About Eren Avsarogullari

JSF 2, PrimeFaces 3, Spring 3 & Hibernate 4 Integration Project

by Eren Avsarogullari on April 9th, 2012 | **Filed in:** Enterprise Java **Tags:** JBoss Hibernate, JSF, PrimeFaces, Spring

This article shows how to integrate JSF2, PrimeFaces3, Spring3 and Hibernate4 Technologies. It provides a general project template for Java developers.

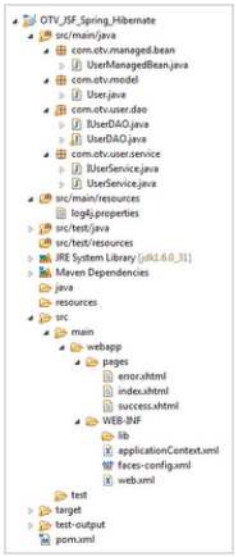
Also if Spring is not used for Business and Data Access layer, JSF – PrimeFaces & Hibernate Integration Project can be offered.

Used Technologies :

- JDK 1.6.0_31
- Spring 3.1.1
- JSF 2.1
- Hibernate 4.1.0
- Primefaces 3.1.1
- MySQL Connector 5.1.17
- MySQL 5.5.8
- c3p0 0.9.1.2
- Tomcat 7.0
- Maven 3.0.2

STEP 1 : CREATE MAVEN PROJECT

A maven project is created as below. (It can be created by using Maven or IDE Plug-in).



STEP 2 : CREATE USER TABLE

A new USER Table is created by executing below script:

```
CREATE TABLE USER (
   id int(11) NOT NULL,
   name varchar(45) NOT NULL,
   surname varchar(45) NOT NULL,
   PRIMARY KEY (`id`)
);
```

STEP 3 : LIBRARIES

Spring, JSF, Hibernate, Primefaces, MySQL and c3p0 dependencies are added to Maven' s pom.xml.

```
<properties>
  <spring.version>3.1.1.RELEASE</spring.version>
 </properties>

 <dependencies>

  <!-- Spring 3 dependencies -->
  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>${spring.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>${spring.version}</version>
  </dependency>


  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-web</artifactId>
   <version>${spring.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-tx</artifactId>
   <version>${spring.version}</version>
  </dependency>

  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-orm</artifactId>
   <version>${spring.version}</version>
  </dependency>

  <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
        </dependency>

  <!-- JSF dependencies -->
  <dependency>
    <groupId>com.sun.faces</groupId>
    <artifactId>jsf-api</artifactId>
    <version>2.1.6</version>
  </dependency>

  <dependency>
    <groupId>com.sun.faces</groupId>
    <artifactId>jsf-impl</artifactId>
    <version>2.1.6</version>
  </dependency>

  <dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>jstl</artifactId>
   <version>1.2</version>
  </dependency>

  <!-- Primefaces dependency -->
   <dependency>
   <groupId>org.primefaces</groupId>
   <artifactId>primefaces</artifactId>
   <version>3.1.1</version>
  </dependency>

  <!-- Hibernate dependencies -->
  <dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate-core</artifactId>
   <version>4.1.0.Final</version>
  </dependency>

  <dependency>
   <groupId>javassist</groupId>
   <artifactId>javassist</artifactId>
   <version>3.12.1.GA</version>
  </dependency>

  <!-- MySQL Java Connector dependency -->
  <dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>5.1.17</version>
  </dependency>

  <!-- c3p0 dependency -->
  <dependency>
   <groupId>c3p0</groupId>
   <artifactId>c3p0</artifactId>
   <version>0.9.1.2</version>
  </dependency>

 </dependencies>
```

