



Servlet & JSP

Stefan Engel

Agenda

- Servlets
- Eclipse + JBoss Tools
- Projekt anlegen
- Ein einfaches Servlet entwickeln
- Webanwendung deployen
- JSP
- JSTL
- Session Management



Servlets

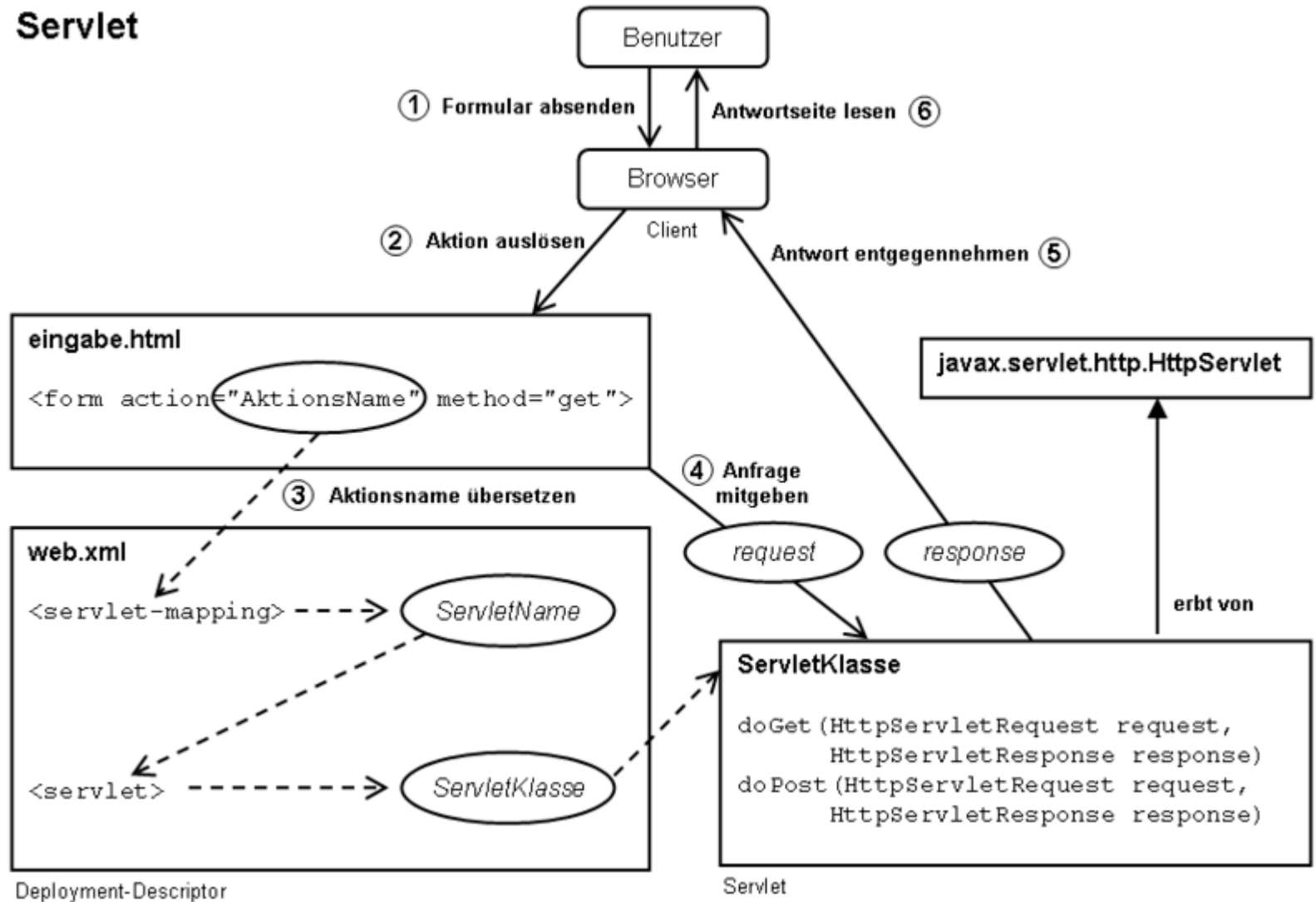
Servlets (1)

- Java-Klassen, die in einem Webcontainer laufen
- Bestandteil von allen Java EE Application Servern
- „Servlet“ = „Server“ + „Applet“

Servlets (2)

- Ein Servlet muss das Interface `javax.servlet.Servlet` implementieren
 - → Meistens: ableiten von `javax.servlet.http.HttpServlet`

