\bin

Népszerű eszköz

A PHP-s webfejlesztés világában a MySQL szinte nélkülözhetetlen, szinte nincs olyan Shared Hosting szolgáltatás, ahol ne a MySQL lenne elérhető mint perzisztens réteg. Különbféle engine-eket támogat, mindegyiknek meg(lehet) a maga előnye/hátránya. Az InnoDB támogat professzionális használatot (hibái ellenére), viszont pl. *role*-okat ez sem kezel még.



 [gradle|maven]\jboss\bookstore\database

```
1 CREATE DATABASE bookstoredb;
```

create-database.sql

```
1 CREATE ROLE bookstore_role WITH NOSUPERUSER NOCREATEDB  
   NOCREATEROLE;
```

create-role.sql

```
1 CREATE USER bookstore_user;  
2 ALTER USER bookstore_user PASSWORD '123topSECret321';  
3 GRANT bookstore_role TO bookstore_user;
```

create-user.sql



```
1 > psql -d postgres -h localhost -p 5432 -U postgres -f
    create-database.sql
2
3 > psql -d postgres -h localhost -p 5432 -U postgres -f
    create-role.sql
4
5 > psql -d postgres -h localhost -p 5432 -U postgres -f
    create-user.sql
```

Csatlakozás a **bookstoredb** adatbázishoz a **bookstore_user** felhasználóval

```
1 > psql -d bookstoredb -h localhost -p 5432 -U bookstore_user
```



```
1 CREATE TABLE bookcategory (  
2     bookcategory_id INTEGER NOT NULL,  
3     bookcategory_name CHARACTER VARYING(100) NOT NULL,  
4     CONSTRAINT PK_BOOKCATEGORY_ID PRIMARY KEY (bookcategory_id)  
5 );  
6  
7 ALTER TABLE bookcategory OWNER TO postgres;  
8  
9 CREATE TABLE book (  
10     book_id SERIAL NOT NULL,  
11     book_isbn CHARACTER VARYING(100) NOT NULL,  
12     book_author CHARACTER VARYING(100) NOT NULL,  
13     book_title CHARACTER VARYING(100) NOT NULL,  
14     book_bookcategory_id INTEGER NOT NULL,  
15     book_price REAL NOT NULL,  
16     book_number_of_pages INTEGER NOT NULL,  
17     CONSTRAINT PK_BOOK_ID PRIMARY KEY (book_id),  
18     CONSTRAINT FK_BOOK_BOOKCATEGORY FOREIGN KEY (book_bookcategory_id)  
19     REFERENCES bookcategory (bookcategory_id) MATCH SIMPLE ON UPDATE RESTRICT ON DELETE  
20     RESTRICT  
21 );  
22 ALTER TABLE book OWNER TO postgres;  
23  
24 CREATE UNIQUE INDEX UI_BOOK_ISBN ON book USING btree (book_isbn);
```

SERIAL: létre fog jönni egy
book_book_id_seq nevű db
sequence.

create-schema.sql



```
1 GRANT SELECT, INSERT, UPDATE, DELETE ON bookcategory, book TO
   bookstore_role;
2
3 GRANT USAGE, SELECT, UPDATE ON book_book_id_seq TO bookstore_role;
```

grant-access.sql

```
1 > psql -d bookstoredb -h localhost -p 5432 -U postgres -f
   create-schema.sql
2
3 > psql -d bookstoredb -h localhost -p 5432 -U postgres -f
   grant-access.sql
```



```
1 INSERT INTO bookcategory (bookcategory_id, bookcategory_name)
  VALUES (0, 'SCIFI');
2 INSERT INTO bookcategory (bookcategory_id, bookcategory_name)
  VALUES (1, 'LITERATURE');
3 INSERT INTO bookcategory (bookcategory_id, bookcategory_name)
  VALUES (2, 'HISTORICAL');
4 INSERT INTO bookcategory (bookcategory_id, bookcategory_name)
  VALUES (3, 'PHILOSOPHY');
5
6 INSERT INTO book (book_isbn, book_author, book_title,
  book_bookcategory_id, book_price, book_number_of_pages)
  VALUES ('978-0441172719', 'Frank Herbert', 'Dune', 0, 3500,
  896);
7 [...]
```

initial-data.sql

```
1 > psql -d bookstoredb -h localhost -p 5432 -U postgres -f
  initial-data.sql
```