STEP 4 : CREATE USER MODEL CLASS

A new User Model Class is created.

```
01 package com.otv.model;
02
03 import javax.persistence.Column;
04 import javax.persistence.Entity;
05 import javax.persistence.Id;
06 import javax.persistence.Table;
07
08 /**
09  *
10  * User Entity
11  *
12  * @author onlinetechvision.com
13  * @since 25 Mar 2012
14  * @version 1.0.0
15  *
16  */
17 @Entity
18 @Table(name="USER")
19 public class User {
20
21  private int id;
22  private String name;
23  private String surname;
24
25  /**
26   * Get User Id
27   *
28   * @return int - User Id
29   */
30  @Id
31  @Column(name="ID", unique = true, nullable = false)
32  public int getId() {
33   return id;
34  }
35
36  /**
37   * Set User Id
38   *
39   * @param int - User Id
40   */
41  public void setId(int id) {
42   this.id = id;
43  }
44
45  /**
46   * Get User Name
47   *
48   * @return String - User Name
49   */
50  @Column(name="NAME", unique = true, nullable = false)
51  public String getName() {
52   return name;
53  }
54
55  /**
56   * Set User Name
57   *
58   * @param String - User Name
59   */
60  public void setName(String name) {
61   this.name = name;
62  }
63
64  /**
65   * Get User Surname
66   *
67   * @return String - User Surname
68   */
69  @Column(name="SURNAME", unique = true, nullable = false)
70  public String getSurname() {
71   return surname;
72  }
73
74  /**
75   * Set User Surname
76   *
77   * @param String - User Surname
78   */
79  public void setSurname(String surname) {
80   this.surname = surname;
81  }
82
83  @Override
84  public String toString() {
85   StringBuffer strBuff = new StringBuffer();
86   strBuff.append("id : ").append(getId());
87   strBuff.append(", name : ").append(getName());
88   strBuff.append(", surname : ").append(getSurname());
89   return strBuff.toString();
90  }
91 }
```

STEP 5 : CREATE USER MANAGED BEAN CLASS

User Managed Bean is created.

```
001 package com.otv.managed.bean;
002
003 import java.io.Serializable;
```

```
004 import java.util.ArrayList;
005 import java.util.List;
006
007 import javax.faces.bean.ManagedBean;
008 import javax.faces.bean.ManagedProperty;
009 import javax.faces.bean.RequestScoped;
010
011 import org.springframework.dao.DataAccessException;
012
013 import com.otv.model.User;
014 import com.otv.user.service.IUserService;
015
016 /**
017  *
018  * User Managed Bean
019  *
020  * @author onlinetechvision.com
021  * @since 25 Mar 2012
022  * @version 1.0.0
023  *
024  */
025 @ManagedBean(name="userMB")
026 @RequestScoped
027 public class UserManagedBean implements Serializable {
028
029  private static final long serialVersionUID = 1L;
030  private static final String SUCCESS = "success";
031  private static final String ERROR   = "error";
032
033  //Spring User Service is injected...
034  @ManagedProperty(value="#{UserService}")
035  IUserService userService;
036
037  List<User> userList;
038
039  private int id;
040  private String name;
041  private String surname;
042
043  /**
044   * Add User
045   *
046   * @return String - Response Message
047   */
048  public String addUser() {
049   try {
050    User user = new User();
051    user.setId(getId());
052    user.setName(getName());
053    user.setSurname(getSurname());
054    getUserService().addUser(user);
055    return SUCCESS;
056   } catch (DataAccessException e) {
057    e.printStackTrace();
058   }
059
060   return ERROR;
061  }
062
063  /**
064   * Reset Fields
065   *
066   */
067  public void reset() {
068   this.setId(0);
069   this.setName("");
070   this.setSurname("");
071  }
072
073  /**
074   * Get User List
075   *
076   * @return List - User List
077   */
078  public List<User> getUserList() {
079   userList = new ArrayList<User>();
080   userList.addAll(getUserService().getUsers());
081   return userList;
082  }
083
084  /**
085   * Get User Service
086   *
087   * @return IUserService - User Service
088   */
089  public IUserService getUserService() {
090   return userService;
091  }
092
093  /**
094   * Set User Service
095   *
096   * @param IUserService - User Service
097   */
098  public void setUserService(IUserService userService) {
099   this.userService = userService;
100  }
101
102  /**
103   * Set User List
104   *
105   * @param List - User List
106   */
```

```
 public void setUserList(List<User> userList) {
  this.userList = userList;
 }

 /**
  * Get User Id
  *
  * @return int - User Id
  */
 public int getId() {
  return id;
 }

 /**
  * Set User Id
  *
  * @param int - User Id
  */
 public void setId(int id) {
  this.id = id;
 }

 /**
  * Get User Name
  *
  * @return String - User Name
  */
 public String getName() {
  return name;
 }

 /**
  * Set User Name
  *
  * @param String - User Name
  */
 public void setName(String name) {
  this.name = name;
 }

 /**
  * Get User Surname
  *
  * @return String - User Surname
  */
 public String getSurname() {
  return surname;
 }

 /**
  * Set User Surname
  *
  * @param String - User Surname
  */
 public void setSurname(String surname) {
  this.surname = surname;
 }

}
```

