JSF - VALUECHANGELISTENER

When user interacts with input components, such as h:inputText or h:selectOneMenu, the JSF fires a valueChangeEvent which can be handled in two ways.

Technique	Description
Method Binding	Pass the name of the managed bean method in *valueChangeListener* attribute of UI Component.
ValueChangeListener	Implement ValueChangeListener interface and pass the implementation class name to *valueChangeListener* attribute of UI Component.

Method Binding

Define a method

```
public void localeChanged(ValueChangeEvent e){
   //assign new value to country
   selectedCountry = e.getNewValue().toString();
}
```

Use above method

```
<h:selectOneMenu value="#{userData.selectedCountry}"  onchange="submit()"
   valueChangeListener="#{userData.localeChanged}" >
   <f:selectItems value="#{userData.countries}" />
</h:selectOneMenu>
```

ValueChangeListener

Implement ValueChangeListener

```
public class LocaleChangeListener implements ValueChangeListener {
   @Override
   public void processValueChange(ValueChangeEvent event)
      throws AbortProcessingException {
     //access country bean directly
     UserData userData = (UserData) FacesContext.getCurrentInstance().
        getExternalContext().getSessionMap().get("userData");
     userData.setSelectedCountry(event.getNewValue().toString());
   }
}
```

Use listener method

```
<h:selectOneMenu value="#{userData.selectedCountry}" onchange="submit()">
   <f:valueChangeListener type="com.tutorialspoint.test.LocaleChangeListener"
      />
   <f:selectItems value="#{userData.countries}" />
</h:selectOneMenu>
```

Example Application

Let us create a test JSF application to test the valueChangeListener in JSF.

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 *UserData.java* file as explained below.
3	Create *LocaleChangeListener.java* file under a package *com.tutorialspoint.test*.Modify it as explained below
4	Modify *home.xhtml* as explained below. Keep rest of the files unchanged.
5	Compile and run the application to make sure business logic is working as per the requirements.
6	Finally, build the application in the form of war file and deploy it in Apache Tomcat Webserver.
7	Launch your web application using appropriate URL as explained below in the last step.

UserData.java

```
package com.tutorialspoint.test;

import java.io.Serializable;
import java.util.LinkedHashMap;
import java.util.Map;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.event.ValueChangeEvent;

@ManagedBean(name = "userData", eager = true)
@SessionScoped
public class UserData implements Serializable {

private static final long serialVersionUID = 1L;

   private static Map<String,String> countryMap;

   private String selectedCountry = "United Kingdom"; //default value

   static{
      countryMap = new LinkedHashMap<String,String>();
      countryMap.put("en", "United Kingdom"); //locale, country name
      countryMap.put("fr", "French");
      countryMap.put("de", "German");
   }

   public void localeChanged(ValueChangeEvent e){
      //assign new value to country
      selectedCountry = e.getNewValue().toString();
   }

   public Map<String, String> getCountries() {
      return countryMap;
   }

   public String getSelectedCountry() {
      return selectedCountry;
   }

   public void setSelectedCountry(String selectedCountry) {
      this.selectedCountry = selectedCountry;
   }
}
```

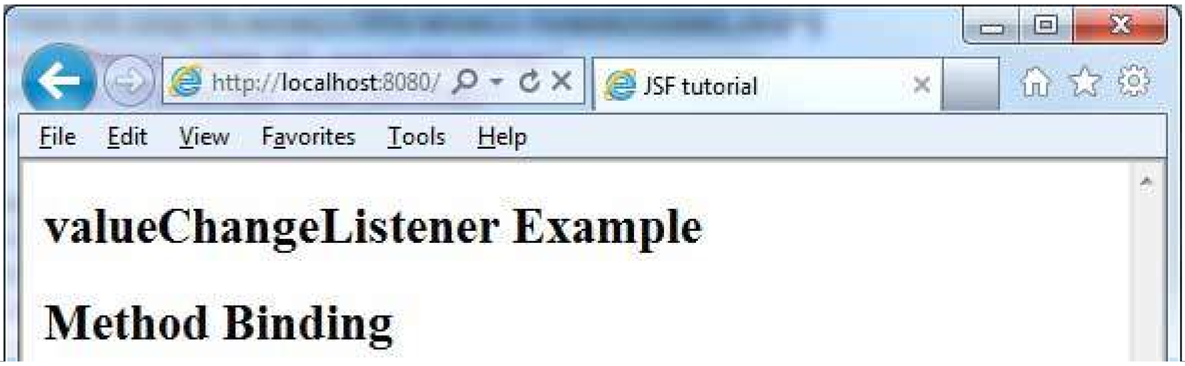
LocaleChangeListener.java

```
package com.tutorialspoint.test;

import javax.faces.context.FacesContext;
import javax.faces.event.AbortProcessingException;
import javax.faces.event.ValueChangeEvent;
import javax.faces.event.ValueChangeListener;

public class LocaleChangeListener implements ValueChangeListener {
   @Override
   public void processValueChange(ValueChangeEvent event)
      throws AbortProcessingException {
      //access country bean directly
      UserData userData = (UserData) FacesContext.getCurrentInstance().
         getExternalContext().getSessionMap().get("userData");

      userData.setSelectedCountry(event.getNewValue().toString());
   }
}
```

