Projekt warstwy integracji

 Autor: Zofia Kruczkiewicz
Design patterns used to build the Integration nad Resources Tiers
D. Alur, J. Crupi, D. Malks, Core J2EE. Design Patterns

Outline of creating the Library Catalogue Java Application

1. Create a database in the Derby database system
2. Creating a Persistence Unit
3. Create Annotations in the object model of the Business Tier
4. Create the following classes of the Integration Tier: TTitle_bookController and the TBookController
5. You may add annotation to your own new classes and create the proper controllers – for higher assessment (5.0 or 5.5)
1. Create a database in the Derby database system

1. Check a directory where you should create an empty database of the Derby database system by using the NetBeans - the default directory is a directory of Windows users \. Netbeans-derby – (slide 1)

2. Select the Services tab and then expand the Databases node. Then right-click on Java DB item and select Create Database – (slide 2)

3. In the form for creating a database (slide 3), enter the data as follows: Database Name, User, Password and the Database Location (default or selected by the user). In the selected directory will be created a directory named database with an empty database (slide 4)

4. Check a directory where you have created from NetBeans an empty database in the system database Derby - - you can see the new katalogue Library1 (slide 4)

5. To connect to the database: Right-click the database connection node and choose the Connect item (slide 6) - `jdbc:derby://localhost:1527/Library1:[Library1 on LIBRARY1] (jdbc:derby://localhost:1527/DataBase_Name:[User on DataBaseSchema] )`

6. Now expand the database connection node to browse the database until disconnect from the database. Then expand the database node to see the empty Library Database (slide 7) p – the Library1 node is lacking.
<table>
<thead>
<tr>
<th>Nazwa</th>
<th>Data modyfikacji</th>
<th>Typ</th>
</tr>
</thead>
<tbody>
<tr>
<td>katalog1</td>
<td>2008-04-14 18:17</td>
<td>Folder plików</td>
</tr>
<tr>
<td>sample</td>
<td>2008-04-14 13:18</td>
<td>Folder plików</td>
</tr>
<tr>
<td>sun-appserv-samples</td>
<td>2008-04-14 00:44</td>
<td>Folder plików</td>
</tr>
<tr>
<td>travel</td>
<td>2008-04-08 19:43</td>
<td>Folder plików</td>
</tr>
<tr>
<td>vir</td>
<td>2008-04-13 03:53</td>
<td>Folder plików</td>
</tr>
<tr>
<td>Zoo</td>
<td>2008-04-11 17:01</td>
<td>Folder plików</td>
</tr>
<tr>
<td>derby</td>
<td>2008-04-16 11:34</td>
<td>Dokument tekstowy</td>
</tr>
<tr>
<td>derby.properties</td>
<td>2008-03-25 14:13</td>
<td>Plik PROPERTIES</td>
</tr>
</tbody>
</table>
3)
5)
2. Creating a Persistence Unit

To create a persistence unit:
1. In the Projects window, right-click a Library1 project node and choose New > Other.
2. Select Persistence Unit in the Persistence category and click Next.
3. Specify a unique Persistence Unit Name. In most cases you can keep the default name suggested by the IDE.
4. Select the EclipseLink library by choosing New Persistence Library.
5. Select a datasource from the drop-down menu. The data source can be the Library1 database connection. To appear in the list, the data source needs to be registered with the IDE.
6. Specify a table generation strategy as Create for your database.
7. Click Finish. When you click Finish, the file persistence.xml opens in the Source Editor.
6)

**New Persistence Unit**

**Steps**

1. Choose File Type
2. Provider and Database

**Provider and Database**

- **Persistence Unit Name**: Library1PU2
- Specify the persistence provider and database for entity classes.
- Persistence Library: EclipseLink (JPA 2.0)
- Database Connection: jdbc:derby://localhost:1527/library1 [library1 on LIBRARY1]
- Table Generation Strategy: 
  - Create
  - Drop and Create
  - None
3. Create Annotations in the object model

Mapping Entity Classes (Help of the NetBeans)

- An entity class is used to represent a table in a database, and the fields in an entity class correspond to columns in that table. In an entity class, you can use annotations to specify how fields in an entity class are mapped to the corresponding database columns and tables.
- If the name of the mapped database column is the same as the field or property name because they are mapped by default.

