Budapesti Műszaki és Gazdaságtudományi Egyetem
Tárgynév (VIMIM147)

JPA, OSGi, REST

Laboranyag

Czotter Márk (U4F0H0)
2012. március 11.

1 Előkészítés

- Mit is csinálunk, sorrend, adatmodell felvázolása
- Projekt letöltése a webről (már fent lesz a gépeken)
- Importálás Eclipse-be
- Target platform magyarázat, Set as Target Platform
- Új Plug-in projekt létrehozása, OSGi Framework kell, dependenciák beállítása
- Jersey server/core/servlet
- org.eclipse.persistence.jpa,
- javax.persistence,
- jaxrs.connector
- org.apache.derby

2 JPA

- Book POJO létrehozása
- id, title, price felvétele
- Generate getter, setter
- Entity annotáció
- Id, Column, GeneratedValue annotáció
- Book (owners a ManyToMany kapcsolatnál elég):

```
@Entity
@NamedQuery(name=Book.GET_BOOKS, query="SELECT b FROM Book b")
public class Book {

	public static final String GET_BOOKS = "GetBooks";

	@Id
	@Column(name = "BOOK_ID")
	@GeneratedValue
	private long id;

	private String title;

	private int price;

	@ManyToMany(mappedBy="ownedBooks")
	private List<User> owners = new ArrayList<User>();

	// getters, setters
}
```

- User POJO létrehozása
- Username felvétele
- Getter, setter
- Entity és Table annotáció
- Id annotáció
- User:

```
@Entity
```

```
@Table(name="T_USER")
public class User {

	@Id
	private String username;

	@ManyToMany
	@JoinTable(name="USER_BOOKS", joinColumns=@JoinColumn(name="USERNAME"),
inverseJoinColumns=@JoinColumn(name="BOOK_ID"))
	private List<Book> ownedBooks = new ArrayList<Book>();

	// getters, setters

}
```

- Kapcsolat felvétele, ManyToMany kapcsolat
- JoinTable felvétele

Persistence xml felvétele az src/META-INF mappába:
FONTOS: átírni az entitások FQN-jét.

```
<?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="osgi.demo.jpa" transaction-type="RESOURCE_LOCAL">
        <class>hu.bme.mit.szolgint.osgi.demo.entities.Book</class>
        <class>hu.bme.mit.szolgint.osgi.demo.entities.User</class>
        <properties>
            <!-- Embedded Derby Login -->
            <property name="javax.persistence.jdbc.driver"
value="org.apache.derby.jdbc.EmbeddedDriver"/>
            <property name="javax.persistence.jdbc.url"
value="jdbc:derby:books;create=true"/>
            <!-- don't need userid/password in embedded Derby -->
            <property name="eclipselink.target-database" value="Derby"/>
            <property name="eclipselink.jdbc.read-connections.min" value="1"/>
            <property name="eclipselink.jdbc.write-connections.min" value="1"/>
            <property name="eclipselink.jdbc.batch-writing" value="JDBC"/>

            <!-- Database Schema Creation -->
            <property name="eclipselink.ddl-generation" value="drop-and-create-
tables"/>
            <property name="eclipselink.ddl-generation.output-mode"
value="database"/>

            <!-- Logging Settings -->
            <property name="eclipselink.logging.level" value="FINE" />
            <property name="eclipselink.logging.thread" value="false" />
            <property name="eclipselink.logging.session" value="false" />
            <property name="eclipselink.logging.exceptions" value="true" />
            <property name="eclipselink.logging.timestamp" value="false"/>
        </properties>
    </persistence-unit>
</persistence>
```

- MANIFEST.MF fájlba beleírni a következőt:
- JPA-PersistenceUnits: osgi.demo.jpa

3 OSGi

- BookManager osztály létrehozása
- createBook és getBooks metódusok felvétele:

```
public Book createBook(String title, int price, long authorId) {
	EntityManagerFactory entityManagerFactory = Persistence
			.createEntityManagerFactory("osgi.demo.jpa");
	EntityManager em = entityManagerFactory.createEntityManager();
	em.getTransaction().begin();
	Book book = new Book();
	book.setTitle(title);
	book.setPrice(price);
	em.persist(book);
	em.getTransaction().commit();
	em.close();
	return book;
}

public Collection<Book> getBooks() {
	EntityManagerFactory entityManagerFactory = Persistence
			.createEntityManagerFactory("osgi.demo.jpa");
	EntityManager em = entityManagerFactory.createEntityManager();
	List results = em.createNamedQuery(Book.GET_BOOKS).getResultList();
	em.close();
	return results;
}
```