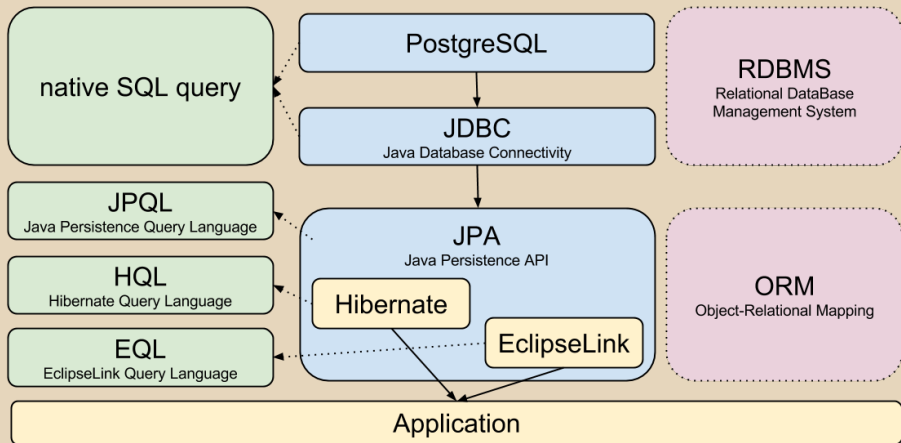
JDBC

Java SE témakörében ez gyakorlatban a `java.sql` csomag osztályainak használatát jelenti. Ezen kívül szükség lesz még a használni kívánt adatbázis **JDBC driver**-ére (**Java DataBase Connectivity**, tipikusan egy `jar` file, mely az ODBC (Open Database Connectivity) driver Java-s változata.

```
1 Class.forName("org.postgresql.Driver");
2 Connection connection =
    DriverManager.getConnection("jdbc:postgresql://localhost:5432/bookstore_user", "123topSEcRet321");
3
4 Statement statement = connection.createStatement();
5 ResultSet rs = statement.executeQuery("SELECT book_id, book_title
    FROM book");
6 while (rs.next()) {
7     long id = rs.getLong("book_id");
8     String title = rs.getString("book_title");
9 }
10 rs.close();
11 statement.close();
12
13 connection.close();
```

A példakód a hibakezelést teljesen mellőzi. Működő példa a shopping projektben megtalálható, de a JPA használatához ennek előzetes megismerése nem szükséges. A JDBC ily használatának legnagyobb hátránya a **nem type-safe** kezelés.

Java Persistence API



- ▷ **Új module** létrehozása: PostgreSQL JDBC driver
- ▷ **Module regisztrálása** a standalone domain-hez
- ▷ **Database driver regisztrálása** a standalone domain-ben
- ▷ **bookstoredb datasource létrehozása** a standalone domain-ben

 environment\modules\org\postgresql\main\

Könyvtár struktúra

```
[JBOSS\_HOME]/modules/  
  org/ → name directory  
    postgresql/ → name directory  
      main/ → slot directory  
        module.xml → descriptor  
        postgresql-42.1.4.jar → JDBC driver https://jdbc.postgresql.org/
```

```
1 <module xmlns="urn:jboss:module:1.1" name="org.postgresql">  
2   <resources>  
3     <resource-root path="postgresql-42.1.4.jar"/>  
4   </resources>  
5   <dependencies>  
6     <module name="javax.api"/>  
7     <module name="javax.transaction.api"/>  
8   </dependencies>  
9 </module>
```


[JBASS_HOME] | **standalone** | configuration | standalone.xml

```
1 <subsystem xmlns="urn:jboss:domain:ee:1.2">
2   <global-modules>
3     <module name="org.postgresql" slot="main"/>
4   </global-modules>
5   [..]
6 </subsystem>
```

standalone.xml

[JBOSS_HOME] | **standalone** | configuration | standalone.xml

```
1 <subsystem xmlns="urn:jboss:domain:datasources:1.2">
2   <datasources>
3     [..]
4     <drivers>
5       [..]
6       <driver name="postgresql" module="org.postgresql">
7         <xa-datasource-class>org.postgresql.xa.PGXADatasource</xa-datasource-class>
8       </driver>
9     </drivers>
10  </datasources>
11 </subsystem>
```

standalone.xml

[JBOSS_HOME] | **standalone** | configuration | standalone.xml

```
1 <subsystem xmlns="urn:jboss:domain:datasources:1.2">
2   <datasources>
3     [...]
4     <datasource jndi-name="java:jboss/datasources/bookstores"
5       pool-name="BookStoreDSPool" enabled="true" use-java-context="true">
6       <connection-url>jdbc:postgresql://localhost:5432/bookstoredb</connection-url>
7       <driver>postgresql</driver>
8       <security>
9         <user-name>bookstore_user</user-name>
10        <password>123topSEcRet321</password>
11      </security>
12    </datasource>
13  </datasources>
14 </subsystem>
```

standalone.xml

JNDI name: java:jboss/datasources/bookstores

JDBC url: jdbc:postgresql://localhost:5432/bookstoredb

[JBOSSE_HOME] | standalone | configuration | standalone.xml

```
1 <subsystem xmlns="urn:jboss:domain:datasources:1.2">
2   <datasources>
3     <datasource jndi-name="java:jboss/datasources/bookstoreDS"
4       pool-name="BookStoreDSPool" enabled="true" use-java-context="true">
5       [..]
6       <validation>
7         <check-valid-connection-sql>SELECT 1</check-valid-connection-sql>
8         <validate-on-match>true</validate-on-match>
9         <background-validation>false</background-validation>
10      </validation>
11      <statement>
12        <share-prepared-statements>false</share-prepared-statements>
13      </statement>
14    </datasource>
15    [..]
16  </datasources>
</subsystem>
```

standalone.xml

- ▷ 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: **bookstoresd**
 - * JNDI name: jdbc/datasource/bookstoresd
 - * Target: "myserver"
 - * Database type: PostgreSQL
 - Next, Next..
 - * Database name: **bookstoredb**
 - * Host name: **localhost**
 - * Port: **5432** (default)
 - * Database user name: **bookstore_user**
 - * Password: **123topSECret321**
 - Érdeemes tesztelni a kapcsolatot a felületen keresztül