STEP 6 : CREATE IUserDAO INTERFACE

IUserDAO Interface provides methods of Data Access Layer. The data access layer manages all the logic to persist and retrieve the data from database.

```
package com.otv.user.dao;

import java.util.List;

import com.otv.model.User;

/**
 *
 * User DAO Interface
 *
 * @author onlinetechvision.com
 * @since 25 Mar 2012
 * @version 1.0.0
 *
 */
public interface IUserDAO {

 /**
  * Add User
  *
  * @param  User user
  */
 public void addUser(User user);

 /**
  * Update User
  *
  * @param  User user
  */
 public void updateUser(User user);

 /**
  * Delete User
  *
  * @param  User user
  */
```

```
 public void deleteUser(User user);

 /**
  * Get User
  *
  * @param  int User Id
  */
 public User getUserById(int id);

 /**
  * Get User List
  *
  */
 public List<User> getUsers();
}
```

STEP 7 : CREATE UserDAO CLASS

UserDAO Class is created by implementing IUserDAO Interface.

```
package com.otv.user.dao;

import java.util.List;

import com.otv.model.User;

import org.hibernate.SessionFactory;

/**
 *
 * User DAO
 *
 * @author onlinetechvision.com
 * @since 25 Mar 2012
 * @version 1.0.0
 *
 */

public class UserDAO implements IUserDAO {

 private SessionFactory sessionFactory;

 /**
  * Get Hibernate Session Factory
  *
  * @return SessionFactory - Hibernate Session Factory
  */
 public SessionFactory getSessionFactory() {
  return sessionFactory;
 }

 /**
  * Set Hibernate Session Factory
  *
  * @param SessionFactory - Hibernate Session Factory
  */
 public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

 /**
  * Add User
  *
  * @param  User user
  */
 @Override
 public void addUser(User user) {
  getSessionFactory().getCurrentSession().save(user);
 }

 /**
  * Delete User
  *
  * @param  User user
  */
 @Override
 public void deleteUser(User user) {
  getSessionFactory().getCurrentSession().delete(user);
 }

 /**
  * Update User
  *
  * @param  User user
  */
 @Override
 public void updateUser(User user) {
  getSessionFactory().getCurrentSession().update(user);
 }

 /**
  * Get User
  *
  * @param  int User Id
  * @return User
  */
 @Override
 public User getUserById(int id) {
  List list = getSessionFactory().getCurrentSession()
            .createQuery("from User where id=?")
                  .setParameter(0, id).list();
  return (User)list.get(0);
```

```
 }

 /**
  * Get User List
  *
  * @return List - User list
  */
 @Override
 public List<User> getUsers() {
  List list = getSessionFactory().getCurrentSession().createQuery("from User").list();
  return list;
 }

}
```

STEP 8 : CREATE IUserService INTERFACE

IUserService Interface provides methods to process the business logic.

```
package com.otv.user.service;

import java.util.List;

import com.otv.model.User;

/**
 *
 * User Service Interface
 *
 * @author onlinetechvision.com
 * @since 25 Mar 2012
 * @version 1.0.0
 *
 */
public interface IUserService {

 /**
  * Add User
  *
  * @param  User user
  */
 public void addUser(User user);

 /**
  * Update User
  *
  * @param  User user
  */
 public void updateUser(User user);

 /**
  * Delete User
  *
  * @param  User user
  */
 public void deleteUser(User user);

 /**
  * Get User
  *
  * @param  int User Id
  */
 public User getUserById(int id);

 /**
  * Get User List
  *
  * @return List - User list
  */
 public List<User> getUsers();
}
```

