Web-Technologien

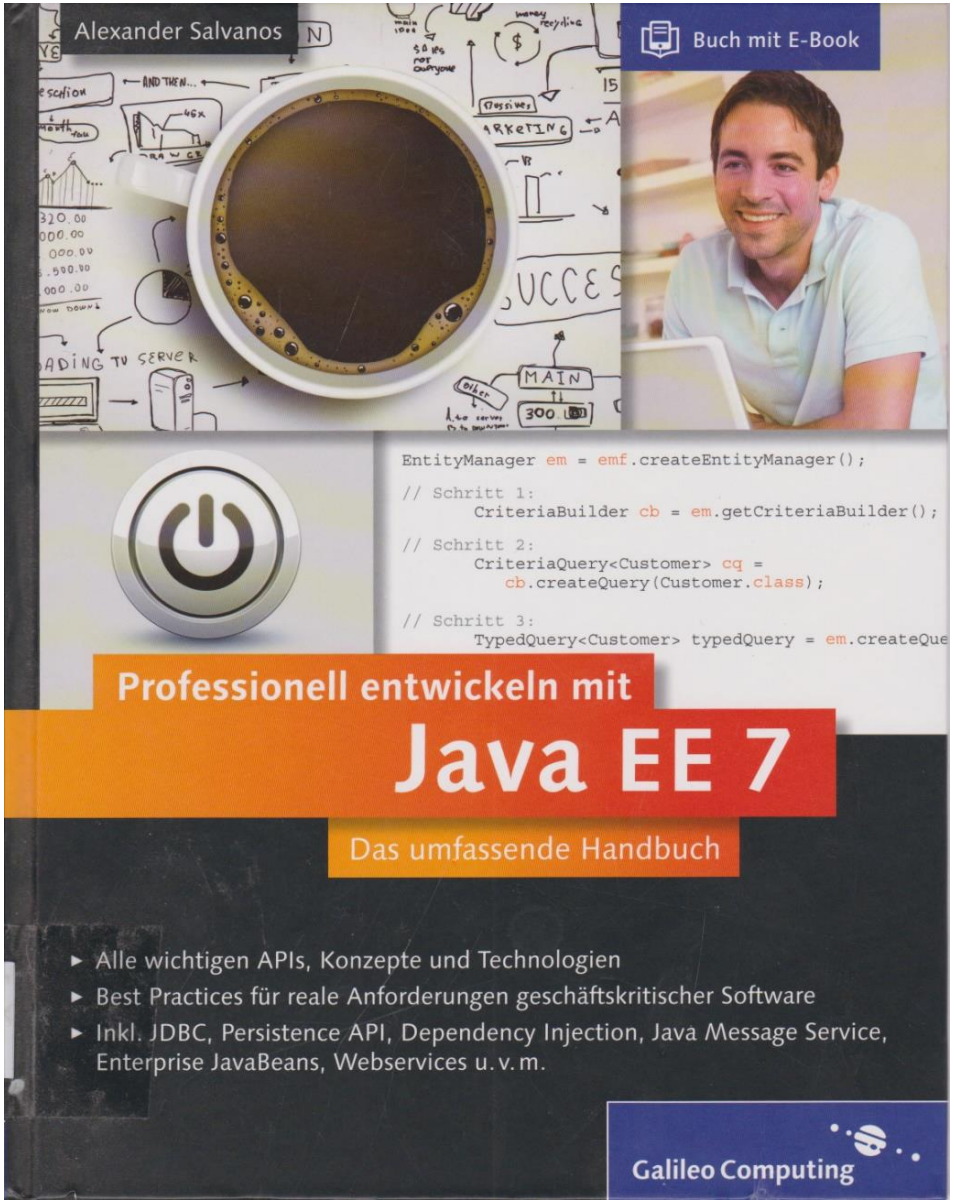
- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://www.miwilhelm.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

1. Einleitung
2. PHP (OOP, I/O, Datenbanken)
3. ASP.net: Web-Forms (kurz)
4. ASP.net: MVC
5. WebPages, WebMatrix
6. jQuery
7. Node.js
8. **JSP**

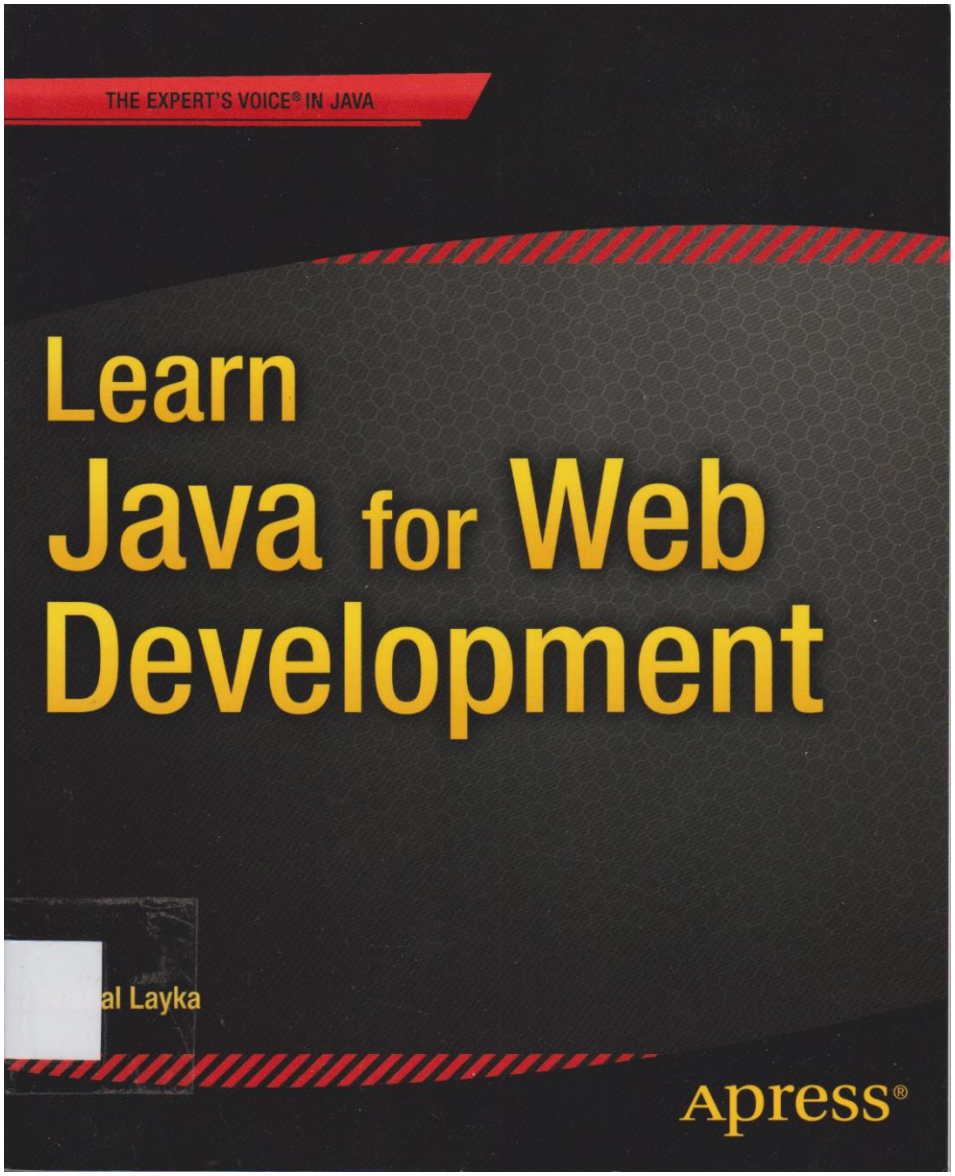
Literatur

- Professional entwickeln mit Java EE7
- Alexander Salvanos
- Galileo Computing
- ISBN: 978-3-8362-2004-0



Literatur

- Learn Java for Web Development
- Vishal Layka
- Apress
- ISBN: 978-1-4302-5983-1