Könyvtár struktúra

```
bookstore/  
  [..]  
  bs-persistence/  
    src/  
      main/  
        java/  
        resources/  
          META-INF/  
            ejb-jar.xml  
            persistence.xml  
      pom.xml  
  [..]
```



```
1 <project [..]>
2   [..]
3   <modules>
4     [..]
5     <module>bs-persistence</module>
6   </modules>
7   [..]
8 </project>
```

pom.xml



```
1 <project [..]>
2   <modelVersion>4.0.0</modelVersion>
3   <artifactId>bs-persistence</artifactId>
4   <packaging>ejb</packaging>
5   <name>BookStore Persistence Services</name>
6
7   <parent>
8     <groupId>hu.qwaevisz.bookstore</groupId>
9     <artifactId>bookstore</artifactId>
10    <version>1.0</version>
11  </parent>
12
13  <dependencies>
14    [..]
15  </dependencies>
16  <build>
17    [..]
18  </build>
19 </project>
```

pom.xml



```
1 <project [...]>
2   [...]
3   <dependencies>
4     <dependency>
5       <groupId>javax</groupId>
6       <artifactId>javaee-api</artifactId>
7     </dependency>
8     <dependency>
9       <groupId>log4j</groupId>
10      <artifactId>log4j</artifactId>
11    </dependency>
12  </dependencies>
13  <build>
14    <plugins>
15      <plugin>
16        <groupId>org.apache.maven.plugins</groupId>
17        <artifactId>maven-ejb-plugin</artifactId>
18      </plugin>
19    </plugins>
20  </build>
21 </project>
```

Mind a három itt hivatkozott *artifact* a parent project pom.xml-jében van definiálva.

JBoss 6.4 runtime a **Hibernate 4.3.10.Final**-t használja mint JPA implementáció, de mi kizárólag a JPA API-t használjuk fejlesztés közben, ezért nincs szükségünk erre a függőségre:
'org.hibernate:hibernate-core:4.3.10.Final'



```
1 <project [..]>
2   [..]
3   <dependencies>
4     <dependency>
5       <groupId>hu.qwaevisz.bookstore</groupId>
6       <artifactId>bs-persistence</artifactId>
7       <version>${project.parent.version}</version>
8       <scope>provided</scope>
9     </dependency>
10    [..]
11  </dependencies>
12
13  <build>
14    [..]
15  </build>
16 </project>
```

A *bs-weblayer* használja a *bs-ejb-service*-t, hogy elérje a *BookFacade* EJB-t.

A *bs-ejb-service* használja *bs-persistence*-t hogy elérje a *BookService* EJB-t.

pom.xml

hibernate.dialect: megadása nem kötelező, de ajánlott. Segítségével a Hibernate képes lesz PostgreSQL specifikus natív query-ket is generálni. Ha itt nem adjuk meg, akkor is működni fog a project (a generált ANSI SQL jelen esetben elegendő).

src | main | **resources** | META-INF | persistence.xml

```
1 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
4         http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
5     version="2.0">
6     <persistence-unit name="bs-persistence-unit"
7         transaction-type="JTA">
8         <jta-data-source>java:jboss/datasources/bookstores</jta-data-source>
9         <properties>
10            <property name="hibernate.dialect"
11                value="org.hibernate.dialect.PostgreSQLDialect"/>
12            <property name="hibernate.show_sql" value="true"/>
13            <property name="hibernate.format_sql" value="true"/>
14        </properties>
15    </persistence-unit>
16</persistence>
```

Persistence unit neve: bs-persistence-unit
JNDI name: java:jboss/datasources/bookstores

src | main | resources | META-INF | persistence.xml

```
1 <?xml version="1.0"?>
2 <persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
5         http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
6     version="2.1">
7     <persistence-unit name="bs-persistence-unit"
8         transaction-type="JTA">
9         <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
10        <jta-data-source>jdbc/datasource/bookstores</jta-data-source>
11        <class>hu.qwaevisz.bookstore.persistence.entity.Book</class>
12        <properties>
13            <property name="eclipselink.logging.level" value="FINE"/>
14        </properties>
15    </persistence-unit>
16</persistence>
```

Az Oracle WebLogic a **EclipseLink**-et (TopLink) használja mint JPA Provider, és a Hibernate beállítása csak úgy lehetséges, ha kicseréljük minden függőségével együtt az elemeket (nem javasolt). Viszont a JPA Provider-ek között is szabadon mozoghatunk a JavaEE világában, egyedül a descriptorokat kell cserélnünk/finomhangolnunk.

Book entitás

Identification mező

Ha nem adjuk meg a `@SequenceGenerator` és `@GeneratedValue` annotációkat, beszúráskor egy `org.hibernate.id.IdentifierGenerationException` fogunk kapni, melyben jelzi a Hibernate hogy: "ids for this class must be manually assigned before calling save()".

```
1 package hu.qwaevisz.persistence.entity;
2     Database table name: book
3 @Entity
4 @Table(name = "book")
5 public class Book implements Serializable {
6     Database sequence name: book_book_id_seq
7     @Id
8     @SequenceGenerator(name = "generatorBook", sequenceName =
9         "book_book_id_seq", allocationSize = 1)
10    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
11        "generatorBook")
12    @Column(name = "book_id", nullable = false)
13    private Long id;
14    Database column name: book_id
15    [...]
16 }
```

Book entitás

További mezők

```
1 @Column(name = "book_isbn", nullable = false)
2 private String isbn;
3
4 @Column(name = "book_author", nullable = false)
5 private String author;
6
7 @Column(name = "book_title", nullable = false)
8 private String title;
9
10
11 @Enumerated(EnumType.ORDINAL)
12 @Column(name = "book_bookcategory_id", nullable = false)
13 private BookCategory category;
14
15 @Column(name = "book_price", nullable = false)
16 private Double price;
17
18 @Column(name = "book_number_of_pages", nullable = false)
19 private Integer numberOfPages;
```

A bookcategory adatbázis szinten egy tábla, ORM szinten viszont egy enum (CRUD műveletek nem értelmezettek rajta).