home.xhtml

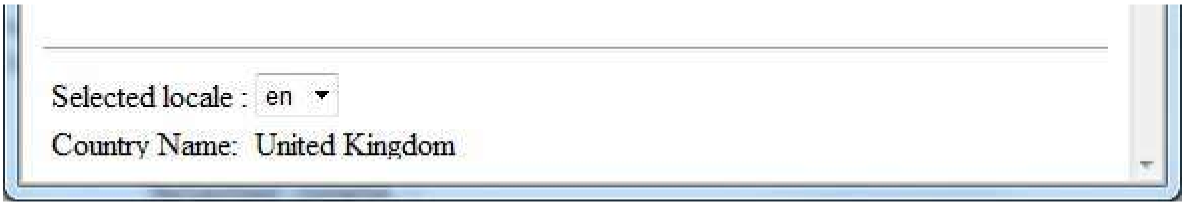
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core">
   <h:head>
      <title>JSF tutorial</title>
   </h:head>
   <h:body>
      <h2>valueChangeListener Examples</h2>
      <h:form>
      <h2>Method Binding</h2>
      <hr/>
      <h:panelGrid columns="2">
         Selected locale :
         <h:selectOneMenu value="#{userData.selectedCountry}"
            onchange="submit()"
            valueChangeListener="#{userData.localeChanged}" >
            <f:selectItems value="#{userData.countries}" />
         </h:selectOneMenu>
         Country Name:
         <h:outputText
            size="20" />
      </h:panelGrid>
      </h:form>
    </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:



valueChangeListener Example

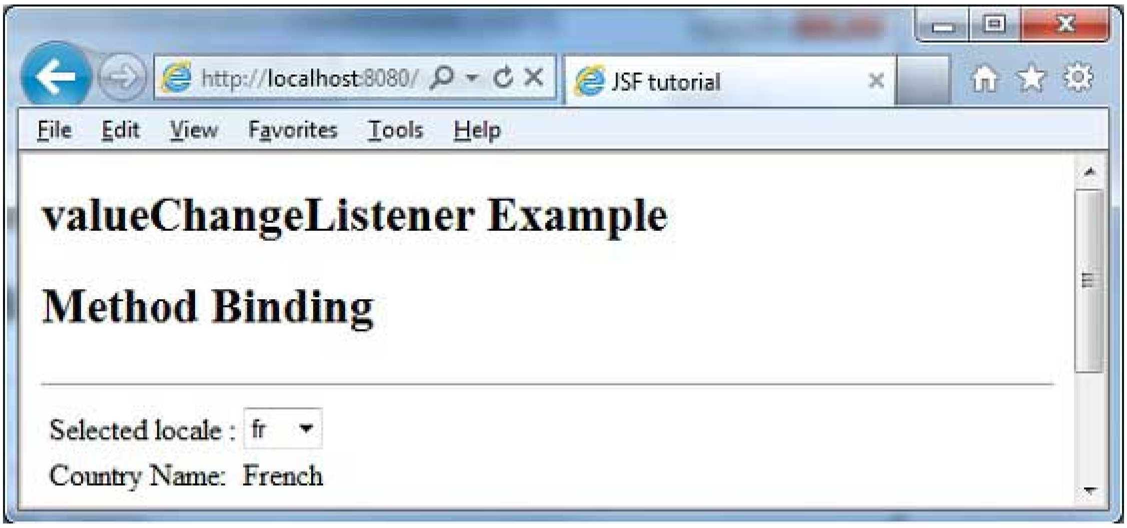
Method Binding



Selected locale : en

Country Name: United Kingdom

Select locale. You will see the following result.



