JSF - SPRING INTEGRATION

Spring provides special class DelegatingVariableResolver to integrate JSF and Spring together in seamless manner.

Following steps are required to integrate Spring Dependency Injection *IOC* feature in JSF

Step 1. Add DelegatingVariableResolver

Add a variable-resolver entry in faces-config.xml to point to spring class **DelegatingVariableResolver**.

```
<faces-config>
   <application>
   <variable-resolver>
      org.springframework.web.jsf.DelegatingVariableResolver
   </variable-resolver>
   ...
</faces-config>
```

Step 2. Add Context Listeners

Add **ContextLoaderListener** and **RequestContextListener** listener provided by spring framework in web.xml

```
<web-app>
   ...
   <!-- Add Support for Spring -->
   <listener>
      <listener-class>
         org.springframework.web.context.ContextLoaderListener
      </listener-class>
   </listener>
   <listener>
      <listener-class>
         org.springframework.web.context.request.RequestContextListener
      </listener-class>
   </listener>
   ...
</web-app>
```

Step 3. Define Dependency

Define bean*s* in applicationContext.xml which will be used as dependency in managed bean

```
<beans>
   <bean
      >
      <property name="message" value="Hello World!" />
   </bean>
</beans>
```

Step 4. Add Dependency

DelegatingVariableResolver first delegates value lookups to the default resolver of the JSF and then to Spring's WebApplicationContext. This allows one to easily inject spring based dependencies into one's JSF-managed beans.

We've injected messageService as spring based dependency here

```
<faces-config>
   ...
   <managed-bean>
      <managed-bean-name>userData</managed-bean-name>
      <managed-bean-class>com.tutorialspoint.test.UserData</managed-bean-class>
      <managed-bean-scope>request</managed-bean-scope>
      <managed-property>
         <property-name>messageService</property-name>
         <value>#{messageService}</value>
      </managed-property>
   </managed-bean>
</faces-config>
```

Step 5. Use Dependency

```
//jsf managed bean
public class UserData {
   //spring managed dependency
   private MessageService messageService;

   public void setMessageService(MessageService messageService) {
      this.messageService = messageService;
   }

   public String getGreetingMessage(){
      return messageService.getGreetingMessage();
   }
}
```

Example Application

Let us create a test JSF application to test spring integration.

Step	Description
1	Create a project with a name *helloworld* under a package *com.tutorialspoint.test* as explained in the *JSF - First Application* chapter.
2	Modify *pom.xml* as explained below.
3	Create *faces-config.xml* in *WEB-INF* folder as explained below.
4	Modify *web.xml* as explained below.
5	Create *applicationContext.xml* in *WEB-INF* folder as explained below.
6	Create *MessageService.java* under package *com.tutorialspoint.test* as explained below.
7	Create *MessageServiceImpl.java* under package *com.tutorialspoint.test* as explained below.
8	Create *UserData.java* under package *com.tutorialspoint.test* as explained below.
9	Modify *home.xhtml* as explained below. Keep rest of the files unchanged.
10	Compile and run the application to make sure business logic is working as per the requirements.
11	Finally, build the application in the form of war file and deploy it in Apache Tomcat Webserver.
12	Launch your web application using appropriate URL as explained below in the last step.

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/maven-v4_0_0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.tutorialspoint.test</groupId>
   <artifactId>helloworld</artifactId>
   <packaging>war</packaging>
   <version>1.0-SNAPSHOT</version>
   <name>helloworld Maven Webapp</name>
   <url>http://maven.apache.org</url>
   <dependencies>
      <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>3.8.1</version>
         <scope>test</scope>
      </dependency>
      <dependency>
         <groupId>com.sun.faces</groupId>
         <artifactId>jsf-api</artifactId>
         <version>2.1.7</version>
      </dependency>
      <dependency>
         <groupId>com.sun.faces</groupId>
         <artifactId>jsf-impl</artifactId>
         <version>2.1.7</version>
      </dependency>
      <dependency>
         <groupId>javax.servlet</groupId>
         <artifactId>jstl</artifactId>
         <version>1.2</version>
      </dependency>
      <dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-core</artifactId>
         <version>3.1.2.RELEASE</version>
      </dependency>
      <dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-web</artifactId>
         <version>3.1.2.RELEASE</version>
      </dependency>
   </dependencies>
   <build>
      <finalName>helloworld</finalName>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.3.1</version>
            <configuration>
               <source>1.6</source>
               <target>1.6</target>
            </configuration>
         </plugin>
         <plugin>
            <artifactId>maven-resources-plugin</artifactId>
            <version>2.6</version>
            <executions>
               <execution>
                  <id>copy-resources</id>
                  <phase>validate</phase>
                  <goals>
                     <goal>copy-resources</goal>
                  </goals>
                  <configuration>
                     <outputDirectory>${basedir}/target/helloworld/resources
                        </outputDirectory>
                     <resources>
```