Literatur

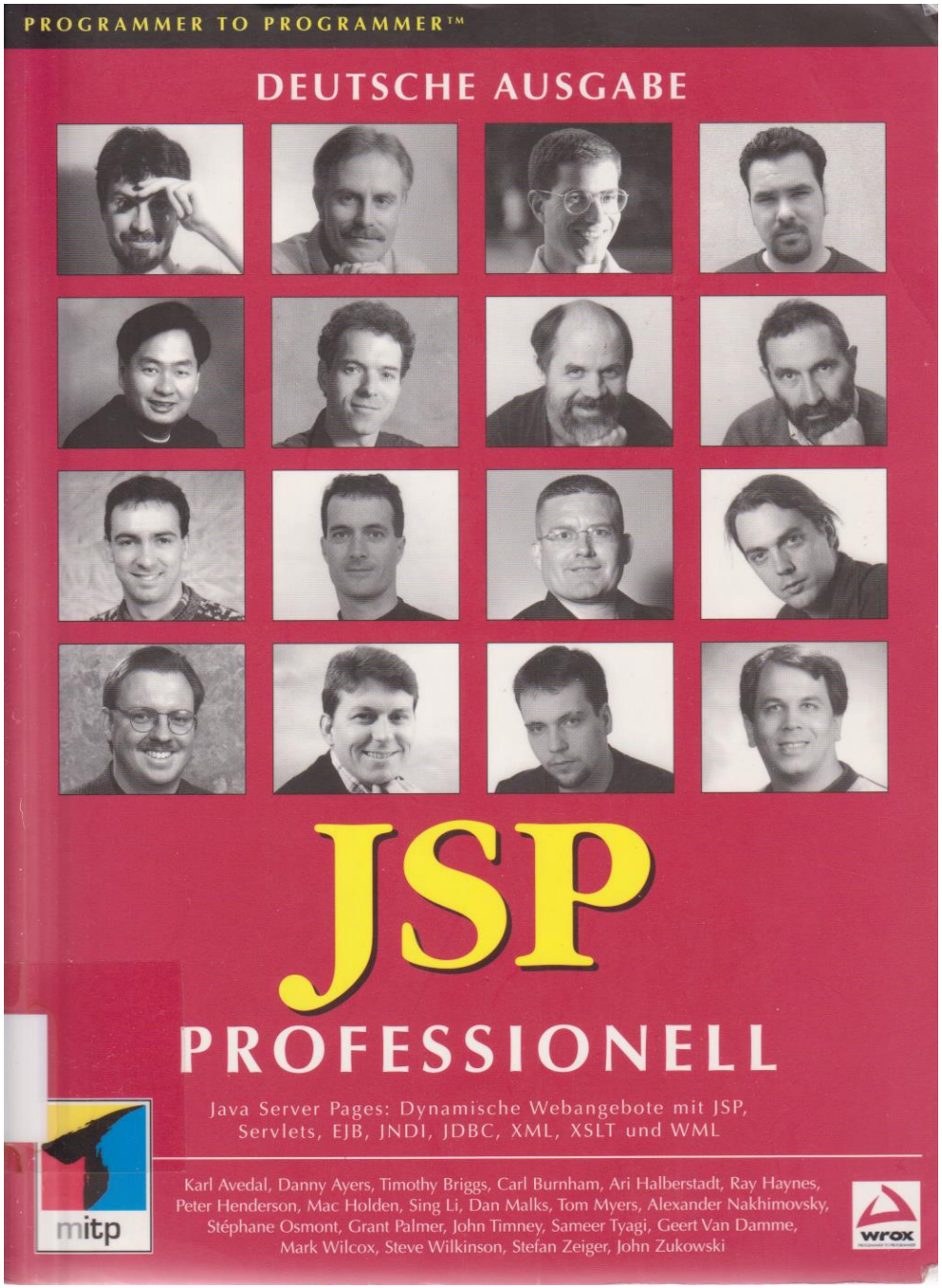
- Tomcat 5
- Lajos Moczar
- Addison Wesley
- ISBN: 3-8273-2202-2

Literatur

- Servlet&JSP von Kopf bis Fuß
- Bryan Basham, Kathy Sierra & Bert Bates
- O‘Reilly
- ISBN: 978-3-89721-873-1

Literatur

- JSP Professional
- Viele Autoren
- MITP Verlag
- ISBN: 3-8266-0660-4



Links

- https://www.java-tips.org/java-tutorials/1507-introduction-to-java-servlets-with-eclipse.html
- https://www.javatpoint.com/creating-servlet-in-eclipse-ide

Java Server Pages

- Ist ähnlich wie PHP, nur anders
- Tag‘s:
  - <% %>
- Compile
  - PHP wird direkt interpretiert
  - JSP-Seiten werden erst übersetzt (Servlets)
- Mächtigkeit
  - PHP ist bewust einfach gehalten
  - Java hat sehr komplexe Strukturen
- Servlets: direkte Ausgabe eines Terms
  - <% out.println(new java.util.Date()); %>
- Ausdruck
  - <%= new java.util.Date() %>
- Zusätzliche Frameworks
  - Struts, Java Server Faces, Spring, Grails

Java Server Pages

- **Directives**
  - <%@ directive {attr="value"}* %>
  - Direktiven erlauben Festlegungen, z.B. import-Befehle.
  - Beispiel: <%@ page language="java" %>
- **declarations**
  - <%! declaration %>
  - Deklaration von Variablen und Methoden. Es ist keine Ausgabe möglich.
  - Beispiel: <%! int i=0 %>
- **scriptlets**
  - <% scriptlet %>
  - Beliebige Java-Befehle.
  - Befehle der Form "System.out.println("Text");" oder "out.println("Cheerio");" erzeugen eine Ausgabe in der HTML-Seite.
  - Beispiel: <% i=i+1; %>

Java Server Pages

- **Expressions, direkte Ausgabe eines Terms**
  - <%= expression %>
  - Gültige Java-Ausdrücke. Das Ergebnis wird in die HTML-Seite eingefügt.
  - Beispiel: <%= preis*0.16 %>
- **comments**
  - <%-- jsp comment --%>
  - Kommentare, sind in der HTML-Seite nicht sichtbar.
  - Beispiel: <%-- Berechnung des Maximums in einer Schleife --%>

Beispiel mit Java Server Pages

```
<html>
<head>
  <title> 01. Beispiel mit JSP </title>
</head>
<body>
   <!--   http://www.miwilhelm.de/scripte/php-neu/bsp_sp1_01.html   -->
  <h2>  Erste JSP-Datei  </h2>
  <%
     // beliebiger Java-Code
    out.println( "Hier ist meine erste JSP-Datei <br />" );
    int nr=12;
    out.println( "Nummer: " + nr );
  %>
</body>

</html>
```

- XAMPP oder Tomcat installieren

1. Apache tomcat (gratis) herunterladen -> LINK
(Die Datei die ihr braucht: Windows Service Installer (pgp, md5)
2. Installieren
3. Installationverzeichnis öffnen:
C:\Programme\Apache Software Foundation\Tomcat 6.0\webapps\ROOT
In diesen Ordner eure HTML, JSP,.. Seite hinein tun.
4. Euren Browser öffnen (IE, Firefox,..)
5. Gebt in der Adresszeile localhost:8080 ein. (8080 ist der Port den ihr bei der Installation angegeben habt).
6. Dann die Datei starten in dem ihr euren Dateinamen dranhängt.
Also zb: http://localhost:8080/meinehompage.html/

Texteingabe

```
<form>
  Geben Sie Ihren Benutzernamen an: <p>
  <input type="text" name="userid" value="anonymous" size="8">
  <input type="submit" value="fertig">
</form>
```

Optionen:

- size (Breite)
- maxlength (Anzahl Zeichen)
- readonly

Texteingabe ansprechen

```
<form>
  <input type="text" name="userid" value="anonymous" size="8">
  <input type="submit" value="fertig">
</form>
```

```
 String userid= request.getParameter("userid");
  if (userid == null) {
    ok=false;
    out.println("Fehlerhafter Parameter userid");
  }
  else {
     out.println(userid);
 }
```

Texteingabe mit als Zahl ansprechen

```
<input type="text" name="zahl" value="12" size="8">
```

```
String str_zahl= request.getParameter("zahl");
if (str_zahl==null) {
   ok=false;
   out.println("Fehlerhafter Parameter zahl");
}
else {
    Integer Zahl = getIntNumber(str_zahl);
    if (Zahl==null) {
      ok=false;
     out.println("Fehlerhafte zahl");
   }
   else {
     zahl = Zahl.intValue();
   }
}
```

Texteingabe mit als Zahl ansprechen

```
<input type="text" name="zahl" value="12" size="8">
String str_zahl= request.getParameter("zahl");
if (str_zahl==null) {
   ok=false;
   out.println("Fehlerhafter Parameter zahl");
}
else {
    Double Zahl = getDoubleNumber(str_zahl);
    if (Zahl==null) {
      ok=false;
     out.println("Fehlerhafte zahl");
   }
   else {
     zahl = Zahl.intValue();
   }
}
```

Beispiel: Checkbox

```
<form>
 Wählen Sie eine oder mehrere Optionen aus:
  <input type="checkbox" name="AutoCAD"> AutoCAD<br />
  <input type="checkbox" value="Catia" checked="checked"> Catia <br />
  <input type="checkbox" value="Eagle"> Eagle<br />
</form>
```

Wählen Sie eine oder mehrere Optionen aus:

- [ ] AutoCAD
- [x] Catia
- [ ] Eagle

CheckBox ansprechen

```
<input type="checkbox" name="chkautocad"> AutoCAD<br />
```

```
boolean ausgabe=false;
String dummy = request.getParameter("chkausgabe");
 // Wenn Checkbox gesetzt, gibt es einen Parameter
 //  sonst NICHTS !!!
 if (dummy!=null) {
   if (dummy.equals("ausgabe")) {
     ausgabe=true;
   }
   else {
     ok=false;
   }
 }
```

Mehrere CheckBoxen ansprechen

```
<input type="checkbox" name="themes" value="php" /> PHP<br />
<input type="checkbox" name="themes" value="jsp" /> JSP<br />
```

```
<%@ page import = "java.util.* " %>
ArrayList<String> liste = new ArrayList<String>();
String[] dummies = request.getParameterValues("themes");
 if (dummies==null) {
  ok=false;
 }
 else {
  for( String item : dummies) {
   liste.add(item);
  }
 }
```

Beispiel 1

localhost:8080/bsp8_form1.xhtml 240% Suchen

JSP CheckBox-Family

Pruefungsthemen

- [x] Resim
- [ ] HTML
- [x] PHP
- [ ] JSP
- [x] Datenbanken

Send Delete

ComboBox JSP-Beispiel

localhost:8080/bsp8_fo 240% Suchen

Checkbox/ Liste

Themen:

1. resim
2. php
3. database

Hauptseite

Auswahlschalter, RadioButton

```
<form>
  <input type="radio" name="rbcad" value="cad" /> AutoCAD<br />
  <input type="radio" name="rbcad" value="catia" checked= "checked" /> Catia <br />
  <input type="radio" name="rbcad" value="eagle"  /> Eagle<br />
</form>
```

Wählen Sie eine der Optionen aus:

- ○ AutoCAD
- ◉ Catia
- ○ Eagle

value ist die Bezeichnung des GUI-Elementes

value ist auch für PHP wichtig

name kennzeichnet die Gruppe

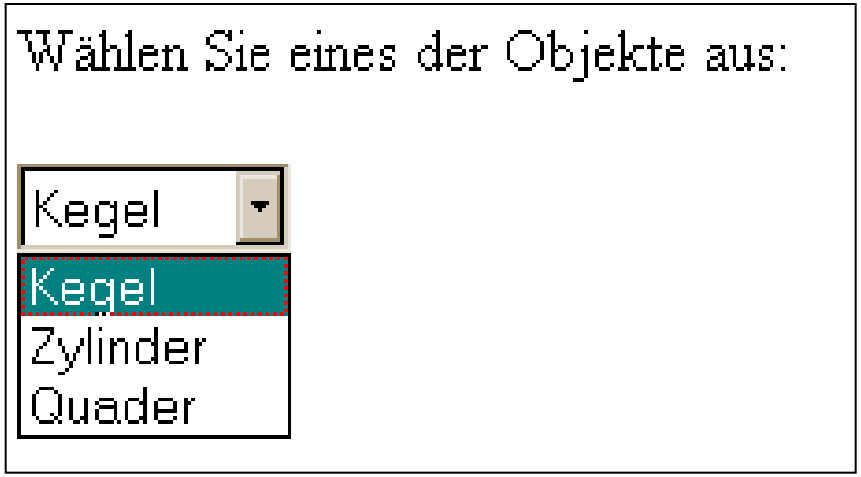
Radiobutton ansprechen

```
<input type="radio" name="rbcad" value="cad" /> AutoCAD<br />
<input type="radio" name="rbcad" value="catia" checked= "checked" /> Catia <br />
```

```
String dummy = request.getParameter("rbcad");
 if (dummy==null) {
   ok=false;
 }
 else {
   if (dummy.equals("cad"))
     cad=0;
    else if (dummy.equals("catia"))
     cad=1;
   else {
     out.println("falscher CAD-Parameter");
     ok=false;
   }
 }
```

Auswahlliste: ComboBox

```
<form>
<p>Wählen Sie eines der Objekte aus: </p>
<select name="geomkoerper">
    <option value="idk"  selected="selected"> Kegel </option>
    <option value="idz" > Zylinder </option>
    <option value="idq"> Quader </option>
</select>
</form>
```



Size: Anzahl der Optionen die sichtbar sind, Liste

multiple: Multi-Select

value="idk" ist der Primarykey, Datenbank, PHP

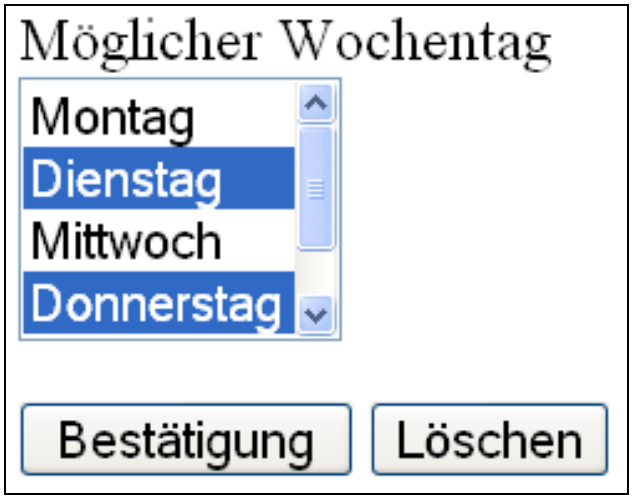
ComboBox ansprechen: Eine Auswahl möglich

```
<select name="geomkoerper"   >
        <option value="idk"  selected="selected"> Kegel </option>
        <option value="idz" > Zylinder </option>
</select>
```

```
String dummy = request.getParameter("geomkorper");
 if (dummy==null) {
   ok=false;
 }
 else {
   if (dummy.equals("idk"))
     geomkoerper=0;
    else if (dummy.equals("idz"))
     geomkoerper=1;
   else {
     out.println("falscher geomkoerper-Parameter");
     ok=false;
   }
 }
```

Auswahlliste: echte Liste

```
<select name="wochentag" size="4" multiple="multiple" >
    <option value="Montag" > Montag </option>
    <option value="Dienstag" selected > Dienstag </option>
    <option value="Mittwoch" > Mittwoch </option>
    <option value="Donnerstag" selected> Donnerstag </option>
    <option value="Freitag"> Freitag </option>
  </select>
```



size: Anzahl der Optionen die sichtbar sind, Liste

multiple: Multi-Select

ComboBox ansprechen: Mehrere Einträge möglich

```
<select name="geomkoerper"   >
        <option value="idk"  selected="selected"> Kegel </option>
        <option value="idz" > Zylinder </option>
</select>
```

```
<%@ page import = "java.util.* " %>
<%
 ArrayList<String> liste = new ArrayList<String>();
 boolean ok=true;
 String[] dummies = request.getParameterValues("geomkoerper");
 if (dummies==null) {
   ok=false;
 }
 else {
   for( String item : dummies) {
     liste.add(item);              // zur Weiterverarbeitung
   }
 }
```

Auswahlliste: Gruppenliste

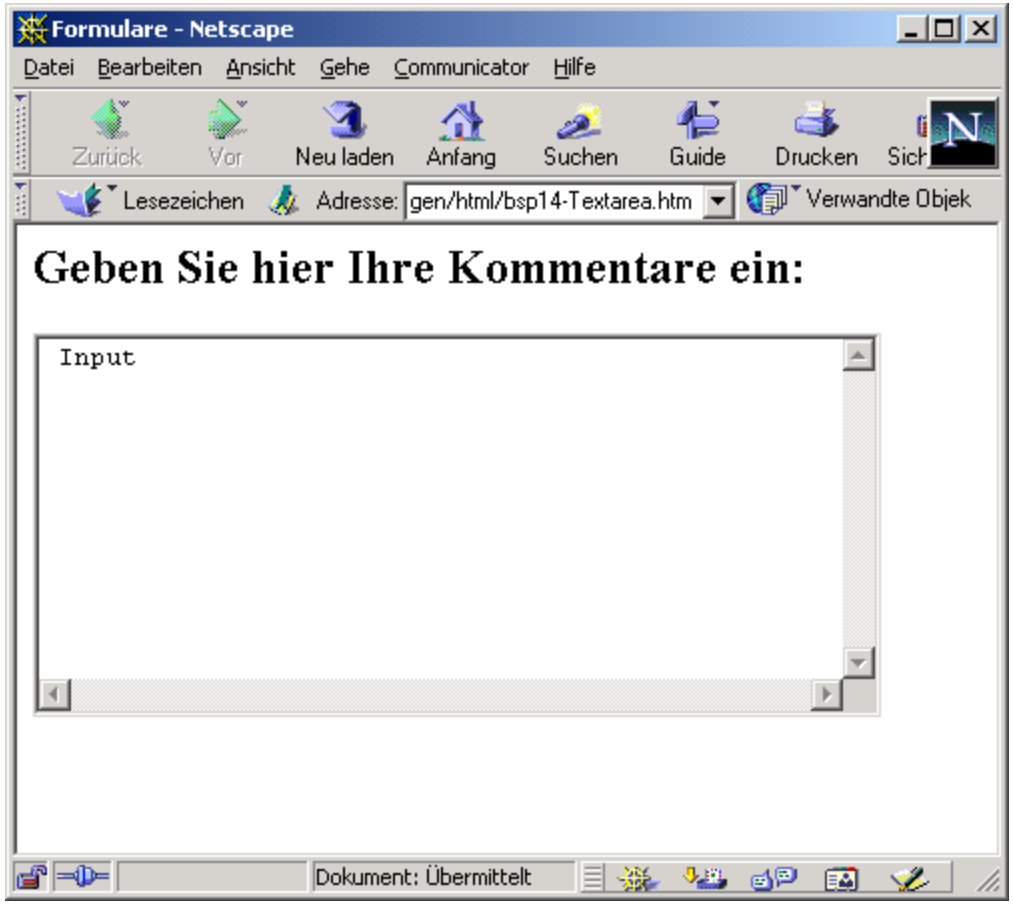
```
<select name="studiengaenge" size="6" multiple="multiple" >
   <optgroup label="AI">
      <option value="ias" > IAS </OPTION>
      <option value="ib"> IB </option>
      <option value="mas" > MAS </option>
      <option value="psc" PSC </option>
   </optgroup>
 <optgroup label="VW">
      <option value="ofvw" > Öffentliche Verwaltung </OPTION>
      <option value="vo"> Verwaltungsökonomie </option>
      <option value="ev" > Europäisches Verwaltungsmanagement
</option>
      <option value="vm" Verwaltungsmanagement / eGovernment
</option>
   </optgroup>
</select>
```

size: Anzahl der Optionen die sichtbar sind, Liste

multiple: Multi-Select

Editor:

```
<form>
 <textArea name="editor" rows="10" cols="50"> Input </textArea>
</form>
```



- wrap="on"
- wrap="off"

Mehrzeiliger Editor ansprechen

```
<form>
  Eingabe des Texte <br />
    <textArea name="editor" rows="10" cols="50"> Input </textArea>
</form>
```

```
String editor= request.getParameter("editor");
 if (editor == null) {
   ok=false;
   out.println("Fehlerhafter Parameter editor");
 }
 else {
    out.println(editor);
}
```

label

- Mit den label-Element werden Beschriftung und Formularelement verknüpft
- Beim Anklicken des label´s wird der Kursor in Element gesetzt

```
<label for="username">Username </label>
 <input id="username" type="text" size="15" />
```

