PrimeFaces

Markus Veijola

2013

Introduction

- PrimeFaces is an open source JSF component suite with various extensions.
  - Rich set of UI components (Accordion, charts etc...)
  - Built-in Ajax based on standard JSF 2.0 Ajax APIs.
  - Lightweight, one jar, zero-configuration and no required dependencies.
  - Mobile UI kit to create mobile web applications.
  - Skinning Framework with 35+ built-in themes

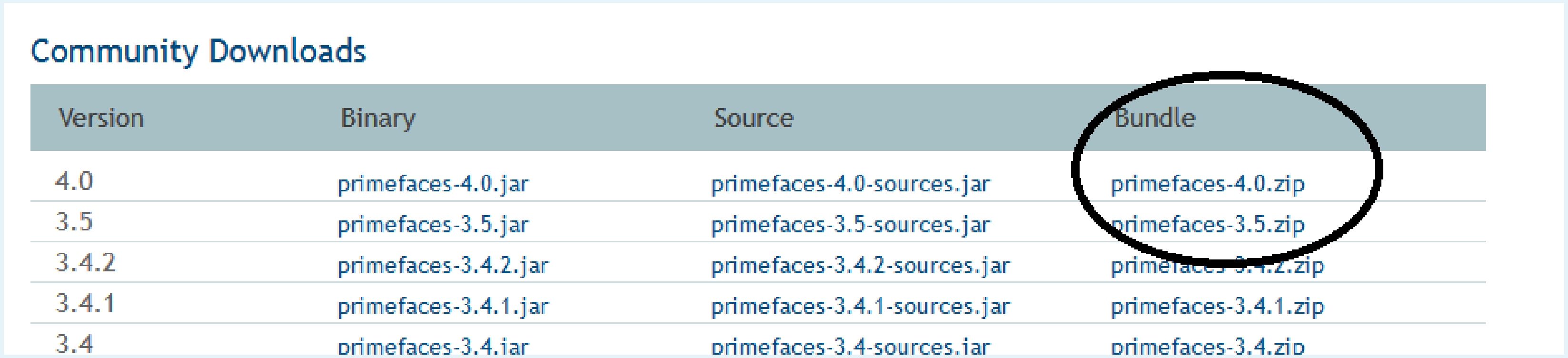
Introduction

- PrimeFaces is *currently the most popular JavaServer Faces component suite* and many of the widgets are p*owered by jQuery UI.*
- You can find prime faces tag library documentation from here

  http://www.primefaces.org/docs/vdl/4.0/
- Icons and other build in themes can be found from here Themes
-

Installation

- Download primefaces zip from here: http://primefaces.org/downloads.html



Community Downloads

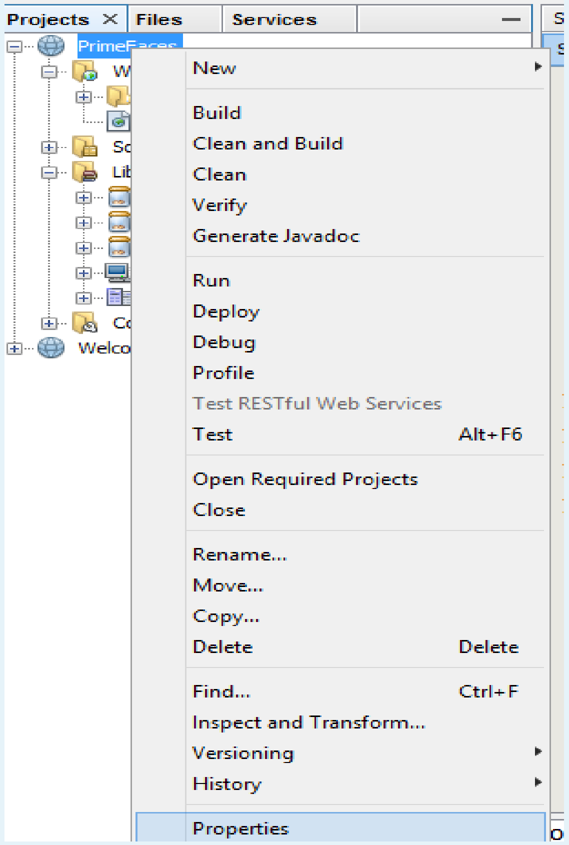
Version	Binary	Source	Bundle
4.0	primefaces-4.0.jar	primefaces-4.0-sources.jar	primefaces-4.0.zip
3.5	primefaces-3.5.jar	primefaces-3.5-sources.jar	primefaces-3.5.zip
3.4.2	primefaces-3.4.2.jar	primefaces-3.4.2-sources.jar	primefaces-3.4.2.zip
3.4.1	primefaces-3.4.1.jar	primefaces-3.4.1-sources.jar	primefaces-3.4.1.zip
3.4	primefaces-3.4.jar	primefaces-3.4-sources.jar	primefaces-3.4.zip

Installation

- Unzip the .zip file from downloads folder (you can unzip it anywhere you want as long as you remember the location).
- Open NetBeans IDE
- Create a JavaEE Web Project with JSF framework included.