```
                        <resource>
                           <directory>src/main/resources</directory>
                           <filtering>true</filtering>
                        </resource>
                     </resources>
                  </configuration>
               </execution>
            </executions>
         </plugin>
      </plugins>
   </build>
</project>
```

faces-config.xml

```
<?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">
   <application>
      <variable-resolver>
         org.springframework.web.jsf.DelegatingVariableResolver
      </variable-resolver>
   </application>
   <managed-bean>
      <managed-bean-name>userData</managed-bean-name>
      <managed-bean-class>com.tutorialspoint.test.UserData</managed-bean-class>
      <managed-bean-scope>request</managed-bean-scope>
      <managed-property>
         <property-name>messageService</property-name>
         <value>#{messageService}</value>
      </managed-property>
   </managed-bean>
</faces-config>
```

web.xml

```
<!DOCTYPE web-app PUBLIC
   "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd" >
<web-app>
   <display-name>Archetype Created Web Application</display-name>

   <context-param>
       <param-name>javax.faces.PROJECT_STAGE</param-name>
       <param-value>Development</param-value>
   </context-param>
   <!-- Add Support for Spring -->
   <listener>
       <listener-class>
          org.springframework.web.context.ContextLoaderListener
       </listener-class>
   </listener>
   <listener>
      <listener-class>
         org.springframework.web.context.request.RequestContextListener
      </listener-class>
   </listener>
   <servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>*.jsf</url-pattern>
   </servlet-mapping>
</web-app>
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
   "http://www.springframework.org/dtd/spring-beans-2.0.dtd">
<beans>
   <bean
      >
      <property name="message" value="Hello World!" />
   </bean>
</beans>
```

MessageService.java

```
package com.tutorialspoint.test;

public interface MessageService {
   String getGreetingMessage();
}
```

MessageServiceImpl.java

```
package com.tutorialspoint.test;

public class MessageServiceImpl implements MessageService {

   private String message;

   public String getGreetingMessage() {
      return message;
   }
   public String getMessage() {
      return message;
   }
   public void setMessage(String message) {
      this.message = message;
   }
}
```

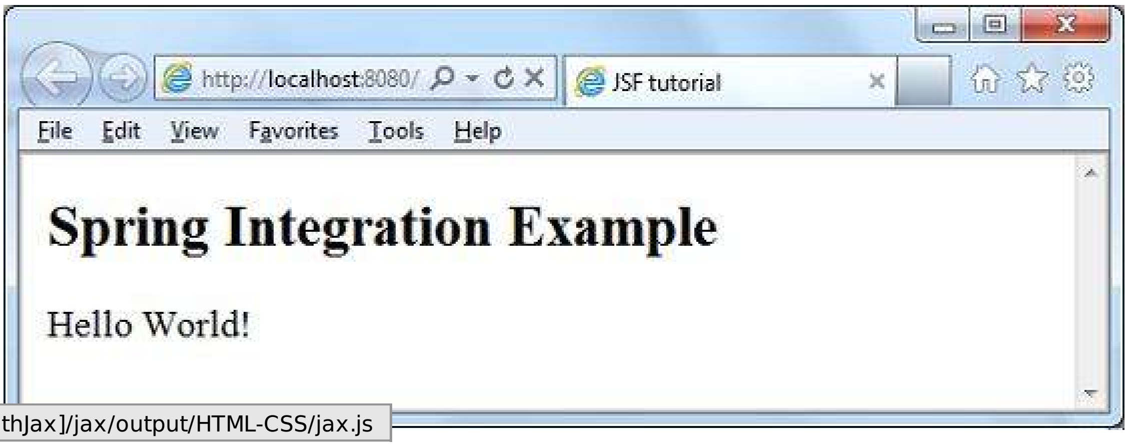
UserData.java

```
package com.tutorialspoint.test;

import java.io.Serializable;

public class UserData implements Serializable {

   private static final long serialVersionUID = 1L;

   private MessageService messageService;

   public MessageService getMessageService() {
      return messageService;
   }

   public void setMessageService(MessageService messageService) {
      this.messageService = messageService;
   }

   public String getGreetingMessage(){
      return messageService.getGreetingMessage();
   }
}
```

home.xhtml

```
<?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:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>JSF Tutorial!</title>
   </h:head>
   <h:body>
   <h2>Spring Integration Example</h2>
    #{userData.greetingMessage}
   </h:body>
</html>
```

Once you are ready with all the changes done, let us compile and run the application as we did in JSF - First Application chapter. If everything is fine with your application, this will produce following result:



Spring Integration Example

Hello World!