Java Code Geeks
Java 2 Java Developers Resource Center

MONDAY, 9 APRIL 2012

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

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

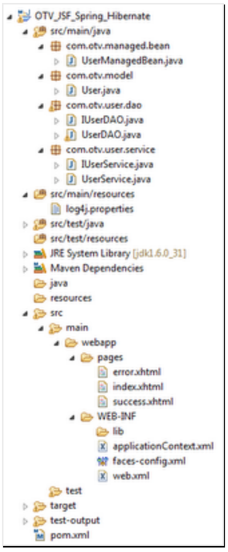
Also if Spring is not used for Business and Data Access layer, JSF – PrimeFaces & Hibernate Integration Project can be offered.

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).



```
OTV_JSF_Spring_Hibernate
  src/main/java
    com.otv.managed.bean
      UserManagedBean.java
    com.otv.model
      User.java
    com.otv.user.dao
      IUserDAO.java
      UserDAO.java
    com.otv.user.service
      IUserService.java
      UserService.java
  src/main/resources
    log4j.properties
  src/test/java
  src/test/resources
  JRE System Library [jdk1.6.0_31]
  Maven Dependencies
  java
  resources
  src
    main
      webapp
        pages
          error.xhtml
          index.xhtml
          success.xhtml
        WEB-INF
          lib
          applicationContext.xml
          faces-config.xml
          web.xml
    test
  target
  test-output
  pom.xml
```

STEP 2 : CREATE USER TABLE

A new USER Table is created by executing below script:

```
CREATE TABLE USER (
```

TECH SEARCH

SPONSORS

JCG FEED

CONNECT WITH US

```
   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>
```

JOIN US

NOTE : Do you have a b
unique and interesting co
Then, check out our JCG
program. You can also b
writer for Java Code Gee

POPULAR TAGS

Enterprise Java
Development Core J
Android Best Of The Week
Java Testing Eclipse Hiber
Tutorial JUnit Android Games
Agile JPA GWT JavaFX Java 7
Performance and Scalability Pl
RESTful Web Service Tomcat
DevOps Maven Security JBos
Best Practices Meta Java Code
Pattern Scala Tutorial JSF JV
WebLogic Books Cloud Commu
JAXB MySQL XML Architectu
Injection Java Netbeans NoSQ
Quartz App Engine Aspect Oriente
Cache EJB3 Git GlassFish Intel
Tutorials Logging Akka AspectJ Ca
Java EE 6 MongoDB OSGi Seleniu
Arquillian CDI CEP Code Analysis
JTA Java 8 MOXy MapReduce S
Security Swing Xuggler ActiveMQ
Reviews Complex Event Proce
Driven Design Gson Heroku Hyp
Messaging Logback Openfire Sm
Tutorial Thread Pool Vaadin Web
findbugs jOOQ

BUZZ

- People
- Recent
- Popular

Most Discussed

- OSGI - Modularizing your app
  Code Geeks

  1 comment · 14 hours ago

- Spring 3: Type safe depender
  Java Code Geeks

  2 comments · 1 day ago

- Software architect mistakes - .
  Geeks

  3 comments · 1 day ago

- JUnit and Hamcrest: Improvin
  assertEquals - Java Code Gee

```
  <!-- 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.

```
package com.otv.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

/**
 *
 * User Entity
 *
 * @author onlinetechvision.com
 * @since 25 Mar 2012
 * @version 1.0.0
 *
 */
@Entity
@Table(name="USER")
public class User {

 private int id;
 private String name;
 private String surname;

 /**
  * Get User Id
  *
  * @return int - User Id
  */
 @Id
 @Column(name="ID", unique = true, nullable = false)
 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
  */
 @Column(name="NAME", unique = true, nullable = false)
 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
  */
 @Column(name="SURNAME", unique = true, nullable = false)
 public String getSurname() {
  return surname;
 }

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

 @Override
 public String toString() {
  StringBuffer strBuff = new StringBuffer();
  strBuff.append("id : ").append(getId());
  strBuff.append(", name : ").append(getName());
  strBuff.append(", surname : ").append(getSurname());
  return strBuff.toString();
 }
}
```

STEP 5 : CREATE USER MANAGED BEAN CLASS

User Managed Bean is created.