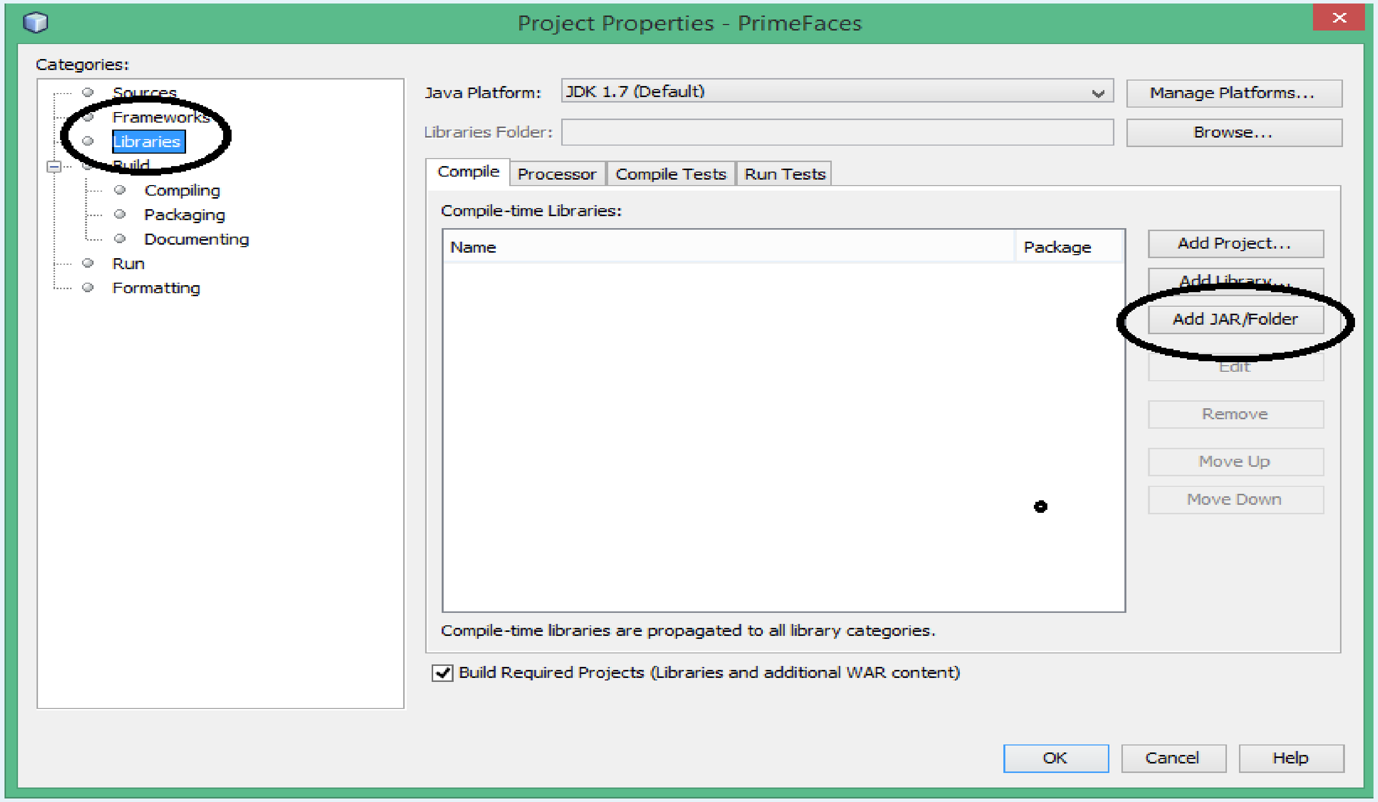
Installation

- Now right click over you project and select properties...



Installation

- Select Libraries and press Add JAR/Folders button



Project Properties - PrimeFaces

Categories:

- Sources
- Frameworks
- Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Formatting

Java Platform: JDK 1.7 (Default)

Libraries Folder:

Compile-time Libraries:

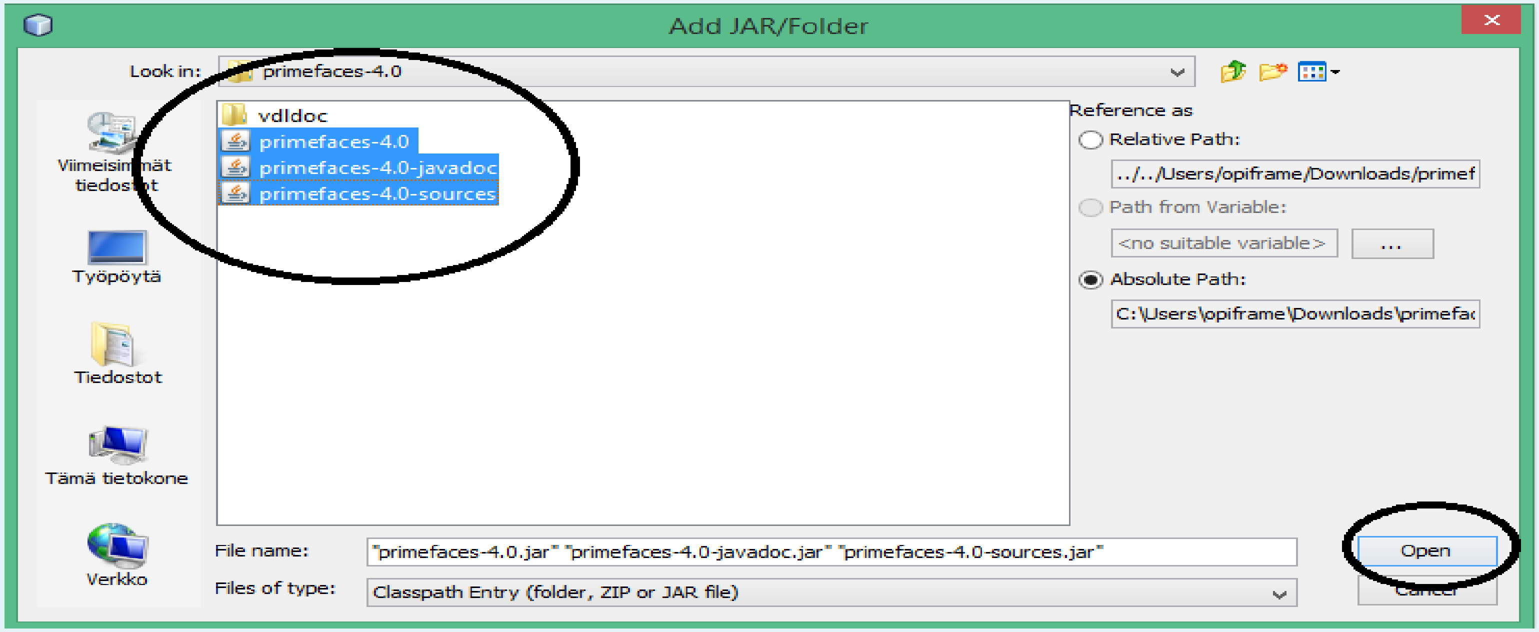
Name	Package

Compile-time libraries are propagated to all library categories.

[✓] Build Required Projects (Libraries and additional WAR content)

Installation

- Now browse to the folder where you unzipped the primefaces.zip file. Select all the .jar files and press Open button.



