ISCG7436 – Enterprise Java Programming
Lab Session 3

Struts Framework + Hibernate

1. Download and setup the Struts + Hibernate example project provided on moodle. Alter the session factory settings in the hibernate.cfg.xml file located in the resources directory to reflect your database and authentication settings.

```
<hibernate-configuration>

	<session-factory>

		<!-- Database connection settings -->
		<property name="connection.driver_class">
			com.mysql.jdbc.Driver
		</property>
		<property name="connection.url">
			jdbc:mysql://localhost:3306/test
		</property>
		<property name="connection.username">root</property>
		<property name="connection.password">test</property>

		<!-- JDBC connection pool (use the built-in) -->
		<property name="connection.pool_size">1</property>

		<!-- SQL dialect -->
		<property name="dialect">
			org.hibernate.dialect.MySQLDialect
		</property>

		<!-- Enable Hibernate's automatic session context management -->
		<property name="current_session_context_class">thread</property>

		<!-- Disable the second-level cache  -->
		<property name="cache.provider_class">
			org.hibernate.cache.NoCacheProvider
		</property>

		<!-- Echo all executed SQL to stdout -->
		<property name="show_sql">true</property>

		<!-- Drop and re-create the database schema on startup -->
		<property name="hbm2ddl.auto">update</property>

		<mapping class="com.jcasey.model.Book" />

	</session-factory>

</hibernate-configuration>
```

2. Have a look at the model object class Book located in the com.jcasey.model package. Note the usage of annotations to setup the Hibernate database bindings ie:
   - `@Entity`
   - `@Table(name="Book")`
   - `@Id`
   - `@GeneratedValue`
   - `@Column(name="book_id")`

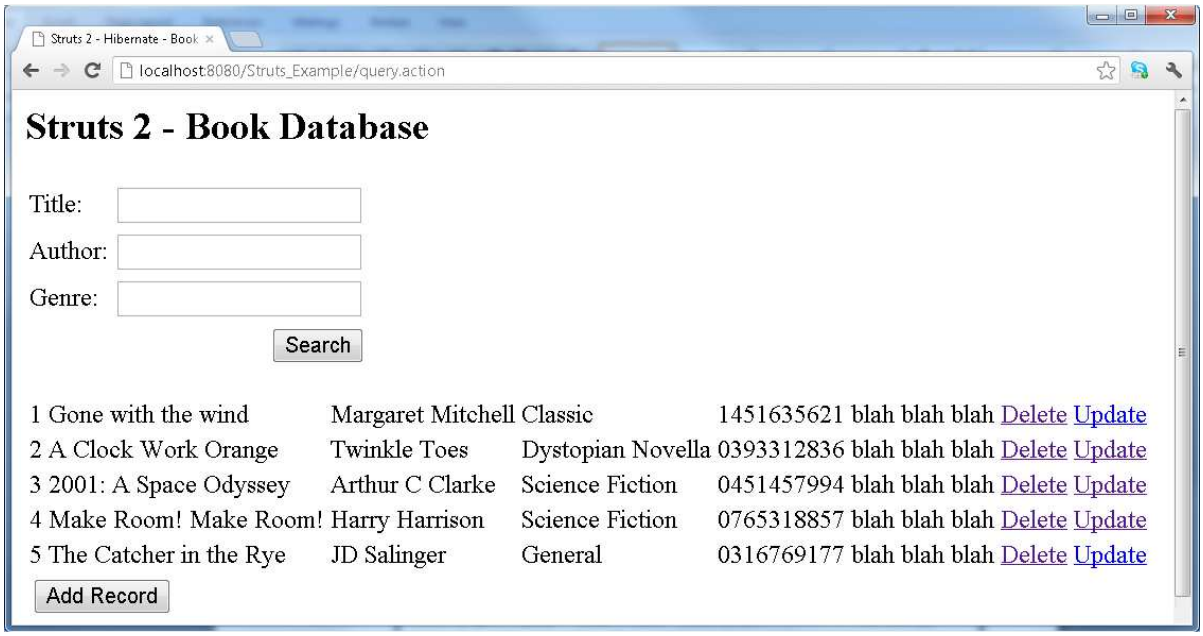
3. Open the BookManager class and have a look a look at the list method. Use the following skeleton code to develop a list method that takes three parameters: a genre, title and author and programmatically add Restrictions for different criteria to Hibernate Criteria object. If a user does not fill in any fields then the list method should return all results.

```
public List<Book> list(String genre, String title, String author)
{
	Session session =
HibernateUtil.getSessionFactory().getCurrentSession();
	session.beginTransaction();
	List<Book> books = null;
	try {

		Criteria criteria = session.createCriteria(Book.class);

		// sample using criteria
		//criteria.add(Restrictions.eq("genre",genre));

		books = (List<Book>)criteria.list();

		// sample with HQL

		// Query query = session.
		// createQuery("from Book b where b.genre = :genre");
		// query.setString("genre", genre);
	} catch (HibernateException e) {
		e.printStackTrace();
		session.getTransaction().rollback();
	}
	session.getTransaction().commit();
	return books;
}
```

- http://docs.jboss.org/hibernate/orm/3.5/api/
- http://docs.jboss.org/hibernate/orm/3.5/api/org/hibernate/Criteria.html
- http://docs.jboss.org/hibernate/orm/3.5/api/org/hibernate/Query.html
- http://docs.jboss.org/hibernate/orm/3.5/api/org/hibernate/criterion/Restrictions.html

4. Update the list() method code so that your Restrictions criteria are fuzzy and make use of the like() method. See links above for more information.

5. Wire the BookManager up to your BookQuery Struts Action POJO and use the list() method to query the book table based on the genre, title and author criteria. Hold onto a reference of the book LinkedList and use this list to drive the s:iterator table as shown below.



6. Using the Delete functionality as a guide, coded by the a href="delete?bookId code block in query.jsp to create a new action mapping to handle record updates. Use add.jsp as a guide when creating the new update.jsp form. Note: the a href tags work completely independently of the form that they exist in so do not rename the form bookAddForm. In addition, extra screens will be required to actually view a selected record before updating so take a look at how the bookAddForm and bookAddRecord action mappings work and interact with the ViewBook class Action and the view.jsp file.

```
<a href="delete?bookId=<s:property value = "bookId"
/>">Delete</a>
```

```
<action name="delete" class="com.jcasey.action.BookQuery"
method="delete">
     <result name="success" type="redirect">query</result>
</action>
```

```
public String delete()
{
     linkController.delete(getBookId());
     return Action.SUCCESS;
}
```