Internationalization

Overview

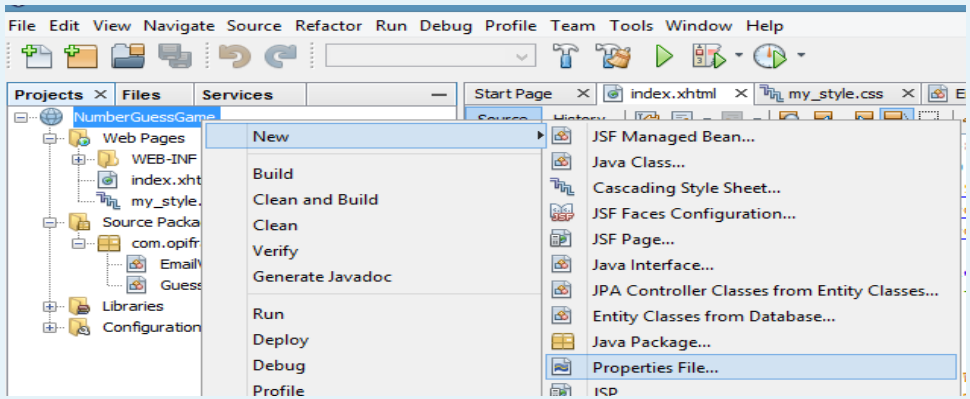
- At this point you should have a basic knowledge of JSF technology, basic UI components and injecting or binding managed Bean data to UI components.
- Now it is time to check how can you localize your web applications.

Getting Started

- Append new folder to your project. Names it as ”resources”, and don't make any misspellings.
- GlassFish server recognizes this resources folder and set it to be the default place to search the images, localization files etc.

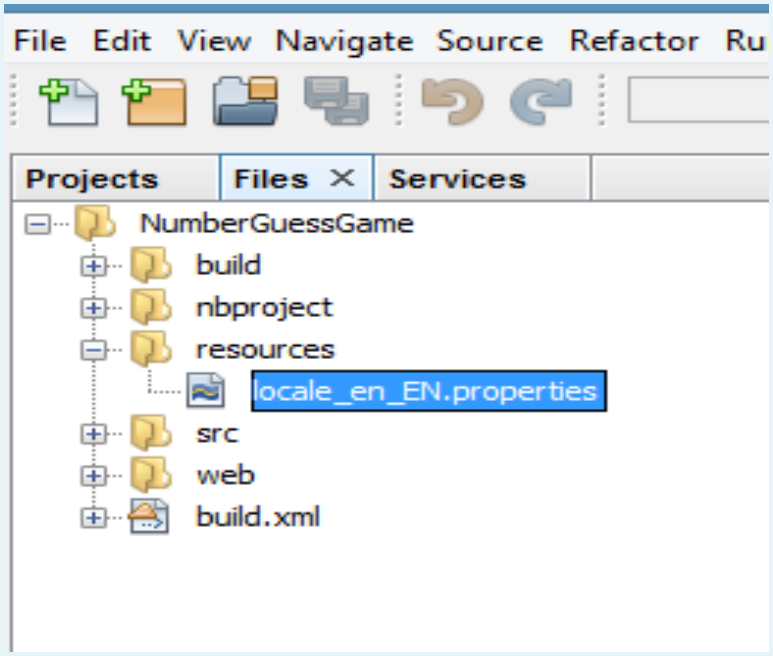
Define default locale

- I will use English as my default locale. To create a localization file right mouse click over the resources folder, select New → Properties file



Define default locale

- Give some name for the file i.e. locale_en_EN



Configuration

- Append faces-conifg.xml file. Right mouse click on the project select New ->JSF Faces Configuration

Faces-config.xml

```
<application>
    <locale-config>
        <default-locale>
            en
        </default-locale>
    </locale-config>
    <resource-bundle>
        <base-name>
            locale
        </base-name>
        <var>
            msg
        </var>
    </resource-bundle>
</application>
```

Localized Text

- Define few string in locale file (locale_en_EN.properties)

```
welcome=Welcome to number guess game!
welcome2=Try to guess number between 1-100.
```

Localization bean

- Next we define a bean that will set the locale depending of the language setting in user browser...

LanguageBean .java

```
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ValueChangeEvent;

@ManagedBean(name="language")
@SessionScoped
public class LanguageBean implements Serializable{

	private static final long serialVersionUID = 1L;

	private String localeCode;

	private static Map<String,Object> countries;
	static{
		countries = new LinkedHashMap<String,Object>();
		countries.put("English", Locale.ENGLISH); //label, value
		countries.put("Chinese", Locale.SIMPLIFIED_CHINESE);
	}
```

LanguageBean .java

```
public Map<String, Object> getCountriesInMap() {
        return countries;
}


public String getLocaleCode() {
        return localeCode;
}


public void setLocaleCode(String localeCode) {
        this.localeCode = localeCode;
}

//value change event listener
public void countryLocaleCodeChanged(ValueChangeEvent e){

        String newLocaleValue = e.getNewValue().toString();

        //loop country map to compare the locale code
        for (Map.Entry<String, Object> entry : countries.entrySet()) {

           if(entry.getValue().toString().equals(newLocaleValue)){

                FacesContext.getCurrentInstance()
                        .getViewRoot().setLocale((Locale)entry.getValue());

           }
        }
}
```

Localized Text

- Use text from localization files in your UI elements

```
<h:head>
    <title>Facelet Title</title>
    <link href="my_style.css" type="text/css" rel="stylesheet"/>
</h:head>
<h:body>
    <h:form id="guessForm">
        <h:outputLabel value="#{msg['welcome']}"></h:outputLabel><br/>
        <h:outputLabel value="#{msg['welcome2']}"></h:outputLabel><br/>
        <!--<h:outputLabel value="Your Email Address"></h:outputLabel><br/>-->
        <!--<h:inputText id="emailText" required="true" value="">
            <f:validator validatorId="com.opiframe.beans.EmailValidator"></f:validator>
        </h:inputText>
        <h:message for="emailText" style="color:red" /><br/>-->
        <h:outputLabel value="Your guess"></h:outputLabel><br/>
        <h:inputText id="guessValue" required="true" value="#{guessBean.userNumber}">
            <f:validateLongRange minimum="1" maximum="100"></f:validateLongRange>
        </h:inputText><br/>
        <h:message for="guessValue"></h:message><br/>
```