Add JAR/Folder

Look in: primefaces-4.0

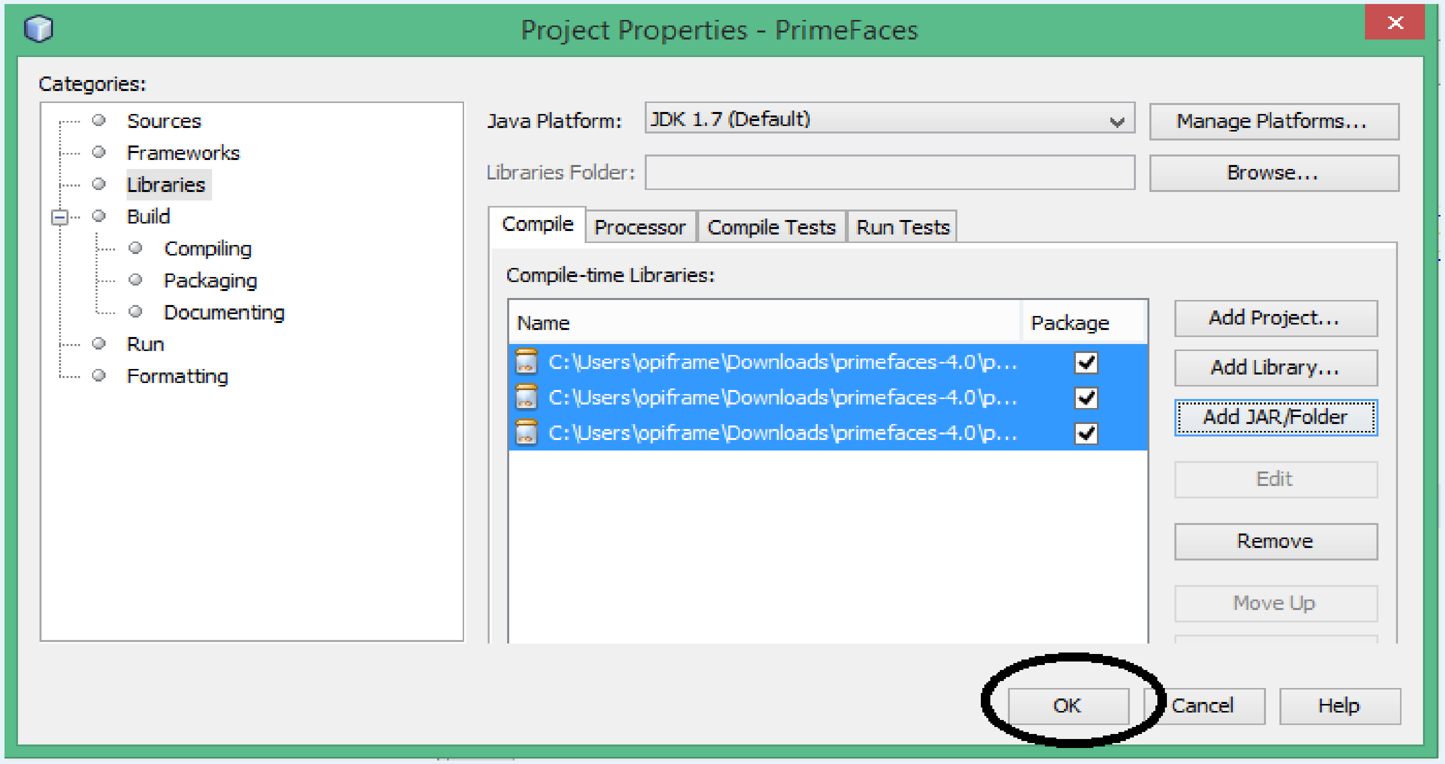
vdldoc
primefaces-4.0
primefaces-4.0-javadoc
primefaces-4.0-sources

Reference as
Relative Path: ../../Users/opiframe/Downloads/primef
Path from Variable: <no suitable variable>
[✓ Absolute Path:] C:\Users\opiframe\Downloads\primefa

File name: "primefaces-4.0.jar" "primefaces-4.0-javadoc.jar" "primefaces-4.0-sources.jar"
Files of type: Classpath Entry (folder, ZIP or JAR file)

Installation

- In the next panel just press Ok...



Project Properties - PrimeFaces

Categories:

- Sources
- Frameworks
- Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Formatting

Java Platform: JDK 1.7 (Default)

Libraries Folder:

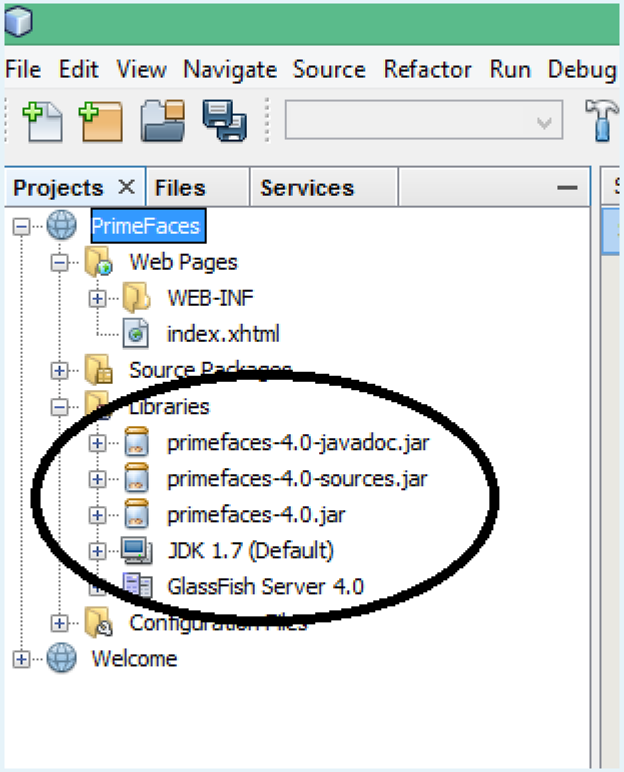
Compile | Processor | Compile Tests | Run Tests

Compile-time Libraries:

Name	Package
C:\Users\opiframe\Downloads\primefaces-4.0\p...	✓
C:\Users\opiframe\Downloads\primefaces-4.0\p...	✓
C:\Users\opiframe\Downloads\primefaces-4.0\p...	✓

