Creating Custom Components

Creating Custom Components

- JavaServer Faces technology offers a basic set of standard, reusable UI components that enable quick and easy construction of user interfaces for web applications.
- But often an application requires a component that has additional functionality or requires acompletely new component.
- JavaServer Faces technology allows extension of standard components to enhance their functionality or to create custom components.

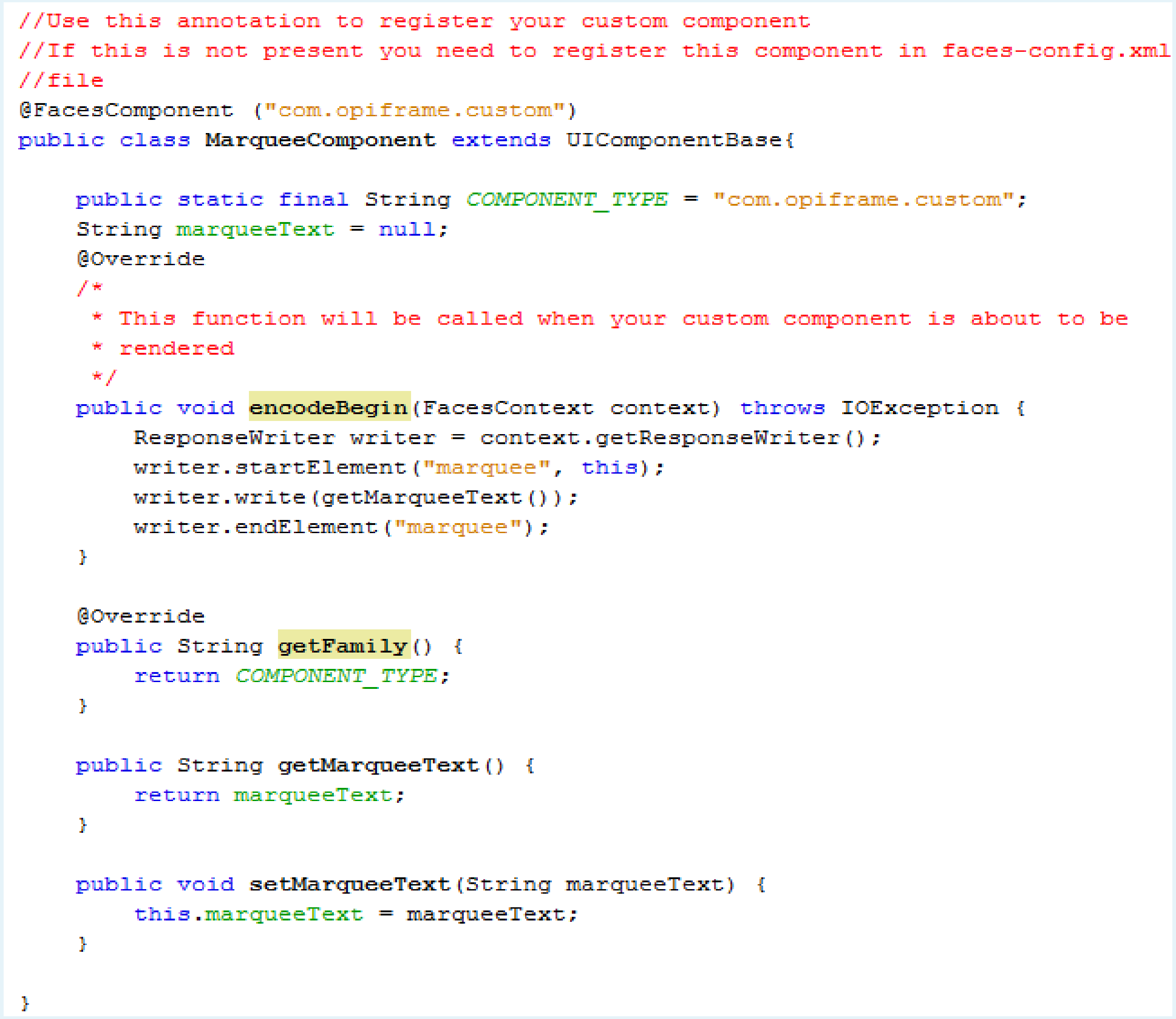
Creating Custom Components

- JavaServer Faces technology provides the ability to create custom components by extending the UIComponentBase class, the base class for all standard UI components.
- A custom component can be used anywhere an ordinary component can be used, such as within a composite component.