Book.java

BookCategory

ORM által használt enum

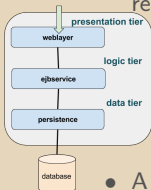
```
1 package hu.qwaevisz.bookstore.persistence.entity.trunk;
2
3 public enum BookCategory {
4
5     SCIFI,
6     LITERATURE,
7     HISTORICAL,
8     PHILOSOPHY;
9
10 }
```

BookCategory.java



▷ A **BookStore** alkalmazás két szinten definiál szolgáltatásokat.

- A `bs-ejbservice` project szintjén definiált EJB service-ek a "customer" (megrendelő) igényei szerint alakítandóak.



- Az interface által megfogalmazott igényekre üzleti *use-case*-ek születnek.
- Az interface által használt típusok a **stub**-ok, melyek technikai szinten publikusan kifelé nézve.
- Az interface által használt kivételek technikai szinten publikusak, vagy publikus hibaüzenetekké alakíthatóak. Technikai részleteket ritkán tartalmaznak (security).

- A `bs-persistence` project szintjén definiált EJB service-ek a fejlesztők belső szabályai alapján, elsősorban technikai követelmények és lehetőségek szerint alakulnak.

- Az interface által megfogalmazott igényeket a fejlesztők maguk alakítják ki, egészséges egyensúlyt tartva a redundancia mentes, karbantartható, tiszta és olykor akár future-proof követelmények alapján.
- Az interface által használt típusok az **entity**-k, melyek az alkalmazás belső használatú DTO-i, kifelé legtöbbször nem publikusak.
- Az interface által használt kivételek részletes technikai információkat is tartalmazhatnak, legtöbbször ezek a "kliens" oldalon nem jelennek meg és nem alakíthatóak közvetlenül publikus hibaüzenetté.

BookService - Persistence SLSB

Local interface

```
1 package hu.qvaevisz.bookstore.persistence.service;
2 [...]
3 @Local
4 public interface BookService {
5     [...]
6     Book read(String isbn) throws PersistenceServiceException;
7
8     List<Book> readAll() throws PersistenceServiceException;
9     [...]
10 }
```

BookService.java

BookService - Persistence SLSB

Implementáció

```
1 @Stateless(mappedName = "ejb/bookService")
2 @TransactionManagement(TransactionManagementType.CONTAINER)
3 @TransactionAttribute(TransactionAttributeType.REQUIRES_NEW)
4 public class BookServiceImpl implements BookService {
5
6     @PersistenceContext(unitName = "bs-persistence-unit")
7     private EntityManager entityManager;
8
9     @Override
10    public Book read(final String isbn) throws PersistenceServiceException {
11        Book result = null;
12        try {
13            result = this.entityManager.createNamedQuery(Book.GET_BY_ISBN,
14                Book.class).setParameter("isbn", isbn).getSingleResult();
15        } catch (final Exception e) {
16            throw new PersistenceServiceException("Unknown error when fetching Book by ISBN
17                (" + isbn + ")! " + e.getLocalizedMessage(), e);
18        }
19        return result;
20    }
21    [..
22 }
```

JPA Named Query: Book.GET_BY_ISBN ("Book.getByIsbn")

Paraméter neve: isbn

getSingleResult(): pontosan egy rekord létezik (különben kivételt dob)

getResultList(): List<Book> lesz a visszatérési érték

JPA Named Query

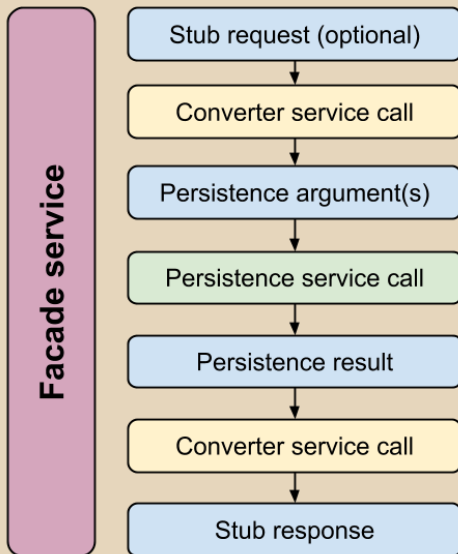
JPQL syntax

```
1 @Entity
2 @Table(name = "book")
3 @NamedQueries(value = { //
4     @NamedQuery(name = Book.GET_BY_ISBN, query = "SELECT b FROM
5         Book b WHERE b.isbn=:isbn"),
6     @NamedQuery(name = Book.GET_ALL, query = "SELECT b FROM Book
7         b ORDER BY b.title"),
8     [...]
9 })
10 public class Book implements Serializable {
11     public static final String GET_BY_ISBN = "Book.getByIsbn";
12     public static final String GET_ALL = "Book.getAll";
13     [...]
14 }
```

Book.java

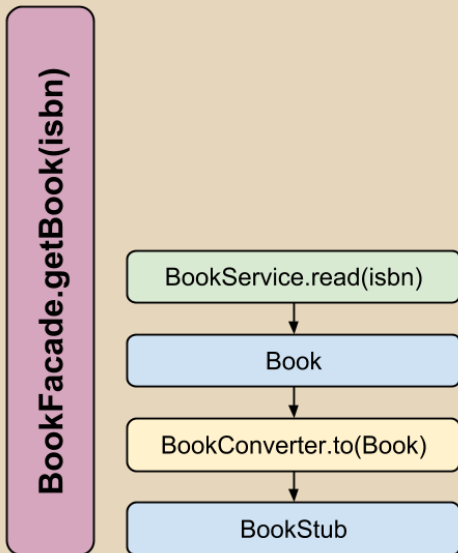
Facade réteg felelőssége

EJB Service project (bs-ejbservice)