- Wenn das Formularelement innerhalb des label-Tags gesetzt wird, werden das *for*- und das *id*-Attribut nicht gebraucht.
- das Eingabefeld und Label sind direkt miteinander verknüpft:

```
 <label>Ergebnis<br />
<textarea cols="40" rows="12"> 4+3=7 </textarea>
 </label>
```

fieldset und legend

Mit den beiden Tags können GUI-Elemente optisch getrennt werden. Fieldset ist die äußere Klammer. Das Tag „legend“ ist die Überschrift der Box.

```
<fieldset>
  <legend>CAD-Auswahl</legend>
  <input type="radio" name="R1" value="AutoCAD" checked>
            AutoCAD<br />
  <input type="radio" name="R1" value="Auto-Sketch" checked>
           Catia <br />
<input type="radio" name="R1" value="AutoCADLT" checked>
          AutoCAD LT<br />
</fieldset>
```

Formulare - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

file:///V:/Daten/html/forms/radio5.html

Google

Meistbesuchte Seiten

Wählen Sie eine CAD und eine GIS-Option aus:

CAD-Auswahl

- AutoCAD
- Catia
- AutoCAD LT

GIS-Auswahl

- ArcView
- Smallworld
- Geomedia

Fertig

Hidden-Element:

```
<form>
<input type="hidden" name="UserBrowser"
        value="userid=1234 password=1234">
</form>
```

- Das Element ist nicht sichtbar
- Es kann aber darauf zugegriffen werden
- Speicherung von Log-ID´s
- Alternativ sind Sessionsvariablen, "Session-ID" in PHP möglich

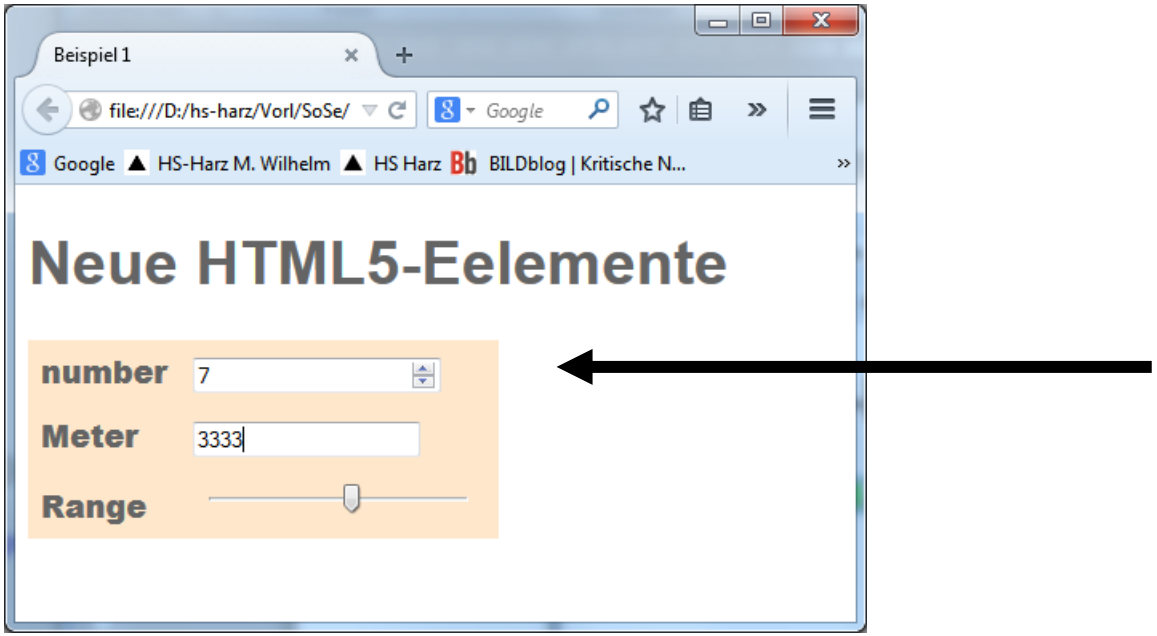
Formularelemente in HTML 5

- Datentyp für Emailadresse(n):
  - <input type="email" />
- Datentyp für URLs:
  - <input type="url" />
- Datentypen für numerische Eingaben
  - <input type="number" />
  - <input type="range" min="10" max="100" step="5" value="55" />
- Auswahl einer Farbe
  - <input type="color" />
- Sucheingabe
  - <input type="search" name="search" />

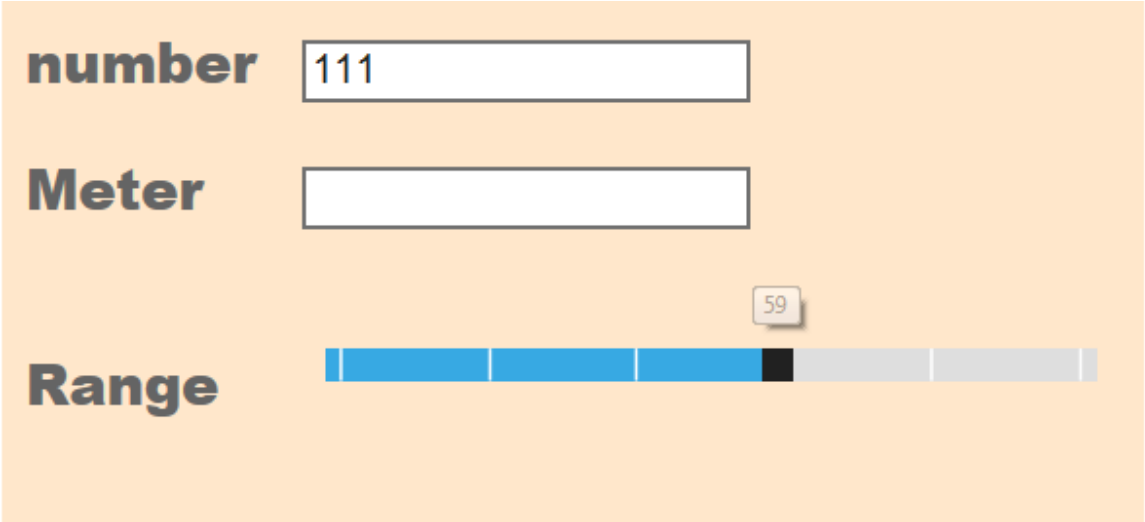
Neue Datums und Zeit-Elemente

Neue Datums und Zeit-Elemente

Firefox
Opera
Chrome



IExplorer



Datalist-Element

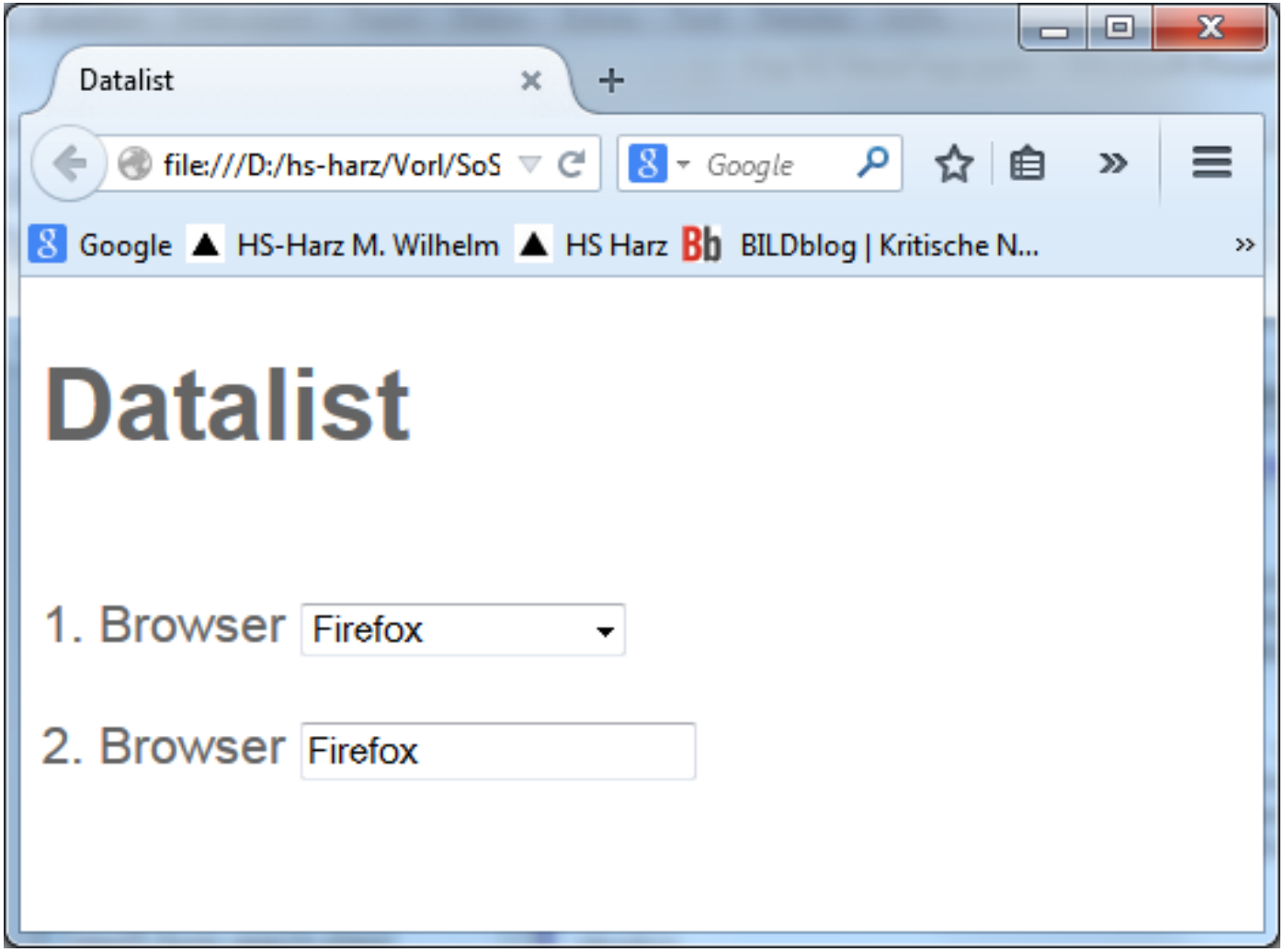
- **Suggestion**
- Optionale Eingaben
- Vergleichbar mit der ComboBox, aber keine festen Vorgaben

