JSP Java Server Pages

Servlet: Java-Code mit eingebettetem HTML (in den out.write()-Befehlen)

JSP: HTML-Seiten/Code mit eingebettetem Java Code (vergleichbar mit PHP-Seiten)

Skriptelemente

1. Scriplets

Syntax

```
<% JAVACODE %>
```

```
<jsp:scriplet>JAVACODE</jsp:scriplet>
```

```
    </head>
<body>
    <h1>Überschrift</h1>
    <p>Klausurnoten</p>

    <%if(Math.random() > 0.5) { %>
    <p>Sie haben bestanden</p>
    <%} else { %>
    <p>Sie haben nicht bestanden</p>
    <%} %>
```

Aus dieser JSP-Datei wird automatisch in ein "Servlet" generiert

```
    </head>
<body>
    <h1>Überschrift</h1>
    <p>Klausurnoten</p>

    <%if(Math.random() > 0.5) { %>
    <p>Sie haben bestanden</p>
    <%} else { %>
    <p>Sie haben nicht bestanden</p>
    <%} %>
```

```
        _jspx_out = out;

        out.write("\r\n");
        out.write("<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01 Transition
        out.write("\t<html>\r\n");
        out.write("\t\t<head>\r\n");
        out.write("\t\t\t<meta http-equiv=\"Content-Type\" content=\"text/h
        out.write("\t\t\t<title>Insert title here</title>\r\n");
        out.write("\t</head>\r\n");
        out.write("<body>\r\n");
        out.write("\t<h1>Überschrift</h1>\r\n");
        out.write("\t<p>Klausurnoten</p>\r\n");
        out.write("\t\r\n");
        out.write("\t");
if(Math.random() > 0.5) {
        out.write("\r\n");
        out.write("\t<p>Sie haben bestanden</p>\r\n");
        out.write("\t");
} else {
        out.write("\r\n");
        out.write("\t<p>Sie haben nicht bestanden</p>\r\n");
        out.write("\t");
}
        out.write("\r\n");
        out.write("\t\r\n");
        out.write("\t<p>");
        out.print( new java.util.Date() );
                                          \n");
                                          >\r\n");
                                          >");
                                          Throwable t) {
```

Dieses Servlet generiert diesen HTML-Code

view-source:localhost:8080/SOTE2-2018-05-04/FirstJSP.jsp

Klicken, um weiterzugehen. Gedrückt halten, um Verlauf aufzurufen

```
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=ISO-885
            <title>Insert title here</title>
    </head>
<body>
    <h1>Überschrift</h1>
    <p>Klausurnoten</p>


    <p>Sie haben nicht bestanden</p>


    <p>Fri May 04 12:11:09 CEST 2018</p>
</body>
</html>
```

localhost:8080/SOTE2-2018-05-04/FirstJSP.jsp

Überschrift

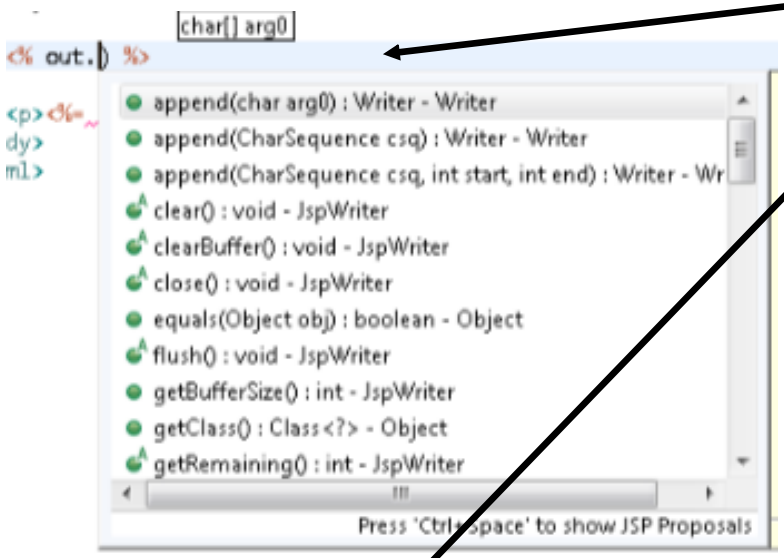
Klausurnoten

Sie haben nicht bestanden

Fri May 04 12:11:03 CEST 2018

... der dann wie hier dargestellt ím Browser angezeigt wird

In den JSP-Dateien sind bereits Objekt impliziert definiert z.B. out:




```
try {
  response.setContentType("text/html; charset=ISO-8859-1"
  pageContext = _jspxFactory.getPageContext(this, request
               null, true, 8192, true);
  _jspx_page_context = pageContext;
  application = pageContext.getServletContext();
  config = pageContext.getServletConfig();
  session = pageContext.getSession();
  out = pageContext.getOut();
  _jspx_out = out;
```

