

## Egyszerű JPA projekt

A következőekben egy JPA entitás beant és műveletek végrehajtására stateless session beant hozunk létre. A JPA példa teszteléséhez egy távoli klienst hozunk létre, az egyszerűség kedvéért ugyanabban a projektben.

### Adatbázis létrehozása

Mindenekelőtt létre kell hozni egy adatbázist és egy táblát. MySQL adatbázis kezelőben új adatbázist a *CREATE DATABASE* utasítással hozhatunk létre, a következőképpen:

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name [create_specification] ...
```

create\_specification:

```
[DEFAULT] CHARACTER SET [=] charset_name
```

```
| [DEFAULT] COLLATE [=] collation_name
```

A *CREATE DATABASE* utasítás létrehozza az adott nevű adatbázist. Az *IF NOT EXIST* opcionális része az utasításnak, megakadályozza hiba keletkezését abban az esetben, ha már létezne az adott nevű adatbázis.

Hozunk létre egy *jpaexampledb* adatbázist a *CREATE DATABASE* utasítással a következőképpen:

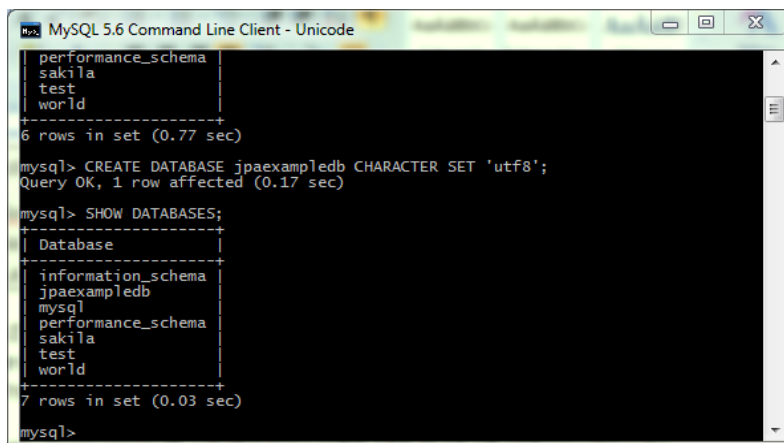
```
CREATE DATABASE jpaexampledb CHARACTER SET 'utf8';
```

A MySQL szerver több karakterkészletet is támogat, ezeket a *SHOW CHARACTER SET* utasítással kérdezhetjük le. A karakterkészletbe tartozó karakter egybevetéseket a *SHOW COLLATION* utasítással lehet lekérdezni, pl:

```
SHOW COLLATION WHERE charset='utf8';
```

A meglévő adatbázisokat a *SHOW DATABASES* utasítással lehet lekérdezni.

```
SHOW DATABASES;
```



```
MySQL 5.6 Command Line Client - Unicode
performance_schema
sakila
test
world
-----
6 rows in set (0.77 sec)

mysql> CREATE DATABASE jpaexampledb CHARACTER SET 'utf8';
Query OK, 1 row affected (0.17 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| jpaexampledb      |
| mysql             |
| performance_schema |
| sakila            |
| test              |
| world             |
+-----+
7 rows in set (0.03 sec)

mysql>
```

Az adatbázis használatához a *USE* utasítás használható:

```
USE jpaexampledb;
```

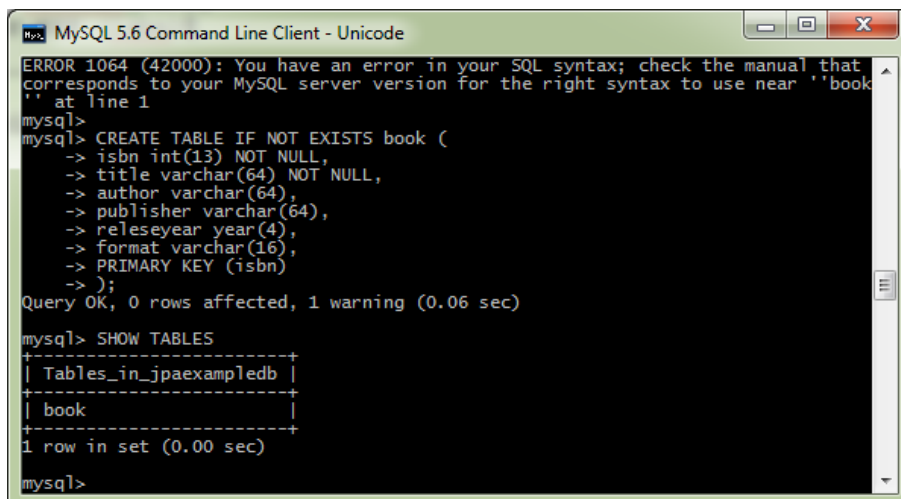
Tábla létrehozására a *CREATE TABLE* utasítást használhatjuk, általános formája:

```
CREATE TABLE [IF NOT EXISTS] table_name(  
column_list  
) engine=table_type
```

Hozunk létre egy *book* táblát, könyvek tárolására:

```
CREATE TABLE IF NOT EXISTS book (  
    isbn int(13) NOT NULL,  
    title varchar(64) NOT NULL,  
    author varchar(64),  
    publisher varchar(64),  
    releaseyear year(4),  
    format varchar(32),  
    PRIMARY KEY (isbn)  
);
```

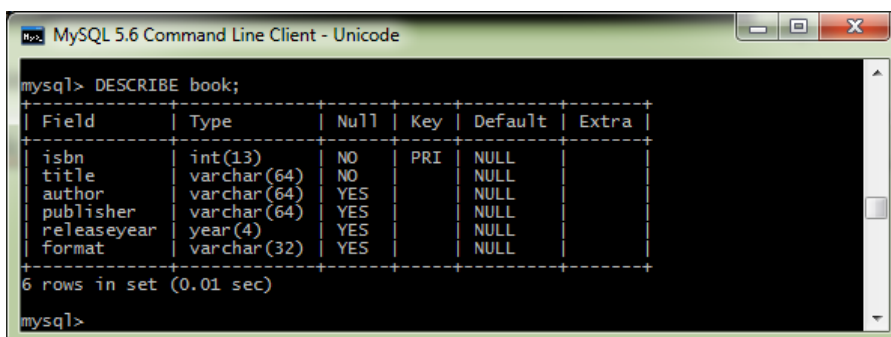
Az elérhető táblákat a *SHOW TABLES* utasítással lehet lekérdezni.



```
MySQL 5.6 Command Line Client - Unicode  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near ''book  
'' at line 1  
mysql>  
mysql> CREATE TABLE IF NOT EXISTS book (  
-> isbn int(13) NOT NULL,  
-> title varchar(64) NOT NULL,  
-> author varchar(64),  
-> publisher varchar(64),  
-> releseyear year(4),  
-> format varchar(16),  
-> PRIMARY KEY (isbn),  
-> );  
Query OK, 0 rows affected, 1 warning (0.06 sec)  
mysql> SHOW TABLES  
+-----+  
| Tables_in_jpaexampledb |  
+-----+  
| book |  
+-----+  
1 row in set (0.00 sec)  
mysql>
```

Egy tábla mezőit a következő utasítással kérdezhetjük le:

```
DESCRIBE book;
```



```
MySQL 5.6 Command Line Client - Unicode  
mysql> DESCRIBE book;  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| isbn  | int(13) | NO   | PRI | NULL    |       |  
| title | varchar(64) | NO   |     | NULL    |       |  
| author | varchar(64) | YES  |     | NULL    |       |  
| publisher | varchar(64) | YES  |     | NULL    |       |  
| releaseyear | year(4) | YES  |     | NULL    |       |  
| format | varchar(32) | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+  
6 rows in set (0.01 sec)  
mysql>
```

## JPA projekt létrehozása

File → New → EJB Project

A megjelenő párbeszédablakban adjuk meg a projekt nevét, ez legyen JPAExample, majd kattintsunk a *Finish* gombra.

**New EJB Project**

Create an EJB Project and add it to a new or existing Enterprise Application.

Project name: JPAExample

Project location

Use default location

Location: C:\Users\Simon\workspaceEE\JPAExample Browse...

Target runtime

JBoss 7.1 Runtime New Runtime...

EJB module version

3.1

Configuration

Default Configuration for JBoss 7.1 Runtime Modify...

A good starting point for working with JBoss 7.1 Runtime runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

Add project to an EAR

EAR project name: EAR New Project...

Working sets

Add project to working sets

Working sets: Select...

? < Back Next > Finish Cancel

## JPA entitás létrehozása

Hozzunk létre egy osztályt *Book* névvel, *org.ait.entities* csomagnévvel.