- UserManager:
- register metódus

```
@Override
public void register(String username) {
	if (username == null || username.isEmpty()) {
		throw new IllegalArgumentException("Invalid username!");
	}
	EntityManagerFactory entityManagerFactory = Persistence
			.createEntityManagerFactory("osgi.demo.jpa");
	EntityManager em = entityManagerFactory.createEntityManager();
	em.getTransaction().begin();
	User user = new User();
	user.setUsername(username);
	em.persist(user);
	em.getTransaction().commit();
	em.close();
}
```

4 REST

- IBookStoreService interfész felvétele
- String addBook(String title, int price)

- String registerUser(String username)
- String listBooks();
- BookStoreService osztály felvétele, implements IBookStoreService
- Metódusok hozzáadása

```
@Path("/bookstore")
public class BookStoreRestService implements IBookStoreService {

	private IBookManager bookManager = new BookManager();
	private IUserManager userManager = new UserManager();

	@PUT
	@Path("/books/{title}/{price}/{authorId}")
	@Produces("text/plain")
	@Override
	public String addBook(@PathParam("title") String title,
			@PathParam("price") int price, @PathParam("authorId") long
authorId) {
		if (bookManager == null) {
			return "Not connected to BookManager!";
		}
		try {
			bookManager.createBook(title, price, authorId);
		} catch (Exception e) {
			e.printStackTrace();
			return e.getMessage();
		}
		return "Successfully added a book to bookStore!";
	}

	@GET
	@Path("/books")
	@Produces("text/plain")
	@Override
	public String listBooks() {
		try {
			if (bookManager == null) {
				return "Not connected to bookstore!";
			}
			Collection<Book> books = this.bookManager.getBooks();
			if (books.isEmpty()) {
				return "No books in the bookstore!";
			}
			return printBooks(books);
		} catch (Exception e) {
			e.printStackTrace();
			return e.getMessage();
		}
	}

	@PUT
	@Path("/users/{username}")
	@Produces("text/plain")
	@Override
	public String register(@PathParam("username") String username) {
		if (userManager == null) {
			return "Not connected to UserManager!";
		}
```

```
		try {
			userManager.register(username);
		} catch (Exception e) {
			e.printStackTrace();
			return e.getMessage();
		}
		return "Registration was successful!";
	}

}
```

- New Component Definition hozzáadása
- component.xml név
- BookStoreService az osztály
- Provided Interfaces: IBookStoreService

5 Launch

- Run Configurations
- New OSGi RunConfig
- Workspace projektek hozzáadása, majd Add Required (érdemes kiszedni az Include Optional….)
- Ezután meggyőződni, hogy a következők bent vannak:
  - jaxrs.connector
  - com.sun.jersey.core/servlet/server
  - javax.persistence
  - javax.servlet
  - org.apache.derby
  - org.eclipse.equinox.ds
  - org.eclipse.equinox.http.jetty/servlet/registry
  - org.eclipse.osgi
  - org.eclipse.persistence.jpa/jpa.osgi/jpa.equinox
  - org.mortbay.jetty.server/util
  - Add required a végén is
  - 28 darab bundle kell
- Port átírása: **-Dorg.osgi.service.http.port=8080**
- **Fontos: MANIFEST.MF-ben a JPA PP neve, illetve az OSGi-INF-ben levő component.xml**

Kipróbálás: Simple REST Client a Chromeban, vagy REST Client a Firefoxban

6 Vásárlás megvalósítása

OSGi:

```
	public void buyBook(String username, long bookId) {
		EntityManagerFactory entityManagerFactory = Persistence
			.createEntityManagerFactory("osgi.demo.jpa");
		EntityManager em = entityManagerFactory.createEntityManager();
		em.getTransaction().begin();
		User user = em.find(User.class, username);
		if (user == null) {
			throw new RuntimeException("User with " + username + " not
registered!");
```

```
            }
            Book book = em.find(Book.class, bookId);
            if (book == null) {
                  throw new RuntimeException("Book with id " + bookId + " not
found in the BookStore!");
            }
            user.getOwnedBooks().add(book);
            book.getOwners().add(user);
            em.merge(user);
            em.merge(book);
            em.getTransaction().commit();
            em.close();
      }
```

REST:

```
      @POST
      @Path("/buy/{username}/{bookId}")
      @Produces("text/plain")
      @Override
      public String buyBook(@PathParam("username") String username,
@PathParam("bookId") long bookId) {
            if (bookManager == null) {
                  return "Not connected to bookstore!";
            }
            try {
                  this.bookManager.buyBook(username, bookId);
                  return "Order was successful!";
            } catch (Exception e) {
                  e.printStackTrace();
                  return e.getMessage();
            }
      }
```