Alte Technik

```
<select name="objekte" >
  <option > Internet Explorer</option >
  <option selected="selected"> Firefox</option>
  <option  > Chrome</option >
  <option  > Opera</option >
  <option  > Safari</option >
</select>
```

Neue Technik

```
<input type="text" list="browsers2" value="Firefox" />

<datalist id="browsers2">
  <option value="Internet Explorer"/>
  <option value="Firefox"/>
  <option value="Chrome"/>
  <option value="Opera"/>
  <option value="Safari"/>
</datalist>
```


Datalist
file:///D:/hs-harz/Vorl/SoS
Google
Google
HS-Harz M. Wilhelm
HS Harz
BILDblog | Kritische N...
Datalist
1. Browser
Firefox
2. Browser
Firefox

Validierung mittels Pattern

- ^ Anfangs des Textes
- $ Ende des Textes
- . Beliebiges Zeichen
- \. Punkt
- [A-Z] ein Zeichen aus A bis Z, [0-9], [a-z]
- [^A-Z] Negation, kein Zeichen aus A bis Z, [^ ' "] \x22
- + 1,n
- * 0,1,n
- {m,n} Wiederholungen {1,} {2}

Validierung mittels Pattern

```
<input id="geburtsdatum"  type="text"
pattern="^(
        31|
        30|
        0[1-9]|
        [12][0-9]|
        [1-9])\.(0[1-9]|
        1[012]|[1-9])\.((18|19|20)\d{2}|\d{2}
        )$">
```

Validierung mittels Pattern

E-mail:

```
<input type="email" name="email"
    pattern= "
        [a-z0-9._%+-]+
        @
        [a-z0-9.-]+
        \.
        [a-z]{2,3}
        $ "
>
```

2. Beispiel mit Java Server Pages

```
<html><body>
<%@ page import = "java.util.*" %>
<b>Parameters:</b><br>
<%
 Enumeration parameterList = request.getParameterNames();
 while( parameterList.hasMoreElements() )
 {
    String   sName     = parameterList.nextElement().toString();
    String[] sMultiple = request.getParameterValues( sName );  // Formular liste
    if( 1 >= sMultiple.length )
        out.println( sName + " = " + request.getParameter( sName ) + "<br>" );
    else
       for( int i=0; i<sMultiple.length; i++ )
          out.println( sName + "[" + i + "] = " + sMultiple[i] + "<br>" );
 }
%>
</body></html>
```

3. Beispiel mit Java Server Pages

```
<%@ page import = "java.util.*" %>
<%
  final String s1 = "<tr bgcolor='#EBEEEE'><td>";
  final String s2 = "</td><td>";
  final String s3 = "</td></tr>\n";
  StringBuffer sb = new StringBuffer();
  Enumeration parameterList = request.getParameterNames();
  while( parameterList.hasMoreElements() )
  {
    String   sName     = parameterList.nextElement().toString();
    String[] sMultiple = request.getParameterValues( sName );
    if( 1 >= sMultiple.length )
      sb.append( s1 + sName + s2 + request.getParameter( sName ) + s3 );
    else
      for( int i=0; i<sMultiple.length; i++ ) {
        sb.append( s1 + sName + "[" + i + "]" + s2 + sMultiple[i] + s3 );
      }
  }
```

3. Beispiel mit Java Server Pages

```
if( sb.length()>0 )
   sb.insert( 0, "<table border=0 cellspacing=3 cellpadding=3>\n"
            + "<tr bgcolor='#EBEEEE'><th colspan='2'>"
            + "<big>Erhaltene Parameter</big></th></tr>\n" )
     .append( "</table>\n" );
%>

<html>
<body>
  <!--  Quelle: http://www.torsten-horn.de/techdocs/jsp-
grundlagen.htm#Einfuehrung  -->
```

3. Beispiel mit Java Server Pages

```
<form action="bsp_jsp_03.jsp?urlParm=seeUrl#Scroll" method="post">
  <input type="hidden" name="hidden" value="hid">
  <table border=0 cellspacing=3 cellpadding=3>
    <tr bgcolor='#EBEEEE'><th colspan='2'>
        <big>Formular</big><br>
        Bitte Eingaben ändern und Submit betätigen</th></tr>
    <tr bgcolor="#EBEEEE"><td>SelectDropDown</td>
      <td>
        <select name="SelectDropDown" size=1>
          <option value="1">Opt. 1</option>
          <option value="2" selected>Opt. 2</option>
          <option value="3">Opt. 3</option>
          <option value="4">Opt. 4</option>
        </select>
      </td></tr>
```

3. Beispiel mit Java Server Pages

```
<tr bgcolor="#EBEEEE"><td>SelectMultiple</td>
  <td>
    <select name="SelectMultiple" size=3 multiple>
      <option value="1">Opt. 1</option>
      <option value="2">Opt. 2</option>
      <option value="3" selected>Opt. 3</option>
      <option value="4" selected>Opt. 4</option>
    </select>
  </td></tr>
 <tr bgcolor="#EBEEEE"><td>Textarea</td>
  <td>
    <textarea name="Textarea" cols=20 rows=3>Text ...</textarea>
  </td></tr>
 <tr bgcolor="#EBEEEE"><td>Textfeld</td>
  <td>
    <input type="text" name="Textfeld" value="Text ..." size=20 maxlength=50>
  </td></tr>
```

3. Beispiel mit Java Server Pages

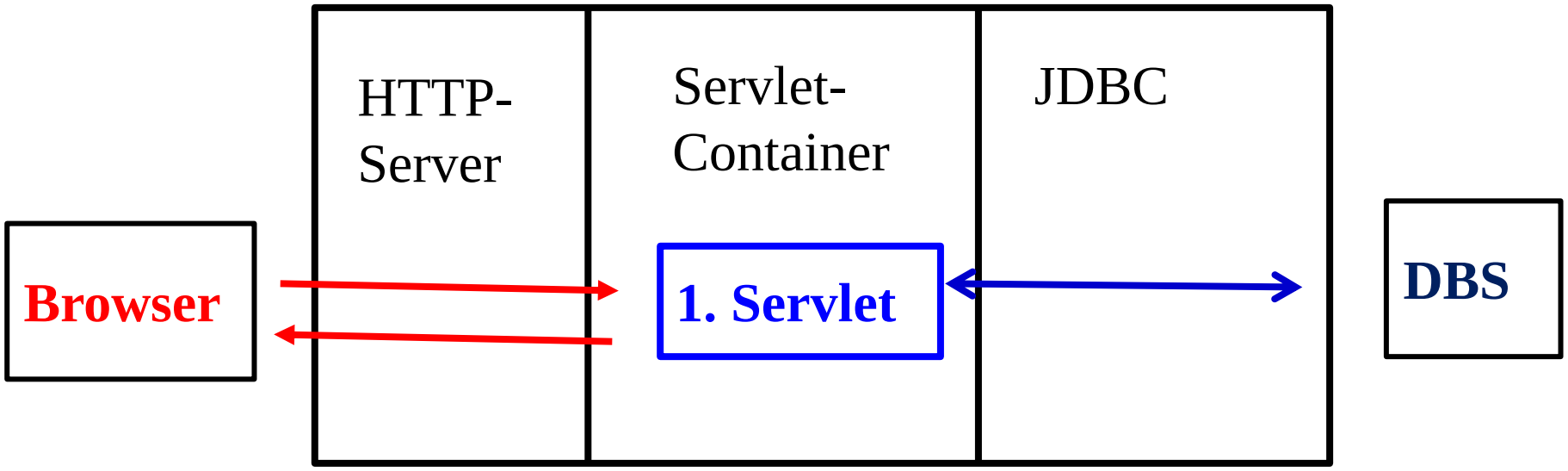
```
<tr bgcolor="#EBEEEE"><td>Passwort</td>
  <td>
    <input type="password" name="Passwort" value="xx" size=20 maxlength=10>
  </td></tr>
 <tr bgcolor="#EBEEEE"><td>Checkboxen cb1...cb3</td>
  <td>
    <input type="checkbox" name="cb1">
    <input type="checkbox" name="cb2" checked>
    <input type="checkbox" name="cb3">
  </td></tr>
 <tr bgcolor="#EBEEEE"><td>Radiobuttons ra</td>
  <td>
    <input type="radio" name="ra" value="1">
    <input type="radio" name="ra" value="2" checked>
    <input type="radio" name="ra" value="3">
  </td></tr>
```

3. Beispiel mit Java Server Pages

```
<tr bgcolor="#EBEEEE"><td>Submit</td>
    <td>
      <button type="submit" name="Submit" value="SubmitImg">
        <img src="http://www.torsten-horn.de/logo.gif" alt="Submit">
      </button>
      <input type="submit" name="Submit" value="Submit1">
      <input type="submit" name="Submit" value="Submit2">
    </td></tr>
  </table>
</form>

<a name="Scroll"></a>
<%= sb.toString() %>

</body>
</html>
```

Servlets

- Java Software-Komponenten zur dynamischen Erweiterung von Web-Servern
- Erzeugung dynamischer HTML-Seiten, z.B. aus Datenbank-Inhalten
- TypischeArchitektur:



Servlets

- Servlets sind Java-Klassen, die innerhalb eines Web-Servers ausgeführt werden.
- Der Web-Server muss Servlet-fähig sein, d.h. über einen Servlet- Container verfügen (z.B. XAMMP, Tomcat, Jetty).
- Container lädt Servlets bei Bedarf dynamisch nach.
- (HTTP-)Servlets werden (u.a.) über die HTTP-Anfragen GET bzw. POST angesprochen.
- Ein Servlet bearbeitet die Anfrage und erzeugt eine HTML-Seite.
- Die Bearbeitung erfolgt durch eigenen Thread im Adressraum des WebServers.