```
package org.ait.entities;

import java.io.Serializable;
import java.sql.Date;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity(name = "book")
public class Book implements Serializable {

    @Id
    private int isbn;
    private String title;
    private String author;
    private String publisher;
    private Date releaseyear;
    private String format;

    public Book() {

    }

    public int getIsbn() {
        return isbn;
    }

    public void setIsbn(int isbn) {
        this.isbn = isbn;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }
}
```

```
public Date getReleaseyear() {
    return releaseyear;
}

public void setReleaseyear(Date releaseyear) {
    this.releaseyear = releaseyear;
}

public String getFormat() {
    return format;
}

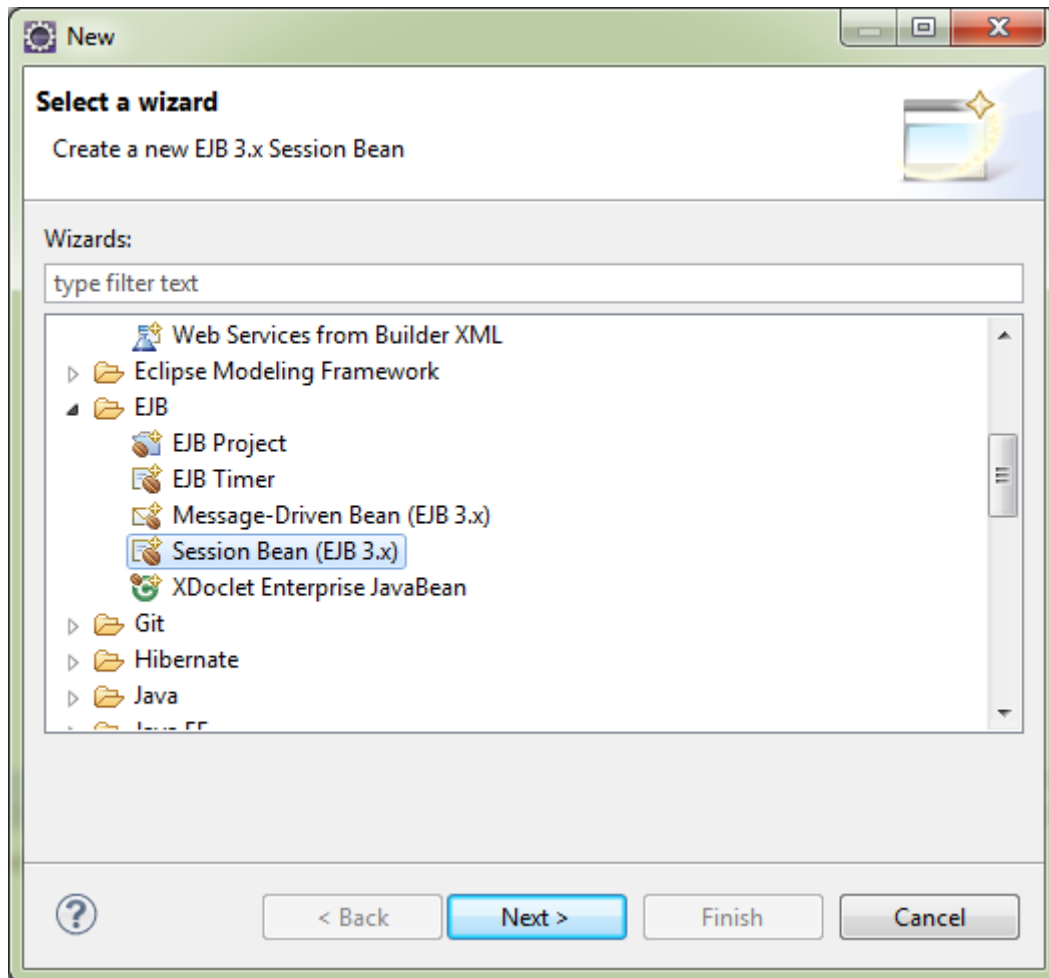
public void setFormat(String format) {
    this.format = format;
}