STEP 9 : CREATE UserService CLASS

UserService Class is created by implementing IUserService Interface.

```
package com.otv.user.service;

import java.util.List;

import org.springframework.transaction.annotation.Transactional;

import com.otv.model.User;
import com.otv.user.dao.IUserDAO;

/**
 *
 * User Service
 *
 * @author onlinetechvision.com
 * @since 25 Mar 2012
 * @version 1.0.0
 *
 */
@Transactional(readOnly = true)
public class UserService implements IUserService {

 // UserDAO is injected...
 IUserDAO userDAO;
```

```
 /**
  * Add User
  *
  * @param  User user
  */
 @Transactional(readOnly = false)
 @Override
 public void addUser(User user) {
  getUserDAO().addUser(user);
 }

 /**
  * Delete User
  *
  * @param  User user
  */
 @Transactional(readOnly = false)
 @Override
 public void deleteUser(User user) {
  getUserDAO().deleteUser(user);
 }

 /**
  * Update User
  *
  * @param  User user
  */
 @Transactional(readOnly = false)
 @Override
 public void updateUser(User user) {
  getUserDAO().updateUser(user);
 }

 /**
  * Get User
  *
  * @param  int User Id
  */
 @Override
 public User getUserById(int id) {
  return getUserDAO().getUserById(id);
 }

 /**
  * Get User List
  *
  */
 @Override
 public List<User> getUsers() {
  return getUserDAO().getUsers();
 }

 /**
  * Get User DAO
  *
  * @return IUserDAO - User DAO
  */
 public IUserDAO getUserDAO() {
  return userDAO;
 }

 /**
  * Set User DAO
  *
  * @param IUserDAO - User DAO
  */
 public void setUserDAO(IUserDAO userDAO) {
  this.userDAO = userDAO;
 }
}
```

STEP 10 : CREATE applicationContext.xml

Spring Application Context' s content is shown as follows :

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd


http://www.springframework.org/schema/tx


http://www.springframework.org/schema/tx/spring-tx-3.0.xsd


http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

 <!-- Beans Declaration -->
 <bean id="User" class="com.otv.model.User"/>

 <!-- User Service Declaration -->
 <bean id="UserService" class="com.otv.user.service.UserService">
  <property name="userDAO" ref="UserDAO" />
 </bean>
```

```
 <!-- User DAO Declaration -->
 <bean id="UserDAO" class="com.otv.user.dao.UserDAO">
  <property name="sessionFactory" ref="SessionFactory" />
 </bean>

 <!-- Data Source Declaration -->
 <bean id="DataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">
  <property name="driverClass" value="com.mysql.jdbc.Driver" />
  <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/Test" />
  <property name="user" value="root" />
  <property name="password" value="root" />
  <property name="maxPoolSize" value="10" />
  <property name="maxStatements" value="0" />
  <property name="minPoolSize" value="5" />
 </bean>

 <!-- Session Factory Declaration -->
 <bean id="SessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
  <property name="dataSource" ref="DataSource" />
  <property name="annotatedClasses">
   <list>
    <value>com.otv.model.User</value>
   </list>
  </property>
  <property name="hibernateProperties">
   <props>
    <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
    <prop key="hibernate.show_sql">true</prop>
   </props>
  </property>
 </bean>

 <!-- Enable the configuration of transactional behavior based on annotations -->
    <tx:annotation-driven transaction-manager="txManager"/>

 <!-- Transaction Manager is defined -->
    <bean id="txManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager">
       <property name="sessionFactory" ref="SessionFactory"/>
    </bean>

</beans>
```

STEP 11 : CREATE faces-config.xml

JSF Configuration is shown as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"

    version="2.0">

   <!-- JSF and Spring are integrated -->
   <application>
     <el-resolver>
      org.springframework.web.jsf.el.SpringBeanFacesELResolver
     </el-resolver>
   </application>

 <!-- configuration of navigation rules -->
 <navigation-rule>
     <from-view-id>/pages/index.xhtml</from-view-id>
     <navigation-case>
         <from-outcome>success</from-outcome>
            <to-view-id>/pages/success.xhtml</to-view-id>
        </navigation-case>
         <navigation-case>
         <from-outcome>error</from-outcome>
            <to-view-id>/pages/error.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>

</faces-config>
```