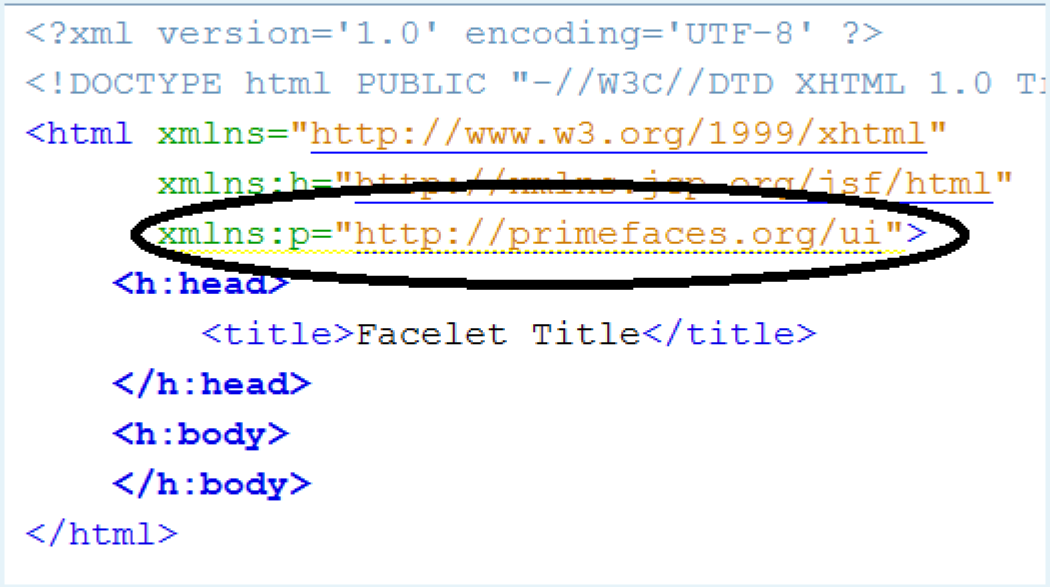
Installation

- Make sure that the libraries are visible in project tree library folder...



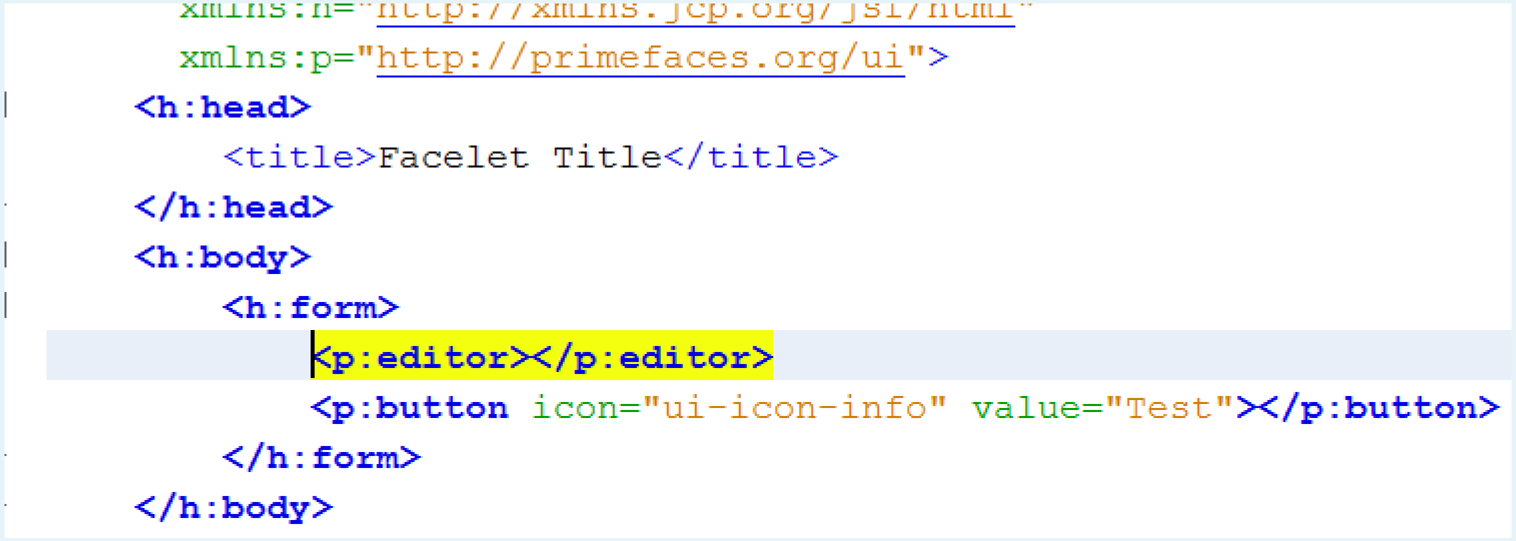
Using Prime Face Components

- Append the prime faces namespace in your .xhtml file

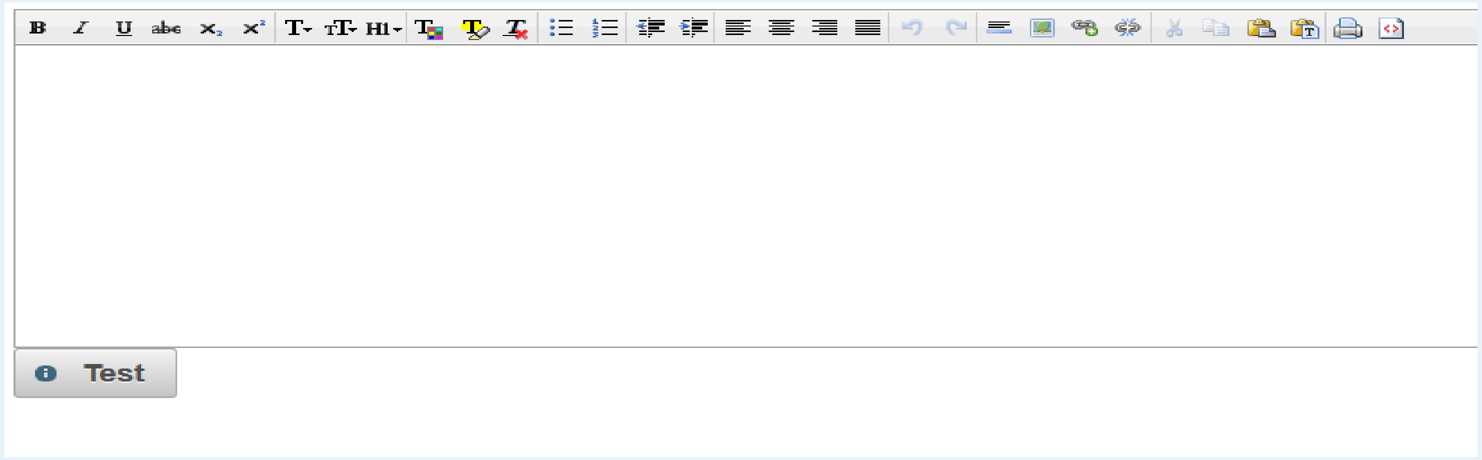


```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 T
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
    </h:body>
</html>
```

Example



```
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <h:form>
            <p:editor></p:editor>
            <p:button icon="ui-icon-info" value="Test"></p:button>
        </h:form>
    </h:body>
```