Servlets: Implementierung von HTTP-Servlets

- Ableiten einer Klasse von **javax.servlet.http.HttpServlet**
- I.d.R. überschreiben einer der Methoden
  - void **doGet**(HttpServletRequestrequest, HttpServletResponseresponse) throws IOException, ServletException
  - Behandlung von HTTP-GET-Anfragen
  - void **doPost**(...): analog für HTTP-POST-Anfragen
- Bei Bedarf überschreiben der Methoden
  - void init(): gerufen wenn Servlet geladen wird voidinit() : gerufen, wenn Servletgeladen wird
  - void destroy() : gerufen, wenn Servlet entfernt wird

Servlets: Beispiel mit Java EE

```
import java.io. *;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloSrv extends HttpServlet {
  private int counter = 0; // Wird bei HTTP−Get−Anfrage aufgerufen , nur Beispiel

  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws IOException,
ServletException {
    counter++;
   String name= request.getParameter("name");
   response.setContentType("text/html"); // Ausgabestrom für die erzeugte HTML−Seite
   PrintWriter out = response.getWriter();
   out.println("<html>");
   out.println("<head><title>Hallo World</title></head>");
   out.println("<body><b>" + counter + ". Hello to " + name "!/b/bd") + "!</b></body>");
   out.println("</html>");
   out.close();
  }
}
```

Servlets: Beispiel mit Java EE

```
<html>
<body>
  <form>
  <p>
    <a href="http://localhost:8080/test/hello?name=Paul">  Say Hello to Paul</a>
  </p>
  <p> Say Hello to:
   <form  method="get"     action="http://localhost:8080/test/hello"  >
   <input type="text"  name="name" size="10"  />
   <input type="submit" value="senden"  />
  </p>
  </form>
</body>
</html>
```

Servlets: Beispiel installieren

Deployment mit Tomcat-Server

- Übersetzen des Servlets
  - javac -classpath $CATALINA_HOME/lib/servlet-api.jar:.HelloSrv.java
- CLASSPATH nur notwendig, wenn J2EE nicht installiert ist
- Erstellen eines Deployment-Deskriptors unter WEBpyp INF/web.xml
- Kopieren der class-Datei(en) nach WEB-INF/classes
  - cp HelloSrv.class WEB-INF/classes
- Erzeugen eines WAR Archivs
  - jar -cvftest.war WEB-INF
- Kopieren des WAR Archivs in das Tomcat-Verzeichnis
  - cp test.war $CATALINA_BASE/webapps

Servlets: Deployment Descriptor

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application2.2//EN"
  "http://javasuncom/j2ee/dtds/web-app_2_2.dtd">
<web-app>
 <servlet>
   <servlet-name>HelloWorld</servlet-name>
   <servlet-class>HelloSrv</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>HelloWorld</servlet-name>
   <url-pattern>/hello</url-pattern>
 </servlet-mapping>
</web-app>
```

Servlets: Deployment mit Tomcat-Server

- Servlet ist nun unter dieser URL ansprechbar:
  - http://localhost:8080/test/hello
- HTML-Datei zum Aufruf des Servlets kann z.B. nach $TomCat/webapps/ROOT/hello.html kopiert werden
  - URL dann http://localhost:8080/hello.html
- **Anmerkungen:**
  - Servlet-Klassen dürfen nicht im CLASSPATH von Tomcat sein!
  - Tomcat nie im Verzeichnis starten in dem die Servlet-Klassen liegen!
  - Tomcat 7.0 kann beim Deployment auch laufen
  - WAR Archiv wird bei Änderung erneut ausgepackt, Klassen werden neu geladen

Servlets: Lebenszyklus

- Beim Start des Servers oder durch Client-Anfrage:
  - Servlet-Klasse wird in Web-Server geladen
  - eine Instanz der Servlet-Klasse wird erzeugt
  - die init()-Methode wird aufgerufen

- Bei einer HTTP-Anfrage:
  - Erzeugung eines neuen Threads, der die Methode doGet() bzw. doPost() ausführt
  - Implementierung der Methoden muss thread-sicher sein!
  - Bei Entfernung des Servlets aus dem Server
  - Aufruf der Methode destroy()

Servlets: Wichtige Methoden

- HttpServletRequest: HTTP-Anfrage
  - String getParameter(String name)
    - liefert Wert des genannten AnfrageParameters
    - z.B. bei GET /buy.html?what=shoe HTTP 1.0
  - HttpSessiongetSession()
    - liefert bzw. erzeugt Sitzungs-Objekt
- HttpServletResponse: HTTP-Antwort
  - void setContentType(String type)
    - setzt MIME-Typ der Antwort (i.d.R. "text/html")
  - PrintWritergetWriter()
    - liefert PrintWriterzum Schreiben der Ausgabe

Eclipse Enterprise Edition

Eclipse Enterprise Edition

Eclipse Enterprise Edition

Tomcat vs. Glassfish

- Tomcat ist keine Java EE Container, sondern nur ein Servlet-Container. Wenn man den vollen Umfang von Java EE ausnutzen will (Security etc.) muss man auf andere Software wechseln.

- Tomcat-Alternativen:
  - Glassfish
  - TomEE
  - WildFly
  - IBM Websphere
  - Oracle Weblogic

Eclipse Enterprise Edition

- **Erstellen eines ersten Projektes unter Java EE**

- New
- File
- Dynamic Web Project

New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: FirstServlet

Project location

☑ Use default location

Location: D:\progs\EclipseEE\FirstServlet Browse...

Target runtime

Apache Tomcat v7.0 New Runtime...

Dynamic web module version

3.0

Configuration

Default Configuration for Apache Tomcat v7.0 Modify...

A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name: FirstServletEAR New Project...

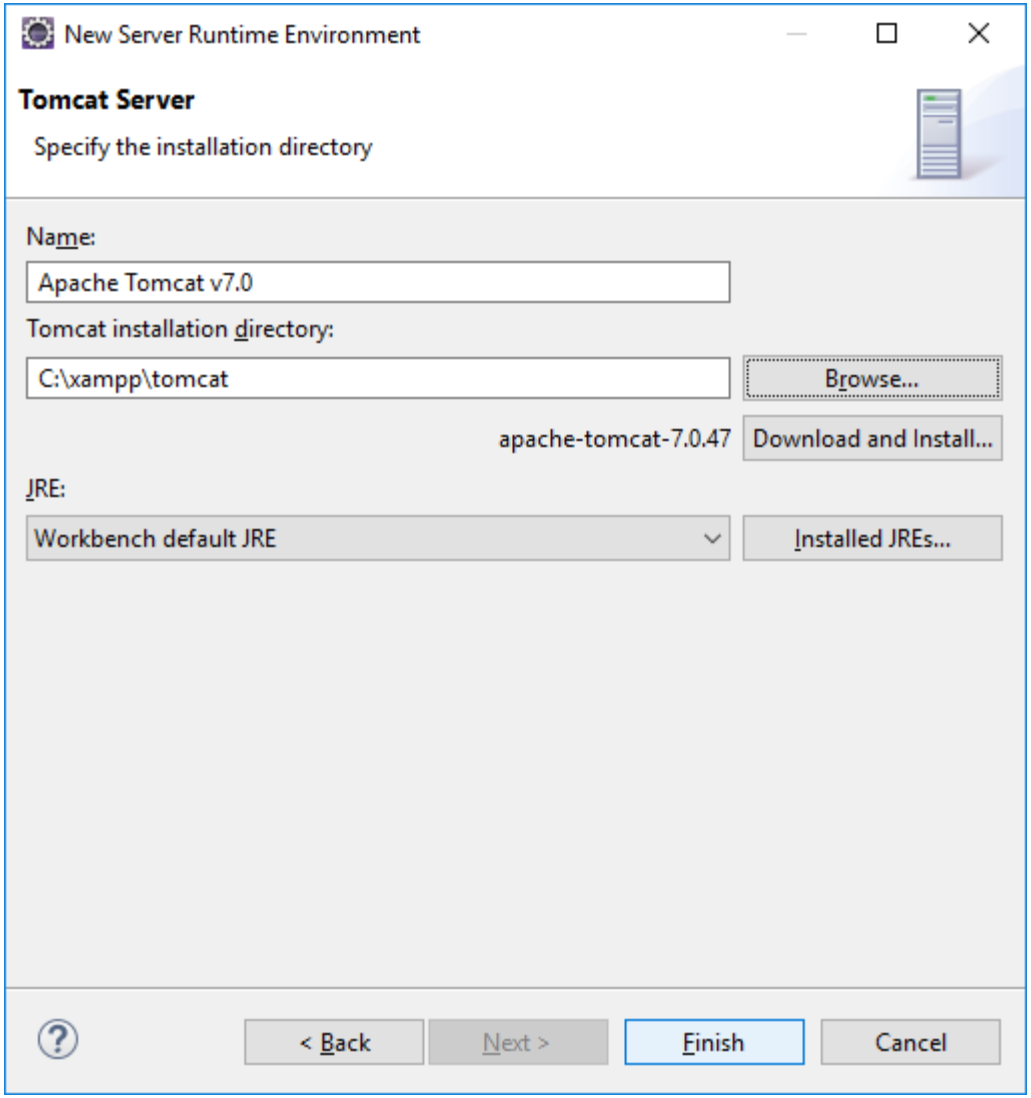
Working sets

☐ Add project to working sets New...

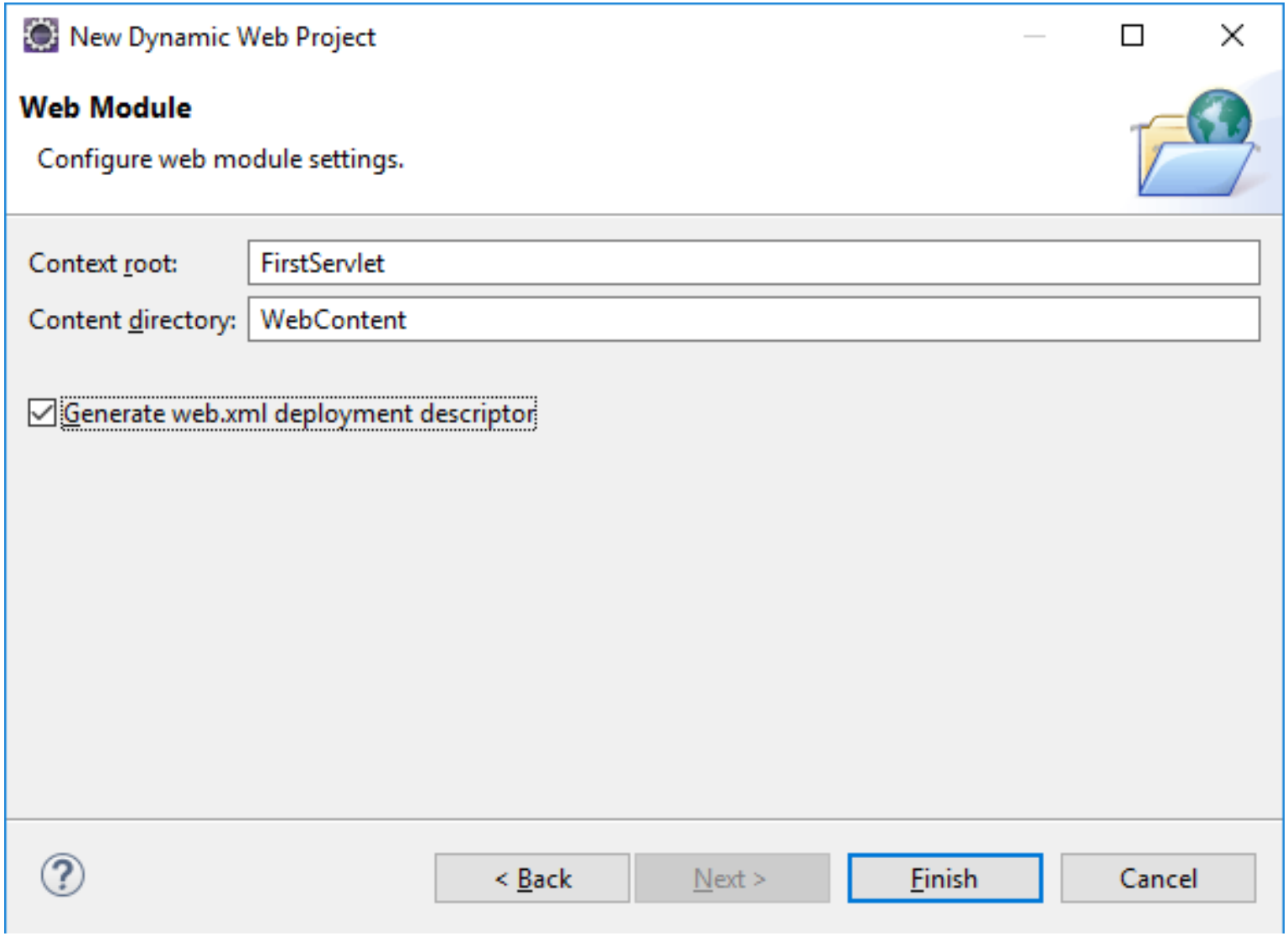
Working sets: Select...

< Back Next > Finish Cancel

Erstellen eines ersten Projektes unter Java EE



Erstellen eines ersten Projektes unter Java EE



Erstellen eines ersten Projektes unter Java EE

EclipseEE - Java EE - Eclipse

File Edit Navigate Search Project Run Window Help

Quick Access

Project Explorer

- FirstServlet
  - Deployment Descriptor: FirstServlet
  - JAX-WS Web Services
  - Java Resources
  - JavaScript Resources
  - build
  - WebContent

An outline is not available.

Welcome

Overview
Get an overview of the feat

Tutorials
Go through tutorials

Samples
Try out the samples

What's New
Find out what is new

Ma... Pr... Se... Da... Sni...

0 items

Description	Resource

FirstServlet

Project Explorer

FirstSe

De

JA

Jav

Jav

bui

We

New
Go Into
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Refactor Alt+Shift+T
Import
Export
Refresh F5
Close Project
Close Unrelated Projects
Validate
Show in Remote Systems view
Run As
Debug As
Profile As
Restore from Local History...
Java EE Tools
Team
Compare With
Configure
Source
Properties Alt+Enter

Project...
File
Folder
SQL File
Annotation
Class
Enum
Interface
Package
Source Folder
Dynamic Web Project
Enterprise Application Project
JavaScript Source File
HTML File
JSP File
Filter
Listener
Servlet
Example...
Other... Ctrl+N

Erstellen eines ersten Projektes unter Java EE

Create Servlet

Create Servlet

Specify class file destination.

Project: FirstServlet

Source folder: \FirstServlet\src — Browse...

Java package: de.test — Browse...

Class name: FirstServlet

Superclass: javax.servlet.http.HttpServlet — Browse...

☐ Use an existing Servlet class or JSP

Class name: FirstServlet — Browse...

< Back | Next > | Finish | Cancel

Erstellen eines ersten Projektes unter Java EE

Create Servlet

Create Servlet

Enter servlet deployment descriptor specific information.

Name: FirstServlet

Description:

Initialization parameters:

Name	Value	Description

Add... Edit... Remove

URL mappings:

/FirstServlet

Add... Edit... Remove

☐ Asynchronous Support

< Back | Next > | Finish | Cancel

Erstellen eines ersten Projektes unter Java EE

Create Servlet

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☑ public ☐ abstract ☐ final

Interfaces: javax.servlet.Servlet

Add... Remove

Which method stubs would you like to create?

☑ Constructors from superclass
☑ Inherited abstract methods

☐ init	☐ destroy	☐ getServletConfig
☐ getServletInfo	☐ service	☑ doGet
☑ doPost	☐ doPut	☐ doDelete
☐ doHead	☐ doOptions	☐ doTrace

< Back Next > Finish Cancel

Quellcode

