2016. április

Informatikai Navigátor

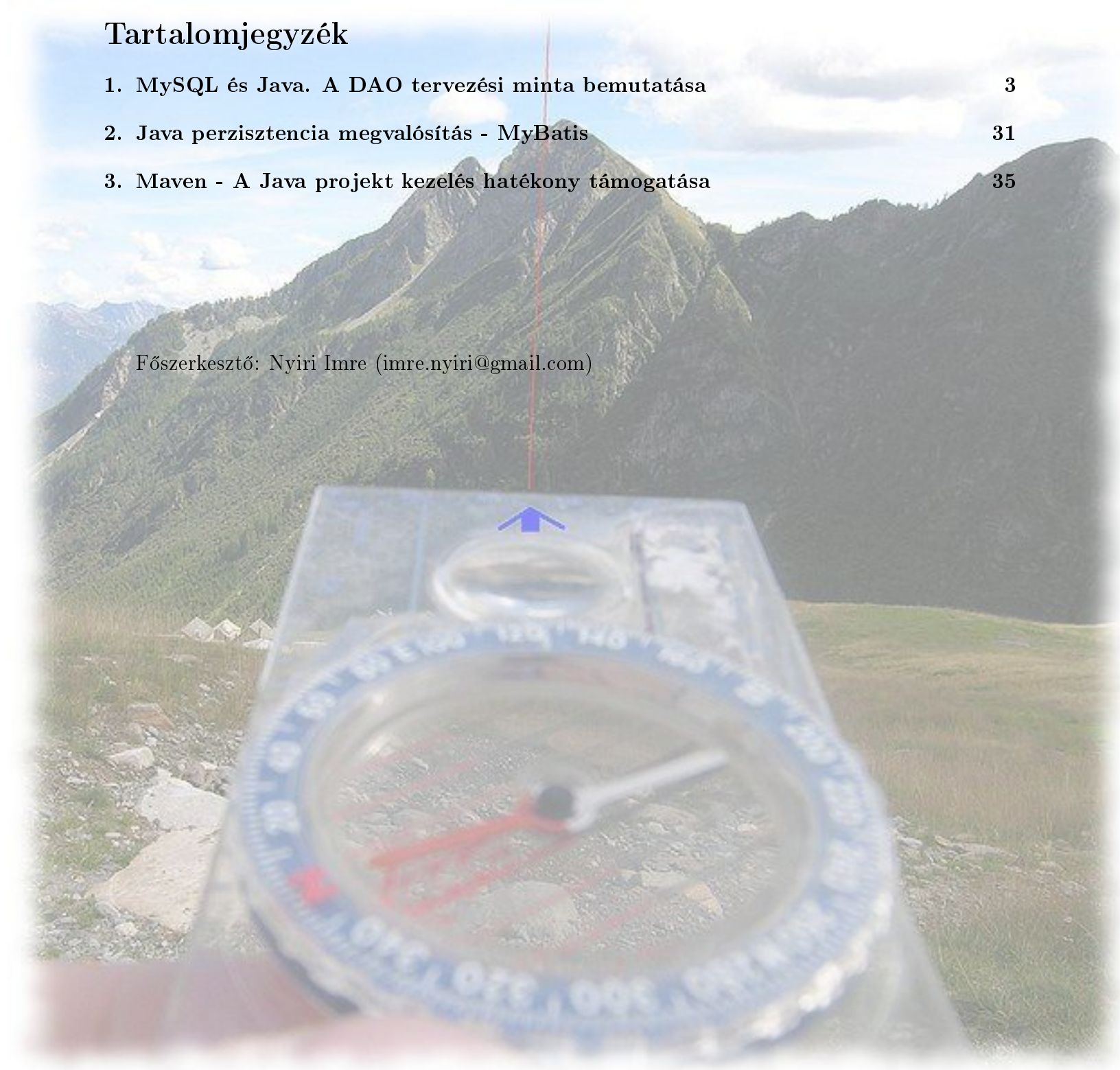
Érdekes Java Programozói könyvtárak - II.

Gondolatok a szoftverek használatáról és fejlesztéséről

12. szám

Tartalomjegyzék

Főszerkesztő: Nyiri Imre (imre.nyiri@gmail.com)

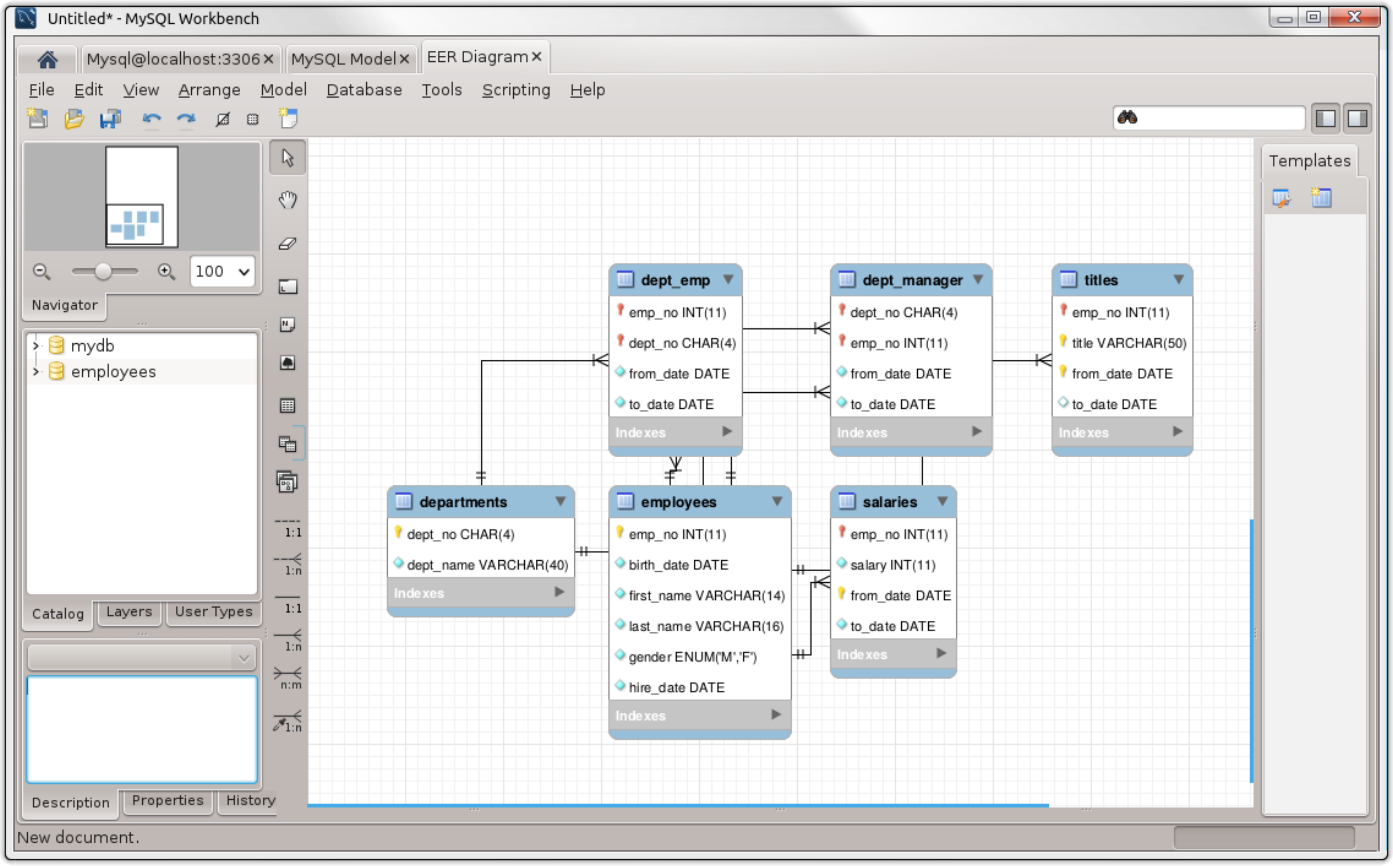


1. MySQL és Java. A DAO tervezési minta bemutatása

Ebben a részben áttekintjük a Java adatbázis-kezelő API-t (JDBC), kitérve a legkorszerűbb elvekre és lehetőségekre. A példákhoz a népszerű MySQL adatbázis-kezelőt használjuk. Külön kiemeljük a bemutatott DAO (Data Access Object) készítési eljárást, ami lényegesen felgyorsíthatja a mindennapi munkánkat.

A példa adatbázis létrehozása

A MySQL oktatóanyag tartalmaz egy példa adatbázist, amit itt tanulmányozhatunk és le is tölthetünk: `https://dev.mysql.com/doc/employee/en/`. Ez egy nagyméretű, milliós nagyságrendű adatbázis, így alkalmas valós példák kipróbálására. Az általunk használt verziót az *employees_db-full-1.0.6.tar.bz2* fájl tartalmazta, amit egy alkönyvtárba kicsomagolhatunk és betölthetjük a futó MySQL szerverünk alá.



1.1. ábra. Az *employees* adatbázis sémája - a MySQL Workbench programmal generálva

A kicsomagolt tartalom telepítő script-jét mutatja az 1-1. Programlista, ezt kell lefuttatnunk a *mysql* nevű kliens programba belépve. Ezután lesz egy *employees* nevű adatbázisunk 6 darab táb-

lával. A külön telepíthető *MySQL Workbench* GUI eszköz (1.1. ábra) segítségével is használhatjuk az adatbázisunkat, az ábrán egy hasznos szolgáltatását látjuk, azaz az adatbázis *ERD* ábrájának előállítását. Például a *departments* és *dept_ emp* között (1, N) kapcsolat van, amit az SQL script 47. sora határoz meg:

```
FOREIGN KEY (dept_no) REFERENCES departments (dept_no) ON DELETE CASCADE
```

A *dept_ emp* egyébként láthatóan egy kapcsoló tábla az *employees* és *departments* táblák között.

1-1. Programlista: employees.sql

```
1
2  CREATE DATABASE IF NOT EXISTS employees CHARACTER SET utf8 COLLATE utf8_general_ci;
3  USE employees;
4
5  SELECT 'CREATING DATABASE STRUCTURE' as 'INFO';
6
7  DROP TABLE IF EXISTS dept_emp,
8                       dept_manager,
9                       titles,
10                      salaries,
11                      employees,
12                      departments;
13
14     set storage_engine = InnoDB;
15 -- set storage_engine = MyISAM;
16 -- set storage_engine = Falcon;
17 -- set storage_engine = PBXT;
18 -- set storage_engine = Maria;
19
20 select CONCAT('storage engine: ', @@storage_engine) as INFO;
21
22 CREATE TABLE employees (
23      emp_no      INT             NOT NULL,
24      birth_date  DATE            NOT NULL,
25      first_name  VARCHAR(14)     NOT NULL,
26      last_name   VARCHAR(16)     NOT NULL,
27      gender      ENUM ('M','F')  NOT NULL,
28      hire_date   DATE            NOT NULL,
29      PRIMARY KEY (emp_no)
30 );
31
32 CREATE TABLE departments (
33      dept_no     CHAR(4)         NOT NULL,
34      dept_name   VARCHAR(40)     NOT NULL,
35      PRIMARY KEY (dept_no),
36      UNIQUE  KEY (dept_name)
37 );
38
39 CREATE TABLE dept_manager (
40     dept_no      CHAR(4)         NOT NULL,
41     emp_no       INT             NOT NULL,
42     from_date    DATE            NOT NULL,
43     to_date      DATE            NOT NULL,
44     KEY          (emp_no),
45     KEY          (dept_no),
46     FOREIGN KEY (emp_no)  REFERENCES employees (emp_no)    ON DELETE CASCADE,
47     FOREIGN KEY (dept_no) REFERENCES departments (dept_no) ON DELETE CASCADE,
48     PRIMARY KEY (emp_no,dept_no)
49 );
50
51 CREATE TABLE dept_emp (
52      emp_no      INT             NOT NULL,
```

```
53      dept_no     CHAR(4)         NOT NULL,
54      from_date   DATE            NOT NULL,
55      to_date     DATE            NOT NULL,
56      KEY         (emp_no),
57      KEY         (dept_no),
58      FOREIGN KEY (emp_no)  REFERENCES employees   (emp_no)  ON DELETE CASCADE,
59      FOREIGN KEY (dept_no) REFERENCES departments (dept_no) ON DELETE CASCADE,
60      PRIMARY KEY (emp_no,dept_no)
61  );
62
63  CREATE TABLE titles (
64      emp_no      INT             NOT NULL,
65      title       VARCHAR(50)     NOT NULL,
66      from_date   DATE            NOT NULL,
67      to_date     DATE,
68      KEY         (emp_no),
69      FOREIGN KEY (emp_no) REFERENCES employees (emp_no) ON DELETE CASCADE,
70      PRIMARY KEY (emp_no,title, from_date)
71  );
72
73  CREATE TABLE salaries (
74      emp_no      INT             NOT NULL,
75      salary      INT             NOT NULL,
76      from_date   DATE            NOT NULL,
77      to_date     DATE            NOT NULL,
78      KEY         (emp_no),
79      FOREIGN KEY (emp_no) REFERENCES employees (emp_no) ON DELETE CASCADE,
80      PRIMARY KEY (emp_no, from_date)
81  );
82
83  SELECT 'LOADING departments' as 'INFO';
84  source load_departments.dump ;
85  SELECT 'LOADING employees' as 'INFO';
86  source load_employees.dump ;
87  SELECT 'LOADING dept_emp' as 'INFO';
88  source load_dept_emp.dump ;
89  SELECT 'LOADING dept_manager' as 'INFO';
90  source load_dept_manager.dump ;
91  SELECT 'LOADING titles' as 'INFO';
92  source load_titles.dump ;
93  SELECT 'LOADING salaries' as 'INFO';
94  source load_salaries.dump ;
```

A MySQL használatának megtanulásához – a hivatalos dokumentáción kívül – ezt a helyet ajánljuk az olvasó figyelmébe: `http://www.tutorialspoint.com/mysql/`

A kliens környezet

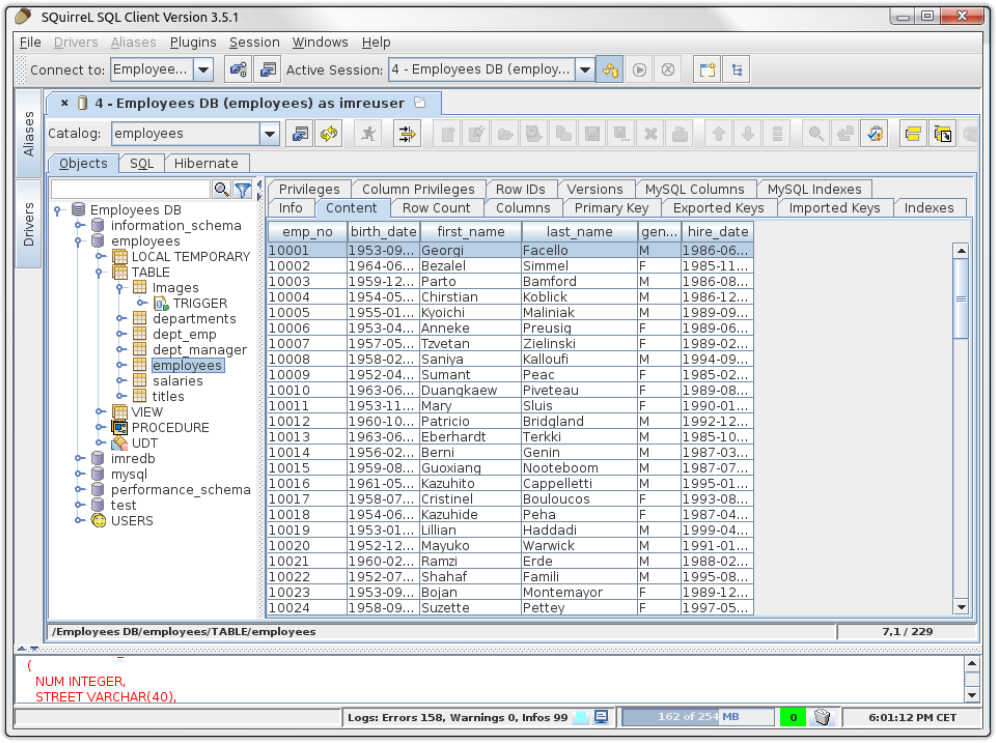
A már említett *MySQL Workbench* program mellett a *mysql* konzol program használatát mindenképpen érdemes megtanulni, ez ugyanazt a szerepet tölti be, mint az Oracle környezetben az *SQL *plus*. Az 1-2. Programlista ízelítőt ad a használatáról.

1-2. Programlista: *mysql* konzol program

```
1  inyiri@europa:~$ mysql -u root -p
2  Enter password:
3  Welcome to the MySQL monitor.  Commands end with ; or \g.
4  Your MySQL connection id is 50
5  Server version: 5.5.37-0ubuntu0.14.04.1 (Ubuntu)
6
7  Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
```

```
8
9  Oracle is a registered trademark of Oracle Corporation and/or its
10 affiliates. Other names may be trademarks of their respective
11 owners.
12
13 Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
14
15 mysql> SHOW DATABASES;
16 +--------------------+
17 | Database           |
18 +--------------------+
19 | information_schema |
20 | mysql              |
21 | performance_schema |
22 | test               |
23 +--------------------+
24
25 mysql> CREATE DATABASE imredb CHARACTER SET utf8 COLLATE utf8_general_ci;
26 Query OK, 1 row affected (0.00 sec)
27
28 mysql> CREATE USER 'imreuser'@'localhost' IDENTIFIED BY '111111';
29 Query OK, 0 rows affected (0.00 sec)
30
31 mysql> use imredb;
32 Database changed
33
34 mysql> GRANT ALL ON imredb.* TO 'imreuser'@'localhost';
35 Query OK, 0 rows affected (0.00 sec)
```

Erősen ajánljuk a *SQuirrel* SQL GUI (webhelye: `http://squirrel-sql.sourceforge.net/`) használatát, a 1.2. ábra éppen az *employees* adatbázisunk *employees* tábláját mutatja.