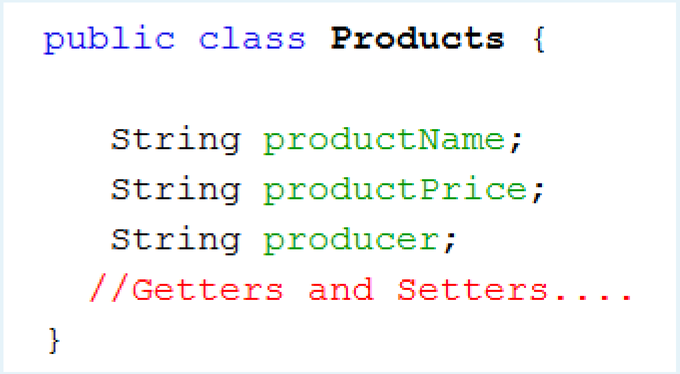
Component Suite

Component Suite

- PrimeFaces library offres you 459 different components you can use to build your application user interface.
- It includes for example: Calendar, Charts, DataTable, Caroucel which you can use to build nice and fluid UIs.

Few Examples:Caroucel

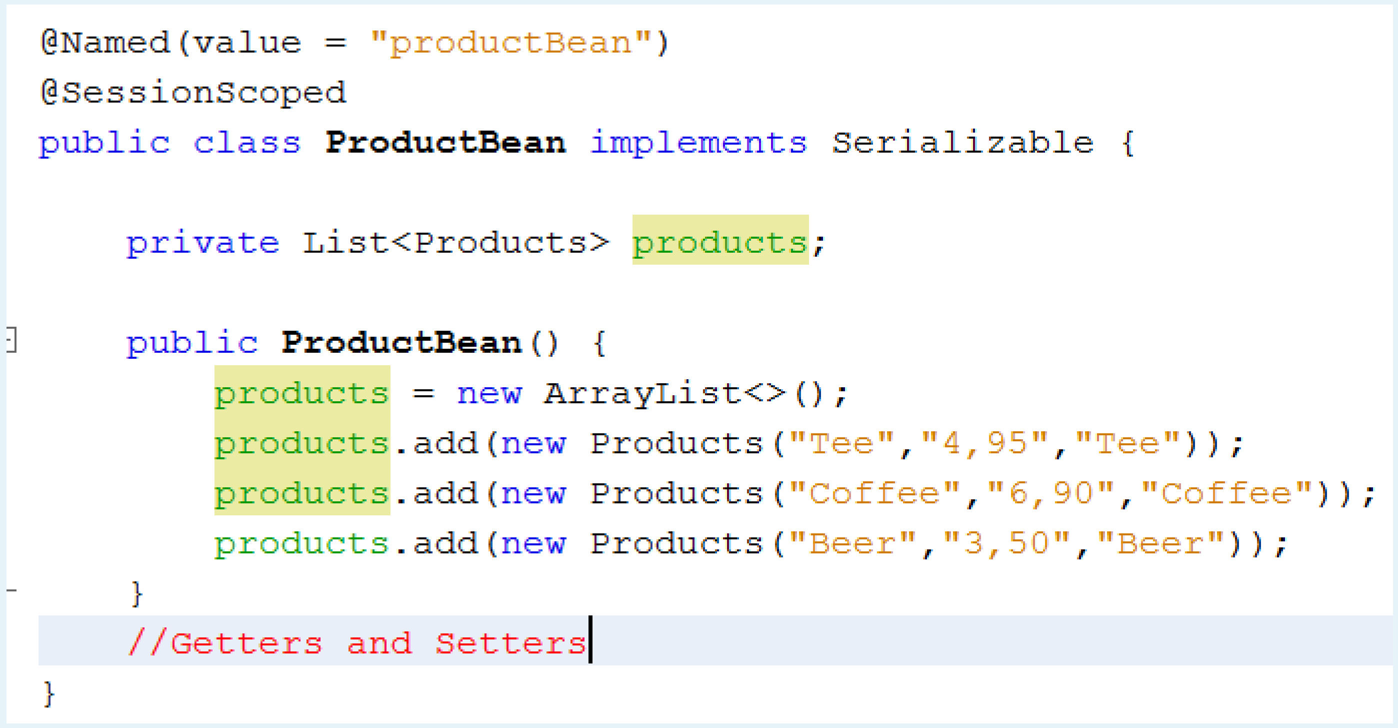
- First the model for our app...



```
public class Products {

   String productName;
   String productPrice;
   String producer;
  //Getters and Setters....
}
```

Few Examples:Caroucel

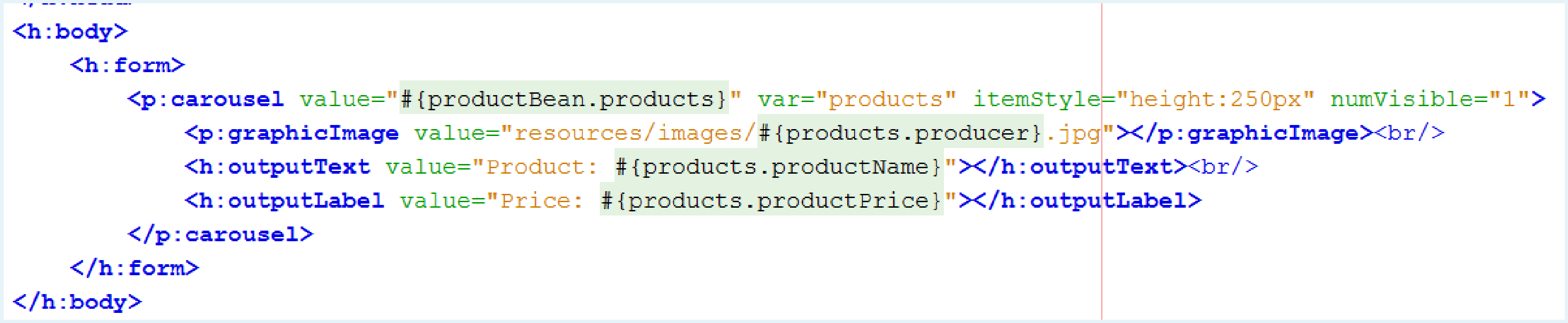
- The managed bean



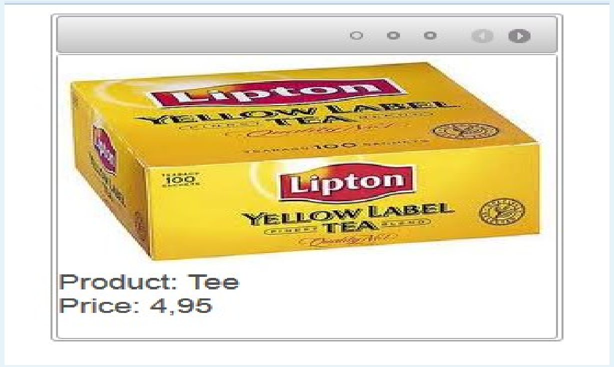
```
@Named(value = "productBean")
@SessionScoped
public class ProductBean implements Serializable {

    private List<Products> products;

    public ProductBean() {
        products = new ArrayList<>();
        products.add(new Products("Tee","4,95","Tee"));
        products.add(new Products("Coffee","6,90","Coffee"));
        products.add(new Products("Beer","3,50","Beer"));
    }
    //Getters and Setters
}
```