Könyv lekérdezése ISBN alapján

EJB Service project (bs-ejbservice)



BookConverter

EJB Service project (bs-ejbservice)

```
1 @Local
2 public interface BookConverter {
3     BookStub to(Book book);
4     List<BookStub> to(List<Book> books);
5 }
```

BookConverter.java

Nagyon jó 3rd party library-k léteznek konverziós célra, pl. **MapStruct** (<http://mapstruct.org/>).

```
1 @Stateless
2 public class BookConverterImpl implements BookConverter {
3
4     @Override
5     public BookStub to(Book book) {
6         final BookCategoryStub category =
7             BookCategoryStub.valueOf(book.getCategory().toString());
8         return new BookStub(book.getIsbn(), book.getAuthor(), book.getTitle(), category,
9             book.getPrice(), book.getNumberOfPages());
10 }
```

BookConverterImpl.java

BookFacade módosítása

EJB Service project (bs-ejbservice)

```
1 @Stateless(mappedName = "ejb/bookFacade")
2 public class BookFacadeImpl implements BookFacade {
3
4     @EJB
5     private BookService service;
6
7     @EJB
8     private BookConverter converter;
9
10    @Override
11    public BookStub getBook(String isbn) throws FacadeException {
12        try {
13            final BookStub stub = this.converter.to(this.service.read(isbn));
14            if (LOGGER.isDebugEnabled()) {
15                LOGGER.debug("Get Book by isbn (" + isbn + ") --> " + stub);
16            }
17            return stub;
18        } catch (final PersistenceServiceException e) {
19            LOGGER.error(e, e);
20            throw new FacadeException(e.getLocalizedMessage());
21        }
22    }
23    [...]
24 }
```

BookFacadeImpl.java



A korábbi `BookPingServlet` mellett készítsünk egy valós megjelenítési réteget Servlet/JSP alapokon, követve a *Modell-View-Controller* (MVC) tervezést.

- ▷ **BookController** servlet, mely URI paraméterben kapja az ISBN számot, mely alapján lekéri az érintett könyvet, majd a `BookStub` példányt elhelyezi a `HttpServletRequest`-ben, és a kérést *forward*-olja egy *JSP* lapnak.
- ▷ **book.jsp** által generált servlet, mely kiveszi a `HttpServletRequest`-ből a `BookStub` példányát és megjeleníti XHTML tagek ölelésében.

BookController

Weblayer project (bs-weblayer)

```
1 package hu.qvaevisz.bookstore.weblayer.servlet;
2 [...]
3 @WebServlet("/Book")
4 public class BookController extends HttpServlet {
5     [...]
6     @EJB
7     private BookFacade facade;
8
9     @Override
10    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
11        String isbn = request.getParameter("isbn");
12        [...]
13        try {
14            BookStub book = this.facade.getBook(isbn);
15            request.setAttribute("book", book);
16        } catch (final FacadeException e) {
17            LOGGER.error(e, e);
18        }
19        RequestDispatcher view = request.getRequestDispatcher("book.jsp");
20        view.forward(request, response);
21    }
22    [...]
23 }
```

BookController.java

book.jsp

Weblayer project (bs-weblayer)

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
2 <%@ page import="hu.qwaevisz.bookstore.ejbsservice.domain.BookStub" %>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>:: Book ::</title>
8 </head>
9 <body>
10 <jsp:useBean id="book" class="hu.qwaevisz.bookstore.ejbsservice.domain.BookStub"
    scope="request" />
11 <h1>
12 <jsp:getProperty name="book" property="author" />:
13 <jsp:getProperty name="book" property="title" />
14 </h1>
15 <div>
16 <label>ISBN: </label>
17 <span><jsp:getProperty name="book" property="isbn" /></span>
18 </div>
19 <div>
20 <label>Number of pages: </label>
21 <span><jsp:getProperty name="book" property="numberOfPages" /></span>
22 </div>
23 [..]
24 </body>
25 </html>
```

book.jsp



<http://localhost:8080/bs-weblayer/Book?isbn=978-0441172719>

További feladatok:

- ▷ Összes könyv listázása (JPA query, persistence service, facade service, converter ejb)
 - Criteria alapján szűrés megvalósítása (opcionális)
- ▷ `list.jsp` és `BookListController` servlet megvalósítása
- ▷ `list.jsp` és `book.jsp` összekötése (a controller servlet-ek összekötése)
- ▷ `page.css` file készítése a weblapokhoz
- ▷ ...



A **BookStore** alkalmazás a lekérdezési lehetőségek mellett teljes CRUD szerkesztési lehetőséget is tartalmaz. Ennek technikai részleteit JPA szempontjából később tekintjük át.

A következő néhány slide-on - csupán előzetesen - az érintett műveletek persistence oldala lesz olvasható. A megoldás nélkülözi a tranzakció- és hibakezelést. Mindezek később a **school** elnevezésű projektben lesznek részletezve.