```
package de.test;
import *.*

@WebServlet("/FirstServlet")
public class FirstServlet extends HttpServlet implements Servlet {
  private static final long serialVersionUID = 1L;
```

Quellcode

```
 public FirstServlet() {
    super();
 }
 protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
            response.getWriter().append("Served at: ").append(
                                    request.getContextPath());
 }
 protected void doPost(HttpServletRequest request,
   HttpServletResponse response)
   throws ServletException, IOException {
        doGet(request, response);
}
```

Quellcode

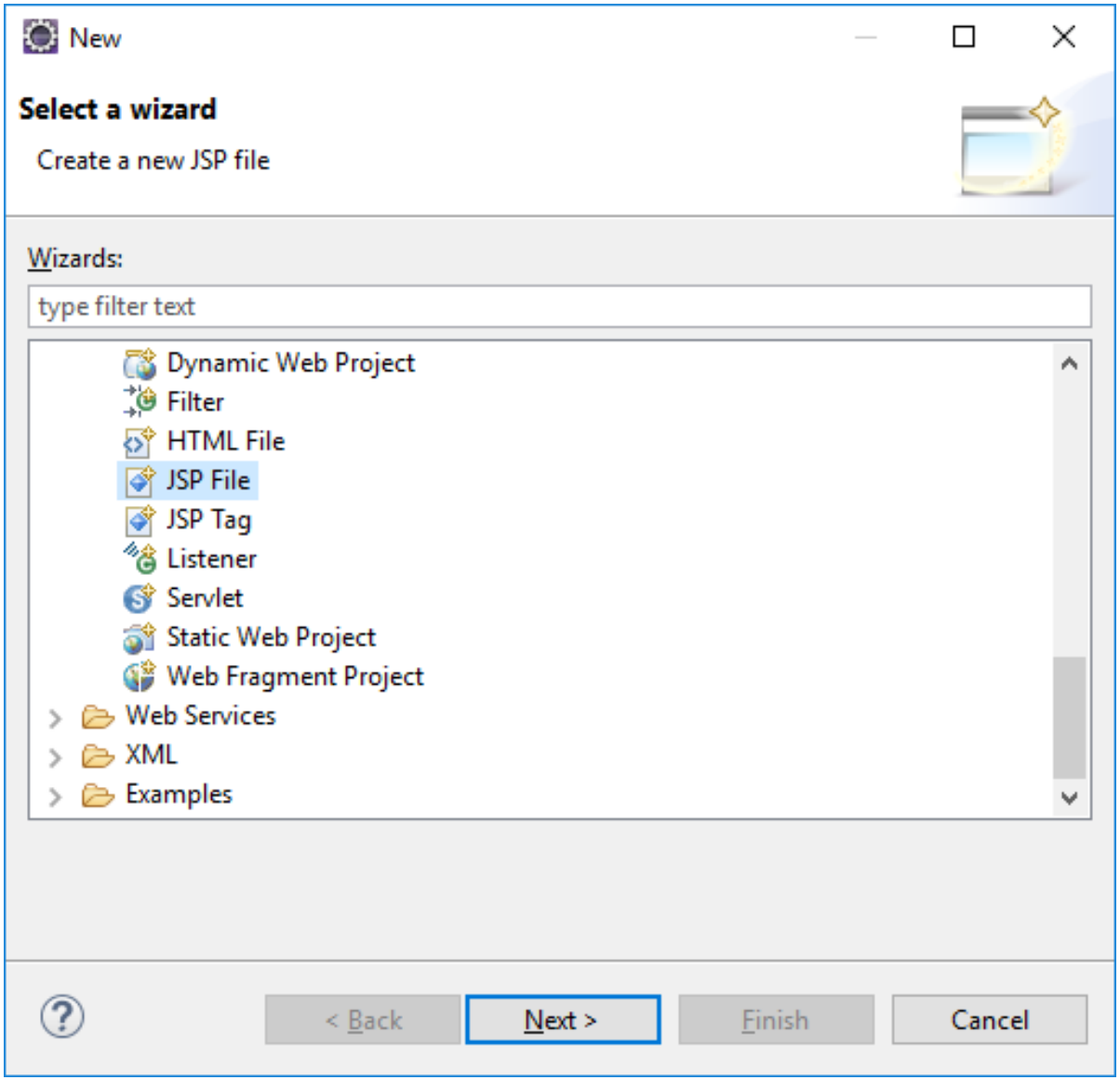
```
protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
            doGet(request, response);
}