STEP 12 : CREATE web.xml

web.xml is configured as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
   id="WebApp_ID"
   version="2.5">

   <display-name>OTV_JSF_PrimeFaces_Spring_Hibernate</display-name>

   <!-- Spring Context Configuration' s Path definition -->
      <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>
      /WEB-INF/applicationContext.xml
```

```
      </param-value>
   </context-param>

   <!-- The Bootstrap listener to start up and shut down Spring's root WebApplicationContext. It is registered to Servlet Container -->
   <listener>
  <listener-class>
   org.springframework.web.context.ContextLoaderListener
  </listener-class>
   </listener>
   <listener>
  <listener-class>
   org.springframework.web.context.request.RequestContextListener
  </listener-class>
   </listener>

   <!-- Project Stage Level -->
   <context-param>
     <param-name>javax.faces.PROJECT_STAGE</param-name>
     <param-value>Development</param-value>
   </context-param>

   <!-- Welcome Page -->
   <welcome-file-list>
     <welcome-file>/pages/index.xhtml</welcome-file>
   </welcome-file-list>

   <!-- JSF Servlet is defined to container -->
   <servlet>
     <servlet-name>Faces Servlet</servlet-name>
     <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
     <load-on-startup>1</load-on-startup>
   </servlet>

   <!-- Mapping with servlet and url for the http requests. -->
   <servlet-mapping>
     <servlet-name>Faces Servlet</servlet-name>
     <url-pattern>*.jsf</url-pattern>
   </servlet-mapping>

   <servlet-mapping>
     <servlet-name>Faces Servlet</servlet-name>
     <url-pattern>*.faces</url-pattern>
   </servlet-mapping>

</web-app>
```

STEP 13 : CREATE index.xhtml

index.xhtml is created as follows :


```
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
     xmlns:p="http://primefaces.org/ui">
 <h:head><title>Welcome to OTV_JSF_Spring_Hibernate_Project</title></h:head>
 <h:body>
     <h:form>
      <table>
       <tr>
        <td><h:outputLabel for="id" value="Id : " /></td>
        <td><p:inputText id="id" value="#{userMB.id}">
          <f:converter converterId="javax.faces.Integer"/>
             <p:ajax event="blur" update="idMsg" />
         </p:inputText>
         <p:message id="idMsg" for="id" display="icon"/>
        </td>
       </tr>
       <tr>
        <td><h:outputLabel for="name" value="Name : " /></td>
        <td><p:inputText id="name" value="#{userMB.name}">
          <f:validateLength minimum="5" />
          <p:ajax event="blur" update="nameMsg" />
         </p:inputText>
         <p:message id="nameMsg" for="name" display="icon"/>
        </td>
       </tr>
       <tr>
        <td><h:outputLabel for="surname" value="Surname : " /></td>
        <td><p:inputText id="surname" value="#{userMB.surname}">
          <f:validateLength minimum="5" />
          <p:ajax event="blur" update="surnameMsg" />
         </p:inputText>
         <p:message id="surnameMsg" for="surname" display="icon"/>
        </td>
       </tr>
          <tr>
        <td><p:commandButton id="addUser" value="Add" action="#{userMB.addUser}" ajax="false"/>
</td>
        <td><p:commandButton id="reset" value="Reset" action="#{userMB.reset}" ajax="false"/></td>
```

```
        </tr>
      </table>
     </h:form>
</h:body>
</html>
```

STEP 14 : CREATE success.xhtml

success.xhtml is created as follows :

USERs :

ID	Name	Surname
12345	Bruce	Willis

