Java EE

J2EE Day 4

```
package com.ubiteck.mysql;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import junit.framework.TestCase;

public class MySQLConnectionTest extends TestCase {

    public void testConnect() {
    String dbUrl = "jdbc:mysql://localhost/mydatabase";
    String dbClass = "com.mysql.jdbc.Driver";
    String query = "Select distinct(table_name) from INFORMATION_SCHEMA.TABLES";
    String username = "root";
    String password = "";
    try {

        Class.forName(dbClass);
        Connection connection = DriverManager.getConnection(dbUrl,
           username, password);
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query);
        while (resultSet.next()) {
        String tableName = resultSet.getString(1);
        System.out.println("Table name : " + tableName);
        }
        connection.close();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    }
}
```

Configuration Step1

In src-> main -> webapp -> WEB-INF -> webflow-config.xml, add the following :

This is to handle the persistence during the webflow flows :

Define the flowExecutor with references to our listeners :

```
<bean id="facesContextListener"
        class="org.springframework.faces.webflow.FlowFacesContextLifecycleListener" />

<webflow:flow-executor id="flowExecutor">
        <webflow:flow-execution-listeners>
                <webflow:listener ref="jpaFlowExecutionListener" />
                <webflow:listener ref="facesContextListener" />
        </webflow:flow-execution-listeners>
</webflow:flow-executor>

<webflow:flow-registry id="flowRegistry"
        flow-builder-services="facesFlowBuilderServices" base-path="/WEB-INF/flows">
        <webflow:flow-location-pattern value="/**/*-flow.xml" />
</webflow:flow-registry>
```

Configuration Step2

Create WEB-INF/flows folder

Configuration Step3

Add the following :

```
<faces:flow-builder-services id="facesFlowBuilderServices" development="true"/>
<faces:resources/>
<bean class="org.springframework.webflow.mvc.servlet.FlowHandlerMapping">
        <property name="order" value="1"/>
        <property name="flowRegistry" ref="flowRegistry" />
        <property name="defaultHandler">
                <bean class="org.springframework.web.servlet.mvc.UrlFilenameViewController" />
        </property>
</bean>
<bean class="org.springframework.faces.webflow.JsfFlowHandlerAdapter">
        <property name="flowExecutor" ref="flowExecutor" />
</bean>
<bean id="faceletsViewResolver" class="org.springframework.web.servlet.view.UrlBasedViewResolver">
        <property name="viewClass" value="org.springframework.faces.mvc.JsfView"/>
        <property name="prefix" value="/WEB-INF/" />
        <property name="suffix" value=".xhtml" />
</bean>
<bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />
<bean id="securityFlowExecutionListener" class="org.springframework.webflow.security.SecurityFlowExecutionListener" />
```

Configuration Step4

Create WEB-INF/faces-config.xml with the following content :

```
<faces-config version="2.0" 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">

        <application>
                <el-resolver>org.springframework.web.jsf.el.SpringBeanFacesELResolver</el-resolver>
        </application>
</faces-config>
```

Configuration Step5

We will do a spring security configuration right now but we can skip for now since we won't be using spring security right now.

Configuration Step6 : Logging Config

For logging configuration, under
src/main/resources, create a general file called log4java.properties

Add the following :

```
### direct log messages to stdout ###
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n
log4j.rootLogger=info, stdout
log4j.category.org.springframework=WARN
log4j.category.org.hibernate=WARN
log4j.category.org.springframework.webflow=DEBUG
log4j.category.org.springframework.binding=DEBUG
log4j.category.org.springframework.transaction=DEBUG
```

Configuration Step7

Under src/main/resources, create a folder called META-INF.