1.2. ábra. *SQuirrel* SQL GUI

Kapcsolódás a MySQL adatbázishoz

Hagyományos adatbázis kapcsolat felépítés

A *MySQLDatabaseSession* osztály (1-3. Programlista) azért készült, hogy az adatbázishoz kapcsolódás és lekapcsolódás, valamint az erőforrások felszabadítására legyen egy újrafelhasználható komponens. Közben ezen osztály segítségével láthatjuk azt, hogy miképpen lehet egy MySQL adatbázishoz kapcsolódni. A *mysql-connector-java-5.1.28.jar* thin JDBC drivert használtuk, ez letölthető a MySQL webhelyéről. A 16. sor mutatja be a használható JDBC connect sztringet. A MySQL szerver a *localhost* 3306-os portról érhető el és most a már bemutatott *employees* adatbázist szeretnénk használni. A kapcsolat URL paraméterek között megadtuk az unicode biztonságos kezeléséhez szükséges *useUnicode* és *characterEncoding* beállításokat is. Az adatbázishoz kapcsolódást a 22-24 sorok között megvalósított *connect()* metódus mutatja be. A 30-34 sorok közötti *disconnect()* pedig megszünteti az adatbázis kapcsolatot. A 40-64 sorok közötti *getMySQLVersion()* kódja lekérdezi az adatbázis szerver verzióját.

1-3. Programlista: MySQLDatabaseSession.java

```
1
2  package org.cs.mysql.samples;
3
4  import java.sql.Connection;
5  import java.sql.DriverManager;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.ResultSetMetaData;
9  import java.sql.SQLException;
10 import java.sql.Statement;
11
12 import org.junit.Test;
13
14 public class MySQLDatabaseSession {
15
16         static final String JDBC_URL = "jdbc:mysql://localhost:3306/employees?useUnicode=true&➥
               characterEncoding=UTF-8";
17         String defaultUser     = "imreuser";
18         String defaultPassword = "111111";
19
20         Connection conn = null;
21
22         public void connect(String jdbcURL, String user, String password) throws SQLException {
23                 conn = DriverManager.getConnection(JDBC_URL, user, password);
24         }
25
26         public void connect() throws SQLException {
27                 connect(JDBC_URL, defaultUser, defaultPassword);
28         }
29
30         public void disconnect() {
31                 try {
32                         conn.close();
33                 } catch (Exception e) { ; }
34         }
35
36         public PreparedStatement getPreparedStatement(String SQLTemplateCommand) throws ➥
               SQLException {
37                 return conn.prepareStatement( SQLTemplateCommand );
38         }
39
```

```
40          public String getMySQLVersion() throws SQLException {
41                  Statement st = null;
42                  ResultSet rs = null;
43                  String versionInfo = "?";
44                  try {
45                          st = conn.createStatement();
46                          rs = st.executeQuery("SELECT VERSION()");
47
48                          if (rs.next()) {
49                                  versionInfo = rs.getString(1);
50                          }
51                  } finally {
52                          try {
53                                  rs.close();
54                          } catch (Exception e) {
55                                  ;
56                          }
57                          try {
58                                  st.close();
59                          } catch (Exception e) {
60                                  ;
61                          }
62                  }
63                  return versionInfo;
64          }
65
66          public void getTableMetadata(String tableName) throws SQLException {
67                  Statement st = null;
68                  ResultSet rs = null;
69                  ResultSetMetaData meta = null;
70
71                  try {
72                          st = conn.createStatement();
73                          rs = st.executeQuery("SELECT * from " + tableName);
74                          meta = rs.getMetaData();
75                          System.out.println( meta.getColumnCount() );
76                          for (int i=1; i <= meta.getColumnCount(); i++) {
77
78                                  System.out.println( meta.getColumnName( i ) );
79                          }
80
81
82                  } finally {
83                          try {
84                                  rs.close();
85                          } catch (Exception e) {
86                                  ;
87                          }
88                          try {
89                                  st.close();
90                          } catch (Exception e) {
91                                  ;
92                          }
93                  }
94          }
95
96          public void releaseAllUtililty(Statement st, PreparedStatement pst, ResultSet rs) {
97
98                  try { if (rs!=null) rs.close(); } catch (Exception e) { ; }
99                  try { if (st!=null) st.close(); } catch (Exception e) { ; }
100                 try { if (pst!=null) pst.close(); } catch (Exception e) { ; }
101         }
102
103         @Test
104         public void testGetTableMetadata() {
```

```
105                 try {
106                         System.out.println("Start test...");
107                         connect();
108                         getTableMetadata("departments");
109                         disconnect();
110                         System.out.println("Finished test");
111                 } catch (SQLException e) {
112                         e.printStackTrace();
113                 }
114         }
115
116         //@Test
117         public void testConnectDisconnect()
118         {
119                 try {
120                         System.out.println("Start test...");
121                         connect();
122                         String ver = getMySQLVersion();
123                         disconnect();
124                         System.out.println("Finished test... MySQL Version ="+ver);
125                 } catch (SQLException e) {
126                         e.printStackTrace();
127                 }
128
129         }
130 }
```

A 116-129 közötti *testConnectDisconnect()* egy *JUnit* metódus, ami teszteli a kapcsolódás, lekapcsolódás műveletét, illetve azért, hogy valami hasznos is történjen, kiírja a MySQL szerver verzióját:

```
Start test...
Finished test... MySQL Version = 5.5.40-0ubuntu1
```

Adatforrás alapú adatbázis kapcsolat felépítés

Egy *Weblogic 12c* szerveren felvettünk egy új *employees_ds* JNDI nevű datasource-t az adatbázisunkra. A *Connection Pool* engedélyezi a *Supports Global Transactions* kapcsolót, így ez az adatforrás részt tud venni egy elosztott (XA) tranzakcióban. Ugyanakkor kiválasztottuk a *One-Phase Commit* opciót is, ez lehetővé teszi, hogy egy egyszerű Java program is használhassa ezt az adatforrást. Az 1-4. Programlista 21-65 sora bemutatja azokat a lépéseket, ahogy JNDI adatforrás segítségével szerezzük meg az adatbázis kapcsolatot.

1-4. Programlista: ConnectTest.java

```
1
2  package org.cs.mysql.samples;
3
4  import java.sql.Connection;
5  import java.sql.DriverManager;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.SQLException;
9  import java.sql.Statement;
10 import java.util.Hashtable;
11
12 import javax.naming.Context;
13 import javax.naming.InitialContext;
14 import javax.sql.DataSource;
```

```
15 import javax.sql.rowset.JdbcRowSet;
16
17 import org.junit.Test;
18 ...
19 public class ConnectTest {
20
21         @Test
22         public void testDatasourceConnect() {
23         System.out.println("Start test");
24         String JNDI_FACTORY = "weblogic.jndi.WLInitialContextFactory";
25         String hostURL = "t3://localhost:7001";
26
27         Hashtable props = new Hashtable();
28         props.put(Context.INITIAL_CONTEXT_FACTORY, JNDI_FACTORY);
29         props.put(Context.PROVIDER_URL, hostURL);
30         props.put(Context.SECURITY_PRINCIPAL, "weblogic");
31             props.put(Context.SECURITY_CREDENTIALS, "weblogicPassword44");
32
33         Context ctx = null;
34
35         Connection conn = null;
36         Statement st = null;
37         ResultSet rs = null;
38         PreparedStatement pstmt = null;
39
40                 try {
41                         System.out.println("S 1");
42
43                         ctx = new InitialContext(props);
44                         System.out.println("S 2");
45                         DataSource ds=(javax.sql.DataSource) ctx.lookup ("employees_ds");
46
47                         conn = ds.getConnection();
48                         st = conn.createStatement();
49                         rs = st.executeQuery("select * from departments order by dept_no");
50
51                         while (rs.next()) {
52                                 System.out.print(rs.getString(1));
53                                 System.out.println(" - " + rs.getString(2));
54                         }
55
56                         rs.close();
57                         st.close();
58                         conn.close();
59
60                         System.out.println("End test");
61
62                 } catch (Exception e) {
63                         e.printStackTrace();
64                 }
65         }
66 ...
67 }
```

Aki ismeri a *JEE* technológiát és a Weblogic-ot, annak ismerősek a fenti sorok. A 43. sorban megszerezzük a JNDI *Context* objektumot, hogy a segítségével a 45. sorban hozzájuthassunk a *DataSource* objektumhoz. Ezzel a 47. sorban már tudunk egy *Connection* objektumot szerezni az adatbázisra. A példa egyébként a *departments* tábla tartalmát listázza ki:

```
d001 - Marketing
d002 - Finance
d003 - Human Resources
d004 - Production
d005 - Development
d006 - Quality Management
```

```
d007 - Sales
d008 - Research
d009 - Customer Service
```

A MySQL és Java típusok áttekintése

Numerikus adatok

- *BIT(N)*: *N=1* esetén a *java.lang.Boolean* típusra képződik, míg *N>1* teljesülésekor *byte[]*-ra.
- *INT:* Egy integer, de lehet előjeles és nem előjeles (ekkor az *UNSIGNED* szóval ezt jelezni kell). Az első esetben ebben a számtartományban: [-2147483648, 2147483647], míg a másodikban [0, 4294967295]. Lehetséges 11 számjegyig specifikálni a szélességét, ennek alakja: *INT(5)*. Használhatjuk az *INTEGER* kulcsszót is. A leképzett Java típus *java.lang.Integer*, illetve előjel nélküli esetben *java.lang.Long*.
- *TINYINT:* Hasonló az INT-hez, de itt maximum 4 számjegyig adhatjuk meg a szélességét. A tartománya előjeles esetben [-128, 127], míg UNSIGNED esetén: [0, 255]. Java reprezentáció: *java.lang.Boolean* (vagy *java.lang.Integer*).
- *SMALLINT:* Értéktartomány: [-32768, 32767] vagy [0, 65535]. A szélesség maximum 5 számjegyig megadható. Java reprezentáció: *java.lang.Integer*.
- *MEDIUMINT:* Értéktartomány: [-8388608, 8388607] vagy [0, 16777215]. A szélesség maximum 9 számjegyig megadható. Java reprezentáció: *java.lang.Integer*.
- *BIGINT:* Értéktartomány: [-9223372036854775808, 9223372036854775807] vagy [0, 18446744073709551
Java reprezentáció: *java.lang.Long* vagy *java.math.BigInteger*.
- *FLOAT(M,D):* Lebegőpontos szám, ahol *M* az összes számjegy, míg *D* a tizedes jegyek száma. Java reprezentáció: *java.lang.Float*.
- *DOUBLE(M,D):* Lebegőpontos szám, ahol *M* az összes számjegy, míg *D* a tizedes jegyek száma. Java reprezentáció: *java.lang.Double*.
- *DECIMAL(M,D):* Decimális szám, az alias neve: *NUMERIC*. Az *M* az összes számjegy, míg *D* a tizedes jegyek száma. Java reprezentáció: *java.math.BigDecimal*.

Dátum és idő

- *DATE:* Egy (év, hó, nap) érték tárolására képes SQL típus (1000-01-01 és 9999-12-31 között). Java reprezentáció: *java.sql.Date*.
- *DATETIME:* Az (év, hó, nap, óra, perc, másodperc) tárolását valósítja meg, java reprezentációja a *java.sql.Timestamp*.
- *TIMESTAMP:* Az (év, hó, nap, óra, perc, másodperc) tárolását valósítja meg, java reprezentációja a *java.sql.Timestamp*.

- *TIME:* Egy (óra, perc, másodperc) érték tárolására alkalmas, java reprezentációja a *java.sql.Time*.
- *YEAR(M):* Egyévet tárol 2 vagy 4 számjegyen. Java reprezentáció: *java.sql.Short*.

Karakteres adatok

- *CHAR(M)*: Ez egy fix hosszúságú sztring 1 és 255 között, ha nem adjuk meg *M* értéket akkor az *M=1*. Amennyiben egy eltárolandó érték rövidebb *M*-nél, úgy az jobbról szóközökkel töltődik fel. A java reprezentáció *java.lang.String*, de *BINARY* beállítás esetén *byte[]* .
- *VARCHAR(M)*: Változó hosszúságú karaktersorozat 1 és 255 között. Az érték nem töltődik fel jobbról szóközökkel. A java reprezentáció *java.lang.String*, de *BINARY* beállítás esetén *byte[]* .

Nagy objektumok

- *TEXT*: Változó hosszúságú karaktersorozat 65535 maximális szélességgel. A java reprezentáció *java.lang.String*.
- *BLOB:* Változó hosszúságú bytesorozat 65535 maximális szélességgel. A java reprezentáció *byte[]*.
- *TINYTEXT*: Változó hosszúságú karaktersorozat 255 maximális szélességgel. A java reprezentáció *java.lang.String*.
- *TINYBLOB:* Változó hosszúságú bytesorozat 255 maximális szélességgel. A java reprezentáció *byte[]*.
- *MEDIUMTEXT*: Változó hosszúságú karaktersorozat 16777215 maximális szélességgel. A java reprezentáció *java.lang.String*.
- *MEDIUMBLOB:* Változó hosszúságú bytesorozat 16777215 maximális szélességgel. A java reprezentáció *byte[]*.
- *LONGTEXT*: Változó hosszúságú karaktersorozat 4294967295 maximális szélességgel. A java reprezentáció *java.lang.String*.
- *LONGBLOB:* Változó hosszúságú bytesorozat 4294967295 maximális szélességgel. A java reprezentáció *byte[]*.

ENUM típus

Amikor „A”, „B”, „C”, ... értékek egy felsorolását akarjuk típusként használni, akkor *ENUM ('A', 'B', 'C')* írható az oszlop típusaként, aminek java reprezentációja *java.lang.String*.

Lekérdezés az adatbázisból

Egyszerű lekérdezés

Az adatbázisból való lekérdezés a JDBC API alapján ismert módon történik, ahogy azt az 1-5. Programlista is bemutatja.

1-5. Programlista: Adatbázis select

```
1
2  package org.cs.mysql.samples;
3
4  import java.sql.Connection;
5  import java.sql.DriverManager;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.SQLException;
9  import java.sql.Statement;
10 import java.util.Hashtable;
11 ...
12     @Test
13         public void testSelect() {
14                 System.out.println("Start test");
15
16                 Connection conn = null;
17                 Statement st = null;
18                 ResultSet rs = null;
19                 PreparedStatement pstmt = null;
20
21                 String connStr = "jdbc:mysql://localhost:3306/employees?useUnicode=true&➥
                       characterEncoding=UTF-8";
22                 String user = "imreuser";
23                 String password = "111111";
24
25                 try {
26                         conn = DriverManager.getConnection(connStr, user, password);
27
28                         st = conn.createStatement();
29                         rs = st.executeQuery("select * from departments order by dept_no");
30
31                         while (rs.next()) {
32                                 System.out.print(rs.getString(1));
33                                 System.out.println(" - " + rs.getString(2));
34                         }
35
36                         rs.close();
37                         st.close();
38                         conn.close();
39
40                 } catch (SQLException e) {
41
42                         e.printStackTrace();
43                 }
44         }
45 ...
46 }
```

Többszörös lekérdezés

Amennyiben a JDBC URL-nél megadjuk az *allowMultiQueries=true* paramétert, úgy lehetővé válik a következő többszörös SQL lekérdezés használata:

```
String query = "SELECT Id, Name FROM Authors WHERE Id=1;"
               + "SELECT Id, Name FROM Authors WHERE Id=2;"
               + "SELECT Id, Name FROM Authors WHERE Id=3";

pst = con.prepareStatement(query);
boolean isResult = pst.execute();
rs = pst.getResultSet();
```

Az adatbázis metaadatok lekérdezése

Nézzük meg ismét az 1-3. Programlistát, annak is 66-94 sorok között lévő *getTableMetadata()* metódusát, ami példát ad arra, hogy mi módon tudjuk az adatbázisunk sémájára vonatkozó információkat visszanyerni. A metódus a paraméterként kapott tábla oszlopainak neveit listázza ki. A 69. sorban deklarált *meta* objektum teszi lehetővé, hogy visszaszerezzünk minden minket érdeklő tábla leíró információt. Esetünkben a *ResultSet* objektum *getMetaData()* metódusával szereztük meg (74. sor) ezt az objektumot.

Az adatbázis változtatása

A DAO és ORM részről szóló pontban részletesen bemutatjuk az *INSERT*, *UPDATE* és *DELETE* műveleteket. Most csak 1 példát mutatunk például a törlésre:

```
...
PreparedStatement pstmt = null;
...
pstmt = conn.prepareStatement("delete from departments where dept_no=?");
pstmt.setString(1, "d0009");
pstmt.execute();
...
```

A *PreparedStatement* objektum kérdőjeleinek pozíciói 1-től kezdődnek, erre mindig figyeljünk oda.