```
<html xmlns="http://www.w3.org/1999/xhtml"
 xmlns:h="http://java.sun.com/jsf/html"
 xmlns:f="http://java.sun.com/jsf/core"
 xmlns:p="http://primefaces.org/ui">
<h:head>
 <title>Welcome to OTV_JSF_Spring_Hibernate_Project</title>
</h:head>
<h:body>
 <h:form>
  <h:outputText value="USERs : "></h:outputText>
  <p:dataTable id="users" var="user" value="#{userMB.userList}" style="width: 10%">
   <p:column>
    <f:facet name="header">
     <h:outputText value="ID" />
    </f:facet>
    <h:outputText value="#{user.id}" />
   </p:column>
   <p:column>
    <f:facet name="header">
     <h:outputText value="Name" />
    </f:facet>
    <h:outputText value="#{user.name}" />
   </p:column>
   <p:column>
    <f:facet name="header">
     <h:outputText value="Surname" />
    </f:facet>
    <h:outputText value="#{user.surname}" />
   </p:column>
  </p:dataTable>
 </h:form>
</h:body>

</html>
```

STEP 15 : CREATE error.xhtml

error.xhtml is created as follows :

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">

 <h:head><title>Welcome to JSF_PrimeFaces_Spring_Hibernate Project</title></h:head>
 <body>
 <f:view>
     <h:form>
         <h:outputText value="Transaction Error has occurred!"></h:outputText>
     </h:form>
 </f:view>
</body>
</html>
```

STEP 16 : DEPLOY PROJECT

After **OTV_JSF_Spring_Hibernate** Project is deployed to Tomcat, index page can be opened via following URL :

http://ip:port/OTV_JSF_Spring_Hibernate_Project-1.0-SNAPSHOT/pages/index.jsf

STEP 17 : DOWNLOAD

OTV_JSF_Spring_Hibernate

Reference: JSF2 + Primefaces3 + Spring3 & Hibernate4 Integration Project from our JCG partner Eren Avsarogullari at the Online Technology Vision blog.

You might also like:

- JSF – PrimeFaces & Hibernate Integration Project
- How cool is integration testing with Spring and Hibernate
- RESTEasy Tutorial Part-2: Spring Integration
- JBoss 4.2.x Spring 3 JPA Hibernate Tutorial Part #2
- Full Web Application Tomcat JSF Primefaces JPA Hibernate – Part 3

Share and enjoy!

21 comments

★ 5

Leave a message...

Best Community Share

James Ngumba • 11 months ago
How can one use the user DAO classes above as generic classes so that there is no repetition for each object in a project?
10 · Reply · Share ›

Alwyn Schoeman • 11 months ago
Is there a way to specify the JSF navigation rules with spring rather than in the faces-config file?
3 · Reply · Share ›

simont • a month ago
I ' don't see the benefit of using Spring since hibernate offer the persistance and jsf-primefaces offer the rich UI components with a big facility and also offer mvc architecture , what do you think ?
0 · Reply · Share ›

Arturo Agudelo • 2 months ago
Nice
0 · Reply · Share ›

ponic • 4 months ago
Nice simple tutorial. Great.
0 · Reply · Share ›

ponic • 4 months ago
I would like to know which maven archtype can be used for creating the project structure for these kind of projects. E.g in the screen shot of your project as per the screen shot which is posted above, have you used any archtype? If so what is the name of it? Thanks
0 · Reply · Share ›

Fredrik • 6 months ago
I had issues with my build path. This solved my issues.
http://stackoverflow.com/quest...
0 · Reply · Share ›

Pablo • 7 months ago
How do you create a maven project with that file structure? (sorry for the dumb question)
0 · Reply · Share ›

Imad Oppa • 7 months ago
unfortunately it does't work for me with this config .. but when I replace Hibernate4 with Hibernate3 now it works, because you used the propriety annotatedClasses here so we need to use the class AnnotationSessionFactoryBean in SessionFactory and not LocalSessionFactoryBean and since it does't exist then it must go through "org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean" and use class "org.springframework.orm.hibernate3.HibernateTransactionManager" in Transaction Manager Bean
0 · Reply · Share ›

Ganesh Kumar Kumaraswamy • 9 months ago
Tried using annotation based (package-scan) implementation, having problem injecting managed bean with Richfaces JSF. I'm not getting any error. However my bean is either not called or not visible in Face Container. Any problem which you might been faced already?
0 · Reply · Share ›

Amine El idrissi • 9 months ago
i get the followinf when i start the project using tomcat:
"FAIL - Application for context path / OTV_JSF_Spring_Hibernate-1.0-SNAPSHOT could not be started"

i have to notice that after deploying the project in tomcat i "functional" is set to "false" is that has an effect.
0 · Reply · Share ›

Ovidiu Drumia • 10 months ago
Hi,

Nice tutorial. You might want to update it with the Spring 3 specification(annotations, package-scan). Also you can

include the jetty plugin in maven to run it with the jetty:run goal.