Servlet



Quelle: <http://upload.wikimedia.org/wikipedia/commons/d/db/Servlet.png>

Servlets (4)

```
12 public class MyServlet extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14
15     public MyServlet() {
16         super();
17     } //ctor
18
19     protected void doGet(HttpServletRequest request, HttpServletResponse response)
20         throws ServletException, IOException {
21
22         System.out.println( "doGet called" );
23
24     } //doGet
25
26     protected void doPost(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28
29         System.out.println( "doPost called" );
30
31     } //doPost
32
33 } //MyServlet
```



Vorbereitungen

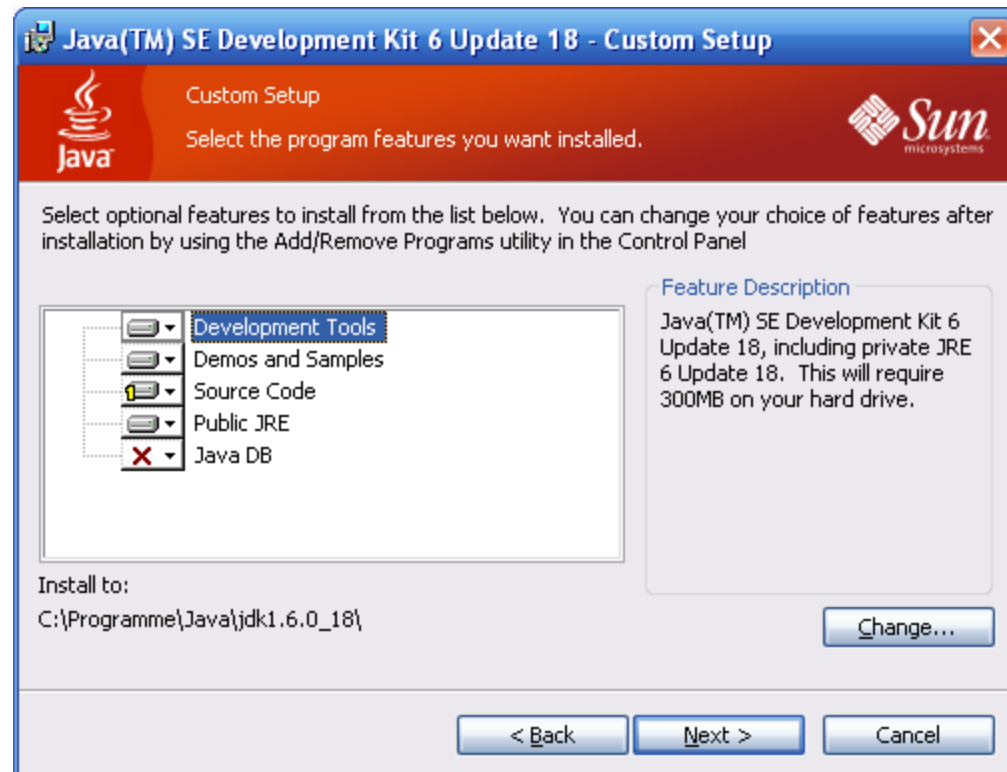
- Wir wollen:
 - Ein einfaches Servlet erstellen und aufrufen
- Dazu muss vorher erledigt werden:
 - JDK6 installieren
 - Eclipse mit JBoss/Hibernate Plugin installieren
 - JBoss in Eclipse integrieren



JDK6 installieren

- Download von JDK6

→ <http://java.sun.com/javase/downloads/widget/jdk6.jsp>





Eclipse installieren

- Download von Eclipse (für Java EE)
→ <http://www.eclipse.org/downloads/>

Eclipse Downloads

[Eclipse Packages](#)[Projects](#)[Development Builds](#)

Galileo Packages (based on Eclipse 3.5 SR1) - [Compare Packages](#)

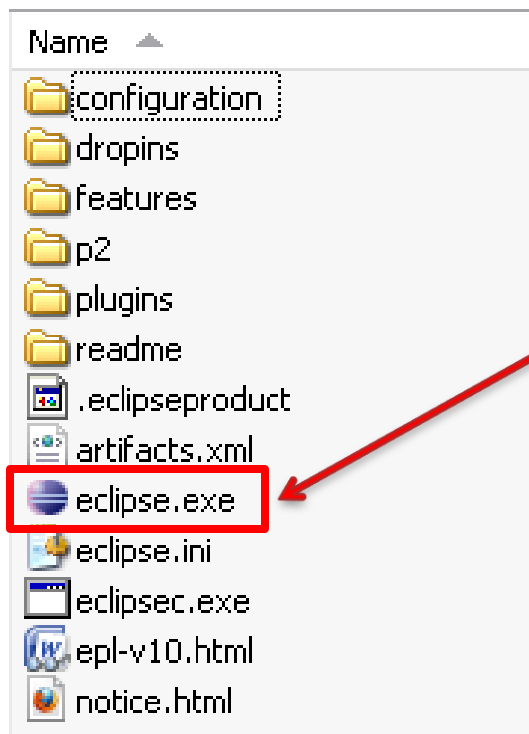
Eclipse IDE for Java EE Developers (189 MB)
Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn and others. [More...](#)
Downloads: 2,135,190