Tetszőleges média tartalom tárolása és lekérdezése

Ebben a részben bemutatunk egy érdekes lehetőséget, történetesen tetszőleges objektumok adatbázisban való tárolását. Ehhez most létrehoztuk az *Images* táblát a következő SQL paranccsal:

```
CREATE TABLE IF NOT EXISTS Images(Id INT PRIMARY KEY AUTO_INCREMENT, mimetype varchar(60), Data MEDIUMBLOB);
```

Az *Id* az objektum egyedi azonosítója, a *mimetype* pedig annak MIME típusa lesz, hiszen aki visszanyeri az objektumunkat, annak ezt ismernie kell. A byte-ok a *Data* oszlopban fognak tárolódni.

1-6. Programlista: MySQLMediumStore.java

```
1
2  package org.cs.mysql.samples;
3
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.io.FileOutputStream;
7  import java.io.InputStream;
8  import java.sql.Blob;
9  import java.sql.PreparedStatement;
```

```
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12
13 import org.junit.Test;
14
15 public class MySQLMediumStore {
16
17         String SQL_INSERT = null;
18         String SQL_READ = null;
19         MySQLDatabaseSession session = null;
20
21         public MySQLMediumStore(String mediaTable) {
22                 SQL_INSERT = "INSERT INTO " + mediaTable + "(mimetype, Data) VALUES(?, ?)";
23                 SQL_READ = "SELECT Data FROM " + mediaTable + " where id=?";
24                 session = new MySQLDatabaseSession();
25         }
26
27         public void saveMediaContent(InputStream is, String mimetype, int size) throws ➥
               SQLException {
28
29                 PreparedStatement pst = null;
30                 try {
31                         session.connect();
32                         pst = session.getPreparedStatement(SQL_INSERT);
33                         pst.setString(1, mimetype);
34                         pst.setBinaryStream(2, is, size);
35                         pst.executeUpdate();
36                 } finally {
37                         session.releaseAllUtililty(null, pst, null);
38                         session.disconnect();
39                 }
40         }
41
42         public byte[] loadMediaContent(int key) throws Exception {
43                 PreparedStatement pst = null;
44                 ResultSet result = null;
45                 byte[] content = null;
46
47                 try {
48                         session.connect();
49                         pst = session.getPreparedStatement(SQL_READ);
50                         pst.setInt(1, key);
51                     result = pst.executeQuery();
52                 result.next();
53                 Blob blob = result.getBlob("Data");
54             int len = (int)blob.length();
55             content = blob.getBytes(1, len);
56                 } finally {
57                         session.releaseAllUtililty(null, pst, null);
58                         session.disconnect();
59                 }
60                 return content;
61         }
62
63         public void testSave() {
64                 File img = new File("/home/inyiri/Letöltések/20141025_114440.jpg");
65                 int size = (int)img.length();
66                 FileInputStream fin = null;
67                 try {
68                         fin = new FileInputStream(img);
69                         saveMediaContent(fin, "image/jpeg", size);
70                 }
71                 catch (Exception e ) {
72                         e.printStackTrace();
73                 }
```

```
74                  finally {
75                          try {
76                                  fin.close();
77                          }
78                          catch (Exception e) { ; }
79                  }
80          }
81
82          public void testLoad() {
83                  File img = new File("/home/inyiri/Letöltések/20141025_114440.jpg");
84                  FileOutputStream fos = null;
85                  byte[] content = null;
86                  try {
87                          fos = new FileOutputStream("/home/inyiri/Letöltések/20141025_114440_COPY➥
                                .jpg");
88                          content = loadMediaContent(1);
89                          fos.write(content, 0, content.length);
90                  }
91                  catch (Exception e ) {
92                          e.printStackTrace();
93                  }
94                  finally {
95                          try {
96                                  fos.close();
97                          }
98                          catch (Exception e) { ; }
99                  }
100         }
101
102         public static void main(String[] args) {
103                 MySQLMediumStore store = new MySQLMediumStore("Images");
104                 //store.testSave();
105                 store.testLoad();
106         }
107 }
```

A *MySQLMediumStore* class (1-6. Programlista) konstruktora a 21-25 sorok között van és 4 dolgot végez el. Rögzíti az *INSERT* és *SELECT* SQL parancs mintáját, illetve megadja, hogy a használt tábla az *Images*, ugyanis az osztály be tudja állítani, hogy erre a célra melyik táblát akarjuk használni. Az implementáció során csak a 3 mező (*id*, *mimetype*, *data*) megléte a fontos. A 4. dolog pedig a *session = new MySQLDatabaseSession()* sor, azaz használjuk az 1-3. Programlistán megadott osztályt.

A 27-40 sorok között létrehozott *saveMediaContent()* metódus egy médium elmentését valósítja meg. Azt érdemes megjegyezni, hogy a *PreparedStatement* objektum *setBinaryStream()* metódusát használjuk, azaz egy *InputStream* a mentésre alkalmas formátum. Egyébként a *session* objektum szolgáltatásait használjuk. A *releaseAllUtililty()* azért hasznos, mert megkímél a sok *try-catch* ág írásától, amikor lezárjuk például a *PreparedStatement* objektumot.

A *loadMediaContent()* metódus (42-61 sorok) a *key* kulcs alapján egy *byte* tömbbe visszaszolgáltatja az objektumot. Figyeljük meg, hogy az 53-55 sorok között mindez technikailag hogyan valósítható meg!

Ezzel az osztály elkészült, de a teszteléshez még készült 2 metódus: *testSave()* és *testLoad()*. Az első egy JPG képet ment az adatbázisba, a második pedig ugyanezt az adatbázisból kiírja egy másik fájlba.

A DAO és ORM tervezési minta használata

A *DAO*[1] egy adatelérési tervezési minta, a célja az, hogy egy adatforrásban lévő adatok eléréséhez kényelmes felületet adjon. Ez azért fontos, mert a programozók osztályokban gondolkodnak és azok működési felületei a legfontosabbak számukra. Az adatelérés során az elemi adatot (vagy rekordot) hordozó struktúrát egy *JDO*[2] osztály tárolja, ez fogalmilag a *Java Bean*-nek (vagy *POJO*[3]) felel meg. Végül megemlítjük, hogy egy *RDBMS* esetén fontos a táblák leképzése a *JDO* objektumokra, amit *ORM*[4] néven szoktunk emlegetni.

A továbbiakban ismertetünk egy jó minőségű *DAO* generátor-t, amit a már említett *SQuirrel* jobb egérgomb, *Generate DAO* funkciójával vehetünk használatba. Az eredmény Java forrásfájlok lesznek, ha egyszerre több adatbázis táblát jelöltünk ki, akkor azok mindegyikére megoldva a *DAO* kódgenerálást. A generáláshoz még a következő input információkat kell megadnunk a megjelenő dialógus ablakban:

- *Destination Directory*: Ide teszi a *DAO4J* plugin a generált forráskódokat
- *Package*: Ebbe a Java csomagba helyeződnek a generált Java osztályok
- *DB Type Definition*: Minden adatbázishoz van egy előre beállított konfiguráció, mi most a *mysql.properties* fájlt használtuk. Ebben a fájlban az adott SQL szerver típusai vannak hozzárendelve Java típusokhoz.

A továbbiakban a példa adatbázisunk *employees* táblájára generált *DAO* kódot és annak használatát mutatjuk be.

Az ORM Bean (JDO)

Az *Employees* osztály (1-7. Programlista) az *employees* tábla 1 sorát reprezentálja, mint JDO objektum, de mivel látjuk azt is, hogy a tábla egy mezője miképpen képződik le egy adattagra és annak Java típusára, ezért ezt ORM Bean-nek is nevezzük. Látható, hogy a *primary key* külön részt képvisel az implementációban, illetve az *equals()*, *clone()* és *toString()* is esetfüggő módon felülírásra került.

1-7. Programlista: Employees.java

```
1
2  /*
3   * This java source file is generated by DAO4J v1.19
4   * Generated on Sat Nov 08 09:44:08 CET 2014
5   * For more information, please contact b-i-d@163.com
6   * Please check http://sourceforge.net/projects/dao4j/ for the latest version.
7   */
8
9  package org.cs.dao;
10
11 /*
```

[1] DAO=Data Access Object
[2] JDO=Java Data Objects
[3] POJO=Plain Old Java Object
[4] ORM=Object/Relational Mapping

```
12  * For Table employees
13  */
14 public class Employees implements java.io.Serializable, Cloneable {
15     private EmployeesKey _key = new EmployeesKey();
16 
17     /* emp_no, PK */
18     protected int empNo;
19 
20     /* birth_date */
21     protected java.util.Date birthDate;
22 
23     /* first_name */
24     protected String firstName;
25 
26     /* last_name */
27     protected String lastName;
28 
29     /* gender */
30     protected Object gender;
31 
32     /* hire_date */
33     protected java.util.Date hireDate;
34 
35     /* Return the key object. */
36     public EmployeesKey getKeyObject() {
37         return _key;
38     }
39 
40     /* emp_no, PK */
41     public int getEmpNo() {
42         return empNo;
43     }
44 
45     /* emp_no, PK */
46     public void setEmpNo(int empNo) {
47         this.empNo = empNo;
48         _key.setEmpNo(empNo);
49     }
50 
51     /* birth_date */
52     public java.util.Date getBirthDate() {
53         return birthDate;
54     }
55 
56     /* birth_date */
57     public void setBirthDate(java.util.Date birthDate) {
58         this.birthDate = birthDate;
59     }
60 
61     /* first_name */
62     public String getFirstName() {
63         return firstName;
64     }
65 
66     /* first_name */
67     public void setFirstName(String firstName) {
68         this.firstName = firstName;
69     }
70 
71     /* last_name */
72     public String getLastName() {
73         return lastName;
74     }
75 
76     /* last_name */
```

```
77      public void setLastName(String lastName) {
78          this.lastName = lastName;
79      }
80
81      /* gender */
82      public Object getGender() {
83          return gender;
84      }
85
86      /* gender */
87      public void setGender(Object gender) {
88          this.gender = gender;
89      }
90
91      /* hire_date */
92      public java.util.Date getHireDate() {
93          return hireDate;
94      }
95
96      /* hire_date */
97      public void setHireDate(java.util.Date hireDate) {
98          this.hireDate = hireDate;
99      }
100
101     /* Indicates whether some other object is "equal to" this one. */
102     public boolean equals(Object obj) {
103         if (this == obj)
104             return true;
105
106         if (obj == null || !(obj instanceof Employees))
107             return false;
108
109         Employees bean = (Employees) obj;
110
111         if (this.empNo != bean.empNo)
112             return false;
113
114         if (this.birthDate == null) {
115             if (bean.birthDate != null)
116                 return false;
117         }
118         else if (!this.birthDate.equals(bean.birthDate))
119             return false;
120
121         if (this.firstName == null) {
122             if (bean.firstName != null)
123                 return false;
124         }
125         else if (!this.firstName.equals(bean.firstName))
126             return false;
127
128         if (this.lastName == null) {
129             if (bean.lastName != null)
130                 return false;
131         }
132         else if (!this.lastName.equals(bean.lastName))
133             return false;
134
135         if (this.gender == null) {
136             if (bean.gender != null)
137                 return false;
138         }
139         else if (!this.gender.equals(bean.gender))
140             return false;
141
```

```
142         if (this.hireDate == null) {
143             if (bean.hireDate != null)
144                 return false;
145         }
146         else if (!this.hireDate.equals(bean.hireDate))
147             return false;
148 
149         return true;
150     }
151 
152     /* Creates and returns a copy of this object. */
153     public Object clone()
154     {
155         Employees bean = new Employees();
156         bean.empNo = this.empNo;
157         if (this.birthDate != null)
158             bean.birthDate = (java.util.Date) this.birthDate.clone();
159         bean.firstName = this.firstName;
160         bean.lastName = this.lastName;
161         bean.gender = this.gender;   // The field (Object) is not cloned actually.
162         if (this.hireDate != null)
163             bean.hireDate = (java.util.Date) this.hireDate.clone();
164         return bean;
165     }
166 
167     /* Returns a string representation of the object. */
168     public String toString() {
169         String sep = "\r\n";
170         StringBuffer sb = new StringBuffer();
171         sb.append(this.getClass().getName()).append(sep);
172         sb.append("[").append("empNo").append("=").append(empNo).append("]").append(sep);
173         sb.append("[").append("birthDate").append("=").append(birthDate).append("]").append(➥
                sep);
174         sb.append("[").append("firstName").append("=").append(firstName).append("]").append(➥
                sep);
175         sb.append("[").append("lastName").append("=").append(lastName).append("]").append(sep)➥
                ;
176         sb.append("[").append("gender").append("=").append(gender).append("]").append(sep);
177         sb.append("[").append("hireDate").append("=").append(hireDate).append("]").append(sep)➥
                ;
178         return sb.toString();
179     }
180 }
```

Az *EmployeesKey* class (1-8. Programlista) az elsődleges kulcsot reprezentáló osztály, már láttuk, hogyan használtuk. Most csak 1 mezőből áll, de általánosságban ez több is lehet.

1-8. Programlista: EmployeesKey.java

```
1 
2 /*
3  * This java source file is generated by DAO4J v1.19
4  * Generated on Sat Nov 08 09:44:08 CET 2014
5  * For more information, please contact b-i-d@163.com
6  * Please check http://sourceforge.net/projects/dao4j/ for the latest version.
7  */
8 
9 package org.cs.dao;
10 
11 public class EmployeesKey implements java.io.Serializable, Cloneable {
12     /* emp_no */
13     protected int empNo;
14 
15     /* emp_no */
```

```
16      public int getEmpNo() {
17          return empNo;
18      }
19
20      /* emp_no */
21      public void setEmpNo(int empNo) {
22          this.empNo = empNo;
23      }
24
25      /* Calculate hash code */
26      public int hashCode() {
27          int hashCode = 0;
28          hashCode += new Integer(empNo).hashCode();
29          return hashCode;
30      }
31
32      /* Indicates whether some other object is "equal to" this one. */
33      public boolean equals(Object obj) {
34          if (this == obj)
35              return true;
36
37          if (obj == null || !(obj instanceof EmployeesKey))
38              return false;
39
40          EmployeesKey key = (EmployeesKey) obj;
41
42          if (this.empNo != key.empNo)
43              return false;
44
45          return true;
46      }
47
48      /* Creates and returns a copy of this object. */
49      public Object clone()
50      {
51          EmployeesKey key = new EmployeesKey();
52          key.empNo = this.empNo;
53          return key;
54      }
55
56      /* Returns a string representation of the object. */
57      public String toString() {
58          String sep = "\r\n";
59          StringBuffer sb = new StringBuffer();
60          sb.append(this.getClass().getName()).append(sep);
61          sb.append("[").append("empNo").append(" = ").append(empNo).append("]");
62          return sb.toString();
63      }
64  }
```

A DAO interface

Az *EmployeesDAO* interface (1-9. Programlista) az a felület, ahogy a táblát rekordonként látjuk és kezeljük. Az egyes metódusok szerepe a következő:

- *create()* metódus: Egy *Employees* objektum mentése az adatbázisba.
- *load()* metódus: Egy *Employees* objektum betöltése az adatbázisból. Itt az objektum azonosítását egy *EmployeesKey* objektummal tesszük meg.
- *update()* metódus: Egy *Employees* objektum módosítása az adatbázisban.

- *delete()* metódus: Egy *Employees* objektum törlése az adatbázisból. Itt az objektum azonosítását egy *EmployeesKey* objektummal tesszük meg.

1-9. Programlista: EmployeesDAO.java

```
1
2  /*
3   * This java source file is generated by DAO4J v1.19
4   * Generated on Sat Nov 08 09:44:08 CET 2014
5   * For more information, please contact b-i-d@163.com
6   * Please check http://sourceforge.net/projects/dao4j/ for the latest version.
7   */
8
9  package org.cs.dao.dao;
10
11 import java.sql.Connection;
12 import java.sql.SQLException;
13 import org.cs.dao.*;
14
15 /**
16  * This interface provides methods to populate DB Table of employees
17  */
18 public interface EmployeesDAO {
19     /**
20      * Create a new record in Database.
21      * @param bean    The Object to be inserted.
22      * @param conn    JDBC Connection.
23      * @exception     SQLException if something is wrong.
24      */
25     public void create(Employees bean, Connection conn) throws SQLException;
26
27     /**
28      * Retrive a record from Database.
29      * @param beanKey    The PK Object to be retrived.
30      * @param conn       JDBC Connection.
31      * @exception        SQLException if something is wrong.
32      */
33     public Employees load(EmployeesKey key, Connection conn) throws SQLException;
34
35     /**
36      * Update a record in Database.
37      * @param bean    The Object to be saved.
38      * @param conn    JDBC Connection.
39      * @exception     SQLException if something is wrong.
40      */
41     public void update(Employees bean, Connection conn) throws SQLException;
42
43     /**
44      * Create a new record in Database.
45      * @param bean    The PK Object to be deleted.
46      * @param conn    JDBC Connection.
47      * @exception     SQLException if something is wrong.
48      */
49     public void delete(EmployeesKey key, Connection conn) throws SQLException;
50 }
```

A DAO implementációja

Az Az *EmployeesDAO* interface-t az ehhez generált *EmployeesDAOImpl* class (1-10. Programlista) implementálja. Található ebben egy *getResults()* metódus is, amit publikussá téve a *findBy...()*

jellegű metódusok helyett tudunk használni, azoknál feltehetően sokkal rugalmasabban. Az implementáció módja triviális, így ennek áttekintését nem részletezzük.