Few Examples:Caroucel

- The UI code.., (for complete example see project PrimeFaces from example folder)



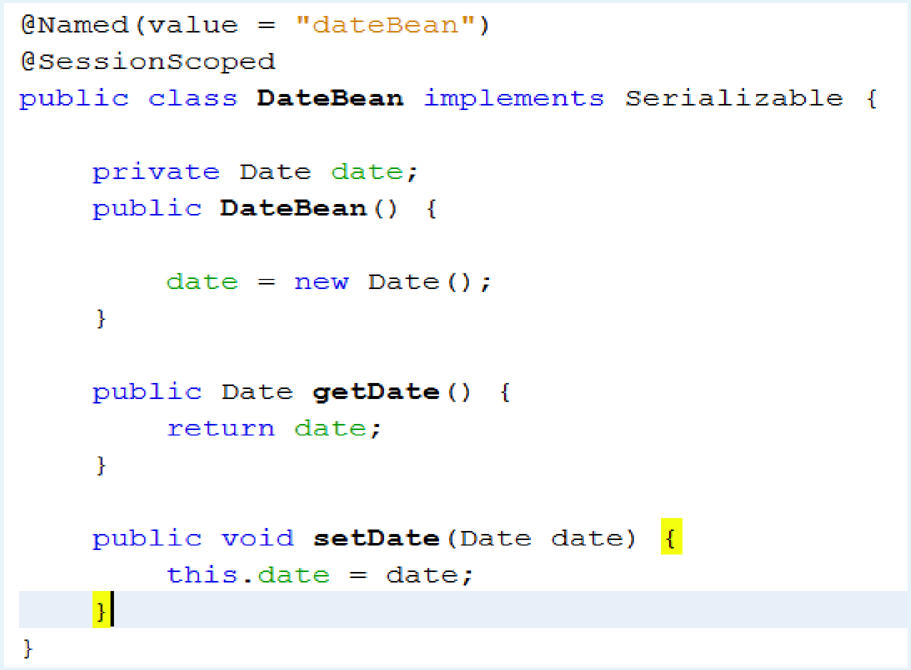
```
<h:body>
    <h:form>
        <p:carousel value="#{productBean.products}" var="products" itemStyle="height:250px" numVisible="1">
            <p:graphicImage value="resources/images/#{products.producer}.jpg"></p:graphicImage><br/>
            <h:outputText value="Product: #{products.productName}"></h:outputText><br/>
            <h:outputLabel value="Price: #{products.productPrice}"></h:outputLabel>
        </p:carousel>
    </h:form>
</h:body>
```



```
Product: Tee
Price: 4,95
```

Calendar

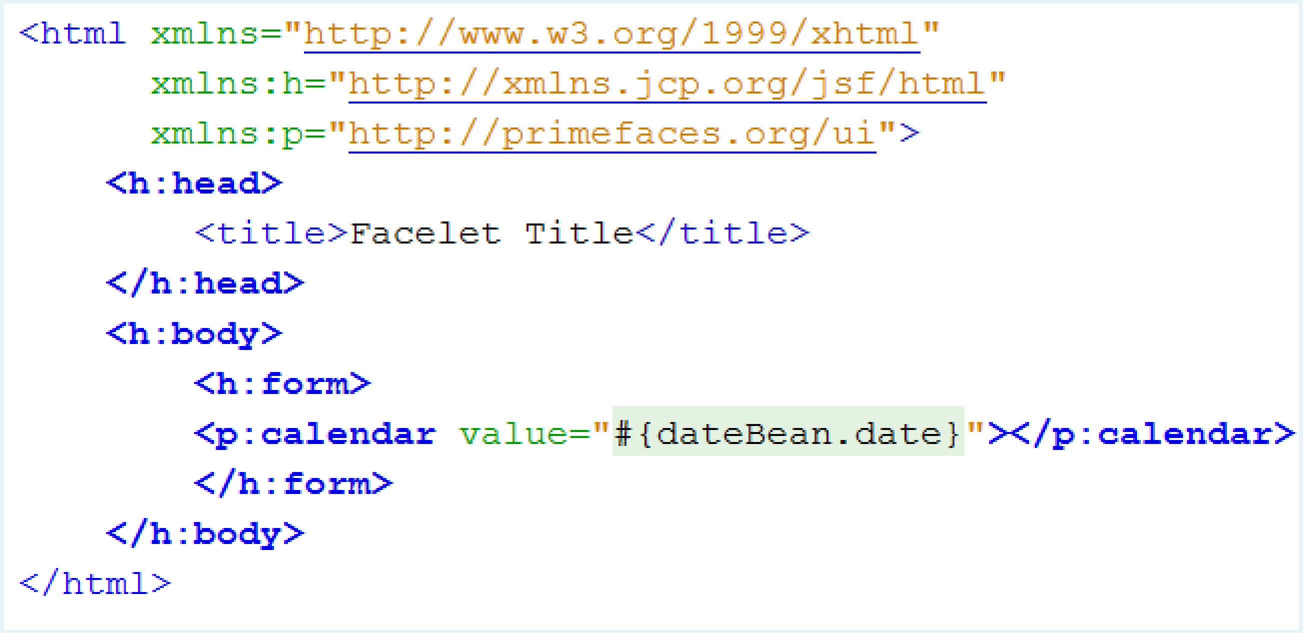
- Value of the calendar should be a java.util.Date!!! The DateBean code