}
```

Eclipse Enterprise Edition: Neue JSP Datei

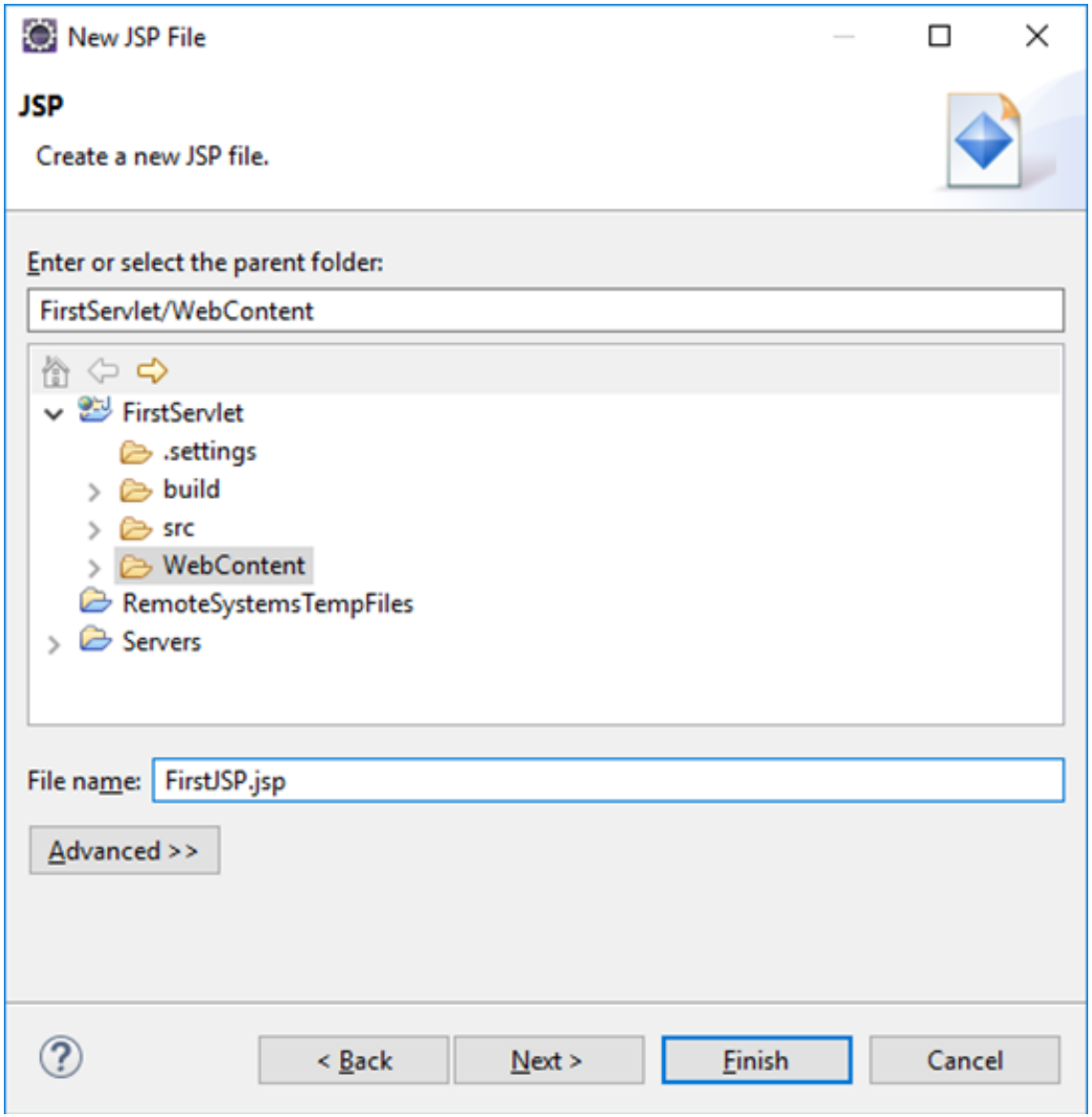
- File
- New
- Other
- Web
- JSP File

oder

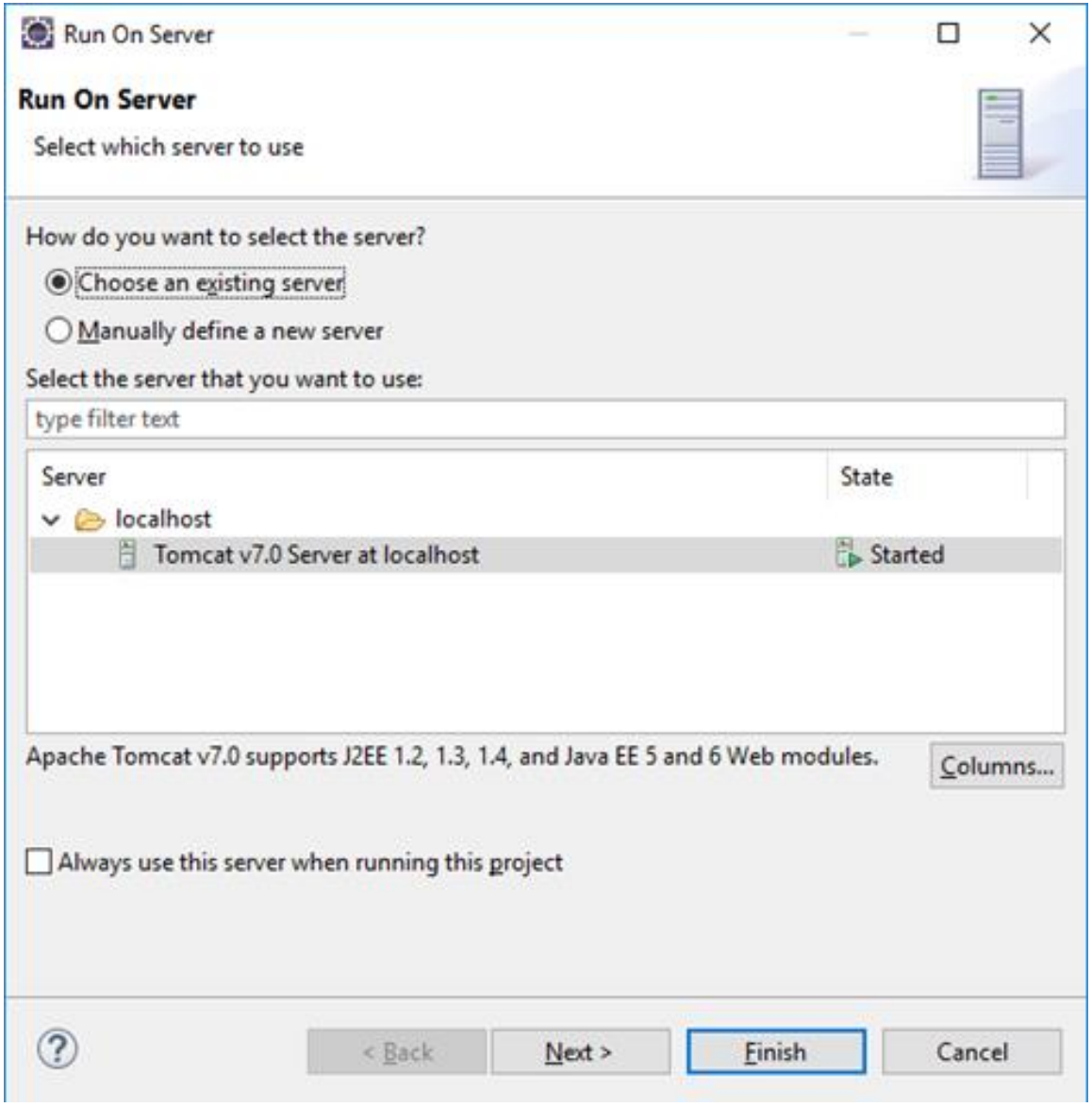
- Strg+N



Eclipse Enterprise Edition: Neue JSP Datei



Eclipse Enterprise Edition: Neue JSP Datei



Eclipse Enterprise Edition: Neue JSP Datei

EclipseEE - Java EE - http://localhost:8080/FirstServlet/FirstJSP.jsp - Eclipse

File Edit Navigate Search Project Run Window Help

Quick Access

Project Expl...

- FirstServlet
  - Deployment Descript
  - JAX-WS Web Services
  - Java Resources
  - JavaScript Resources
  - build
  - WebContent
- Servers
  - Tomcat v7.0 Server at

FirstServlet.java | FirstJSP.jsp | Insert title here

http://localhost:8080/FirstServlet/FirstJSP.jsp

My first JSP-Formular-Sample

Number 12

Send Delete

An outline is not available.

Markers | Properties | Servers | Data Source Explorer | Snippets | Console

Tomcat v7.0 Server at localhost [Started, Synchronized]

Fertig