The following annotations are commonly used when mapping entity classes:

- @Entity defines the Entity class
- @Id Specifies the primary key property or field of an entity.
- @GeneratedValue Allows you to specify the strategy that automatically generates the values of primary keys. Used with @Id.
- @Column Specifies a mapped column for a persistent property or field.
- @ManyToMany Defines a many-valued association with many-to-many multiplicity.
- @ManyToOne Defines a single-valued association to another entity class that has many-to-one multiplicity.
- @OneToMany Defines a many-valued association with one-to-many multiplicity. For more on using annotations and annotation elements to map entities in an enterprise application, see the Java EE 5 Tutorial

Insert the annotations shown in the next slides to the TTitle_book, TTitle_book_on_tape, TBook, TBook_period
Click the Add Class and select all classes as the Entity classes (slide 5)
@Entity

public class TTitle_book implements Serializable {

private static final long serialVersionUIDUID = 1L;

private String publisher;
private String ISBN;
private String title;
private String author;

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;

public Long getId() { return id; }
public void setId(Long val) { id = val; }

@OneToMany(mappedBy = "mTitle_book")
private Collection<TBook> books;

public Collection<TBook> getBooks() { return books; }
public void setBooks(Collection<TBook> val) { books = val; }

public TTitle_book() { id = null; }

@Transient
private ArrayList<TBook> mBooks = new java.util.ArrayList<TBook>();

public ArrayList<TBook> getmBooks() { return mBooks; }
public void setmBooks(ArrayList<TBook> books) { mBooks = books; }

package sub_business_tier.entities;

import java.io.Serializable;
import javax.persistence.Entity;

@Entity
public class TTitle_book_on_tape extends TTitle_book implements Serializable {
    private static final long serialVersionUID = 1L;

    @Override
    public String toString() {
        String help = super.toString();
        help += " Actor: " + getActor();
        return help;
    }
}

@Override
public String getActor() { return actor; }

@Override
public void setActor(String val) {
    actor = val;
}
```java
package sub_business_tier.entities;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.PersistenceContext;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import org.springframework.stereotype.Service;

@Entity
public class TBook implements Serializable {
    private static final long serialVersionUID = 1L;
    private int number;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    public void setId(Long id) { this.id = id; }
    public Long getId() { return id; }

    @ManyToOne
    private TTitle_book mTitle_book;
    public TTitle_book getTitle_book() { return mTitle_book; }
    public void setTitle_book(TTitle_book title_book) { mTitle_book = title_book; }

    public TBook() { id = null; }
}
```
```java
@Entity
public class TBook_period extends TBook implements Serializable {

    private static final long serialVersionUID = 1L;

    @Temporal(javax.persistence.TemporalType.DATE)
    private Date period;

    @Override
    public Date getPeriod() {
        return period;
    }
}
```
The names of packages on this slide are different than on other slides – you must select the entities from your own packages of your program. In real situation, the tool only allows you to select classes with Entity annotation.
4. Create the following classes of the Integration Tier: TTitle_bookController and the TBookController for persisting of objects.

**Facade of the Integration Tier**

- **TBookController**
  - Attributes
    - private EntityManagerFactory emf
    - EntityManager
  - Operations
    - private EntityManager getEntityManager()
    - public TBook[0..*] getTBooks()
    - public TBook[0..*] getTBook()
    - public boolean addTBook(TBook book)
    - public boolean addTBooks(TTitle_book titles[0..*])
    - public boolean findTBook(TBook book)
    - public boolean removeTBook(TBook book)
    - public boolean updateTBook(TBook book)
    - public EntityManagerFactory getEntityManager()
    - public void setEntityManager(EntityManagerFactory val)

- **EntityManagerFactory**
  - Attributes
    - private EntityManager getEntityManager()
    - public TTitle_book getTTitle_book()
    - public TTitle_bookTitles getTTitle_bookTitles()
    - public boolean addTTitle_book(TTitle_book title_book)
    - public boolean addTTitle_bookTitles(TTitle_bookTitles title_bookTitles)
    - public boolean findTTitle_book(TTitle_book title_book)
    - public boolean removeTTitle_book(TTitle_book title_book)
    - public void setEntityManager(EntityManagerFactory val)
    - public void close()
    - public void createEntityManager()

- **EntityManager**
  - Attributes
    - public EntityManager getEntityManager()
    - public Transaction getTransaction()
    - public void setTransaction(Transaction transaction)
    - public Query getQuery()
    - public Object getQuery(Query query)
    - public void setQuery(Query query)
    - public void persist(TTitle_book title_book)
    - public void close()
    - public void createQuery(String s)

- **Query**
  - Attributes
    - public Query getQuery()
    - public Object getResultSet()

- **Transaction**
  - Attributes
    - public Transaction getTransaction()
    - public void begin()
    - public void commit()
4.1. Creating the new package integration_tier in the source packages in the Library1 project

To create a new package:
1) In the Projects window, right-click a Library1 Source Package node and choose New >.
2) Select Java Package.
3) Specify a unique Package Name – integration_tier. In most cases you can keep the default name suggested by the IDE.
4) Click Finish.