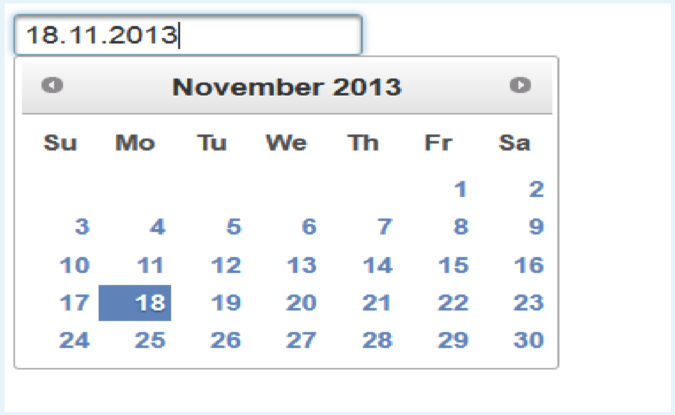
```
@Named(value = "dateBean")
@SessionScoped
public class DateBean implements Serializable {

    private Date date;
    public DateBean() {

        date = new Date();
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }
}
```

Calendar

- The UI…



```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <h:form>
        <p:calendar value="#{dateBean.date}"></p:calendar>
        </h:form>
    </h:body>
</html>
```



18.11.2013

November 2013

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Calendar

- For complete example see PrimeCalendar project from examples folder.

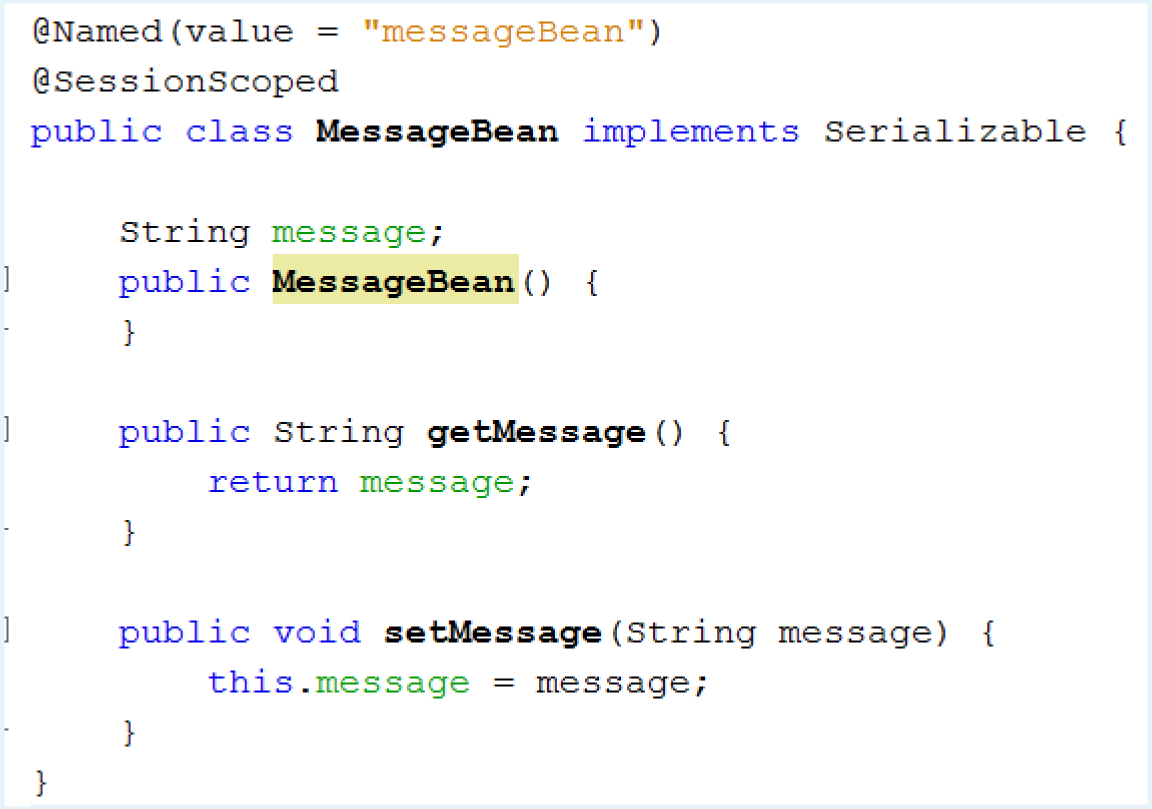
AjaxBehavior

AjaxBehavior

- AjaxBehavior is an extension to standard f:ajax.
- It is very easy to use...
- AjaxBehavior is attached to the component to ajaxify.
- See the next example..