Under src/main/resources/META-INF, create an xml file called persistence.xml

Configuration Step8

We have to tell hibernate which entities are able to persist :

Add the common namespace configuration to persistence.xml that is required by hibernate for ORM as follows :

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence 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_1_0.xsd" version="1.0">

        <persistence-unit name="default" transaction-type="RESOURCE_LOCAL">

                <class>com.example.j2eeapp.domain.UserEntity</class>

        </persistence-unit>
</persistence>
```

Configuration Step9

Spring Elements is empty because we didn’t tell Spring framework about the existence of the configuration files so it will be easier to debug them.

So right-click on Spring Elements -> Properties

Leave defaults on Project Validators

Configuration Step10

In the Spring Elements -> Properties -> Spring -> Beans Support -> Add XML Config -> Open src -> main -> webapp -> WEB-INF ->

Select applicationContext.xml,

datasource-config.xml

and webflow-config.xml

Check Enable support for <import/> element in configuration files so we don’t receive warning from Spring Elements about import tags

Configuration Step11

Now you can expand all the beans under Spring Elements.

There will be a few warnings. To avoid warnings, create a Config Set that will make the config files aware of each other.

Under Spring Elements -> Properties -> Spring -> Beans Support -> Config Sets -> New -> Create a new config set with Name : Config and some of the warnings will go away

Also, if you expand Config under Spring Elements -> Config, you will see the beans that are defined in the config files

Configuration Step12

Folders and Files contain letters which show which ones are Spring managed configuration

The files which are related to flows will be marked as F instead of S

Configuration Step13

Create a folder called css under src->main->webapp->css

Configuration Step14

Create a folder called main under src->main->webapp->flows

Inside this folder, create a webflow definition file using Spring -> Spring Web Flow Definition File called main-flow.xml

Uncheck the 2nd checkbox -> Type in the flow id -> Finish

Configuration Step15

The folder for the flow files will be marked with an F and it will be added to Spring Elements in Project Explorer

Configuration Step16

We will define a view state and give it a unique id

This is a view that the user sees in the current moment.

Hover over view-state to get more information on view-state

Configuration Step17

Change the view-state id to welcome instead of start

We will be required to create an xhtml view file with Java Server Faces inside with the same name in the main flow folder so that it can be shown to the client.

The Spring plugins allow us to open flows in Spring Web Flow XML Editor

Configuration Step18

We can create view-states and subflow-states and decision-states but it is easier to edit the source files

Configuration Step19

Create a view with name welcome using

src->main->webapp->WEB-INFO->flows->main->File->New->Web -> HTML File -> welcome.xhtml

The templates should be New XHTML File (1.0 transitional)

Configuration Step20

Add the following namespaces to welcome.xhmtl :

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
        xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:p="http://primefaces.org/ui"
        template="/WEB-INF/templates/general.xhtml">
```

These are namespaces with facelets with the ui tag

primefaces is where we attach the jsf implementation

To use primefaces, we use the p tag

template has content which is common for webapplication that is shared between flows such as menu-bars, headers and footers.

Configuration Step21

In welcome.xhtml and other xhtml files, we can use html as well as jsf tags

In src->main->webapp->WEB-INF->templates, create a templates folder and then create a general.xhtml under the templates folder

Configuration Step22

Add the following JSF markup to general.xhtml to look as follows :

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<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:c="http://java.sun.com/jsp/jstl/core"
        xmlns:p="http://primefaces.org/ui"
    xmlns:ui="http://java.sun.com/jsf/facelets">
<f:view contentType="text/html">
      <h:head>
      </h:head>
      <h:body>
      </h:body>
</f:view>
</html>
```

Configuration Step23

Add the following inside <h:head> tag in general.xhtml

```
<h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>
    <!-- The following means that we can use the attribute name to insert title dynamically

    It's good practice to use the same name as the tagname

    We can use the title from any flow by referencing the name from any flow
    -->
    <ui:insert name="title" />

    </title>
</h:head>
```

Configuration Step24

Add the following inside <h:body> tag in general.xhtml

```
<h:body>
    <h1>WebFlow Facelets + </h1>
    <ui:insert name="body" />
</h:body>
```

Configuration Step25

Add the following into ui:composition tag in welcome.xhtml :

```
    <ui:define name="title"><h:outputText value="Main Flow" /></ui:define>

    <ui:define name="body"><h:outputText value="<h1>JSF Test Case</h1>" /></ui:define>
```

Configuration Step26

Change the index.jsp under src->main->webapp to redirect to the flow which we created with the views and templates

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<%@page import="java.util.Date"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>

<meta http-equiv="Refresh" content="0; URL=app/main" />
</head>
<body>



</body>
</html>
```

Configuration Step27

Check that the project uses JRE System Library (JavaSE-1.6) JDK instead of JRE

Configuration Step28