```
package com.otv.managed.bean;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;

import org.springframework.dao.DataAccessException;

import com.otv.model.User;
import com.otv.user.service.IUserService;

/**
 *
 * User Managed Bean
 *
 * @author onlinetechvision.com
 * @since 25 Mar 2012
 * @version 1.0.0
 *
 */
@ManagedBean(name="userMB")
@RequestScoped
public class UserManagedBean implements Serializable {

 private static final long serialVersionUID = 1L;
 private static final String SUCCESS = "success";
 private static final String ERROR   = "error";

 //Spring User Service is injected...
 @ManagedProperty(value="#{UserService}")
 IUserService userService;

 List<User> userList;

 private int id;
 private String name;
 private String surname;

 /**
  * Add User
  *
  * @return String - Response Message
  */
 public String addUser() {
  try {
   User user = new User();
   user.setId(getId());
   user.setName(getName());
   user.setSurname(getSurname());
   getUserService().addUser(user);
   return SUCCESS;
  } catch (DataAccessException e) {
   e.printStackTrace();
  }

  return ERROR;
 }

 /**
  * Reset Fields
  *
  */
 public void reset() {
  this.setId(0);
  this.setName("");
  this.setSurname("");
 }

 /**
  * Get User List
  *
  * @return List - User List
  */
 public List<User> getUserList() {
  userList = new ArrayList<User>();
  userList.addAll(getUserService().getUsers());
  return userList;
 }

 /**
  * Get User Service
  *
  * @return IUserService - User Service
  */
 public IUserService getUserService() {
  return userService;
 }

 /**
  * Set User Service
  *
  * @param IUserService - User Service
  */
 public void setUserService(IUserService userService) {
  this.userService = userService;
 }

 /**
  * Set User List
  *
  * @param List - User List
  */
 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-metho
d="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.LocalSessionFacto
ryBean">
  <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.HibernateTransactio
nManager">
       <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>
```

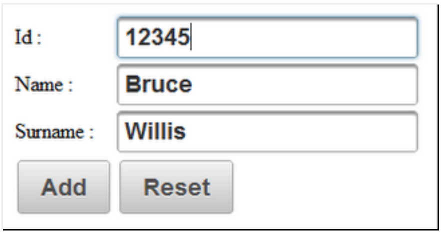
STEP 12 : CREATE web.xml

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/ja
vaee/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 WebApplicationC
ontext. 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 :



Id : 12345

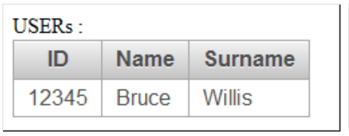
Name : Bruce

Surname : 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>
      <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="f
alse"/></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.

3

Share

0 0

Curtir

Posted by Nikos Maravitsas at 9:00 AM 9 Comments and 9 Reactions
Labels: Hibernate, JSF, PrimeFaces, Spring

Like and 3 others liked this.

Add New Comment Login

Showing 5 of 9 comments Sort by popular now ▼

bbkumscap

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

2 weeks ago Like Reply

Ghazi Hakim

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

2 weeks ago Like Reply

bbkumscap

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>
```