Note: If you can’t see the Java Package item, you select Other and click it. In the New File dialog box select the Java Categories item and the Java Package as the File Types item. Click the Next button. In the New Java Package dialog box you should go to the p.3.
To create a new Java Class file:
1) In the Projects window, right-click a Library1 Source Package node and choose New.
2) Select Java Class.
3) Specify a unique Class Name – TTile_bookController. In most cases you can keep the default name suggested by the IDE.
4) Click Finish.
5) Repeat these activities to create the TBookController Java Class (p.4.3).

Note: If you can’t see the Java Class item, you select Other and click it. In the New File dialog box select the Java Categories item and the Java Class as the File Types item. Click the Next button. In the New Java Class dialog box you should go to the p.3
4.2. Creating the TBookController.java in the domainstore package for persisting TTitle_book and TTitle_book_on_tape objects. Applying of design patterns Domain Store i Transfer Object

(1) The TTTitle_bookController class:
creating the EntityManager object
private EntityManager getEntityManager()
public class TTitle_bookController {

    private EntityManagerFactory emf = null;

    private EntityManager getEntityManager() {
        if (emf == null) {
            emf = Persistence.createEntityManagerFactory("Library1PU");
        }
        return emf.createEntityManager();
    }
}
The `TTitle_bookController` class: Adding a title in the database

```
public boolean addTTitle_book(TTitle_book title_book)
```

Diagram:

```
self : TTitle_bookController

title_book : TTitle_book
	em : EntityManager

private EntityManager getEntityManager()

public Transaction getTransaction()

assert

public void begin()

public void persist(TTitle_book title_book)

public Transaction getTransaction()

public void commit()

public void close()

[Finally]
```
(2) The TTitle_bookController class: Adding a title in the database

```java
public boolean addTTitle_book(TTitle_book title_book) {
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        em.persist(title_book);
        em.getTransaction().commit();
    } finally {
        em.close();
        return false;
    }
}
```
(1) The TTitle_bookController class: Adding titles in the database

public boolean addTTitle_books (ArrayList<TTitle_book> titles)
(2) The TTitle_bookController class: Adding titles in the database

```java
public boolean addTTitle_books (ArrayList<TTitle_book> titles) {
    EntityManager em = getEntityManager();
    TTitle_book newTTitle_book = null;
    try {
        Iterator it = titles.iterator();
        em.getTransaction().begin();
        while (it.hasNext()) {
            newTTitle_book = (TTitle_book) it.next();
            if (newTTitle_book.getId() == null) {
                em.persist(newTTitle_book);
            }
        }
        em.getTransaction().commit();
    } finally {
        em.close();
        return false;
    }
}
```
(1) Getting TTitle_book and TTitle_book_on_tape objects from the database
   (design pattern Transfer Object – TTitle_book[0..*])

   public List<TTitle_book> getTTitle_books()

   public TTitle_book[] getTTitle_books_()
(2) Getting TTTitle_book and TTTitle_book_on_tape objects from the database
   (design pattern Transfer Object – TTTitle_book[0..*])

public TTTitle_book[] getTTTitle_books_() {
    return (TTTitle_book[]) getTTTitle_books().toArray(new TTTitle_book[0]);
}

public List<TTTitle_book> getTTTitle_books() {
    EntityManager em = getEntityManager();
    try {
        javax.persistence.Query q =
            em.createQuery("select c from TTTitle_book as c");
        return q.getResultList();
    } finally {
        em.close();
    }
} // end of TTTitle_bookController
package integration_tier;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.List;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import sub_business_tier.entities.ITitle_book;

public class TTitle_bookController {

    private EntityManagerFactory emf = null;

    private EntityManager getEntityManager() {
        if (emf == null) {
            emf = Persistence.createEntityManagerFactory("Library1PU");
        }
        return emf.createEntityManager();
    }

    public TTitle_book[] getTTitle_books() {
        return (TTitle_book[]) getTTitle_books().toArray(new TTitle_book[0]);
    }