valueChangeListener Example

Method Binding

Selected locale : fr

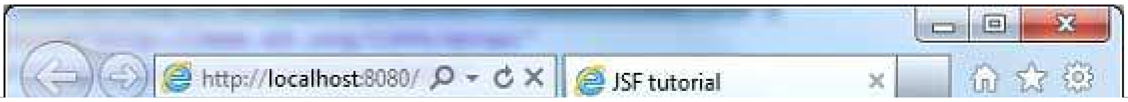
Country Name: French

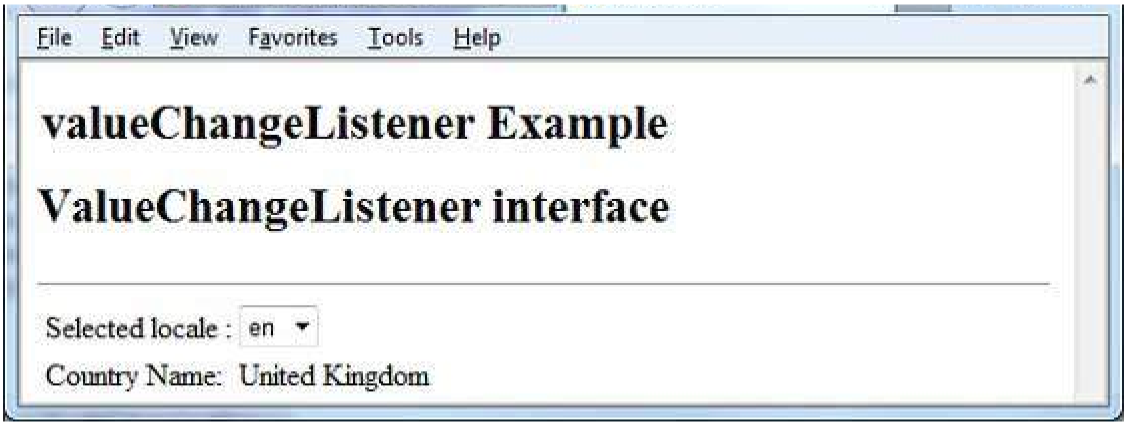
Modify *home.xhtml* again in deployed directory where you've deployed the application as explained below. Keep rest of the files unchanged.

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:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core">
   <h:head>
      <title>JSF tutorial</title>
   </h:head>
   <h:body>
      <h2>valueChangeListener Examples</h2>
      <h:form>
      <h2>ValueChangeListener interface</h2>
      <hr/>
      <h:panelGrid columns="2">
         Selected locale :
         <h:selectOneMenu value="#{userData.selectedCountry}"
            onchange="submit()">
            <f:valueChangeListener
            type="com.tutorialspoint.test.LocaleChangeListener" />
            <f:selectItems value="#{userData.countries}" />
         </h:selectOneMenu>
         Country Name:
         <h:outputText
            size="20" />
      </h:panelGrid>
</h:form>
</h:body>
</html>
```

Once you are ready with all the changes done, refresh the page in browser. If everything is fine with your application, this will produce following result:





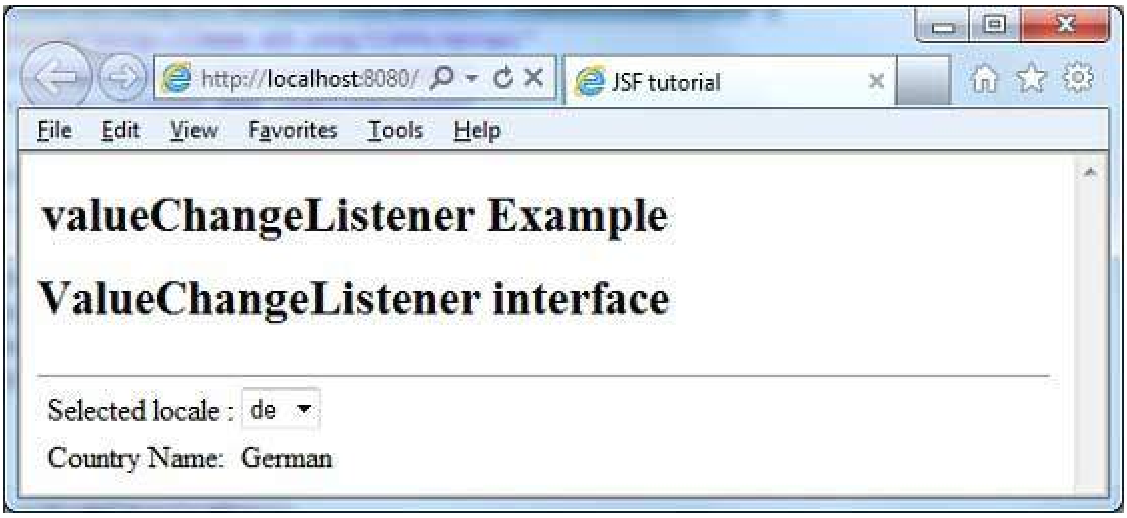
valueChangeListener Example

ValueChangeListener interface

Selected locale : en

Country Name: United Kingdom

Select locale. You will see the following result.



valueChangeListener Example

ValueChangeListener interface

Selected locale : de

Country Name: German