Creating Custom Components

- Step1: Create class extending UIComponentBase

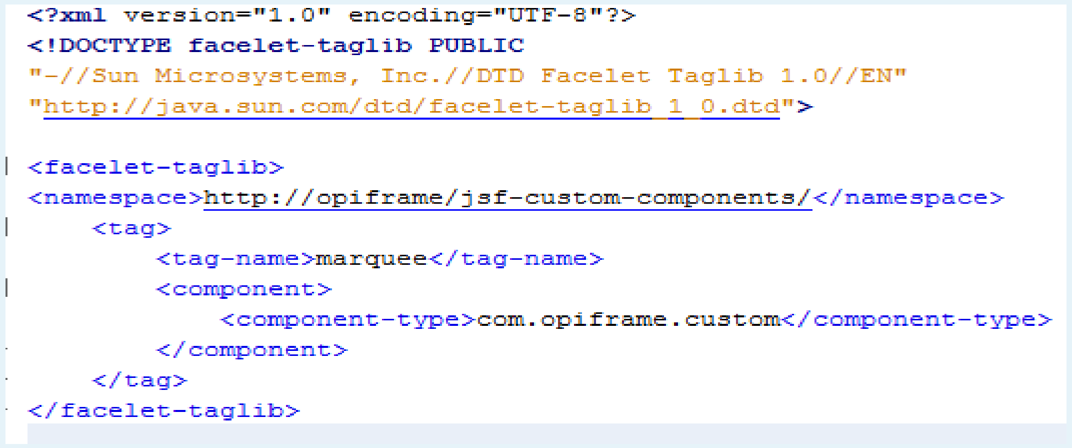
Creating Custom Components



```java
//Use this annotation to register your custom component
//If this is not present you need to register this component in faces-config.xml
//file
@FacesComponent ("com.opiframe.custom")
public class MarqueeComponent extends UIComponentBase{

    public static final String COMPONENT_TYPE = "com.opiframe.custom";
    String marqueeText = null;
    @Override
    /*
     * This function will be called when your custom component is about to be
     * rendered
     */
    public void encodeBegin(FacesContext context) throws IOException {
        ResponseWriter writer = context.getResponseWriter();
        writer.startElement("marquee", this);
        writer.write(getMarqueeText());
        writer.endElement("marquee");
    }

    @Override
    public String getFamily() {
        return COMPONENT_TYPE;
    }

    public String getMarqueeText() {
        return marqueeText;
    }

    public void setMarqueeText(String marqueeText) {
        this.marqueeText = marqueeText;
    }

}
```