@Override
public String toString() {
    return "Book [isbn=" + isbn + ", title=" + title + ", author=" +
author + ", publisher=" + publisher + ", releaseyeare=" + releaseyear + ",
format=" + format + " ]";
}
}
```

## Session Bean létrehozása

Új Session Bean létrehozását a következő menüpontban lehet megtenni:

File → New → Other ... itt a következőket kell választani: EJB → Session Bean



A csomag név legyen *org.ait.businesslogic*, az osztály neve pedig *BookBean*.

Az állapotok közül válasszuk a *Stateless*-t.

Az üzleti interfészek részénél jelöljük be a Remote interfész létrehozását.

Majd kattintsunk a *Next* gombra, majd a *Finish*-re.

**Create EJB 3.x Session Bean**  
Specify class file destination.

Project: JPAExample

Source folder: \JPAExample\ejbModule

Java package: org.ait.businesslogic

Class name: BookBean

Superclass:

State type: Stateless

Create business interface

Remote org.ait.businesslogic.BookBeanRemote

Local org.ait.businesslogic.BookBeanLocal

No-interface View

*BookBeanRemote* interfész törzse legyen a következő:

```
package org.ait.businesslogic;

import java.util.List;
import javax.ejb.Remote;
import org.ait.entities.Book;

@Remote
public interface BookBeanRemote {
    public void saveBook(Book book);
    public Book findBook(Book book);
    public List<Book> retrieveAllBooks();
}
```

A *BookBean* törzse legyen a következő:

```
package org.ait.businesslogic;

import java.util.List;

import javax.ejb.LocalBean;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

import org.ait.entities.Book;

@Stateless
public class BookBean implements BookBeanRemote {

    @PersistenceContext(unitName = "jpaexampledb")
    private EntityManager entityManager;

    public BookBean() {
    }

    @Override
    public void saveBook(Book book) {
        entityManager.persist(book);
    }

    @Override
    public Book findBook(Book book) {
        Book b = entityManager.find(Book.class, book.getIsbn());
        return b;
    }

    @Override
    public List<Book> retrieveAllBooks() {
        String q = "SELECT b from" + Book.class.getName() + " p";
        Query query = entityManager.createQuery(q);
        List<Book> books = query.getResultList();
        return books;
    }
}
```



Honnan tudja az *EntityManager* API hogy melyik adatbázist használja? A *persistence.xml* fájlban kell konfigurálni az *EntityManager*-t.

A *persistence.xml* fájlnek a *META-INF* könyvtárban kell lennie. Definiálnia kell egy *persistence-unit*-ot egy egyedi névvel, amit az *EntityManager* használhat.

Kattintsunk jobb gombbal a *META-INF* könyvtárra -> *New* -> *Other* -> *XML* -> *XML file*, fájl névnek adjuk meg a *persistence.xml*-t és írjuk bele a következőt:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="jpaexampledb">
    <jta-data-source>java:/MySQL</jta-data-source>
    <properties>
      <property name="showSql" value="true"/>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQLDialect" />
    </properties>
  </persistence-unit>
</persistence>
```

## MySQL adatforrás konfigurálása JBoss AS 7-ben

MySQL connectort a <http://dev.mysql.com/downloads/connector/j/> címről lehet letölteni.

A JBoss AS 7-ben létre kell hozni egy új modult, melyik tartalmazza a MySQL Connector J jart.

A JBoss AS 7 root könyvtárában létre kell hozni a következő könyvtár hierarchiát:

*modules/com/mysql/main*. Majd a MySQL Connector J jart másoljuk a *main* könyvtárba.

Ezután hozzunk létre egy *module.xml* fájlt a következő tartalommal:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.26-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
```

## Adatforrás hozzáadása a driverhez

Adjunk hozzá egy `<datasources>` elemet a `standalone.xml`-hez, amely a `JBossAS_HOME/standalone/configuration` könyvtárban található.