1-10. Programlista: EmployeesDAOImpl.java

```
1
2  /*
3   * This java source file is generated by DAO4J v1.19
4   * Generated on Sat Nov 08 09:44:08 CET 2014
5   * For more information, please contact b-i-d@163.com
6   * Please check http://sourceforge.net/projects/dao4j/ for the latest version.
7   */
8
9  package org.cs.dao.orm;
10
11 import java.sql.Connection;
12 import java.sql.PreparedStatement;
13 import java.sql.ResultSet;
14 import java.sql.Statement;
15 import java.sql.SQLException;
16 import java.sql.Types;
17 import java.util.List;
18 import java.util.ArrayList;
19 import org.cs.dao.*;
20 import org.cs.dao.dao.EmployeesDAO;
21
22 /**
23  * This class provides methods to populate DB Table of employees
24  */
25 public class EmployeesDAOImpl implements EmployeesDAO {
26     /* SQL to insert data */
27     private static final String SQL_INSERT =
28         "INSERT INTO employees ("
29         + "emp_no, birth_date, first_name, last_name, gender, hire_date"
30         + ") VALUES (?, ?, ?, ?, ?, ?)";
31
32     /* SQL to select data */
33     private static final String SQL_SELECT =
34         "SELECT "
35         + "emp_no, birth_date, first_name, last_name, gender, hire_date "
36         + "FROM employees WHERE "
37         + "emp_no = ?";
38
39     /* SQL to update data */
40     private static final String SQL_UPDATE =
41         "UPDATE employees SET "
42         + "birth_date = ?, first_name = ?, last_name = ?, gender = ?, hire_date = ? "
43         + "WHERE "
44         + "emp_no = ?";
45
46     /* SQL to delete data */
47     private static final String SQL_DELETE =
48         "DELETE FROM employees WHERE "
49         + "emp_no = ?";
50
51     /**
52      * Create a new record in Database.
53      * @param bean    The Object to be inserted.
54      * @param conn    JDBC Connection.
55      * @exception     SQLException if something is wrong.
56      */
57     public void create(Employees bean, Connection conn) throws SQLException {
58         PreparedStatement ps = null;
59         try {
```

```
60              ps = conn.prepareStatement(SQL_INSERT);
61              ps.setInt(1, bean.getEmpNo());
62              if (bean.getBirthDate() != null)
63                  ps.setDate(2, new java.sql.Date(bean.getBirthDate().getTime()));
64              else
65                  ps.setNull(2, Types.DATE);
66              ps.setString(3, bean.getFirstName());
67              ps.setString(4, bean.getLastName());
68              ps.setObject(5, bean.getGender());
69              if (bean.getHireDate() != null)
70                  ps.setDate(6, new java.sql.Date(bean.getHireDate().getTime()));
71              else
72                  ps.setNull(6, Types.DATE);
73              ps.executeUpdate();
74          }finally {
75              close(ps);
76          }
77      }
78
79      /**
80       * Retrive a record from Database.
81       * @param beanKey    The PK Object to be retrived.
82       * @param conn       JDBC Connection.
83       * @exception        SQLException if something is wrong.
84       */
85      public Employees load(EmployeesKey key, Connection conn) throws SQLException {
86          PreparedStatement ps = null;
87          ResultSet rs = null;
88          try {
89              ps = conn.prepareStatement(SQL_SELECT);
90              ps.setInt(1, key.getEmpNo());
91              rs = ps.executeQuery();
92              List results = getResults(rs);
93              if (results.size() > 0)
94                  return (Employees) results.get(0);
95              else
96                  return null;
97          }finally {
98              close(rs);
99              close(ps);
100         }
101     }
102
103     /**
104      * Update a record in Database.
105      * @param bean    The Object to be saved.
106      * @param conn    JDBC Connection.
107      * @exception     SQLException if something is wrong.
108      */
109     public void update(Employees bean, Connection conn) throws SQLException {
110         PreparedStatement ps = null;
111         try {
112             ps = conn.prepareStatement(SQL_UPDATE);
113             if (bean.getBirthDate() != null)
114                 ps.setDate(1, new java.sql.Date(bean.getBirthDate().getTime()));
115             else
116                 ps.setNull(1, Types.DATE);
117             ps.setString(2, bean.getFirstName());
118             ps.setString(3, bean.getLastName());
119             ps.setObject(4, bean.getGender());
120             if (bean.getHireDate() != null)
121                 ps.setDate(5, new java.sql.Date(bean.getHireDate().getTime()));
122             else
123                 ps.setNull(5, Types.DATE);
124             ps.setInt(6, bean.getEmpNo());
```

```
125             ps.executeUpdate();
126         }finally {
127             close(ps);
128         }
129     }
130 
131     /**
132      * Create a new record in Database.
133      * @param bean   The PK Object to be deleted.
134      * @param conn   JDBC Connection.
135      * @exception    SQLException if something is wrong.
136      */
137     public void delete(EmployeesKey key, Connection conn) throws SQLException {
138         PreparedStatement ps = null;
139         try {
140             ps = conn.prepareStatement(SQL_DELETE);
141             ps.setInt(1, key.getEmpNo());
142             ps.executeUpdate();
143         }finally {
144             close(ps);
145         }
146     }
147 
148     /**
149      * Populate the ResultSet.
150      * @param rs      The ResultSet.
151      * @return        The Object to retrieve from DB.
152      * @exception     SQLException if something is wrong.
153      */
154     protected List<Employees> getResults(ResultSet rs) throws SQLException {
155         List results = new ArrayList<Employees>();
156         while (rs.next()) {
157             Employees bean = new Employees();
158             bean.setEmpNo(rs.getInt("emp_no"));
159             bean.setBirthDate(rs.getDate("birth_date"));
160             bean.setFirstName(rs.getString("first_name"));
161             bean.setLastName(rs.getString("last_name"));
162             bean.setGender(rs.getObject("gender"));
163             bean.setHireDate(rs.getDate("hire_date"));
164             results.add(bean);
165         }
166         return results;
167     }
168 
169     /**
170      * Close JDBC Statement.
171      * @param stmt  Statement to be closed.
172      */
173     protected void close(Statement stmt) {
174         if (stmt != null) {
175             try {
176                 stmt.close();
177             }catch(SQLException e){}
178         }
179     }
180 
181     /**
182      * Close JDBC ResultSet.
183      * @param rs  ResultSet to be closed.
184      */
185     protected void close(ResultSet rs) {
186         if (rs != null) {
187             try {
188                 rs.close();
189             }catch(SQLException e){}
```

```
190         }
191     }
192 }
```

A DAO tesztelése

A következőkben a generált *DAO* forráskódok használatát mutatjuk be. Első lépésként importáljuk be ezeket a Java forrásfájlokat a projektünkbe. Az 1-11. Programlista egy objektum betöltését mutatja, amit a *testLoadObject()* valósít meg. A 27. sorban beállítjuk azt a kulcsot, aminek megfelelő objektumot akarjuk megkapni. Az objektum betöltése a 31. sorban valósul meg. Láthatjuk, hogy a *Connection* objektumot mindig paraméterként adjuk át, ezzel sokkal rugalmasabban lehet a saját kódunkat szervezni és a tranzakciókezelést is a magunk hatáskörében tudjuk lerendezni.

1-11. Programlista: TestDAO - Objektum betöltése

```
1
2  package org.cs.mysql.samples;
3
4
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.util.Date;
9  import java.util.List;
10 import org.cs.dao.Employees;
11 import org.cs.dao.EmployeesKey;
12 import org.cs.dao.dao.EmployeesDAO;
13 import org.cs.dao.orm.EmployeesDAOImpl;
14 import org.junit.Test;
15
16 import beans.Employee;
17
18 public class TestDAO {
19 ...
20         @Test
21         public void testLoadObject() {
22
23                 MySQLDatabaseSession session = new MySQLDatabaseSession();
24                 EmployeesDAO edao = new EmployeesDAOImpl();
25
26                 EmployeesKey key = new EmployeesKey();
27                 key.setEmpNo(10006);
28
29                 try {
30                         session.connect();
31                         Employees employee = edao.load(key, session.conn);
32
33                         System.out.println(employee.getFirstName());
34
35                 } catch (SQLException e) {
36                         e.printStackTrace();
37                 } finally {
38                         session.disconnect();
39                 }
40         }
41 ...
42 }
```

A *testLoadObjects()* metódus (1-12. Programlista) több objektum betöltési lehetőségét szemlélteti, ezeket 31. sorban lévő SQL select-tel tudtuk megadni, ami igen rugalmas lehetőségeket biztosít.

1-12. Programlista: TestDAO - Több objektum betöltése

```
1
2  package org.cs.mysql.samples;
3
4
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.util.Date;
9  import java.util.List;
10 import org.cs.dao.Employees;
11 import org.cs.dao.EmployeesKey;
12 import org.cs.dao.dao.EmployeesDAO;
13 import org.cs.dao.orm.EmployeesDAOImpl;
14 import org.junit.Test;
15
16 import beans.Employee;
17
18 public class TestDAO {
19 ...
20         @Test
21         public void testLoadObjects() {
22
23                 MySQLDatabaseSession session = new MySQLDatabaseSession();
24                 EmployeesDAOImpl dao = new EmployeesDAOImpl();
25
26                 EmployeesKey key = new EmployeesKey();
27                 key.setEmpNo(10006);
28
29                 try {
30                         session.connect();
31                         String selectSQL = "select * from employees where emp_no <= 10006";
32                         PreparedStatement pst = session.getPreparedStatement(selectSQL);
33                         ResultSet rs = pst.executeQuery();
34                         List<Employees> emberek = dao.getResults(rs);
35                         session.releaseAllUtililty(null, pst, rs);
36
37                         for (Employees employee : emberek) {
38                                 System.out.println(employee.getFirstName());
39                         }
40
41                 } catch (SQLException e) {
42                         e.printStackTrace();
43                 } finally {
44                         session.disconnect();
45                 }
46         }
47 ...
48 }
```

A *testCreateObject()* metódus (1-13. Programlista) egy új *Employees* objektum adatbázisba mentését szemlélteti.

1-13. Programlista: TestDAO - Új objektum eltárolása

```
1
2  package org.cs.mysql.samples;
3
```

```
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.util.Date;
8  import java.util.List;
9  import org.cs.dao.Employees;
10 import org.cs.dao.EmployeesKey;
11 import org.cs.dao.dao.EmployeesDAO;
12 import org.cs.dao.orm.EmployeesDAOImpl;
13 import org.junit.Test;
14
15 import beans.Employee;
16
17 public class TestDAO {
18 ...
19         @Test
20         public void testCreateObject() {
21
22                 MySQLDatabaseSession session = new MySQLDatabaseSession();
23                 EmployeesDAO edao = new EmployeesDAOImpl();
24
25                 try {
26                         session.connect();
27                         Employees ember = new Employees();
28                         ember.setEmpNo(10000);
29                         ember.setFirstName("Imre");
30                         ember.setLastName("Nyiri");
31
32                         java.util.Calendar cal = new java.util.GregorianCalendar();
33                         cal.set(1963, 1, 5);
34                         ember.setBirthDate( cal.getTime() );
35                         ember.setGender("M");
36                         cal.set(2000, 1, 5);
37                         ember.setHireDate( cal.getTime() );
38                         edao.create(ember, session.conn);
39                 } catch (SQLException e) {
40                         e.printStackTrace();
41                 } finally {
42                         session.disconnect();
43                 }
44         }
45 ...
46 }
```

A *testDeleteObject()* (1-14. Programlista) példát ad egy megadott kulccsal rendelkező sor törlésére.

1-14. Programlista: TestDAO - Egy objektum törlése

```
1
2  package org.cs.mysql.samples;
3
4
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.util.Date;
9  import java.util.List;
10 import org.cs.dao.Employees;
11 import org.cs.dao.EmployeesKey;
12 import org.cs.dao.dao.EmployeesDAO;
13 import org.cs.dao.orm.EmployeesDAOImpl;
14 import org.junit.Test;
15
```

```
16 import beans.Employee;
17
18 public class TestDAO {
19 ...
20         @Test
21         public void testDeleteObject() {
22
23                 MySQLDatabaseSession session = new MySQLDatabaseSession();
24                 EmployeesDAO edao = new EmployeesDAOImpl();
25
26                 try {
27                         session.connect();
28                         EmployeesKey key = new EmployeesKey();
29                         key.setEmpNo(10000);
30                         edao.delete(key, session.conn);
31                 } catch (SQLException e) {
32                         e.printStackTrace();
33                 } finally {
34                         session.disconnect();
35                 }
36         }
37 ...
38 }
```

Végül a *tesUpdateObject()* metódus (1-15. Programlista) egy objektum adatbázisbeli perzisztenciájának módosítását tanítja meg számunkra.

1-15. Programlista: TestDAO - Egy objektum módosítása

```
1
2  package org.cs.mysql.samples;
3
4
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.util.Date;
9  import java.util.List;
10 import org.cs.dao.Employees;
11 import org.cs.dao.EmployeesKey;
12 import org.cs.dao.dao.EmployeesDAO;
13 import org.cs.dao.orm.EmployeesDAOImpl;
14 import org.junit.Test;
15
16 import beans.Employee;
17
18 public class TestDAO {
19 ...
20         @Test
21         public void tesUpdateObject() {
22
23                 MySQLDatabaseSession session = new MySQLDatabaseSession();
24                 EmployeesDAO edao = new EmployeesDAOImpl();
25
26                 EmployeesKey key = new EmployeesKey();
27                 key.setEmpNo(10000);
28
29                 try {
30                         session.connect();
31                         Employees employee = edao.load(key, session.conn);
32                         System.out.println(employee.getFirstName());
33                         java.util.Calendar cal = new java.util.GregorianCalendar();
34                         cal.set(1963, 0, 5);
35                         employee.setBirthDate(  cal.getTime() );
```

```
36                          edao.update(employee, session.conn);
37
38                  } catch (SQLException e) {
39                          e.printStackTrace();
40                  } finally {
41                          session.disconnect();
42                  }
43          }
44  ...
45  }
```

Tranzakciókezelés és egyéb JDBC lehetőségek

A fejezet befejezéseként szeretnénk kiemelni, hogy a MySQL Java programozása lehetővé teszi az ismert JDBC lehetőségek használatát. Lehetséges az elosztott tranzakciókezelés, azaz a MYSQL driver képes részt venni *JTA* (Java Transaction API) tranzakciós kontextusokban is. Az SQL kurzorban való mozgás lehetséges. Példák:

```
resultSet.next(); // Régóta ismert
resultSet.previous();
resultSet.beforeFirst();
resultSet.relative(-4);
resultSet.first();
resultSet.last();
```

Használhatóak a következő modern JDBC API osztályok, amik a kapcsolat lezárása után is elérhetővé hagyják az adatokat:

- *JdbcRowSet*
- *CachedRowSet*
- *WebRowSet*
- *JoinRowSet*
- *FilteredRowSet*

Lehetőség van arra, hogy egy műveletet ne végeztessünk el egyből, hanem az *addBatch()* hívással csak jegyezzük fel, majd összegyűjtve azokat 1 menetben az *executeBatch()* hívással küldjük el az adatbázis szerver felé.

```
preparedStatement.setInt(1, 101);
preparedStatement.setString(2, "mkyong101");
preparedStatement.setString(3, "system");
preparedStatement.setTimestamp(4, getCurrentTimeStamp());
preparedStatement.addBatch();

preparedStatement.setInt(1, 102);
preparedStatement.setString(2, "mkyong102");
preparedStatement.setString(3, "system");
preparedStatement.setTimestamp(4, getCurrentTimeStamp());
preparedStatement.addBatch();
preparedStatement.executeBatch();
```

Lehetőség van a JDBC listener-ek használatára is.

2. Java perzisztencia megvalósítás - MyBatis

A *MyBatis* egy Java perzisztenciát megvalósító keretrendszer. Mindezt a már említett *ORM* alapú megközelítéssel valósítja meg, hiszen a Java objektumok és a relációs adatbázisok között kell az oda-vissza (író és olvasó) kapcsolatot kialakítania. Webhely: `mybatis.org`.