```
1 @Stateless(mappedName = "ejb/bookService")
2 public class BookServiceImpl implements BookService {
3     [...]
4     @PersistenceContext(unitName = "bs-persistence-unit")
5     private EntityManager entityManager;
6
7     @Override
8     public Book create(final String isbn, final String author, final String title, final
9         int numberOfPages, final double price, final BookCategory category)
10        throws PersistenceServiceException {
11        [...]
12        final Book book = new Book(isbn, author, title, numberOfPages, price, category);
13        this.entityManager.persist(book);
14        [...]
15        return book;
16    }
17    [...]
18 }
```

BookServiceImpl.java



```
1 @Stateless(mappedName = "ejb/bookService")
2 public class BookServiceImpl implements BookService {
3     [...]
4     @PersistenceContext(unitName = "bs-persistence-unit")
5     private EntityManager entityManager;
6
7     @Override
8     public Book update(final String isbn, final String author, final String title, final
9         int numberOfPages, final double price, final BookCategory category)
10         throws PersistenceServiceException {
11         final Book book = this.read(isbn);
12         book.setAuthor(author);
13         book.setTitle(title);
14         book.setNumberOfPages(numberOfPages);
15         book.setPrice(price);
16         book.setCategory(category);
17         [...]
18         return this.entityManager.merge(book);
19     }
20 }
```

BookServiceImpl.java



```
1 @Stateless(mappedName = "ejb/bookService")
2 public class BookServiceImpl implements BookService {
3     [...]
4     @PersistenceContext(unitName = "bs-persistence-unit")
5     private EntityManager entityManager;
6
7     @Override
8     public void delete(final String isbn) throws PersistenceServiceException {
9         this.entityManager.createNamedQuery(Book.REMOVE_BY_ISBN).setParameter("isbn",
10             isbn).executeUpdate();
11     }
12     [...]
13 }
```

BookServiceImpl.java

A hivatkozott JPQL az alábbi:

```
DELETE FROM Book b WHERE b.isbn=:isbn
```

A kódolás - mint fejlesztési folyamat - szét nem választható egységet képez a teszteléssel. Jelen diasorozatnak azonban nem scope-ja a kódolást a minden ponton vele együtt járó teszteléssel együtt bemutatni.

- ▷ **egység** vagy komponens- teszt: az objektum-orientált fejlesztésben egy osztály (egy felelősség) önálló, független tesztjét értjük egység teszt alatt. Minden környezeti és közel minden programozott függőséget (elsősorban az aggregációkat és kompozíciókat) ki kell küszöbölni, meg kell kerülni. Célja legtöbbször a programhibák leggyorsabb és legpontosabb felderítése, illetve relatíve magas *coverage* esetén a refactorálás támogatása.
- ▷ **integrációs** teszt: néhány komponens együttes tesztelése, célja legtöbbször a fejlesztés menetének meggyorsítása. Mindig helyben dönthető el, hogy mekkora környezeti függőséget engedélyezünk a teszt számára.
- ▷ **system** vagy rendszer- teszt: tipikusan a fejlesztők által végzett legmagasabb tesztelés, melyben a rendszer vagy annak egy önállóan működő része "minden" függőségével együtt kerül tesztelésre (itt képke kerülhetnek szimulátorok, melyeket szintjén a fejlesztőknek kell lefejlesztenie).
- ▷ **end-to-end** vagy átvételi- teszt: egy jól működő projektben az end-to-end tesztelést külön, a fejlesztőktől független csapat végzi. Mérnökökből álló tesztelői (integrációs és verifikációs) csapat automata tesztekben gondolkodik, míg sok helyen végzik ezt a szintet manuális lépések segítségével. Webalkalmazás fejlesztés során gyakoriak pl. a *Selenium* tesztek.

- ▷ Körülményes egy-egy named query tesztelése miatt deploy-olni és manuális tesztelni (nagyon sok idő, sok elgépelési lehetőség).
- ▷ Adatbázis műveletek, named query-k tesztelése nem unit-test felelősség.
- ▷ A JPA API használható *standalone* alkalmazásban is. Ez is egy mód a tesztelésre, de ez egy elkülönített kódrész lesz, mely automata tesztelésre legtöbbször nem alkalmas (de a fejlesztés támogatására igen).
- ▷ Talán a legelegánsabb megoldás **integration test**-eket készíteni a persistence réteg tesztelésére, melyben a megírt named query-k és entity manager műveletek valóban lefutnak az adatbázison, azok hatékonysága és teljesítménye mérhető és ellenőrizhető lesz, függetlenül az alkalmazás szervertől (az alkalmazás szervert kivesszük a tesztelésből, de az adatbázist nem).
- ▷ Az integrációs tesztek tipikusan nem futnak az egység tesztekkel együtt, így a külön futásukat illendő megoldani (többféle megoldás lehetséges).
- ▷ A persistence réteg osztályait (ahol értelme van, és nem boilerplate algoritmusok érintettek) természetesen egység tesztekkel le lehet fedni. Az egység tesztelésről a **school** elnevezésű projekt fog részletesebben foglalkozni.

Standalone Persistence Unit az integrációs tesztekhez

src | main | resources | META-INF | persistence.xml