    public List<TTitle_book> getTTitle_books() {
        EntityManager em = getEntityManager();
        try {
            javax.persistence.Query q =
                em.createQuery("select o from TTitle_book as o");
            return q.getResultList();
        } finally {
            em.close();
        }
    }
}
```java
public boolean addTTtitle_book(TTitle_book title_book) {
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        em.persist(title_book);
        em.getTransaction().commit();
    } finally {
        em.close();
        return false;
    }
}

public boolean addTTtitle_books(ArrayList<TTitle_book> titles) {
    EntityManager em = getEntityManager();
    TTitle_book newTTitle_book = null;
    try {
        Iterator it = titles.iterator();
        em.getTransaction().begin();
        while (it.hasNext()) {
            newTTitle_book = (TTitle_book) it.next();
            if (newTTitle_book.getId() == null) {
                em.persist(newTTitle_book);
            }
        }
        em.getTransaction().commit();
    } finally {
        em.close();
        return false;
    }
}
```
4.3. Creating the TBookController.java in the integration_tier package

```java
import javax.persistence.Persistence;
import library1.TBook;

public class TBookController {
    private EntityManagerFactory emf = null;

    private EntityManager getEntityManager() {
        if (emf == null) {
            emf = Persistence.createEntityManagerFactory("Library1PU");
        }
        return emf.createEntityManager();
    }
}
```
The TBookController class: Getting books from the database

```java
public TBook[] getTBooks_(), public List<TBook> getTBooks()

public TBook[] getTBooks_() {
    return (TBook[]) getTBooks().toArray(new TBook[0]);
}

public List<TBook> getTBooks() {
    EntityManager em = getEntityManager();
    try {
        javax.persistence.Query q = em.createQuery("select c from TBook as c");
        return q.getResultList();
    } finally {
        em.close();
    }
}
```
The TBookController class: Adding a book in the database

public boolean addTTtitle_books (ArrayList<TTitle_book> titles)

public boolean addTBook(TBook book) {
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        em.persist(book);
        em.getTransaction().commit();
    } finally {
        em.close();
        return false; }
}
public boolean addTBooks (ArrayList<TTitle_book> titles) {
    EntityManager em = getEntityManager();
    TBook newBook = null;
    Iterator it = titles.iterator();
    em.getTransaction().begin();
    try {
        while (it.hasNext()) {
            TTitle_book newTitle_book = (TTitle_book) it.next();
            if (newTitle_book.getId() == null)
                continue;
            Iterator it_ = newTitle_book.getmBooks().iterator();
            while (it_.hasNext()) {
                newBook = (TBook) it_.next();
                if (newBook.getId() == null)
                    em.persist(newBook);
            }
        }  
        em.getTransaction().commit();
    } finally {
        em.close();
        return false; 
    }
}
// end of TBookController
package integration_tier;

import sub_business_tier.entities.TBook;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import sub_business_tier.entities.TTitle_book;

/**...*/

// RUN AS ("administrator")

public class TBookController {
    private EntityManagerFactory emf = null;

    private EntityManager getEntityManager() {
        if (emf == null) {
            emf = Persistence.createEntityManagerFactory("Library1PU");
        }
        return emf.createEntityManager();
    }

    public TBook[] getTBooks() {
        return (TBook[]) getTBooks().toArray(new TBook[0]);
    }

    public List<TBook> getTBooks() {
        EntityManager em = getEntityManager();
        try {
            javax.persistence.Query q = em.createQuery("select c from TBook as c");
            return q.getResultList();
        } finally {
            em.close();
        }
    }
}
public boolean addTBook(TBook book) {
    EntityManager em = getEntityManager();
    try {
        em.getTransaction().begin();
        em.persist(book);
        em.getTransaction().commit();
    } finally {
        em.close();
        return false;
    }
}

public boolean addTBooks(ArrayList<TTitle_book> titles) {
    EntityManager em = getEntityManager();
    TBook newBook = null;
    Iterator it = titles.iterator();
    em.getTransaction().begin();
    try {
        while (it.hasNext()) {
            TTitle_book newTitle_book = (TTitle_book) it.next();
            if (newTitle_book.getId() == null) {
                continue;
            }
            Iterator it_ = newTitle_book.getBooks().iterator();
            while (it_.hasNext()) {
                newBook = (TBook) it_.next();
                if (newBook.getId() == null) {
                    em.persist(newBook);
                }
            }
        }
        em.getTransaction().commit();
    } finally {
        em.close();
        return false;
    }
}
5. You may add annotation to your own new classes and create the proper controllers – for higher assessment (5.0 or 5.5)