2. Expressions

Syntax:

```
<%= TEXT %>
```

```
<jsp:expression>Text</jsp:expression>
```

Ist eine Kurzschreibweise /Alternative zu

```
<% out.write("TEXT"); %>
```

```
17      <%} %>
18
19      <% out.write("" + new java.util.Date()); %>
20
21      <p><%= new java.util.Date() %></p>
22  </body>
23  </html>
```

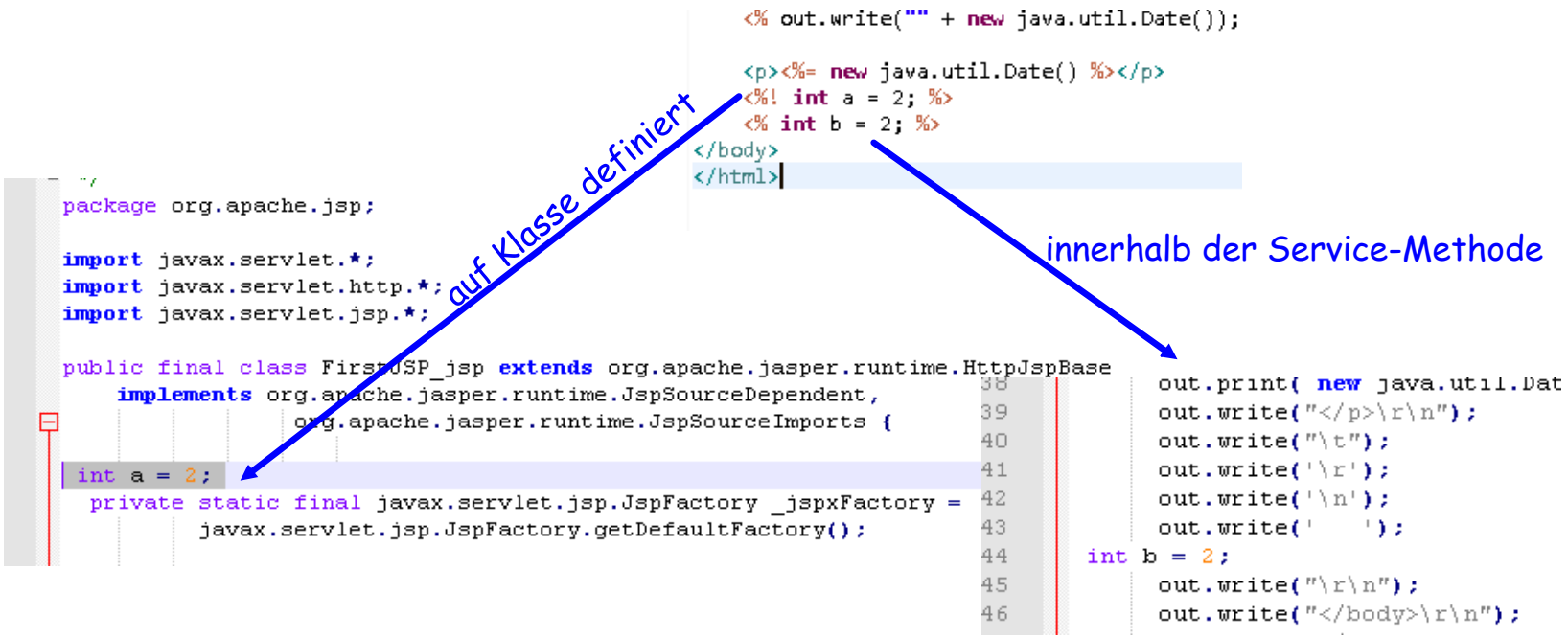
3. Deklarations

Syntax

```
<%! int a = 2; %>
```

```
<jsp:declaration>int a = 0 2;</jsp:declaration>
```

Deklarationen (mit !) nutzt man, um Instanzvariablen zu definieren, also Variablen außerhalb der service-Methode. Damit können andere JSP-Dateien auf diese Variable zugreifen.