Creating Custom Components

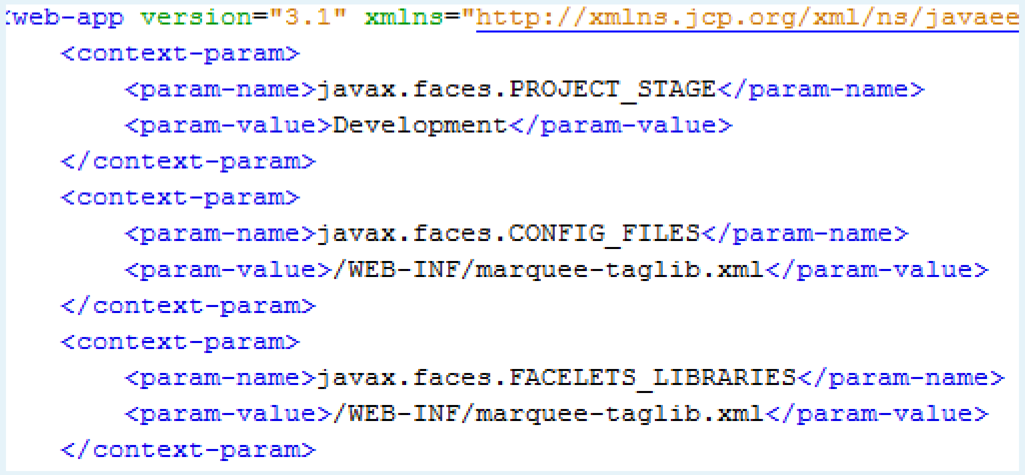
- Define a tag library for your component
  - Create an .xml file under WEB-INF folder
  - You can name the file as you like. In the example it is named marquee-taglib.xml
  - Append next content to file



```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE facelet-taglib PUBLIC
"-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"
"http://java.sun.com/dtd/facelet-taglib_1_0.dtd">

<facelet-taglib>
<namespace>http://opiframe/jsf-custom-components/</namespace>
    <tag>
        <tag-name>marquee</tag-name>
        <component>
            <component-type>com.opiframe.custom</component-type>
        </component>
    </tag>
</facelet-taglib>
```

Creating Custom Components

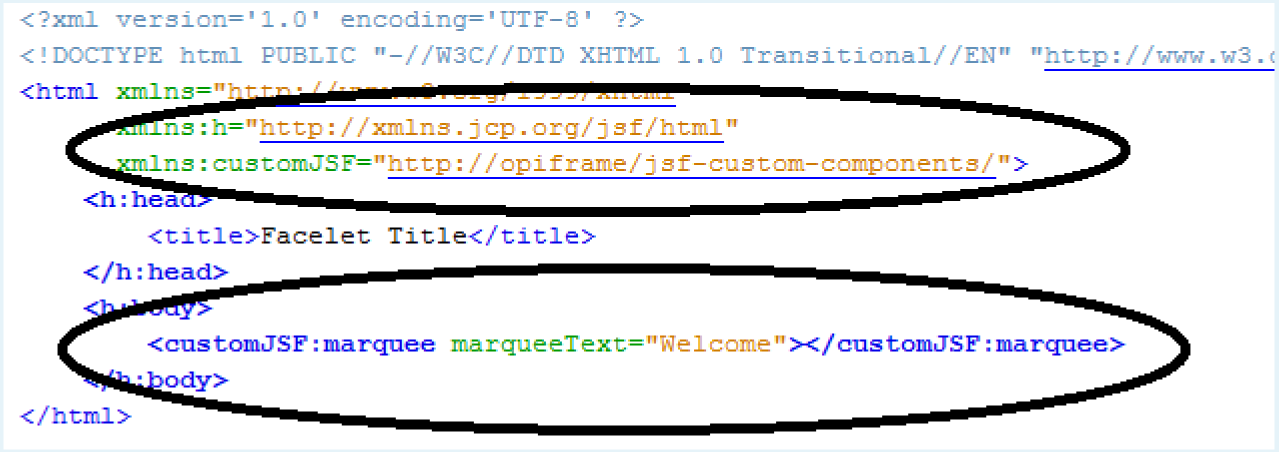
- Expose your tag library by defining it to web.xml file



```
:web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <context-param>
        <param-name>javax.faces.CONFIG_FILES</param-name>
        <param-value>/WEB-INF/marquee-taglib.xml</param-value>
    </context-param>
    <context-param>
        <param-name>javax.faces.FACELETS_LIBRARIES</param-name>
        <param-value>/WEB-INF/marquee-taglib.xml</param-value>
    </context-param>
```

Creating Custom Components

- Then use your component...



```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:customJSF="http://opiframe/jsf-custom-components/">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <customJSF:marquee marqueeText="Welcome"></customJSF:marquee>
    </h:body>
</html>
```