A MyBatis szerepe és koncepciója

Ennek van egy szöveges alapú

Egy teljes példa a MyBatis használatára

1-7. Programlista: Employees.java

2-1. Programlista: EmployeesMapper.java

```
1
2  package org.cs.mybatis.test;
3
4  import java.util.List;
5
6  public interface EmployeesMapper {
7      List<Employees> findAllEmployees();
8      Employees findEmployeesById(Integer id);
9      void insertEmployees(Employees employee);
10     void updateEmployees(Employees employee);
11 }
```

2-2. Programlista: mybatis-config.xml

```
1
2  <?xml version="1.0" encoding="UTF-8" ?>
3  <!DOCTYPE configuration
4    PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
5    "http://mybatis.org/dtd/mybatis-3-config.dtd">
6  <configuration>
7
8        <typeAliases>
9         <package name="org.cs.mybatis.test"/>
10
11       </typeAliases>
12
13       <environments default="development">
14         <environment id="development">
15           <transactionManager type="JDBC"/>
16           <dataSource type="POOLED">
17             <property name="driver" value="com.mysql.jdbc.Driver"/>
18             <property name="url" value="jdbc:mysql://localhost:3306/employees?useUnicode=true&amp;
                   characterEncoding=UTF-8"/>
19             <property name="username" value="imreuser"/>
20             <property name="password" value="111111"/>
21           </dataSource>
22         </environment>
23       </environments>
24
```

```
25      <mappers>
26        <mapper url="file:/home/inyiri/workspace/MyBatisTestProject/resources/mybatis/mappers/➥
              EmployeesMapper.xml"/>
27      </mappers>
28  </configuration>
```

2-3. Programlista: EmployeesMapper.xml

```
1
2  <?xml version="1.0" encoding="UTF-8" ?>
3  <!DOCTYPE mapper
4    PUBLIC "-//mybatis.org//DTD␣Mapper␣3.0//EN"
5    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
6
7  <mapper namespace="org.cs.mybatis.test.EmployeesMapper">
8
9      <resultMap type="Employees" id="EmployeesResult">
10       <id property="empNo" column="emp_no" />
11       <result property="birthDate" column="birth_date" />
12       <result property="firstName" column="first_name" />
13       <result property="lastName" column="last_name" />
14       <result property="gender" column="gender" />
15       <result property="hireDate" column="hire_date" />
16     </resultMap>
17
18     <select id="findAllEmployees" resultMap="EmployeesResult">
19       select emp_no, birth_date, first_name, last_name, gender, hire_date from employees where ➥
             emp_no &lt; 10008
20     </select>
21
22     <select id="findEmployeesById" parameterType="java.lang.Integer" resultType="Employees" ➥
         resultMap="EmployeesResult">
23       SELECT emp_no, birth_date, first_name, last_name, gender, hire_date FROM
24       employees WHERE emp_no = #{empNo}
25     </select>
26
27     <insert id="insertEmployees" parameterType="Employees">
28       INSERT INTO employees (emp_no, birth_date, first_name, last_name, gender, hire_date)
29       VALUES (#{empNo}, #{birthDate}, #{firstName}, #{lastName}, #{gender}, #{hireDate})
30     </insert>
31
32     <update id="updateEmployees" parameterType="Employees">
33       UPDATE employees SET birth_date = #{birthDate}, first_name = #{firstName},
34       last_name = #{lastName}, gender = #{gender}, hire_date = #{hireDate}
35       WHERE emp_no = #{empNo}
36     </update>
37
38 </mapper>
```

2-4. Programlista: TestMyBatis.java

```
1
2  package org.cs.mybatis.test;
3
4  import java.io.IOException;
5  import java.io.InputStream;
6  import java.util.List;
7
8  import org.apache.ibatis.io.Resources;
9  import org.apache.ibatis.session.SqlSession;
10 import org.apache.ibatis.session.SqlSessionFactory;
11 import org.apache.ibatis.session.SqlSessionFactoryBuilder;
```

```
12
13 public class TestMyBatis {
14
15    SqlSessionFactory sqlSessionFactory = null;
16
17    public SqlSessionFactory getSqlSessionFactory()
18    {
19       if(sqlSessionFactory==null)
20       {
21          InputStream inputStream = null;
22          SqlSession sqlSession = null;
23          try
24          {
25             //inputStream = Resources.getResourceAsStream("mybatis-config.xml");
26             inputStream = Resources.getUrlAsStream("file:/home/inyiri/workspace/➥
                   MyBatisTestProject/resources/mybatis/mybatis-config.xml");
27             sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
28
29             sqlSession = sqlSessionFactory.openSession();
30
31             EmployeesMapper employeesMapper = sqlSession.getMapper(EmployeesMapper.class);
32
33             Employees e = null;
34             System.out.println( e == null );
35             e = employeesMapper.findEmployeesById(10000);
36
37             e.setFirstName("Imre (x)");
38             employeesMapper.updateEmployees(e);
39             sqlSession.commit();
40
41             List<Employees> resEmps = employeesMapper.findAllEmployees();
42             for ( Employees emp : resEmps ) {
43                System.out.println( emp.getFirstName() );
44             }
45
46             System.out.println( e.getFirstName() );
47
48          }catch (IOException e)
49          {
50             e.printStackTrace();
51          }finally {
52             if(inputStream != null){
53                try {
54                   inputStream.close();
55                   sqlSession.close();
56                } catch (IOException e) {
57                }
58             }
59          }
60       }
61       return sqlSessionFactory;
62    }
63
64    public static void main(String[] args) {
65       TestMyBatis test = new TestMyBatis();
66       test.getSqlSessionFactory();
67       System.out.println("The program is completted");
68    }
69 }
```

3. Maven - A Java projekt kezelés hatékony támogatása

Manapság az Apache Maven eszközt használjuk projektjeink építéséhez, menedzseléséhez. A Maven használatával egy egyszerű xml fájlban tudjuk megadni a projekt felépítéséhez szükséges összes információt. A különböző függőségek összevadászásával sem kell többet bajlódnunk, hiszen azokat a Maven tölti le egy repository-ból az xml fájlban megadott információk alapján. A *Maven Central Repository* rengeteg projektet tesz könnyen elérhetővé, de ezen felül léteznek más gyártók által készített repository-k is, sőt saját magunk is létrehozhatunk ilyet.

Mi a Maven?

Az Apache Maven (röviden Maven) egy szoftver, amelyet szoftverprojektek menedzselésére és a build folyamat automatizálására lehet használni. Támogatja az összeállítást (*build*), függőségkezelést (*dependency*), kiadásokat (*release*), terjesztéseket (*distribution*), a verziókezelést, dokumentálást és jelentéskészítés. Egy nagyobb szoftver projekt több framework-öt, programcsomagot tartalmazhat, melyeket összefoglaló néven függőségeknek (*dependency*) szoktunk nevezni. Egy java projekt függőségei általában *jar* fájlok szoktak lenni. A Maven bevezeti a *POM*, azaz a *Project Object Model* fogalmát. Egy POM egy build-elendő projektet ír le, annak függőségeivel és build-elési eljárásaival együtt. Az egyes lépéseket céloknak, angolul *goal*-oknak nevezik. Vannak előre definiált célok a tipikus feladatokra, mint például a kód fordítása és csomagolása, de a felhasználónak lehetősége van saját célokat is definiálni a projektspecifikus lépések végrehajtására. A Maven hálózatképes, tehát szükség esetén dinamikusan is le tud tölteni komponenseket. *Repository* névvel illetik a különböző hosztok fájlrendszereinek azon mappáit, ahol a letölthető komponensek találhatók. A Maven nem csak a repository-kból való letöltést támogatja, hanem az elkészült szoftvercsomag feltöltését is.

Telepítés

Az *M2_HOME* környezeti változónak arra a könyvtárra kell mutatnia, ahova a maven-t kicsomagoltuk. Az *export PATH=${M2}/bin:$PATH* beállítása után az eszköz bárhonnan elérhető. A maven sikeres telepítésének ellenőrzéseként egy konzol ablakban futtassuk le az *mvn –version* parancsot:

```
inyiri@earth:/home/tanulas/maven$ mvn -version

Apache Maven 3.3.3
Maven home: /usr/share/maven
Java version: 1.8.0_60, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-8-oracle/jre
Default locale: hu_HU, platform encoding: UTF-8
OS name: "linux", version: "4.2.0-27-generic", arch: "amd64", family: "unix"
```

A maven első futtatása után létrejön egy *.m2* mappa (ez Linuxon a *~/.m2*), mely tartalmazni fogja a letöltött függőségeket, valamint az itt található *settings.xml*-ben különböző beállításokat adhatunk meg. Az Eclipse (*m2eclipse*, `http://www.sonatype.org/m2eclipse`) és a NetBeans is beépített maven támogatással rendelkezik.

A maven első használata

Minden maven projekt alapja a *pom.xml* fájl, mely minden fontos információt tartalmaz: a projekt neve, verziója, csomagolási formátuma, a fordítási beállítások. Itt kell megadni azt is, hogy milyen függőségei vannak a projektnek, milyen *jar*-okat töltsön le a maven fordítás során. Már itt megjegyezzük, hogy ezen letöltött függőségek a *~/.m2/repository* mappába kerülnek, csomagokba és verziókba (mappa hierarchiába) szervezve. Érdemes megtekinteni a saját gépünkön! A maven segít az induló projektek létrehozásában, amihez az *archetype* nevű projekt minta mechanizmus használja. Adjuk ki például a következő *mvn archetype:generate...* kezdetű parancsot, ami egy egyszerű java projektet hoz létre:

```
mvn archetype:generate -DgroupId=org.cs.maven -DartifactId=Test -DarchetypeArtifactId=maven-archetype-quickstart➥
    -DinteractiveMode=false

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.4:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.4:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.4:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: /home/tanulas/maven
[INFO] Parameter: package, Value: org.cs.maven
[INFO] Parameter: groupId, Value: org.cs.maven
[INFO] Parameter: artifactId, Value: Test
[INFO] Parameter: packageName, Value: org.cs.maven
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: /home/tanulas/maven/Test
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.514 s
[INFO] Finished at: 2016-02-14T11:20:28+01:00
[INFO] Final Memory: 15M/201M
[INFO] ------------------------------------------------------------------------
```

A képernyőre írt szövegből most a *basedir* property-t emeljük ki, ez mindig az a hely, ahol a projekt tárolódik. A létrejött projekt neve *Test* (megegyezik az *artifactId* nevével), a szerkezetét nézzük meg a *tree* paranccsal:

```
tree /home/tanulas/maven/Test

Test/
   pom.xml
   src
       main
           java
                org/sbin/rcvboxdrv setup
                    cs
                       maven
                          App.java
       test
           java
                org
                    cs
                        maven
                           AppTest.java

11 directories, 3 files
```

A *basedir* segítségével kifejezve nézzük meg a maven projektek mappa szerkezetét:

- *${basedir}/src/main/java* → java forráskód
- *${basedir}/src/main/resources* → erőforrás fájlok

- *${basedir}/src/test* → a unit teszt forráskódja
- *${basedir}/target* → a legyártott artifact ide kerül
- *${basedir}/target/classes* → a lefordított fájlok (java class) helye

A fenti szerkezet a *maven-archetype-quickstart* nevű archetype formátuma, ez egy egyszerű java library számára mindig megfelelő projekt felépítés. Az automatikusan létrehozott *pom.xml* fájl így néz ki:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.cs.maven</groupId>
  <artifactId>Test</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Test</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Az *artifactId* a projektünk neve, míg a *groupId* tag-ek között található a projektet készítő névtér (vagy csomag) azonosítója, ez jelen esetben a *org.cs.maven* név. A *packaging* részben adhatjuk meg, hogy milyen típusú állományt készítsen a maven a projektünkből. A *name* tag-ek között található a projekt neve, a *version* tag-ek közt pedig jelenlegi verziószáma. A *dependencies* részben tudjuk felsorolni a projekt függőségeit. A projekteket le tudjuk fordítani konzolból az *mvn clean install* paranccsal, de az IDE-kből is lehetséges a fordítás és a futtatás is. Fordítás előtt a maven letölti a projekt függőségeit az említett *~/.m2/repository* előre mappába, majd a fordítás eredménye a projekt könyvtárában létrejött *target* mappában lesz megtalálható (*jar*, *war*, *ear*, ... fájl, attól függően, hogy mit állítottunk be). Nézzünk a függőségekre még erre egy példát, itt az ismert Common Math csomagot tesszük elérhetővé a projekt számára:

```
<dependency>
   <groupId>org.apache.commons</groupId>
   <artifactId>commons-math3</artifactId>
   <version>[3.3,)</version>
   <scope>compile</scope>
</dependency>
```

Most a második számot kihagytuk, s ez által nem adtunk meg felső határt, mivel jelenleg nem tudjuk, hogy mely verziót szeretnénk majd a jövőben letölteni. Dependenciák legfrissebb verziója a 3.3-as, így célszerű azt megadni legkisebb verziószámnak.

Függőségek (dependenciák) kezelése

A függőség hatóköre (Dependencia scope)

Lehetőségünk van beállítani, hogy az egyes dependenciák, alkalmazásunk életciklusának mely szakaszában lépjenek életbe. Erre azért lehet szükség, mert korlátozni szeretnénk az egyes függőségek élettartalmát, vagy befolyásolni akarjuk a fordítási folyamataink menetét. Összesen 6 ilyen scope létezik:

- *compile*: Ez az alapvető scope. Amennyiben nem állítunk be scope-ot, ebben a módban használja a maven a függőséget. Ebben az esetben a dependencia minden fordításnál betöltődik a projektünkbe, illetve letöltődik, ha nincs még fenn a gépünkön.
- *runtime*: Ezt a scope-ot akkor használjuk, ha a dependenciára nincs szükség fordítás során, csak futtatás közben.
- *test*: Ezt a scope-ot akkor használjuk, ha csak a tesztjeinknek van szüksége az adott függőségre.
- *provided*: Build időben elérhető, de nem lesz benne az artifact-ban (a szerver tartalmazza majd ezeket a jar fájlokat)
- *system*: Megadjuk, hogy hol található a *jar* a fájlrendszerben.

A projekt által ismert függőségeket kilistázhatjuk ezzel a paranccsal:

```
mvn dependency:tree
```

A projekt build függőségeinek lekérdezésére pedig ezt a parancsot érdemes használni:

```
mvn dependency:build-classpath
```

A verziók kezelése

Minden függőségnél megadhatjuk, hogy mely verziót töltse le belőle a maven. Általában érdemes konkrét verziószámot megadni, de a maven lehetőséget biztosít arra, hogy verziószámok tartományát adjuk meg. Tartomány megadásánál zárójelek közé kell két verziószámot írni. Az első szám jelképezi az intervallum kezdetét, a második a végét. Amennyiben a zárójel szögletes, a megadott szám részét képezi az intervallumnak, sima zárójel esetén viszont nincs benne (Példa.: *[2.1, 2.9]*). A tartományok egyik leghasznosabb felhasználási módja, hogy rávehetjük a mavent arra, hogy mindig a dependencia legfrissebb verzióját használja.

A repository-k használata

A repository az a tárhely ahonnan a maven le tudja tölteni az ott megtalálható függőségeket. A Maven rendelkezik egy úgynevezett central avagy központi repository-val (`http://mvnrepository.com/`), melyet alapértelmezetten használ minden Maven projekt. Természetesen léteznek kisebb repository-k is, melyek például egy-egy gyártó termékeit tartalmazhatják, de akár mi is létrehozhatunk egy sajátot. Példa egy új repository felvételére:

```
<repositories>
   <repository>
      <id>central</id>
      <url>http://repo1.maven.org/maven2</url>
   </repository>
</repositories>
```

Amint láthatjuk az egyes repository-kat a *repository* tag-ek között kell felsorolni (ezt a részt az átláthatóság kedvéért célszerű a *pom.xml* fájl elejére írni), a repository tulajdondonságait pedig a tag-ek közt. Mindenképpen meg kell adni a repository id-át, valamint az URL-t, amin elérhető. Ebből a repository-ból a *Spring framework* verzióit lehet letölteni:

```
<repository>
   <id>repository.springframework.maven.release</id>
   <name>Spring Framework Maven Release Repository</name>
   <url>http://maven.springframework.org/release</url>
</repository>
```

Írunk majd a maven plugin-jairól (kiegészítőiről), de már most érdemes megjegyezni, hogy a plugin-eknek is vannak repository-ai, amit *PluginRepository*-nak hívnak, s az alábbi módon hivatkozhatunk rájuk a *pom* fájlban:

```
<pluginRepositories>
   <pluginRepository>
      <id>...</id>
      <url>...</url>
   </pluginRepository>
</pluginRepositories>
```

Amint láthatjuk a *PluginRepository*-kat gyakorlatilag úgy kell felsorolni, mint a sima repository-kat, csak a tag-ek neve tér el. Végezetül fontos szót ejtenünk a gépünkön megtalálható lokális repository-ról. Ez gyakorlatilag az a könyvtár, ahova a Maven az összes projektünk által használt függőséget letölti, s a beállításokat is tartalmazó *.m2* mappában található. A maven fordításonként csak azokat a függőségeket tölti le, melyek nem találhatóak meg a lokális repository-ban, ezek az új, illetve a megváltozott verziószámú dependenciák. Az első fordítás során a jar fájl letöltődik, de a későbbi fordítások során csak akkor fog egy új fájl letöltődni, ha újabb jar verzió került fel a központi repository-ba. Lehetőség van rá, hogy kézzel kitöröljük a dependenciákat, s arra is, hogy egy, a gépen található jar-t telepítsük a saját lokális repository-ba. Ez akkor jöhet jól, ha többen dolgozunk egy projekten, nincs saját repository, viszont van egy általunk készített dependencia, melyet mindannyian használni szeretnénk. Például, ha egy */home/jars/mylib.jar* nevű dependenciát szeretnék a lokális repository-ba telepíteni, akkor azt az alábbi paranccsal tehetjük meg:

```
mvn install:install-file -Dfile=/home/jars/mylib.jar -DgroupId=org.cs.csomag -DartifactId=Valami -Dversion=1.0 -➥
    Dpackaging=jar
```

Build setup plugins

Most a build plugin-eket ismerjük meg, melyek lehetővé teszik különböző kiegészítő építő eszközök használatát.

Property

A property-kre *${property_név}* formában tudunk hivatkozni. Segítségükkel egyszerűbbé tehetjük maven projektjeink karbantartását, valamint megkönnyíthetjük hordozhatóságát. Például, ha néhány dependencia szorosan kapcsolódik egymáshoz, s azt szeretnénk, ha mind ugyanazt a verziószámú verziót használják, akkor azt a verziószámot beleírhatjuk egy property-be, így a későbbiekben csak egy helyen kell megváltoztatni azt. De a property-k segítségével használhatjuk projektünk környezeti változóit is (például a *pom* fájlt tartalmazó könyvtár), annak érdekében, hogy a projekten dolgozó összes fejlesztő változtatás nélkül tudja használni a pom fájlt. Végső soron többféle property különböztethető meg:

- Beépített property-k:

– *${basedir}* a pom fájlt tartalmazó könyvtár-t reprezentálja, már megismertük korábban

${version} a projekt verziószámát reprezentálja

- Projekt property-k: A projektünkre vonatkozó, automatikusan létrejövő property-k. Például: *${project.name}*
- Rendszerünk környezeti változóira is hivatkozhatunk. Például: *${java.home}* vagy *${env.M2_HOME}*
- Saját magunk is hozhatunk létre property-ket a *pom* fájlban a *<properties>...</properties>* tag-ek között.