```
<datasource jndi-name="java:/MySQL" pool-name="MySQL" enabled="true" use-java-
context="true">
  <connection-url>jdbc:mysql://localhost:3306/test</connection-url>
  <driver>mysqlDriver</driver>
  <security>
    <user-name>MYSQL-USERNAME</user-name>
    <password>MYSQL-PASSWORD</password>
  </security>
</datasource>
```

## Teszt kliens létrehozása

Hozzunk létre egy osztályt amellyel tesztelhetjük a létrehozott beant. Ehhez először hozzunk létre egy osztályt, amely inicializálja beant.

Válasszuk a `New` menüben a `Class` menüpontot, majd a megjelenő ablakban adjuk meg az osztály nevét ez legyen `JNDILookupClass`, csomagnevét, amely legyen `org.ait.clientutility`.

```
package org.ait.clientutility;

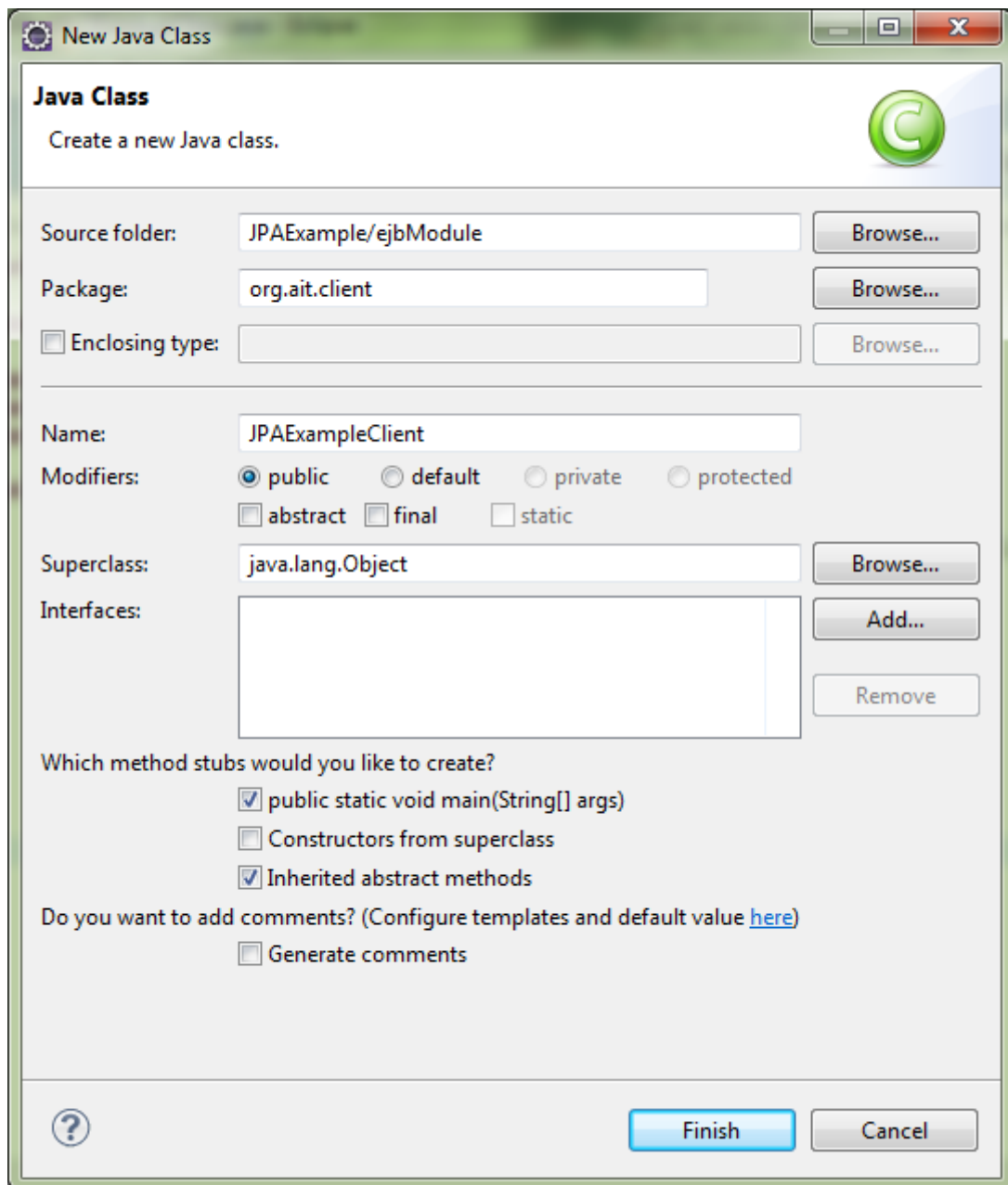
import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class JNDILookupClass {
    private static Context initialContext;
    private static final String PKG_INTERFACES = "org.jboss.ejb.client.naming";