2 weeks ago in reply to Ghazi Hakim Like Reply

bbkumscap

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
org.postgresql.jdbc2.AbstractJdbc2Statement.execute(AbstractJdbc2Statement.java:512) at
org.postgresql.jdbc2.AbstractJdbc2Statement.executeWithFlags(AbstractJdbc2Statement.java:388) at
org.postgresql.jdbc2.AbstractJdbc2Statement.executeUpdate(AbstractJdbc2Statement.java:334) at
com.mchange.v2.c3p0.impl.NewProxyPreparedStatement.executeUpdate(NewProxyPreparedStatement.java:105)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57) at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at
java.lang.reflect.Method.invoke(Method.java:601) at
org.hibernate.engine.jdbc.internal.proxy.AbstractStatementProxyHandler.continueInvocation(AbstractStatementProx
yHandler.java:122) ... 56 more
```

thanks

2 weeks ago Like Reply

bbkumscap

solved

2 weeks ago in reply to bbkumscap Like Reply

Subscribe by email RSS

Load more comments

Reactions

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

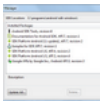
WHAT'S HOT! **HALL OF FAME!** **ARCHIVE**

“Android Full Application Tutorial” series
Continuing our tutorials on Android here at JavaCodeGeeks, I am going to provide a number of articles on how to build a full Android appli...

Android Location Based Services Application – GPS location
With the incorporation of GPS devices in smartphones, Location Based Services (LBS) have become pretty hot the past few years. The iPhone ...

Android Game Development Tutorials
Some months ago we received an email from a fellow Java developer, Tamas Jano , asking to be part of our JCG partners program. To our surp...

What Refactoring is, and what it isn’t
Sometimes a programmer will come to me and explain that they don’t like the design of something and that “we’re gonna need to do a whole bu...


Android Google Maps Tutorial
The Android platform provides easy and tight integration between Android applications and Google Maps . The well established Google Maps A...

Software architect mistakes
I think that to get up in the morning and brew a good cup of coffee is one of the best way to start the day. You know, the heady fragrance ...

The Ultimate Java Resource Collection
Hello fellow Java Geeks! Here at Java Code Geeks we are striving to create the ultimate Java to Java developers resource center. In that ...

Serious about your software career? Leave your job
I recently resigned my position as senior software engineer and technical lead for a middleware services group at Wells Fargo. The job was ...

Humour Time: Programming Puns
Though not strictly puns, I tried to make them sort-of funny: - What are you doing on this bench with a bank slip and a marker? - Benchm...

Android Interview Questions and Answers
Here are some basic Android related interview questions and their respective answers. What is Android? Android is a stack of software for...

Funny Source Code Comments
I recently stumbled upon an awesome Stack Overflow thread, entitled "What is the best comment in source code you have ever encountered?" . ...

“Android Full Application Tutorial” series
Continuing our tutorials on Android here at JavaCodeGeeks, I am going to provide a number of articles on how to build a full Android appli...

GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial
This step by step guide will present how to develop a simple web application using Google's Web Toolkit (GWT) for the rich client and Spri...

Android Google Maps Tutorial
The Android platform provides easy and tight integration between Android applications and Google Maps . The well established Google Maps A...

Things Every Programmer Should Know
At ui-programming , one of our JCG program participant sites, articles about "Things Every Programmer Should Know" are occasionally posted....

Android Location Based Services Application – GPS location
With the incorporation of GPS devices in smartphones, Location Based Services (LBS) have become pretty hot the past few years. The iPhone ...

10 Tips for Proper Application Logging
CP partner , Tomasz Nurkiewicz , has submitted a number of posts describing the basic principles of proper application logging ...

GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial – Eclipse and Maven 2 showcase
A little while ago a friend and colleague of mine winged at me saying „Only half of the world is using Maven “. His statement struck me lik...

Java Best Practices – Vector vs ArrayList vs HashSet
Continuing our series of articles concerning proposed practices while working with the Java programming language, we are going to perform a...

Hate Java? You’re fighting the wrong battle.
One of the most interesting trends I’ve seen lately is the unpopularity of Java around blogs, DZone and others. It seems some people are e...

▼ 2012 (397)
► May (26)
▼ April (108)
JavaOne 2012 Analysis - Submitt Sp...
Writing modules for Play 2, part 2
JavaFX 2 GameTutorial Part 1
Word Count MapReduce with Akk
Frameworks vs Libraries as Inher Composit...
Runtime vs Compile-Time Classp
TeamCity Build Dependencies
HPROF - Memory leak analysis tu
Java Thread CPU analysis on Wir
The True Story of the Grid Engine
DBUnit, Spring and Annotations fo testin...
Using the final keyword on metho
Keep As Much Stuff As Possible I ...
High Performance Webapps - Dat
Aleri - Complex Event Processing
Maven Does Not Suck . . . but the
Which Private Cloud is Best and H
OSGI - Modularizing your applicat
Key Success Factors for Open So
JMeter: Load Testing Relational D
Hibernate Tip: Sort and Order
Java Swing to Android
Integrating Spring & JavaServer F T...
Java Concurrency with ReadWrite
JSON with GSON and abstract cla
Why Developers Keep Making Ba Choices
SOA example application
Setting up JNDI with Jetty (Embed
FXML: Custom components using
The Greatest Developer Fallacy C Words...
Complex Event Processing - a be
Being a better enterprise architec
What is ActiveMQ?
GlassFish 3.1.2 is Full of MOXy (E
ANTLR Tutorial - Hello Word
Connect to RabbitMQ (AMQP) us A...
AOP made easy with AspectJ and
Open Source Culture and Ideals
Java Enum puzzler
Application Lifecycle Managemen
JNDI and JPA without J2EE Conta
Lucene Overview Part One: Creat
MOXy as Your JAX-RS JSON Pro
Cost of Ownership Analysis: Orac Server...
Can you get more out of Static An
Fork and join in Java 7 - JSR 166
TestNG and Maven configuration
Writing modules for Play 2, part 1
Integration Testing with Selenium
IBM JVM tuning - gencon GC poli
Java EE Revisits Design Patterns
What is Spring Integration?

Threading stories: about robust th

Monitoring : Making sense of the

MOXy as Your JAX-RS JSON Pro

JRockit jrcmd tutorial

Play!ing (2.0) with Twitter Bootstra

Arquillian 1.0.0.Final released! Re

The Java EE 6 Example - Galleria

IntelliJ IDEA Tip: Configuration Pr

Are Static Imports Becoming Incre ...

Quartz scheduler misfire instructio

The Java EE 6 Example - Galleria

Serious about your software care job...

Drools Guvnor - Manage access

Taking over the Codebase, Solvin Cr...

Using slf4j with logback tutorial

Using Delayed queues in practice

What additional features does Jav mo...

End of ERP as we know it?

Lazy JSF Primefaces Datatable P

Why Math.round(0.499999999999 1 o...

Infrastructure, Technical Debt, an Test...

What Refactoring is, and what it is

Quartz scheduler plugins - hidden

Lazy JSF Primefaces Datatable P

Humour Time: Programming Puns

JAXB and Unmapped Properties

Java 7 #8: NIO.2 File Channels o

JSF 2, PrimeFaces 3, Spring 3 & Integr...

ADF: Using UI Categories with Dy

New JCG Features: Your Voice M

Beyond JUnit - Testing Framewor

5 Ways to Contribute to GlassFish

Exception Handling Guidelines & I

Building High Performance Applic

Spring Remoting Support with Htt

Exposing Functionality Over HTTP U...

The all-new Play Module Reposito

Protect a REST service using HM

The Strategy Pattern

RabbitMQ: Scheduled Message D

5 useful methods JSF developers

Play framework 2 quicktip: Scala c

Configuring Quartz with JDBCJob

Java 7's Support for Suppressed

MANIFEST.MF and feature.xml ve

JavaFX 2 Form with Tron effect

Java EE Revisits Design Patterns

Spring Remoting Support and Dev Service...

Seven NetBeans Hints for Modern

Oracle Propels JavaFX Forward b ...

Virtual Host + Apache httpd serve mod_...

Introduction to mutation testing wi

Java 7 - Project Coin Decompiled