Nézzük meg a következő példán keresztül a property létrehozását és használatát:

```
<properties>
        <vaadin.version>7.3.2</vaadin.version>
</properties>

<dependencies>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-server</artifactId>
            <version>${vaadin.version}</version>
        </dependency>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-client-compiled</artifactId>
            <version>${vaadin.version}</version>
        </dependency>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-themes</artifactId>
            <version>${vaadin.version}</version>
        </dependency>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-maven-plugin</artifactId>
            <version>${vaadin.version}</version>
        </dependency>
</dependencies>
```

A property-k továbbá a resource-ok kezelésében is fontos szerepet játszanak. Mik is azok a resource-ok? A resource-nak hívjuk a projektünk által használt, nem java forrásállományokat, például különböző xml leírókat, property fájlokat, vagy akár képeket is. A property-k segítségével a resource-ok könnyen paraméterezhetőek lesznek, ez által projektünk hordozhatóbbá válik.

A Build Plugin

A maven alapvetően egy pluginekkel modularizált rendszer, vagyis a *pom.xml* értelmezésétől eltekintve, projektjeink gyakorlatilag plugin-ek futtatásának eredményeképp jönnek létre. Például az *install* plugin segítségével lefordíthatjuk és lokálisan installálhatjuk projektünket. Ám rengeteg olyan plugin létezik, melyeket alapvetően nem használnak a projektek, de kézzel beköthetjük őket. Pom fájlunk *<build>...</build>* tag-ekkel határolt részében adhatunk meg mindenféle, a fordítási folyamattal kapcsolatos beállítást, például *build* plugin-eket is itt húzhatunk be a projektbe. Példaként nézzünk 2 gyakori plugint, hogy képet kapjunk a használatukról:

- A *maven-jar-plugin* segítségével többek közt megadhatjuk, hogy mely osztályunkban található a main metódus, mivel a plugin használata nélkül a *jar* fájlok futtatásakor kellene ezt megadni (példa: *mvn exec:java -Dexec.mainClass="org.cs.TestClass"*).

- A *maven-dependency-plugin* a dependenciák kezeléséért felel. A *copy-dependencies* goal felel azért, hogy a repository-ból, projektünk egy meghatározott könyvtárába kerüljenek a felhasznált dependenciák.

A fenti 2 plugin használata így néz ki:

```
<build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>2.4</version>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>hu.zerotohero.MavenDemo</mainClass>
                            <addClasspath>true</addClasspath>
                            <classpathPrefix>lib/</classpathPrefix>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <version>2.8</version>
                <executions>
                    <execution>
                        <id>copy-dependencies</id>
                        <phase>package</phase>
                        <goals>
                            <goal>copy-dependencies</goal>
                        </goals>
                        <configuration>
                            <outputDirectory>${project.build.directory}/lib</outputDirectory>
                            <overWriteReleases>false</overWriteReleases>
                            <overWriteSnapshots>false</overWriteSnapshots>
                            <overWriteIfNewer>true</overWriteIfNewer>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
 </build>
```

Amint látható a *<plugins>...</plugins>* tag-ek között kell felsorolni a különböző plugineket. Hasonlóan a dependenciákhoz, mindig meg kell adni plugin-ek *groupId-ját*, *artifactId*-ját és verziószámát. A *configuration* részben a plugin különböző tulajdonságait állíthatjuk be, míg az *executions* részben a build folyamatot lehet befolyásolni.

Egy példa Build Plugin az XML Beans használatára

A következő példa azt mutatja be, hogy miképpen tudjuk automatizálni azt a build-elési lépést, amikor egy XSD→JAR építést végzünk az XML Beans könyvtár segítségével. Ehhez az *xmlbeans-maven-plugin* build plugin-t használjuk. A *schemaDirectory* tag azt a mappát határozza meg, ahol az XSD fájlok vannak, míg a *sourceGenerationDirectory* a generált java forráskód helyét specifikálja. A függőségek közé betettük magának a plugin-nak az artifact-ját és az XML Beans könyvtárat is.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>org.cs.maven</groupId>
        <artifactId>Test</artifactId>
        <packaging>jar</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>Test</name>
        <url>http://maven.apache.org</url>

        <dependencies>
                <dependency>
```

```
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>

                <dependency>
                        <groupId>org.codehaus.mojo</groupId>
                        <artifactId>xmlbeans-maven-plugin</artifactId>
                        <version>2.3.3</version>
                </dependency>

                <dependency>
                        <groupId>org.apache.xmlbeans</groupId>
                        <artifactId>xmlbeans</artifactId>
                        <version>2.6.0</version>
                        <scope>compile</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.codehaus.mojo</groupId>
                                <artifactId>xmlbeans-maven-plugin</artifactId>
                                <executions>
                                        <execution>
                                                <id>Build-XMLBeans</id>
                                                <phase>generate-sources</phase>
                                                <goals>
                                                        <goal>xmlbeans</goal>
                                                </goals>
                                        </execution>
                                </executions>
                                <inherited>true</inherited>
                                <configuration>
                                        <memoryInitialSize>32m</memoryInitialSize>
                                        <memoryMaximumSize>64m</memoryMaximumSize>
                                        <schemaDirectory>src/main/resources/xsd</schemaDirectory>
                                        <sourceGenerationDirectory>src/main/java</sourceGenerationDirectory>
                                        <verbose>true</verbose>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Több modulból álló projektek

Miért is jó, ha több modulba szervezzük projektjeinket? Alapvetően a Java projektek osztályait package-ekbe szokás szervezni, ezzel könnyebben átláthatóvá téve a projektet. Ám egy-egy nagyobb projekt több ezer osztályt, s több száz package-et is tartalmazhat, s ez indokolttá tehet egy újabb absztrakció bevezetését a projekt szerkezetébe. Egy projekten belül lehetőségünk van rá, hogy az alkalmazás fő funkcionalitásai mentén modulokba szervezzük a package-eket és az osztályokat. Másfelől különféle formátuma van a *web*, *ejb* és egyéb projekteknek, ezeket is érdemes szétválasztani őket egymástól. A multi modulos maven projekt egy fő, azaz pom vagy parent projektből és a beágyazott modul projektekből áll.

Eclipse használata

Eclipse esetén vegyük fel a következő 2 repository-t a *Window→Preferences→Maven→Archetypes* helyen az *Add Remote Catalog...* gomb megnyomásával:

- Catalog file: `http://repo.maven.apache.org/maven2` és Description: *Maven Central*
- Catalog file: `http://download.java.net/maven/2` és Description: *Java.Net*

Először a parent projektet kell létrehozni a *File→New→Other→Maven* helyen lévő *Maven Project* választással. A megjelenő *Filter* mezőben szűrhetünk, nekünk egy *pom* típusú projekt kell. A lehetséges választásokból válasszuk a *org.codehaus.mojo.archetypes:pom-root:1.1* archetype-ot, ahol az artifactId *pom-root*, majd adjuk meg a projekt koordinátákat (példánkban ezt adtuk meg: *TestAppGroup*, *TestAppArtifact*, 0.1 verzió, *org.cs.test* csomag). A következő lépés az EAR project létrehozása. Álljunk az Eclipse-ben a *TestAppArtifact* projektre, nyomjuk meg a jobb egérgombot és válasszuk ki a *Maven→New Maven Module Project* menüpontot. Ekkor a parent project a *TestAppArtifact* lesz és a module neveként megadhatjuk például az *EAR-Project* nevet. A next gomb megnyomása után a *Filter* mezőbe írjuk be, hogy *ear*, majd a megjelent listából válasszuk ki a *org.codehaus.mojo.archetypes* group megfelelő *artifactId-jét*, esetünkben a példában az *ear-javaee6* (version 1.5) lesz használva. Az EAR projektünk *groupId*-ja marad *TestAppGroup* (és a package is *org.cs.test* marad). A következő lépés egy web projekt (azaz module) beiktatása az alkalmazásunkba. Ismét álljunk rá a *TestAppArtifact* projektre és a *Maven→New Maven Module Project* menüpont segítségével adjunk hozzá például egy *WEB-Project* nevű modult. A next utáni képernyőn a *Filter* mezőbe most a *web* szócskát írjuk be és az archetype csoport azonosítója szintén az *org.codehaus.mojo.archetypes* maradjon, innen mi a *webapp-javaee6* (version 1.5) *artifactId*-t választottuk. A csoport azonosító marad *TestAppGroup* (a verzió is marad 0.1). Ezzel a parent projekt immár 2 modullal rendelkezik és jelenleg így néz ki:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:➥
    schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>TestAppGroup</groupId>
  <artifactId>TestAppArtifact</artifactId>
  <version>0.1</version>
  <packaging>pom</packaging>
  <name>TestAppArtifact</name>
  <modules>
    <module>EAR-Project</module>
    <module>WEB-Project</module>
  </modules>
</project>
```

Az egyes modulok *pom.xml* fájljai is tartalmazzák azt a bejegyzést, ami referál a parent projektre:

```
...
 <parent>
    <artifactId>TestAppArtifact</artifactId>
    <groupId>TestAppGroup</groupId>
    <version>0.1</version>
  </parent>
...
```

Bár az *mvn package* parancs már hibátlanul lefut a parent project *pom.xml*-re, de a keletkezett *EAR-Project-0.1.ear* még semmit sem tartalmaz a *META-INF* könyvtáron kívül, pedig a *WEB-Project-0.1.war* fájl is létrejött. A megoldás az, hogy az EAR-project *pom.xml*-be be kell írni ezt a függőség részt, azaz az EAR függ a WAR-tól:

```
...
  <dependencies>
        <dependency>
                <groupId>TestAppGroup</groupId>
                <artifactId>WEB-Project</artifactId>
                <version>0.1</version>
                <type>war</type>
        </dependency>
  </dependencies>
...
```

Ezután már a keletkezett ear tartalmazni fogja a war fájlt is. Végül vegyünk fel egy egyszerű java module projektet is az EAR-hoz, *LIB-Project* néven. Az EAR *pom.xml* függőségbe tegyük be ezt a dependenciát:

```
...
                <dependency>
                        <groupId>TestAppGroup</groupId>
                        <artifactId>LIB-Project</artifactId>
                        <version>0.1</version>
                        <type>jar</type>
                <dependency>
...
```

Az *mvn package* parancs a *lib* alkönyvtárba befordítja a keletkezett *LIB-Project-0.1.jar* fájlt is, azaz minden rendben működik.

Maven parancssor használata

Az eclipse-ben elvégzett lépéseket természetesen a maven parancsorban is el tudjuk végezni. A parent projekt generálása így néz ki:

```
mvn archetype:generate -DarchetypeGroupId=org.codehaus.mojo.archetypes -DarchetypeArtifactId=pom-root -➥
    DarchetypeVersion=RELEASE -DgroupId=org.cs.test -DartifactId=TestAppArtifact -Dversion=0.1 -Dpackage=org.cs➥
    .test
```

Belépve a parent projekt gyökerébe, az EAR projekt generálása így történhet:

```
mvn archetype:generate -DarchetypeGroupId=org.codehaus.mojo.archetypes -DarchetypeArtifactId=ear-javaee6 -➥
    DarchetypeVersion=RELEASE -DgroupId=org.cs.test -DartifactId=EAR-Project -Dversion=0.1 -Dpackage=org.cs.➥
    test
```

A webes alkalmazás generálása is hasonló, ugyanazokat at archetype korrdinátákat használtuk, mint az eclipse-ben:

```
mvn archetype:generate -DarchetypeGroupId=org.codehaus.mojo.archetypes -DarchetypeArtifactId=webapp-javaee6 -➥
    DarchetypeVersion=RELEASE -DgroupId=org.cs.test -DartifactId=WEB-Project -Dversion=0.1 -Dpackage=org.cs.➥
    test
```

Végül ehhez a projekthez is tegyünk hozzá egy könyvtár modult:

```
mvn archetype:generate -DgroupId=org.cs.test -DartifactId=LIB-Project -DarchetypeArtifactId=maven-archetype-➥
    quickstart -DinteractiveMode=false
```

A konfiguráció többi lépése, azaz a dependenciák felvétele, teljesen megegyezik az előzőekben leírtakkal.

Egy saját archetype létrehozása

A maven lehetőséget kínál úgynevezett archetype-ok készítésére, használatára, melyek segítségével gyakorlatilag felépíthetünk egy újra felhasználható projekt vázat, template-et. A már fentebb mutatott *maven-archetype-quickstart* sem más, mint egy ilyen archetype. Az archetype-okban meghatározhatjuk, hogyan fog kinézni egy projekt könyvtárstruktúrája, definiálhatjuk a *pom* fájl(ok) alapvető szerkezetét, valamint olyan osztályokat is előre létrehozhatunk, melyekről tudjuk, hogy szükségünk lesz rájuk a projektjeink során. Ezek általános template-k lesznek, ugyanis ahány projekt, annyi különböző eszközt használunk, és igazából ezekben a template-ekben a metszeteket kellene meghatároznunk, különben túl sok utómunkálattal járhat a generált projekt aktualizálása. Nézzük meg, hogyan készíthetünk saját archetype-ot!

Egy minta projekt készítése - az archetype alapjaként szolgáló projekt

Először hozzunk létre egy *ejb* projektet (a neve ez lesz: *EJB-Project*) a következő paranccsal:

```
mvn archetype:generate -DarchetypeGroupId=org.codehaus.mojo.archetypes -DarchetypeArtifactId=ejb-javaee6 -➥
    DarchetypeVersion=RELEASE -DgroupId=org.cs.test -DartifactId=EJB-Project -Dversion=0.1 -Dpackage=org.cs.➥
    test
```

A *pom.xml* fájlban ezek lesznek az arifact koordinátái:

```
...
  <groupId>TestAppGroup</groupId>
  <artifactId>EJB-Project</artifactId>
  <version>0.1</version>
  <packaging>ejb</packaging>
...
```

Ezután alakítsuk ki ez alapján a váz alapján azt a projekt szerkezetet és tartalmat, amit látni szeretnénk, ha ez alapján akarunk egy projektet legenerálni. Mi most csak annyit tettünk, hogy egy session bean-t is elhelyeztünk a projektbe.

Az archetype legenerálása a minta projekt alapján

Menjünk be a most létrehozott *EJB-Project* projekt gyökerébe és adjuk ki ezt a parancsot:

```
mvn archetype:create-from-project
```

A most létrehozott *EJB-Project/target/generated-sources/archetype* könyvtárba létrejött tartalom a projektünkből absztrahált archetype. Ezt az alkönyvtár fát mentsük el egy munkakönyvtárba, az archetype létrehozásához már csak erre lesz szükségünk. Az archetype könyvtár *src*, *target* alkönyvtárakat és a *pom.xml* fájlt tartalmazza. A *target* könyvtár nem kell, ezért adjuk ki az *mvn clean* parancsot, hogy megszűnjön. Ezen a ponton eljutottunk oda, hogy van egy archetype típusú forráskódunk, ennek lefordításával kapunk egy jar-t, ami már egy *archetype* és telepíteni tudjuk. A projektnek így néz ki a *pom.xml* fájlja (vegyük észre, hogy a packaging értéke: *maven-archetype*):

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:➥
    schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>TestAppGroup</groupId>
  <artifactId>EJB-Project-archetype</artifactId>
  <version>0.1</version>
  <packaging>maven-archetype</packaging>

  <name>EJB-Project-archetype</name>

  <build>
    <extensions>
      <extension>
        <groupId>org.apache.maven.archetype</groupId>
        <artifactId>archetype-packaging</artifactId>
        <version>2.4</version>
      </extension>
    </extensions>

    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-archetype-plugin</artifactId>
          <version>2.4</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

A fordítást és a local repository-ba való telepítést ezzel a paranccsal véhezhetjük el:

```
mvn install
```

Ennek hatására létrejön a maven local repository-ban az új bináris és használható archetype:

```
~/.m2/repository/TestAppGroup/EJB-Project-archetype/0.1/EJB-Project-archetype-0.1.jar
```

Az archetype használata

A lenti *archetype:generate* parancs a most létrehozott archetype (a neve: *EJB-Project-archetype*) felhasználásával egy új projekt létrehozását mutatja, annak képernyő kimenetével együtt:

```
inyiri@earth:/home/tanulas/maven/xxx/ttt$ mvn archetype:generate -DarchetypeGroupId=TestAppGroup ➥
    DarchetypeArtifactId=EJB-Project-archetype -DarchetypeVersion=0.1

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.4:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.4:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.4:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [TestAppGroup:EJB-Project-archetype:0.1] found in ➥
    catalog local
Define value for property 'groupId': : AlmaGrp
Define value for property 'artifactId': : Artif
Define value for property 'version':  1.0-SNAPSHOT: :
Define value for property 'package':  AlmaGrp: :
Confirm properties configuration:
groupId: AlmaGrp
artifactId: Artif
version: 1.0-SNAPSHOT
package: AlmaGrp
 Y: :
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: EJB-Project-archetype:0.1
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: AlmaGrp
[INFO] Parameter: artifactId, Value: Artif
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: AlmaGrp
[INFO] Parameter: packageInPathFormat, Value: AlmaGrp
[INFO] Parameter: package, Value: AlmaGrp
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: groupId, Value: AlmaGrp
[INFO] Parameter: artifactId, Value: Artif
[INFO] project created from Archetype in dir: /home/tanulas/maven/xxx/ttt/Artif
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