```
1 <persistence [..]>
2   [..]
3   <persistence-unit name="bs-persistence-test-unit" transaction-type="RESOURCE_LOCAL">
4     <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
5     <class>hu.qwaevisz.bookstore.persistence.entity.Book</class>
6     <properties>
7       <property name="javax.persistence.jdbc.url"
8         value="jdbc:postgresql://localhost:5432/bookstoredb" />
9       <property name="javax.persistence.jdbc.user" value="bookstore_user" />
10      <property name="javax.persistence.jdbc.password" value="123topSEcRet321" />
11      <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
12
13      <property name="hibernate.dialect"
14        value="org.hibernate.dialect.PostgreSQLDialect"/>
15      <property name="hibernate.show_sql" value="true"/>
16      <property name="hibernate.format_sql" value="true"/>
17      <property name="hibernate.use_managed_defaults" value="false" />
18    </properties>
19  </persistence-unit>
20 </persistence>
```

A JPA a classpath fix helyén keresi a persistence.xml-t. E miatt a jboss.as.jpa.managed értékét false-ra állítva elérhetjük hogy runtime a JBoss ne foglalkozzon ezzel a standalone persistence unit-tal (ClassNotFoundException hibát okoz e nélkül). JNDI név helyett most definiálva van a **provider** class valamint a csatlakozás adatai. Mivel nincs container aki át nézné a kódot annotációk után, an entity-ket fel kell sorolni.

Persistence unit neve: bs-persistence-test-unit

Integration teszt classpath

Az integrációs tesztek JavaEE container nélkül futnak (pl. gradle/maven/eclipse JVM-je futtatja őket), így nem elégséges csupán a interface-ek/enum-ok halmazát tartalmazó függőség beállítása. Érdeemes a futtató környezet által használt runtime függőségeket felsorolni, de a JavaEE szabványos világában akár más runtime környezetet is felépíthetünk.

Szükséges runtime függőségek:

▷ PostgreSQL

- `org.postgresql:postgresql:9.4+`

▷ Hibernate

- `org.hibernate:hibernate-core:4.3.10.Final`
- `org.hibernate:hibernate-entitymanager:4.3.10.Final`

▷ JavaEE 6.0 implementáció

- `org.jboss.spec:jboss-javaee-6.0:3.0.3.Final`
- **Fontos!** Ezzel egy időben nem szabad a classpath-ra tenni a JavaEE 6.0 API-t (a teszt classpath része a production classpath):
 - `javax:javaee-api:6.0`



```
1 <project [..]>
2   [..]
3   <properties>
4     [..]
5     <version.plugin.surefire>2.19.1</version.plugin.surefire>
6     <version.testng>6.11</version.testng>
7     <version.postgresql>9.4.1212</version.postgresql>
8     <version.hibernate>4.3.10.Final</version.hibernate>
9     <version.jbossjavaee>3.0.3.Final</version.jbossjavaee>
10  </properties>
11  <dependencyManagement>
12    <dependencies>
13      [..]
14      <dependency>
15        <groupId>org.testng</groupId>
16        <artifactId>testng</artifactId>
17        <version>${version.testng}</version>
18        <scope>test</scope>
19      </dependency>
20    </dependencies>
21  </dependencyManagement>
22  [..]
23 </project>
```

pom.xml

Maven Persistence project

Módosítás



```
1 <dependencies>
2   [..]
3   <dependency>
4     <groupId>org.testng</groupId><artifactId>testng</artifactId>
5   </dependency>
6   <dependency>
7     <groupId>org.postgresql</groupId><artifactId>postgresql</artifactId>
8     <version>${version.postgresql}</version><scope>test</scope>
9   </dependency>
10  <dependency>
11    <groupId>org.hibernate</groupId><artifactId>hibernate-core</artifactId>
12    <version>${version.hibernate}</version><scope>test</scope>
13  </dependency>
14  <dependency>
15    <groupId>org.hibernate</groupId><artifactId>hibernate-entitymanager</artifactId>
16    <version>${version.hibernate}</version><scope>test</scope>
17  </dependency>
18  <dependency>
19    <groupId>org.jboss.spec</groupId><artifactId>jboss-javaee-all-6.0</artifactId>
20    <version>${version.jbossjavaee}</version><scope>test</scope>
21  </dependency>
22 </dependencies>
```

A testng mellett (mely a *parent*-ben van definiálva) felvesszük a további függőségeket is **test** scope-pal (kizárólag a tesztek futásakor pakolja fel ezeket a Maven a classpath-ra).

Maven Surefire Plugin

Test goal konfigurálása



A **test** goal konfigurálására és riport generálásra használható. Ami az integrációs tesztek szempontjából lényeges:

- ▷ lehetőséget ad kizárni classpath elemeket, melyek a teszt fázisban nem szabad hogy a classpath-ra kerüljenek
- ▷ lehetőséget ad TestNG suite.xml konfiguráció használatára, mely egy lehetséges módja pl. az integrációs és egység tesztek szétválasztásának

```
1 <build>
2   [..]
3   <plugins>
4     <plugin>
5       <groupId>org.apache.maven.plugins</groupId>
6       <artifactId>maven-surefire-plugin</artifactId>
7       <version>2.19.1</version>
8       <configuration>
9         [..]
10      </configuration>
11    </plugin>
12  </plugins>
13 </build>
```

Maven Persistence project

Surefire plugin konfiguráció



A kizárandó classpath entry *artifact* URI-ját megadva elérhetjük hogy ne akadjon össze a JavaEE API és implementáció az integrációs tesztek futása közben.