    public static Context getInitialContext() throws NamingException {
        if(initialContext == null) {
            Properties properties = new Properties();
            properties.put(Context.URL_PKG_PREFIXES, PKG_INTERFACES);
            initialContext = new InitialContext(properties);
        }
        return initialContext;
    }
}
```

Ezután hozzuk létre a klienst. Válasszuk a `New` menüben a `Class` menüpontot, majd a megjelenő ablakban adjuk meg az osztály nevét ez legyen `JPAExampleClient`, csomagnevét, amely legyen `org.ait.client` és jelöljük be a main metódus vázának a legenerálását.



Az *JPAExampleClient* tartalma legyen a következő:

```
package org.ait.client;

import java.sql.Date;
import java.util.List;

import javax.naming.Context;
import javax.naming.NamingException;

import org.ait.businesslogic.BookBean;
import org.ait.businesslogic.BookBeanRemote;
import org.ait.clientutility.JNDILookupClass;
import org.ait.entities.Book;
```

```

public class JPAExampleClient {

    public static void main(String[] args) {
        BookBeanRemote bean = doLookup();

        Book b1 = new Book();
        b1.setIsbn(new java.lang.Long("9789631424607"));
        b1.setTitle("A Pendragon legenda");
        b1.setAuthor("Szerb Antal");
        b1.setPublisher("Magvető Könyvkiadó");
        b1.setReleaseyear(new Date(2007, 1, 1));
        b1.setFormat("papír/puha kötés");

        Book b2 = new Book();
        b2.setIsbn(new java.lang.Long("9789631194074"));
        b2.setTitle("A kis herceg");
        b2.setAuthor("Antoine de Saint-Exupéry");
        b2.setPublisher("Móra Kiadó");
        b2.setReleaseyear(new Date(2013, 1, 1));
        b2.setFormat("kemény kötés");

        bean.saveBook(b1);
        bean.saveBook(b2);

        System.out.println("List of books:");
        List<Book> books = bean.retrieveAllBooks();
        for(Book book : books) {
            System.out.println(book);
        }
    }

    private static BookBeanRemote doLookup() {
        Context context = null;
        BookBeanRemote bean = null;
        try {
            context = JNDILookupClass.getInitialContext();

            String lookupName = getLookupName();

            bean = (BookBeanRemote) context.lookup(lookupName);
        } catch (NamingException e) {
            e.printStackTrace();
        }
        return bean;
    }

    private static String getLookupName() {
        String appName = "";

        String moduleName = "JPAExample";

        String distinctName = "";

        String beanName = BookBean.class.getName();

        final String interfaceName = BookBeanRemote.class.getName();
    }
}

```

```
        String name = "ejb:" + appName + "/" + moduleName + "/" +  
distinctName + "/" + beanName + "!" + interfaceName;  
  
        return name;  
    }  
}
```

A host paraméterinek megadásához hozzunk létre egy *jboss-ejb-client.properties* fájlt, a következő tartalommal:

```
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false  
remote.connections=default  
remote.connection.default.host=localhost  
remote.connection.default.port = 4447  
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS  
=false
```

Ha még nincs a classpath-ban a *jboss-client.jar*, akkor adjuk hozzá, majd futtathatjuk a klienst.