További lehetőségek

Az archetype projekt forrásának *src/main/resources/META-INF/maven/archetype-metadata.xml* fájlja leírja, hogy a projekt milyen fájlokat és tegyen bele a legenerálandó projektbe, illetve milyen *velocity* template szövegdarabkákat oldjon fel (makrohelyettesítés) azokban. Példa egy *archetype-metadata.xml* fájlra:

```
<archetype-descriptor name="maven_archetype_demo">
   <fileSets>
      <fileSet filtered="true">
          <directory></directory>
          <includes>
              <include>pom.xml</include>
          </includes>
      </fileSet>
      <fileSet filtered="true">
          <directory>src/main/java/__packageInPathFormat__</directory>
```

```
        </fileSet>
        <fileSet filtered="true">
            <directory>src/test/java/__packageInPathFormat__</directory>
        </fileSet>
        <fileSet filtered="true">
            <directory>src/main/resources</directory>
        </fileSet>
        <fileSet filtered="true">
            <directory>src/test/resources</directory>
        </fileSet>
    </fileSets>
</archetype-descriptor>
```

Az archetype-ból generált projekt tartalmazni fogja az *archetype-resources* mappában definiált struktúrát. A legkülső pom.xml az az archetype projekt pom-ja. Az *archetype-metadata.xml* az archetype leíró fájlja, s itt kell meghatározni az archetype-hoz tartozó metainformációkat, például, hogy hol találhatók azok a fájlok, melyeknek a projektek létrehozásánál létre kell jönniük. Először a pom fájl helyét határoztuk meg, majd a java fájlok és a resource fájlok helyét adtuk meg. A *__packageInPathFormat__* egy változó, melynek helyére a projekt létrehozásakor megadott package név fog bekerülni. Az archetype által definiált projekthez tartozó pom.xml, amely majd a generált projekt *pom.xml*-je lesz, jelenleg így néz ki:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:➥
    schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
    <artifactId>TestAppArtifact</artifactId>
    <groupId>TestAppGroup</groupId>
    <version>0.1</version>
  </parent>

    <groupId>${groupId}</groupId>
    <artifactId>${artifactId}</artifactId>
    <version>${version}</version>
    <packaging>ejb</packaging>

    <name>${artifactId}</name>

    <properties>
        <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-api</artifactId>
            <version>6.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.2</version>
                <configuration>
                    <source>1.6</source>
                    <target>1.6</target>
                    <compilerArguments>
                        <endorseddirs>${endorsed.dir}</endorseddirs>
                    </compilerArguments>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-ejb-plugin</artifactId>
                <version>2.3</version>
                <configuration>
                    <ejbVersion>3.1</ejbVersion>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <version>2.1</version>
                <executions>
```

```
                    <execution>
                        <phase>validate</phase>
                        <goals>
                            <goal>copy</goal>
                        </goals>
                        <configuration>
                            <outputDirectory>${endorsed.dir}</outputDirectory>
                            <silent>true</silent>
                            <artifactItems>
                                <artifactItem>
                                    <groupId>javax</groupId>
                                    <artifactId>javaee-endorsed-api</artifactId>
                                    <version>6.0</version>
                                    <type>jar</type>
                                </artifactItem>
                            </artifactItems>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

A pom fájlba betehetjük azokat a dependenciákat, plugin-eket, profilokat, melyekről úgy ítéljük meg, hogy szükségesek lesznek a generált projektünk számára. Az archetype fájljai gyakorlatilag velocity template-ként kezelhetők, emiatt változókat is rakhatunk a pom fájlba *${...}* formátumban, melyek helyére a projekt létrehozásakor megadott értékek fognak kerülni. A fenti példában a *${groupId}*, *${artifactId}* és a *${version}* ilyen változók. A java fájlokba is rakhatunk velocity változókat *${...}* formátumban. Léteznek előre definiált változók, mint például a minta java fájlban szereplő *${package}* változó. A java fájl így néz ki:

```
package ${package};
  public class MavenDemo {
    public static void main(String[] args) {
    }
  }
```

Fontos lehetőség az új archetype változók bevezetésének lehetősége. Az *archetype-metadata.xml* fájlba beírhatjuk, hogy mely változók kötelezőek, ezt az XML fájlba így kell megadni, ahol a példánkban a változó neve most *dbHost*:

```
...
  <requiredProperties>
        <requiredProperty key="dbHost"/>
  </requiredProperties>
...
```

Ezen változó alapértelmezett értékét a *.../src/test/resources/projects/basic/archetype.properties* fájlba is be kell jegyezni:

```
#Sun Apr 03 17:34:44 CEST 2016
package=it.pkg
version=0.1-SNAPSHOT
groupId=archetype.it
artifactId=basic
dbHost=AAA
```

Ezek után például a template-be lévő *java* fájlokba beírhatjuk, hogy *${dbHost}* és amikor az archtype generálás folyik, akkor ennek az értékére is rákérdez a rendszer, a generált projektben pedig az itt megadott szöveget helyettesíti a létrejött projektbe.

A pom.xml fájl összeállításának gyakorlati elemei

A következőkben egy tipikus multimodulos maven projek felépítését és annak a gyakorlatban hasznos néhány belső elemét mutatjuk be.

A projekt pom fájl hasznos elemei

A parent pom.xml fájlba érdemes betenni a lentiekben látott tag-eket is (*description*, *name*). A verzió egy property legyen.

```
...
        <modelVersion>4.0.0</modelVersion>
        <groupId>org.cs.test</groupId>
        <artifactId>MyProject</artifactId>
        <version>${app.version}</version>
        <packaging>pom</packaging>
        <name>MyApp</name>
        <description>Nem túl hosszú leírás a projektről</description>

        <properties>
                <app.version>v20160317-01</app.version>
        </properties>
...
```

Az is jó gyakorlat, ha a fejlesztők nevét és elérhetőségét is feltüntetjük:

```
...
<developers>
        <developer>
            <name>Imre Nyiri</name>
            <email>imre.nyiri@gmail.com</email>
            <organization>CreedSoft.org</organization>
            <roles>
                <role>Owner, CEO</role>
            </roles>
        </developer>
    </developers>
...
```

A modulok neveit - ahogy már láttuk - így kell feltüntetni:

```
...
        <modules>
                <module>MyProject_ear</module>
                <module>MyProject_ejb</module>
                <module>MyProject_war</module>
                <module>MyProject_ejbclient</module>
        </modules>
...
```

A parent pom.xml-be így kell felvenni a függőségeket, ezeket a gyerekek öröklik:

```
...
        <dependencyManagement>
                <dependencies>
                        <dependency>
                                <groupId>javaee</groupId>
                                <artifactId>javaee-api</artifactId>
                                <version>5</version>
                                <scope>provided</scope>
                        </dependency>
                        <dependency>
                                <groupId>log4j</groupId>
                                <artifactId>log4j</artifactId>
                                <version>1.2.17</version>
                        </dependency>
                        <dependency>
                                <groupId>org.jasypt</groupId>
                                <artifactId>jasypt</artifactId>
                                <version>1.9.2</version>
                                <scope>compile</scope>
                        </dependency>
                        <dependency>
                                <groupId>com.google.code.gson</groupId>
                                <artifactId>gson</artifactId>
                                <version>2.3.1</version>
                        </dependency>
                        <dependency>
                                <groupId>org.quartz-scheduler</groupId>
                                <artifactId>quartz</artifactId>
                                <version>2.1.7</version>
                        </dependency>
                        <dependency>
                                <groupId>org.apache.httpcomponents</groupId>
                                <artifactId>httpclient</artifactId>
                                <version>4.5.2</version>
```

```
                        </dependency>
                        <dependency>
                                <groupId>commons-io</groupId>
                                <artifactId>commons-io</artifactId>
                                <version>2.4</version>
                        </dependency>
                </dependencies>
        </dependencyManagement>
...
```

Szinte mindig elengedhetetlen, hogy a használt java forrás és bináris target verziót közvetlenül előírjuk:

```
...
        <build>
                <plugins>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source>1.6</source>
                                        <target>1.6</target>
                                        <encoding>UTF-8</encoding>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
...
```

Az EAR fájlok előállításánál fontos lehet annak működését szabályozni. Itt a példa azt mutatja meg, hogy a manifest fájlba bekerül egy sor, ami a *Weblogic-Application-Version* property és annak értéke.

```
...
        <build>
                <plugins>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-ear-plugin</artifactId>
                                <version>2.10</version>
                                <configuration>
                                        <defaultLibBundleDir>lib/</defaultLibBundleDir>
                                        <archive>
                                                <manifestEntries>
                                                        <Weblogic-Application-Version>${project.version}</➥
                                                            Weblogic-Application-Version>
                                                </manifestEntries>
                                        </archive>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
...
```

Természetesen az ejb plugin is konfigurálható, ha szükséges:

```
...
        <build>
                <plugins>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-ejb-plugin</artifactId>
                                <version>2.5</version>
                                <configuration>
                                        <ejbVersion>3.0</ejbVersion>
                                        <archive>
                                                <manifest>
                                                        <addClasspath>true</addClasspath>
                                                        <classpathPrefix>lib/</classpathPrefix>
                                                </manifest>
                                        </archive>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
...
```

Végül a web alkalmazások build-elését végző war plugin konfigurálása is megoldható:

```
...
        <build>
                <plugins>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-war-plugin</artifactId>
                                <version>2.6</version>
                                <!--<configuration> <failOnMissingWebXml>false</failOnMissingWebXml>
                                        </configuration> -->
                        </plugin>
                </plugins>
        </build>
...
```

Az EAR modul pom fájl hasznos beállításai

Az alábbi példa egy jól átgondolt EAR modul konfigurálását mutatja, természetesen azok az elemek találhatóak meg benne függőségként, amiket az EAR-hoz használnunk kell. Vegyük észre, hogy a war projekt típusa *war*, míg az ejb projekté *ejb*!

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:➥
    schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.cs.test</groupId>
                <artifactId>MyProject</artifactId>
                <version>${app.version}</version>
        </parent>

        <artifactId>MyProject_ear</artifactId>
        <packaging>ear</packaging>
        <name>MyProject_EAR</name>

        <dependencies>
                <dependency>
                        <groupId>org.cs.test</groupId>
                        <artifactId>MyProject_ejb</artifactId>
                        <version>${project.version}</version>
                        <type>ejb</type>
                </dependency>

                <dependency>
                        <groupId>org.cs.test</groupId>
                        <artifactId>MyProject_war</artifactId>
                        <version>${project.version}</version>
                        <type>war</type>
                </dependency>

                <dependency>
                        <groupId>org.cs.test</groupId>
                        <artifactId>MyProject_ejbclient</artifactId>
                        <version>${project.version}</version>
                </dependency>

                <dependency>
                        <groupId>log4j</groupId>
                        <artifactId>log4j</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.jasypt</groupId>
                        <artifactId>jasypt</artifactId>
                </dependency>

                <dependency>
                        <groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.quartz-scheduler</groupId>
                        <artifactId>quartz</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.apache.httpcomponents</groupId>
                        <artifactId>httpclient</artifactId>
                </dependency>

                <dependency>
```

```
                        <groupId>commons-io</groupId>
                        <artifactId>commons-io</artifactId>
                </dependency>

        </dependencies>
</project>
```

Az EJB modul pom fájl hasznos beállításai

Egy multi modulos maven projekt tipikus ejb moduljának a *pom.xml* fájl-ja valahogy így néz ki:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:➥
    schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.cs.test</groupId>
                <artifactId>MyProject</artifactId>
                <version>${app.version}</version>
        </parent>
        <artifactId>MyProject_ejb</artifactId>
        <packaging>ejb</packaging>
        <name>MyProject_EJB</name>

        <dependencies>
                <dependency>
                        <groupId>log4j</groupId>
                        <artifactId>log4j</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.cs.test</groupId>
                        <artifactId>MyProject_ejbclient</artifactId>
                        <version>${project.version}</version>
                </dependency>

                <dependency>
                        <groupId>org.jasypt</groupId>
                        <artifactId>jasypt</artifactId>
                </dependency>

                <dependency>
                        <groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                </dependency>

                <dependency>
                        <groupId>javaee</groupId>
                        <artifactId>javaee-api</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.apache.httpcomponents</groupId>
                        <artifactId>httpclient</artifactId>
                </dependency>

                <dependency>
                        <groupId>commons-io</groupId>
                        <artifactId>commons-io</artifactId>
                </dependency>
        </dependencies>
</project>
```

A WAR (web) modul pom fájl hasznos beállításai

Egy multi modulos maven projekt tipikus war moduljának a *pom.xml* fájl-ja valahogy így néz ki:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.cs.test</groupId>
                <artifactId>MyProject</artifactId>
                <version>${app.version}</version>
        </parent>
        <artifactId>MyProject_war</artifactId>
        <packaging>war</packaging>
        <name>MyProject_WAR</name>
```

```
        <dependencies>
                <dependency>
                        <groupId>org.cs.test</groupId>
                        <artifactId>MyProject_ejbclient</artifactId>
                        <version>${project.version}</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.quartz-scheduler</groupId>
                        <artifactId>quartz</artifactId>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>javaee</groupId>
                        <artifactId>javaee-api</artifactId>
                </dependency>
                <dependency>
                        <groupId>log4j</groupId>
                        <artifactId>log4j</artifactId>
                        <scope>provided</scope>
                </dependency>



        </dependencies>
</project>
```

A fentiekben érdemes elgondolkodnunk azon, hogy a scope-ok beállításai miért olyanok, amiket látunk.

Az EJB kliens modul pom fájl hasznos beállításai

A teljesség kedvéért itt egy ejb kliens modul-t is bemutatunk, például a war modulból használhatjuk:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:➥
    schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.cs.test</groupId>
                <artifactId>MyProject</artifactId>
                <version>${app.version}</version>
        </parent>
        <artifactId>bci_ejbclient</artifactId>
        <name>MyProject_EJBClient</name>

        <dependencies>
                <dependency>
                        <groupId>log4j</groupId>
                        <artifactId>log4j</artifactId>
                </dependency>

                <dependency>
                        <groupId>javaee</groupId>
                        <artifactId>javaee-api</artifactId>
                </dependency>
        </dependencies>
</project>
```

Alap Vaadin projekt

Az újabb tapasztalatszerzés érdekében most egy olyan multi modulos projektet szeretnénk megmutatni, aminek a web projektje a Vaadin technológiára épül. Semmi új nincs benne, célunk az volt, hogy a pom.xml-ek áttekintésével további tapasztalatokat szerezzünk. Íme a parent pom:

3-x. Programlista: Vaadin - Fő pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>hu.mol.eai.mosec</groupId>
    <artifactId>mosec</artifactId>
    <version>1.0</version>
    <packaging>pom</packaging>

    <properties>
        <project.encoding>UTF-8</project.encoding>
        <project.source.version>1.6</project.source.version>
        <project.target.version>1.6</project.target.version>
    </properties>

    <description>MOL Sales eContracting Web Application</description>

    <developers>
        <developer>
            <name>Arnold Somogyi</name>
            <email>arnold.somogyi@gmail.com</email>
            <organization>Enterprise Application Integration Team, MOL Plc.</organization>
            <roles>
                <role>IT System Integration Expert</role>
            </roles>
        </developer>
    </developers>

    <modules>
        <module>mosec-commons</module>
        <module>mosec-restapi</module>
        <module>mosec-webui</module>
    </modules>
</project>
```

Ez egy közös komponens lesz az ear-ban:

3-x. Programlista: Vaadin - Common pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>mosec</artifactId>
        <groupId>hu.mol.eai.mosec</groupId>
        <version>1.0</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <artifactId>mosec-commons</artifactId>
    <packaging>jar</packaging>

    <build>
        <plugins>
            <!-- sets java version -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <encoding>${project.encoding}</encoding>
                    <source>${project.source.version}</source>
                    <target>${project.target.version}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <!-- apache commons lang 3 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.4</version>
        </dependency>
        <!-- json marshalling/unmarshalling -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-annotations</artifactId>
            <version>2.6.2</version>
        </dependency>
        <!-- generates random data -->
        <dependency>
            <groupId>org.fluttercode.datafactory</groupId>
```

```
            <artifactId>datafactory</artifactId>
            <version>0.8</version>
        </dependency>
    </dependencies>
</project>
```

A restful szervizek module projektje:

3-x. Programlista: Vaadin - Restful project

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>mosec</artifactId>
        <groupId>hu.mol.eai.mosec</groupId>
        <version>1.0</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <artifactId>mosec-restapi</artifactId>
    <packaging>war</packaging>

    <dependencies>
        <!-- mosec-commons.jar -->
        <dependency>
            <groupId>hu.mol.eai.mosec</groupId>
            <artifactId>mosec-commons</artifactId>
            <version>1.0</version>
        </dependency>
        <!-- json marshalling/unmarshalling -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.6.2</version>
        </dependency>
        <!-- dependency for log4j, provided by application server  -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
        <!-- rest-api is created with spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>4.2.1.RELEASE</version>
        </dependency>
        <!-- for java ee api, provided by application server -->
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-api</artifactId>
            <version>7.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- sets java version -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <encoding>${project.encoding}</encoding>
                    <source>${project.source.version}</source>
                    <target>${project.target.version}</target>
                </configuration>
            </plugin>
            <!-- eliminates the 'web.xml file is missing' error message -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>2.5</version>
                <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Végül maga a Vaadin projekt:

3-x. Programlista: Vaadin - Web UI project

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>mosec</artifactId>
        <groupId>hu.mol.eai.mosec</groupId>
        <version>1.0</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <artifactId>mosec-webui</artifactId>
    <packaging>war</packaging>

    <properties>
        <vaadin.version>7.5.5</vaadin.version>
        <jetty.version>9.2.3.v20140905</jetty.version>
    </properties>

    <dependencies>
        <!-- mosec-commons.jar -->
        <dependency>
            <groupId>hu.mol.eai.mosec</groupId>
            <artifactId>mosec-commons</artifactId>
            <version>1.0</version>
        </dependency>
        <!-- vaadin -->
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-server</artifactId>
            <version>${vaadin.version}</version>
        </dependency>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-push</artifactId>
            <version>${vaadin.version}</version>
        </dependency>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-client</artifactId>
            <version>${vaadin.version}</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>com.vaadin</groupId>
            <artifactId>vaadin-themes</artifactId>
            <version>${vaadin.version}</version>
        </dependency>
        <!-- http request/response handler library -->
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.1</version>
        </dependency>
        <!-- apache commons lang 3 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.4</version>
        </dependency>
        <!-- json marshalling/unmarshalling -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.6.2</version>
        </dependency>
        <!-- dependency for log4j, provided by application server  -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
        <!-- servlet api -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
```

```
    <build>
        <plugins>
            <!-- sets java version -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <encoding>${project.encoding}</encoding>
                    <source>${project.source.version}</source>
                    <target>${project.target.version}</target>
                </configuration>
            </plugin>
            <!-- eliminates the 'web.xml file is missing' error message -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>2.6</version>
                <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                    <packagingExcludes>WEB-INF/classes/VAADIN/gwt-unitCache/**,WEB-INF/classes/VAADIN/widgetsets➥
                        /WEB-INF/**</packagingExcludes>
                </configuration>
            </plugin>
            <!-- plugin for vaadin -->
            <plugin>
                <groupId>com.vaadin</groupId>
                <artifactId>vaadin-maven-plugin</artifactId>
                <version>${vaadin.version}</version>
                <configuration>
                    <extraJvmArgs>-Xmx512M -Xss1024k</extraJvmArgs>
                    <webappDirectory>${basedir}/target/classes/VAADIN/widgetsets</webappDirectory>
                    <draftCompile>false</draftCompile>
                    <compileReport>false</compileReport>
                    <style>OBF</style>
                    <strict>true</strict>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>clean</goal>
                            <goal>resources</goal>
                            <goal>update-theme</goal>
                            <goal>update-widgetset</goal>
                            <goal>compile-theme</goal>
                            <goal>compile</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <!-- clean up any pre-compiled themes -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-clean-plugin</artifactId>
                <version>2.6.1</version>
                <configuration>
                    <filesets>
                        <fileset>
                            <directory>src/main/webapp/VAADIN/themes</directory>
                            <includes>
                                <include>**/styles.css</include>
                                <include>**/styles.scss.cache</include>
                            </includes>
                        </fileset>
                    </filesets>
                </configuration>
            </plugin>
            <!-- jetty plugin allows us to easily test the development build by running jetty:run -->
            <plugin>
                <groupId>org.eclipse.jetty</groupId>
                <artifactId>jetty-maven-plugin</artifactId>
                <version>${jetty.version}</version>
                <configuration>
                    <scanIntervalSeconds>1</scanIntervalSeconds>
                    <webApp>
                        <extraClasspath>${basedir}/../mosec-commons/target/mosec-commons-1.0.jar</extraClasspath➥
                        >
                    </webApp>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Maven használata Weblogic környezetben

A támogatott lehetőségek

A weblogic maven támogatás a következő lehetőségekkel rendelkezik, amelyek *groupId* és *artifactId* értékeit zárójelben adtuk meg:

- WebLogic Server plug-in (*com.oracle.weblogic*, *weblogic-maven-plugin*)
- Basic WebApp archetype (*com.oracle.weblogic.archetype*, *basic-webapp*)
- WebApp with EJB archetype (*com.oracle.weblogic.archetype*, *basic-webapp-ejb*)
- Basic MDB archetype (*com.oracle.weblogic.archetype*, *basic-mdb*)
- Basic WebServices archetype (*com.oracle.weblogic.archetype*, *basic-webservice*)

Az első pont egy általános weblogic szerver kezelő plugin, míg a többi 4 különféle projektek generálását támogató maven archetype.

Az archetype támogatás telepítése

Az *.../Oracle12c/wlserver/server/lib* könyvtárban megtalálhatóak a *wls-maven-plugin.jar* és *pom.xml* fájlok. Ezeket telepítsük a maven local repository-na:

```
mvn install -Dfile=wls-maven-plugin.jar -DpomFile=pom.xml
```

Menjünk ebbe a könyvtárba:

```
.../Oracle12c/oracle_common/plugins/maven/com/oracle/maven/oracle-maven-sync/12.1.3
```

Tegyük a lokális maven repository-ba az itt található 2 fájlt ezzel a paranccsal:

```
mvn install:install-file -DpomFile=oracle-maven-sync.12.1.3.pom -Dfile=oracle-maven-sync.12.1.3.jar
```

A *push* parancs további pluginokat telepít vel, ami segíti a még kényelmesebb munkát:

```
mvn com.oracle.maven:oracle-maven-sync:push -Doracle-maven-sync.oracleHome=/opt/Oracle/Middleware/Oracle_Home/➥
    wlserver -Doracle-maven-sync.testingOnly=false
```

Az eredményt ide helyezi el, ami után használhatóvá válnak az weblogic archetype-ok:

```
$HOME/.m2/repository/com/oracle
```

A különféle weblogic projektek generálása

Egy weblogic induló web projektet a következő maven paranccsal tudunk legenerálni:

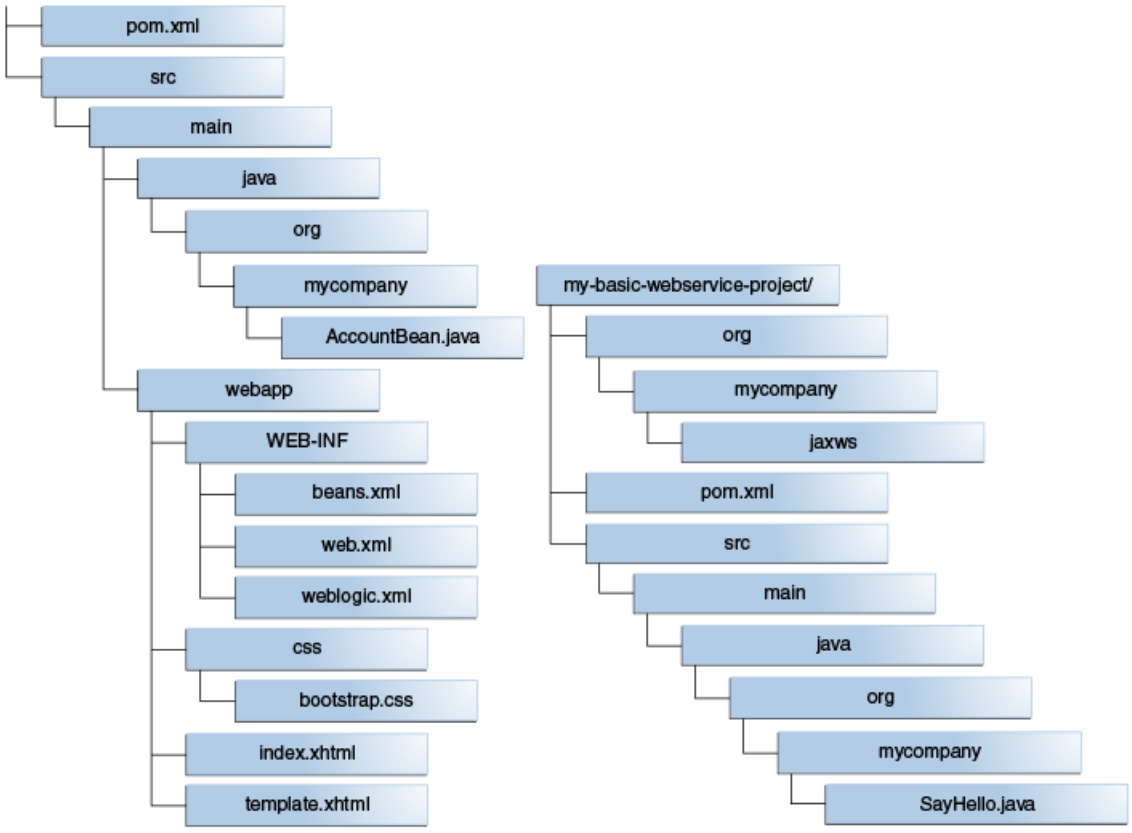
```
mvn archetype:generate -DarchetypeGroupId=com.oracle.weblogic.archetype -DarchetypeArtifactId=basic-webapp -➥
    DarchetypeVersion=12.1.3-0-0
 -DgroupId=org.cs.test -DartifactId=test-webapp-project -Dversion=1.0-SNAPSHOT
```

A paraméterek magyarázata:

- *archetypeGroupId*: A project generáláshoz használt archetype plugin group ID-ja, ami kötelezően *com.oracle.weblogic*

- *archetypeArtifactId*: A project generáláshoz használt archetype plugin artifact ID-ja, ami kötelezően *basic-webapp*
- *archetypeVersion*: A project generáláshoz használt archetype plugin verziója
- *groupId*: A generált projektünk group ID-ja
- *artifactId*: A generált projektünk artifact ID-ja
- *version*: A A generált projektünk induló verziószáma

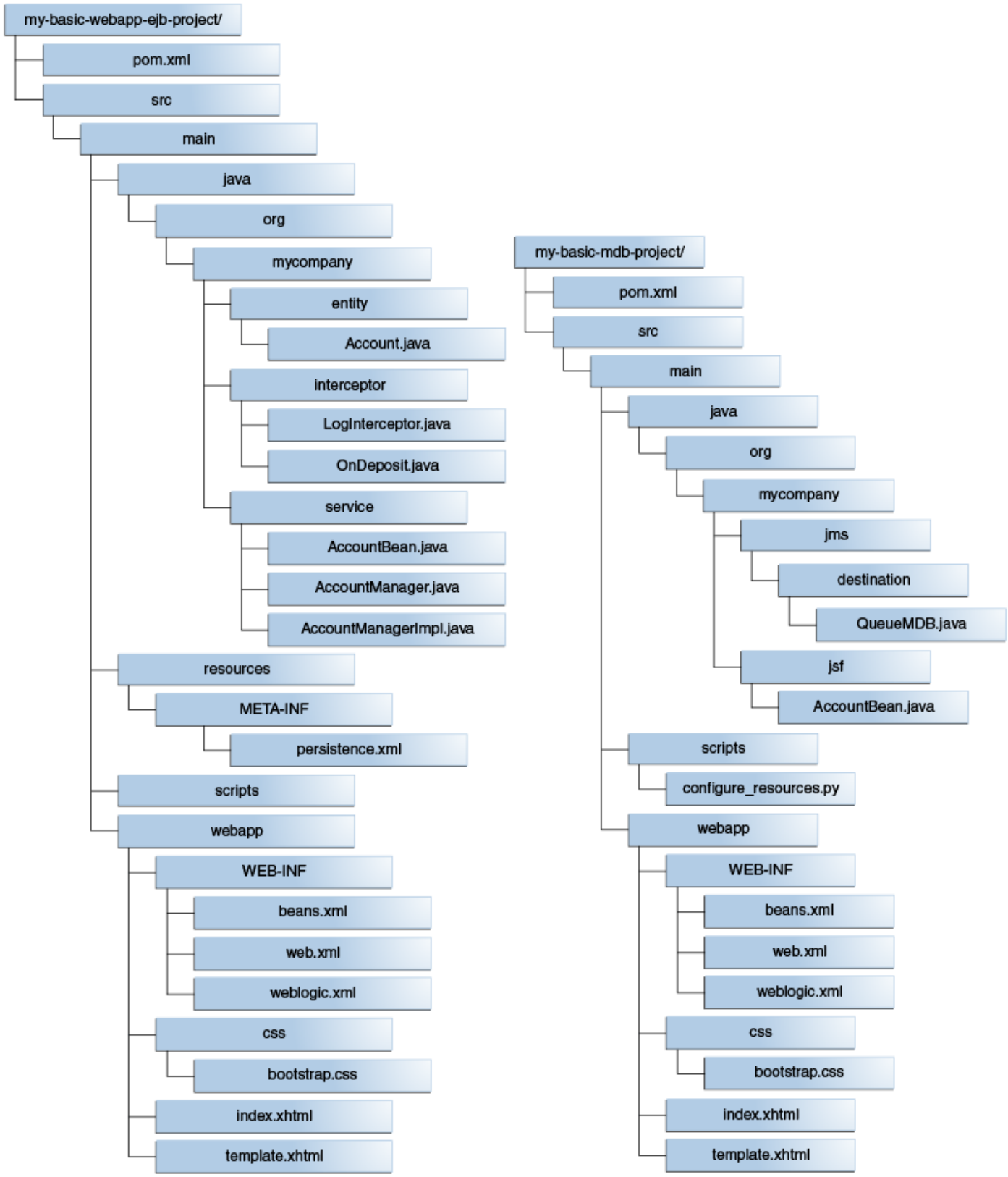
A projekt felépítését a 3.1. ábra mutatja.



3.1. ábra. Weblogic WEB és webservice projectek felépítése

Egy web és ejb lehetőséggel rendelkező projekt generálása:

```
mvn archetype:generate -DarchetypeGroupId=com.oracle.weblogic.archetype -DarchetypeArtifactId=basic-webapp-ejb -➥
    DarchetypeVersion=12.1.3-0-0
-DgroupId=org.cs.test -DartifactId=my-basic-webapp-ejb-project -Dversion=1.0-SNAPSHOT
```



3.2. ábra. Weblogic EJB és MDB projectek felépítése

Webservice projekt:

```
mvn archetype:generate
    -DarchetypeGroupId=com.oracle.weblogic.archetype
    -DarchetypeArtifactId=basic-webservice
    -DarchetypeVersion=12.1.3-0-0
    -DgroupId=org.mycompany
    -DartifactId=my-basic-webservice-project
    -Dversion=1.0-SNAPSHOT
```

A webservice és MDB projekt felépítését a 3.2. ábra mutatja.
Message Driven Bean projekt:

```
mvn archetype:generate
    -DarchetypeGroupId=com.oracle.weblogic.archetype
    -DarchetypeArtifactId=basic-mdb
    -DarchetypeVersion=12.1.3-0-0
    -DgroupId=org.mycompany
    -DartifactId=my-basic-mdb-project
    -Dversion=1.0-SNAPSHOT
```

Fordítás, csomagolás és telepítés

A weblogic maven project a szokásos *mvn compile* paranccsal fordítható, illetve a csomagoláshoz az *mvn package* használható. A teszteléshez a *mvn pre-integration-test* utasítással tudjuk feltenni az alkalmazást a lokális weblogic szerverünkre. Az alkalmazásunkat (ha nem írtunk át semmit a konfigurációs XML-ekben) a *http://servername:7001/basicWebapp/index.xhtml* címen tudjuk tesztelni.

A weblogic maven plugin telepítése és használata

A weblogic 10.3 verzióig csak ez a plugin létezett, ezzel a szervert lehetett kezelni. A WebLogic JarBuilder Tool segítségével a *MW_HOME/wlserver_10.3/server/lib/* könyvtárban állva adjuk ki a következő parancsot:

```
java -jar wljarbuilder.jar -profile weblogic-maven-plugin
```

A létrejött *weblogic-maven-plugin.jar* segítségével használható maven koordináták:

```
groupId=com.oracle.weblogic
artifactId=weblogic-maven-plugin
version=10.3.4
packaging=maven-plugin
```

Szedjük ki a *pom.xml* fájlt a *weblogic-maven-plugin.jar* fájlból:

```
jar xvf MW_HOME/wlserver_10.3/server/lib/weblogic-maven-plugin.jar META-INF/maven/com.oracle.weblogic/weblogic-➥
    maven-plugin/pom.xml
```

Utána a *pom.xml* fájlt másoljuk be a *.../server/lib* folder alá:

```
cp MW_HOME/wlserver_10.3/server/lib/META-INF/maven/com.oracle.weblogic/weblogic-maven-plugin/pom.xml MW_HOME/➥
    wlserver_10.3/server/lib
```

Végül telepítsük a plugint a local repository-ba:

```
mvn install:install-file -Dfile=MW_HOME/wlserver_10.3/server/lib/weblogic-maven-plugin.jar -DpomFile=pom.xml
```

A maven *~/.m2/settings.xml* fájlba írjuk be ezeket a sorokat:

```
<pluginGroups>
    <pluginGroup>com.oracle.weblogic</pluginGroup>
</pluginGroups>
```

A plugin feltelepítése után például így deploy-álhatunk egy *sample.war* alkalmazást:

```
mvn com.oracle.weblogic:weblogic-maven-plugin:deploy -Dadminurl=t3://myhost:7001 -Duser=weblogic
 -Dpassword=mypassword -Dtargets=AdminServer -Dsource=/home/deploy-pool/sample.war
```