JSF - CUSTOM CONVERTER

 Copyright © tutorialspoint.com

We can create our own Custom convertor in JSF.

Defining a custom converter in JSF is a three step process

Step No.	Description
1	Create a converter class by implementing *javax.faces.convert.Converter* interface.
2	Implement getAsObject and getAsString methods of above interface.
3	Use Annotation @FacesConvertor to assign a unique id to the custom convertor.

Step 1: Create a converter class : UrlConverter.java

```
public class UrlConverter implements Converter {
...
}
```

Step 2: Implement Converter interface methods : UrlConverter.java

Create a simple class to store data: UrlData. This class will store a URL string.

```
public class UrlData {

   private String url;

   public UrlData(String url){
      this.url = url;
   }
   ...
}
```

Use UrlData in getAsObject method.

```
public class UrlConverter implements Converter {
   @Override
   public Object getAsObject(FacesContext facesContext,
      UIComponent component, String value) {
      ...
      UrlData urlData = new UrlData(url.toString());
      return urlData;
   }

   @Override
   public String getAsString(FacesContext facesContext,
      UIComponent component, Object value) {
      return value.toString();
   }
}
```

Step 3: Annotate to register the convertor : UrlConverter.java

```
@FacesConverter("com.tutorialspoint.test.UrlConverter")
public class UrlConverter implements Converter {
}
```

Use the convertor in JSF page

```
<h:inputText  >
   <f:converter converterId="com.tutorialspoint.test.UrlConverter" />
</h:inputText>
```

Example Application

Let us create a test JSF application to test the above tag.

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

UrlData.java

```
package com.tutorialspoint.test;

public class UrlData {
   private String url;

   public UrlData(String url){
      this.url = url;
   }

   public String getUrl() {
      return url;
   }

   public void setUrl(String url) {
      this.url = url;
   }

   public String toString(){
      return url;
   }
}
```

UrlConvertor.java

```
package com.tutorialspoint.test;

import java.net.URI;
```

```
import java.net.URISyntaxException;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.ConverterException;
import javax.faces.convert.FacesConverter;

@FacesConverter("com.tutorialspoint.test.UrlConverter")
public class UrlConverter implements Converter {

   @Override
   public Object getAsObject(FacesContext facesContext,
      UIComponent component, String value) {

      StringBuilder url = new StringBuilder();

      if(!value.startsWith("http://", 0)){
         url.append("http://");
      }
      url.append(value);

      try {
         new URI(url.toString());
      } catch (URISyntaxException e) {
         FacesMessage msg = new FacesMessage("Error converting URL",
            "Invalid URL format");
         msg.setSeverity(FacesMessage.SEVERITY_ERROR);
         throw new ConverterException(msg);
      }

      UrlData urlData = new UrlData(url.toString());
      return urlData;
   }

   @Override
   public String getAsString(FacesContext facesContext,
      UIComponent component, Object value) {
         return value.toString();
   }
}
```

UserData.java

```
package com.tutorialspoint.test;

import java.io.Serializable;

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

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

private static final long serialVersionUID = 1L;

   public UrlData data;

   public UrlData getData() {
      return data;
   }

   public void setData(UrlData data) {
      this.data = data;
   }
}
```

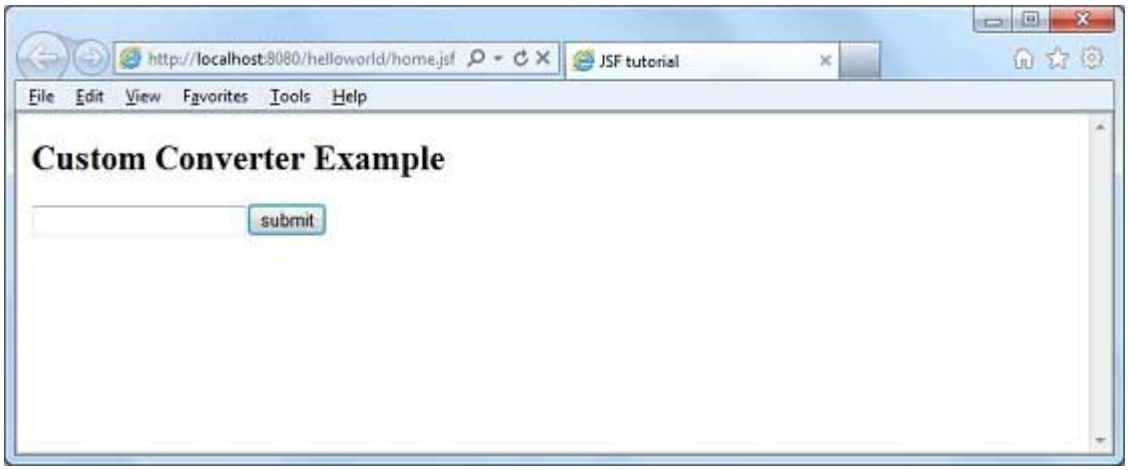
home.xhtml

```
<?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>Custom Converter Example</h2>
      <h:form>
         <h:inputText
            label="URL" >
            <f:converter converterId="com.tutorialspoint.test.UrlConverter" />
         </h:inputText>
         <h:commandButton value="submit" action="result"/>
         <h:message for="urlInput" style="color:red" />
      </h:form>
   </h:body>
</html>
```

result.xhtml

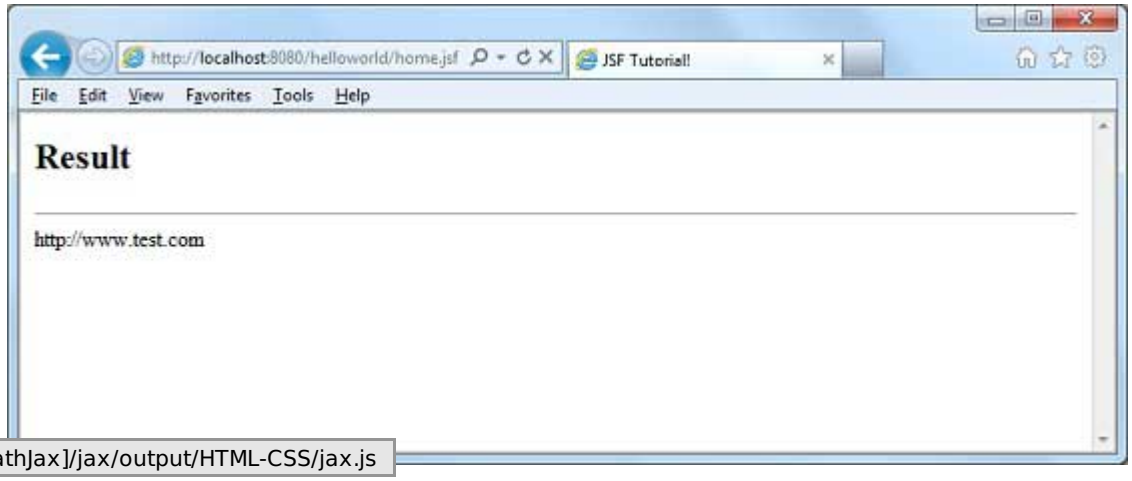
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets">
   <h:body>
      <h2>Result</h2>
      <hr />
      #{userData.data}
   </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:



Enter any invalid value and press submit button.See the error message.

Enter any valid value and press submit button.See the result.



Loading [MathJax]/jax/output/HTML-CSS/jax.js