AjaxBehavior

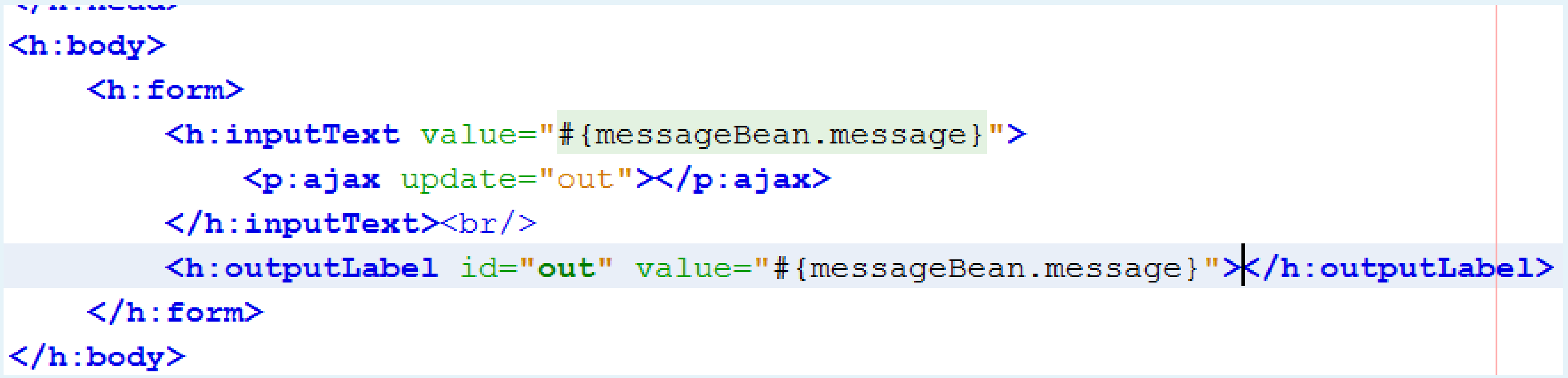
- The managed bean....



```
@Named(value = "messageBean")
@SessionScoped
public class MessageBean implements Serializable {

    String message;
    public MessageBean() {
    }

    public String getMessage() {
        return message;
    }

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

AjaxBehavior

- The UI...



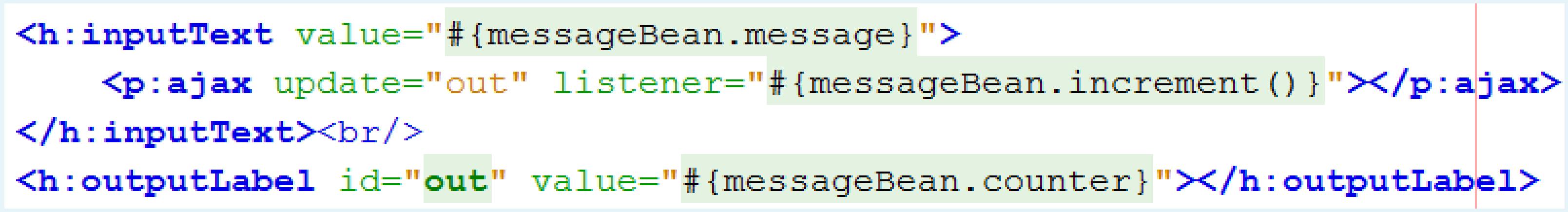
```
<h:body>
    <h:form>
        <h:inputText value="#{messageBean.message}">
            <p:ajax update="out"></p:ajax>
        </h:inputText><br/>
        <h:outputLabel id="out" value="#{messageBean.message}"></h:outputLabel>
    </h:form>
</h:body>
```

AjaxBehaviour

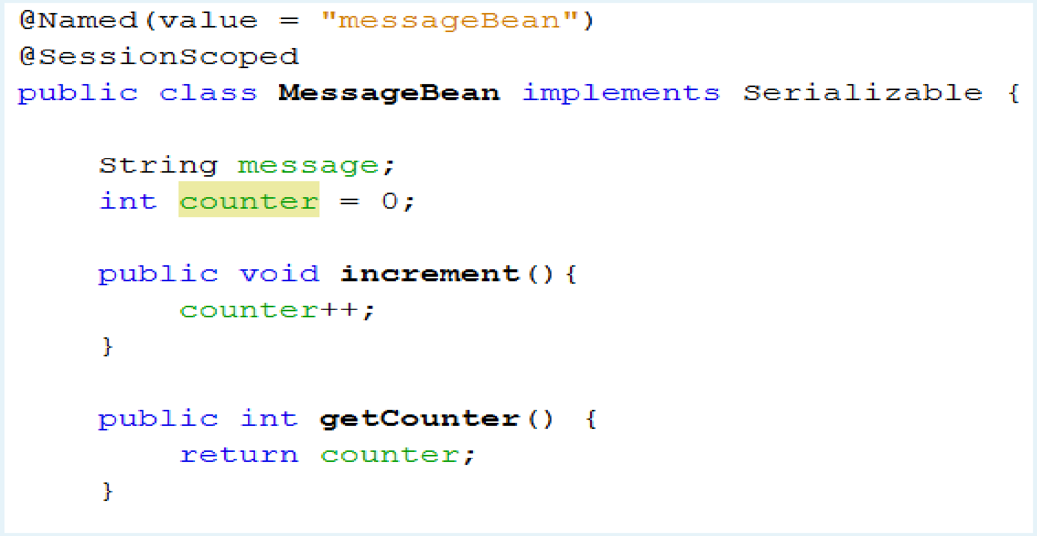
- The previous example works so that, each time the input changes (lost focus), an ajax request is sent to the server. When the response is received output text with id "out" is updated with value of the input.

AjaxBehaviour and Listeners

- In case you need to execute a method on a backing bean, define a listener



```
<h:inputText value="#{messageBean.message}">
    <p:ajax update="out" listener="#{messageBean.increment()}"></p:ajax>
</h:inputText><br/>
<h:outputLabel id="out" value="#{messageBean.counter}"></h:outputLabel>
```

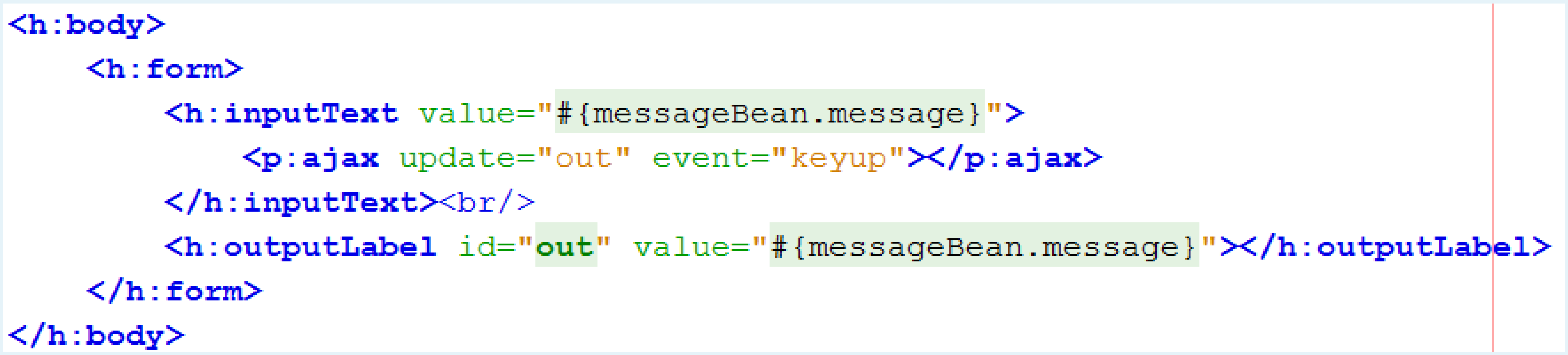


```
@Named(value = "messageBean")
@SessionScoped
public class MessageBean implements Serializable {

    String message;
    int counter = 0;

    public void increment(){
        counter++;
    }

    public int getCounter()  {
        return counter;
    }
```

AjaxBehaviour and Events

- Default client side events are defined by components that support client behaviors, for input components it is onchange and for command components it is onclick.
- In order to override the dom event to trigger the ajax request use event option. In following example, ajax request is triggered when key is up on input field.

AjaxBehaviour and Events



```
<h:body>
    <h:form>
        <h:inputText value="#{messageBean.message}">
            <p:ajax update="out" event="keyup"></p:ajax>
        </h:inputText><br/>
        <h:outputLabel id="out" value="#{messageBean.message}"></h:outputLabel>
    </h:form>
</h:body>
```

Finally

- I Appended an great book of PrimeFaces in dropbox. You can find it in Extras folder.