Anyway it seems u forgot to put the:
<servlet-mapping>
<servlet-name>Faces Servlet</servlet-name>
<url-pattern>*.xhtml</url-pattern>
</servlet-mapping>
in the web.xml.

Best of luck,
Ovidiu

0 · Reply · Share ›

---

bbkumscap · a year ago
Hi
I solved my probleme.first, it's not necessery to add some dependancies.2nd, in postgres, User is a reserved keyword , we should not use this as a table name.Thanks
0 · Reply · Share ›

Ghazi Hakim · a year ago
Hi
I have a warning in faces-config.xml in this tag:
<application>
<el-resolver>
org.springframework.web.jsf.el.SpringBeanFacesELResolver
</el-resolver>
</application> the warning is:Class org.springframework.web.jsf.el.SpringBeanFacesELResolver must extend the type javax.el.ELResolver. Any solution ?
Thank you
0 · Reply · Share ›

---

bbkumscap ↗ Ghazi Hakim · a year ago
add this in your pom
<dependency>
<groupid>org.apache.tomcat</groupid>
<artifactid>tomcat-el-api</artifactid>
<version>${tomcat.version}</version>
<scope>provided</scope>
</dependency>

Or
<dependency>
<groupid>javax.el</groupid>
<artifactid>el-api</artifactid>
<version>2.2</version>
<scope>provided</scope>
</dependency>
0 · Reply · Share ›

---

bbkumscap · a year ago
hi
i download and test this sample, i little changed it for using postgres instead mysql, when i compiled it, it made errors, i resolved the problems by adding dependancies
<dependency> <groupid>dom4j</groupid> <artifactid>dom4j</artifactid> <version>1.6.1</version> <scope>compile</scope> </dependency> <dependency> <groupid>org.jboss.logging</groupid> <artifactid>jboss-logging</artifactid> <version>3.1.0.GA</version> </dependency> <dependency> <groupid>org.hibernate</groupid> <artifactid>hibernate-infinispan</artifactid> <version>4.1.1.Final</version> </dependency> <dependency> <groupid>org.infinispan</groupid> <artifactid>infinispan-core</artifactid> <version>5.1.2.CR1</version> </dependency> <dependency> <groupid>org.hibernate.javax.persistence</groupid> <artifactid>hibernate-jpa-2.0-api</artifactid> <version>1.0.1.Final</version> </dependency> <dependency> <groupid>javax.transaction</groupid> <artifactid>jta</artifactid> <version>1.1</version> <type>jar</type> </dependency>

When i add the new user, i have a transaction error
Caused by: org.postgresql.util.PSQLException: ERREUR: erreur de syntaxe sur ou près de « user » Position : 13 at org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:2103) at org.postgresql.core.v3.QueryExecutorImpl.processResults(QueryExecutorImpl.java:1836) at org.postgresql.core.v3.QueryExecutorImpl.execute(QueryExecutorImpl.java:257) at

see more

0 · Reply · Share ›

---

bbkumscap ↗ bbkumscap · a year ago
solved
0 · Reply · Share ›

Ghazi Hakim • a year ago
I want to use "run on server" option in Eclipse instead of deploying the war file manually in Tomcat Manager,
How can I do that ?
Thank you
0 · Reply · Share ›

Byron Kiourtzoglou Mod • a year ago
The download link is fixed now ... sorry for the inconvenience
0 · Reply · Share ›

Ghazi Hakim → Byron Kiourtzoglou • a year ago
Thank you :)
0 · Reply · Share ›

Ghazi Hakim • a year ago
thank you!
but I can't download it!!
0 · Reply · Share ›