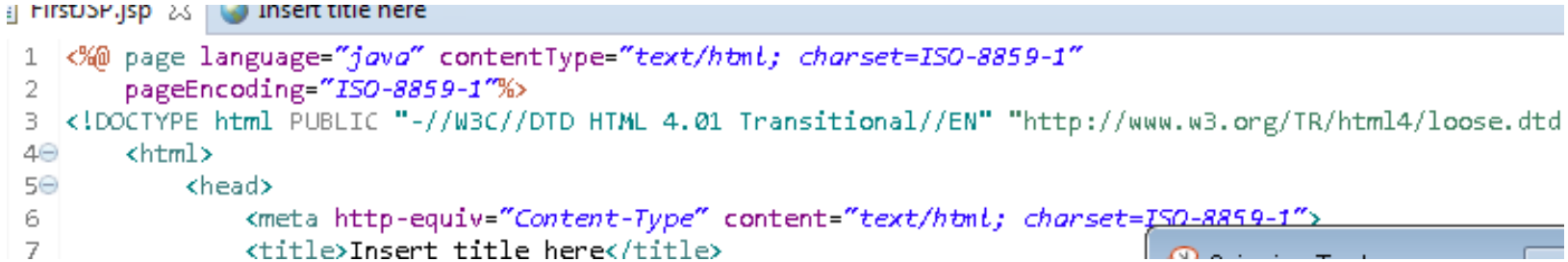


4. Direktiven

Syntax

```
<%@ NAME-DIREKTIVE attribut1="wert1" attribut2="wert2", ... %>
```

```
<jsp:directive attribut1="wert1" attribut2="wert2", ...</jsp:directive>
```



4.1 Page Direktiven

z.B. für Imports

```
<% page import="java.util.*" %>
```

z.B. für Erben von Klassen

```
<% page extends="myPackage.MyClass" %>
```

4.2 Include Direktive

<%@ include file="footer.txt">

```
<%@ include file="footer.txt" %>
</body>
</html>
```

Servlet wird generiert: Der Text aus der Datei (hier: footer.txt) wird in das Servlet reingeneriert und damit fest kompiliert. D.h. Änderungen an der Datei werden zur Laufzeit ignoriert.

```
out.write("\r\n");
out.write("\t\r\n");
out.write("\t");
out.write("<h3>Footer</h3>\r\n");
out.write("<p>Impressum</p>");
out.write("\r\n");
out.write("</body>\r\n");
out.write("</html>");
```

5. Actions

5.1. Include

<jsp:include page =" footer.txt"/>

Mit <jsp:include> wird die Datei (hier footer.txt) zur Laufzeit d.h. mit jedem Aufruf eingelesen und der Inhalt in die Seite eingefügt:

```
        out.write("<h3>Footer</h3>\r\n");
        out.write("<p>Impressum 2</p>");
        out.write("\r\n");
        out.write("\t\r\n");
        out.write("\t");
        org.apache.jasper.runtime.JspRuntimeLibrary.include(request, response, "footer.txt", out, false);
        out.write("\r\n");
        out.write("</body>\r\n");
        out.write("</html>");
    } catch (java.lang.Throwable t) {
```

5.2. JavaBeans

Syntax

```
<jsp:useBean id="instanzname"/>
```

Instanz der Klasse

```
30
31      <!-- Buch sparbuch = new Buch(); -->
32      <jsp:useBean id="sparbuch" class="hochschule.Buch"></jsp:useBean>
33
34      <!-- sparbuch.getTitel(); -->
35      <jsp:getProperty property="titel" name="sparbuch"/>
36
37      <!-- sparbuch.setTitel("Momo"); -->
38      <jsp:setProperty property="titel" name="sparbuch" value="Momo"/>
39      <jsp:getProperty property="titel" name="sparbuch"/>
40  </body>
41  </html>
```

Wert auslesen

Wert setzen

JavaBeans sind "normale" Java-Klassen, die einer Namenskonvention für die getter und setter genügen und die über einen Default-Konstruktor verfügen.

JavaBean helfen, das "Backend" (in Java implementiert) vom Frontend (HTML, CSS, JavaScript) zu trennen, in dem das Backend keinen HTML-Code und das Frontend keinen Java-Code mehr enthält.

JavaBeans entsprechen im Sinne eines MVC-Patterns dem Modell (enthält die Daten), die JSP der View (stellt die Daten dar).

Nachtrag: Webprojekte ohne Eclipse laufen lassen:

1. Projekt als war-Datei (gezippte Datei mit allen Artefakten) exportieren
2. Diese Datei in das Verzeichnis "webapps" des Tomcat kopieren (dieser entpackt die Datei automatisch.
3. Auf das neue Webprojekt zugreifen (http://<server>/<namedatei>