JPA Example

Hozzunk létre egy dinamikus web projektet JPA néven.

1.) Hozzuk létre a persistence.xml nevű file-t a src/META-INF könyvtárban a következő tartalommal:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
     <persistence-unit name="customers">
     <jta-data-source>java:jboss/datasources/ExampleDS</jta-data-source>
        <properties>
            <property name="hibernate.hbm2ddl.auto" value="create-drop" />
            <property name="showSql" value="true"/>
            <property name="hibernate.dialect"
                    value="org.hibernate.dialect.MySQLDialect" />
        </properties>
     </persistence-unit>
</persistence>
```

2.) Készítsük el az org.me.CustomerManager class-t:

```
  package org.me;
  import java.util.List;
  import javax.ejb.EJB;
  import javax.ejb.Stateful;
  import javax.ejb.Stateless;
  import javax.persistence.EntityManager;
  import javax.persistence.PersistenceContext;
  import javax.persistence.Query;
  @Stateless
  public class CustomerManager {
    @PersistenceContext(unitName = "customers")
    EntityManager em;
    public void createCustomer(String name, String address) {
        Customer customer = new Customer(name, address);
            em.persist(customer);
    }
    public List<Customer> listCustomers() {
        String q = "SELECT b from " + Customer.class.getName() + " b";
        Query query = em.createQuery(q);
        return query.getResultList();
    }
  }
```

3. Készítsük el a class org.me.Customer class-t

```
  package org.me;
```

```
  import java.io.Serializable;
  import javax.persistence.Entity;
  import javax.persistence.Id;
  @Entity(name = "customer")
  public class Customer implements Serializable {
    @Id
    private String name;
    private String address;
    public Customer() {
    }
    public Customer(String name, String address) {
        this.name = name;
        this.address = address;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
  }
```

4.) Készítsünk egy org.me.MyServlet class-t.

```
  package org.me;
  import java.io.IOException;
  import java.io.PrintWriter;
  import java.util.List;
  import javax.ejb.EJB;
  import javax.servlet.ServletException;
  import javax.servlet.annotation.WebServlet;
  import javax.servlet.http.HttpServlet;
  import javax.servlet.http.HttpServletRequest;
  import javax.servlet.http.HttpServletResponse;
  @WebServlet("/MyServlet")
  public class MyServlet extends HttpServlet {
      @EJB
      private CustomerManager cm;
      public MyServlet() {
          super();
      }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String method = request.getParameter("method");
```

```
        String name = request.getParameter("name");
        String address = request.getParameter("address");
        PrintWriter writer = response.getWriter();
            writer.print("<html><body>");
        if(method.equalsIgnoreCase("create")) {
            cm.createCustomer(name, address);
            writer.print("result = OK, saved");
        } else if(method.equalsIgnoreCase("list")) {
            List<Customer> persons = cm.listCustomers();
            for (Customer customer : persons) {
                writer.print("customer.name=" + customer.getName() + "
address=" + customer.getAddress() + "</br>");
            }
        }
            writer.print("</body> </html>");
        writer.flush();
        writer.close();
    }
  }
```

5.) Deployment után a kipróbálásként hozzunk létre 3 customer példányt.

http://localhost:8080/JPA/MyServlet?method=create&name=john1&address=miskolc

http://localhost:8080/JPA/MyServlet?method=create&name=john2&address=miskolc

http://localhost:8080/JPA/MyServlet?method=create&name=john3&address=miskolc

Mentett elemek listázása:

http://localhost:8080/JPA/MyServlet?method=list&name=john1&address=miskolc

Gyakorlatok

1. Töltsük le a DB nézegető eszközt: h2console.zip. Tömörítsük ki ide: ..\jbdevstudio\runtimes\jboss-eap\standalone\deployments\ és futtassuk: http://localhost:8080/h2console/
    - Connection string: jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
    - User/Pass: sa/sa

1. Készítsünk egy html form-ot (index.html) a customer-ek manipulálására

egy lehetséges megoldás:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
```

```
<body>
    <form action="MyServlet">
     Name: <input type="text" name="name"> <br>
     Address: <input type="text" name="address"> <br>
     <input type="submit" name="method" value="create">
     <input type="submit" name="method" value="list">
     <input type="submit" name="method" value="delete">
    </form>
</body>
</html>
```

1. Készítsünk egy megoldást a mentett customerek törlésére. A lekérdező nyelv leírása itt található: http://docs.oracle.com/javaee/6/tutorial/doc/bnbtl.html#bnbtm

Egy lehetséges megoldás:

```
    public int delete(String name) {
      String q = "DELETE from " + Customer.class.getName()
            + " b WHERE b.name = '" + name + "'";
      Query query = em.createQuery(q);
      return query.executeUpdate();
    }
```