```
1 <build>
2   [..]
3   <plugins>
4     <plugin>
5       <groupId>org.apache.maven.plugins</groupId>
6       <artifactId>maven-surefire-plugin</artifactId>
7       <version>${version.plugin.surefire}</version>
8       <configuration>
9         <classpathDependencyExcludes>
10          <classpathDependencyExclude>javax:javaee-api</classpathDependencyExclude>
11        </classpathDependencyExcludes>
12        <suiteXmlFiles>
13          <suiteXmlFile>src/test/resources/testng-integration.xml</suiteXmlFile>
14        </suiteXmlFiles>
15      </configuration>
16    </plugin>
17  </plugins>
18 </build>
```

pom.xml

src | test | resources | log4j.xml

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
3 <log4j:configuration debug="true"
4   xmlns:log4j='http://jakarta.apache.org/log4j/'>
5
6   <appender name="console" class="org.apache.log4j.ConsoleAppender">
7     <layout class="org.apache.log4j.PatternLayout">
8       <param name="ConversionPattern"
9         value="%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n" />
10      </layout>
11    </appender>
12
13    <root>
14      <level value="DEBUG" />
15      <appender-ref ref="console" />
16    </root>
17
18 </log4j:configuration>
```

log4j.xml

A részletes log nagyon sokat segít a fejlesztésben és a tesztelésben is. És pontosan ezen részletek miatt lesz nagyon gyors és hatékony pl. a named query-k optimalizálása. Mivel az integrációs tesztek futása során nincs JavaEE container aki biztosítja a naplózást, nekünk kell ezt is konfigurálnunk.

Integrációs teszt

Előkészületek

```
1 package hu.qvaevisz.bookstore.persistence.service;
2 [..]
3 public class BookServiceImplIntegrationTest {
4
5     private static final String PERSISTENCE_UNIT = "bs-persistence-test-unit";
6
7     private BookServiceImpl object;
8
9     @BeforeClass
10    private void setUp() {
11        final EntityManagerFactory factory =
12            Persistence.createEntityManagerFactory(PERSISTENCE_UNIT);
13        final EntityManager entityManager = factory.createEntityManager();
14
15        this.object = new BookServiceImpl();
16        this.object.setEntityManager(entityManager);
17    }
18    [..]
19
20    @AfterClass
21    private void tearDown() {
22        this.object.getEntityManager().close();
23    }
24 }
```

A lokális entityManager "befecskendezésének" megkerülése miatt egy default/protected setEntityManager() mutator metódust készítenünk kell a production kódban:-)

BookServiceImplIntegrationTest.java

Integrációs teszt

A read() mögötti named query működésének tesztelése.

Az integration csoportosításra a unit-test-ektől való megkülönböztetés miatt van szükség.

```
1 public class BookServiceImplIntegrationTest {
2     [...]
3     @Test(groups = "integration")
4     private void readSampleBookFromDatabase() throws PersistenceServiceException {
5         final Book book = this.object.read("978-0441172719");
6         this.assertBook(book, "978-0441172719", "Frank Herbert", "Dune",
7             BookCategory.SCIFI, 896, 3500d);
8     }
9     private void assertBook(final Book book, final String isbn, final String author,
10        final String title, final BookCategory category,
11        final Integer numberOfPages, final double price) {
12        Assert.assertEquals(book.getIsbn(), isbn);
13        Assert.assertEquals(book.getAuthor(), author);
14        Assert.assertEquals(book.getTitle(), title);
15        Assert.assertEquals(book.getCategory(), category);
16        Assert.assertEquals(book.getNumberOfPages(), numberOfPages);
17        Assert.assertEquals(book.getPrice(), price);
18    }
19    [...]
20 }
```

BookServiceImplIntegrationTest.java

Integrációs teszt

Adatmanipuláció tranzakcióban

```
1 public class BookServiceImplIntegrationTest {
2     [..]
3     private static final String NEW_BOOK_ISBN = "42-NEW-BOOK-ISBN";
4
5     @Test(groups = "integration")
6     private void createABook() throws PersistenceServiceException {
7         if (this.object.exists(NEW_BOOK_ISBN)) {
8             this.object.getEntityManager().getTransaction().begin();
9             this.object.delete(NEW_BOOK_ISBN);
10            this.object.getEntityManager().getTransaction().commit();
11        }
12
13        this.object.getEntityManager().getTransaction().begin();
14        this.object.create(NEW_BOOK_ISBN, "Lorem Ipsum", "Sample book", 142, 999,
15            BookCategory.HISTORICAL);
16        this.object.getEntityManager().getTransaction().commit();
17
18        final Book book = this.object.read(NEW_BOOK_ISBN);
19        this.assertBook(book, NEW_BOOK_ISBN, "Lorem Ipsum", "Sample book",
20            BookCategory.HISTORICAL, 142, 999);
21    }
22    [..]
23 }
```

Itt most nincs EJB container, "ami" gondoskodna a tranzakciókról. Ezt nekünk kell megtennünk minden adatmanipulációs esetben! Illetve valamiképpen a *rerunnable* képességről is gondoskodnunk kell.

Integration TestNG suites

src | test | resources | testng-*.xml

```
1 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
2 <suite name="integration-test-suite" verbose="1" >
3   <test name="bookservice">
4     <classes>
5       <class name="hu[..].service.BookServiceImplIntegrationTest" />
6       <class name="hu[..].service.BookSearchImplIntegrationTest" />
7     </classes>
8   </test>
9 </suite>
```

testng-integration.xml

```
1 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
2 <suite name="unit-test-suite" verbose="1">
3   <test name="bookservice">
4     <groups>
5       <run>
6         <exclude name="integration" />
7       </run>
8     </groups>
9     <packages>
10    <package name="hu[..].service.*" />
11    </packages>
12  </test>
13 </suite>
```

Az összes teszt case unit test ami a megadott package-ben megtalálható, kivéve az ami az integration csoportba tartozik.

testng-unit.xml