Eclipse IDE for Java Developers (92 MB)
The essential tools for any Java developer, including a Java IDE, a CVS client, XML Editor and Mylyn. [More...](#)
Downloads: 1,066,851



Eclipse installieren (2)

- Verzeichnis entpacken – fertig!

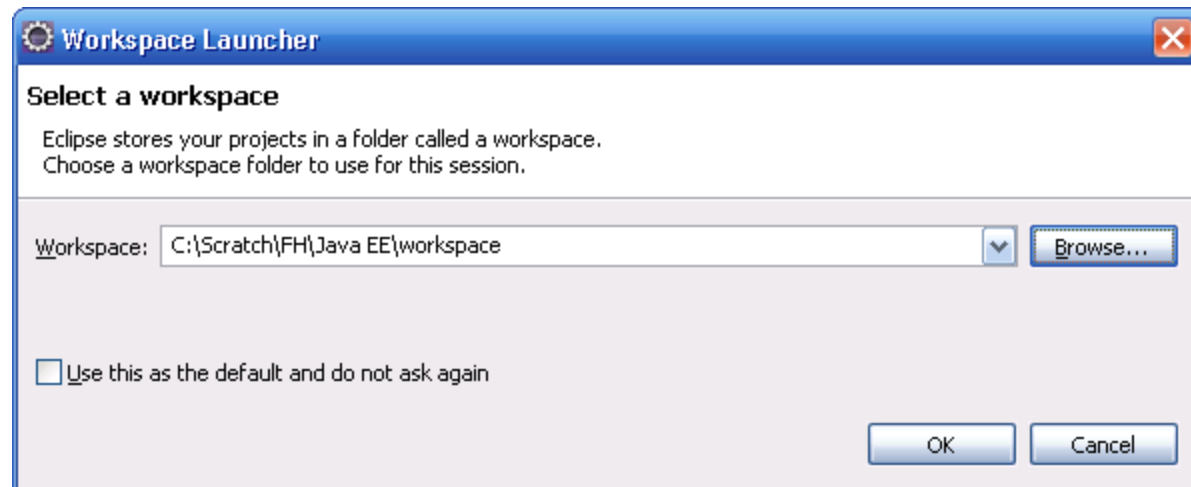


hier starten!



Eclipse starten

- Beim ersten Start muss ein workspace angegeben werden (frei wählbar)





JBosstools installieren

- Download von:
→ <http://www.jboss.org/tools/download.html>

Downloads

To install JBoss Tools you must first install to **appropriate version of Eclipse** - see Ir

Stable Releases

JBoss Tools 3.0 :: [Eclipse 3.4.2](#)

JBoss Tools 2.1 :: [Eclipse 3.3.2](#)

Development Milestones

JBoss Tools 3.1 :: [Eclipse 3.5.0](#)

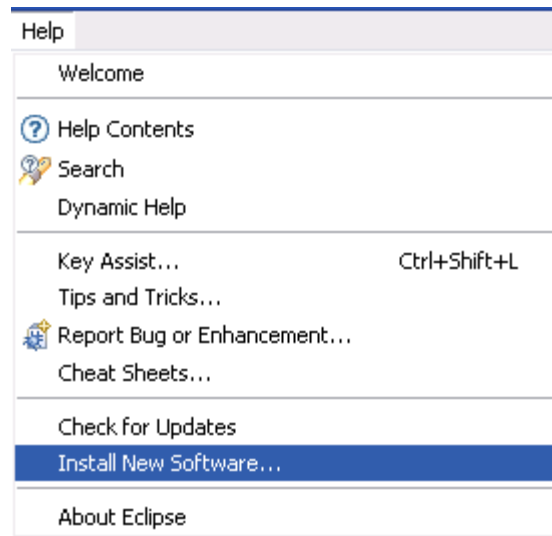
JBoss Tools 3.0 :: [Eclipse 3.4.2](#)

Description	Version	Download
All Plugins (repo) - A p2 repo (update site) bundle of all JBoss Tools plugins - install instructions	3.1.0.v200912250601M	All platforms (87.33 MB)



JBosstools installieren (2)

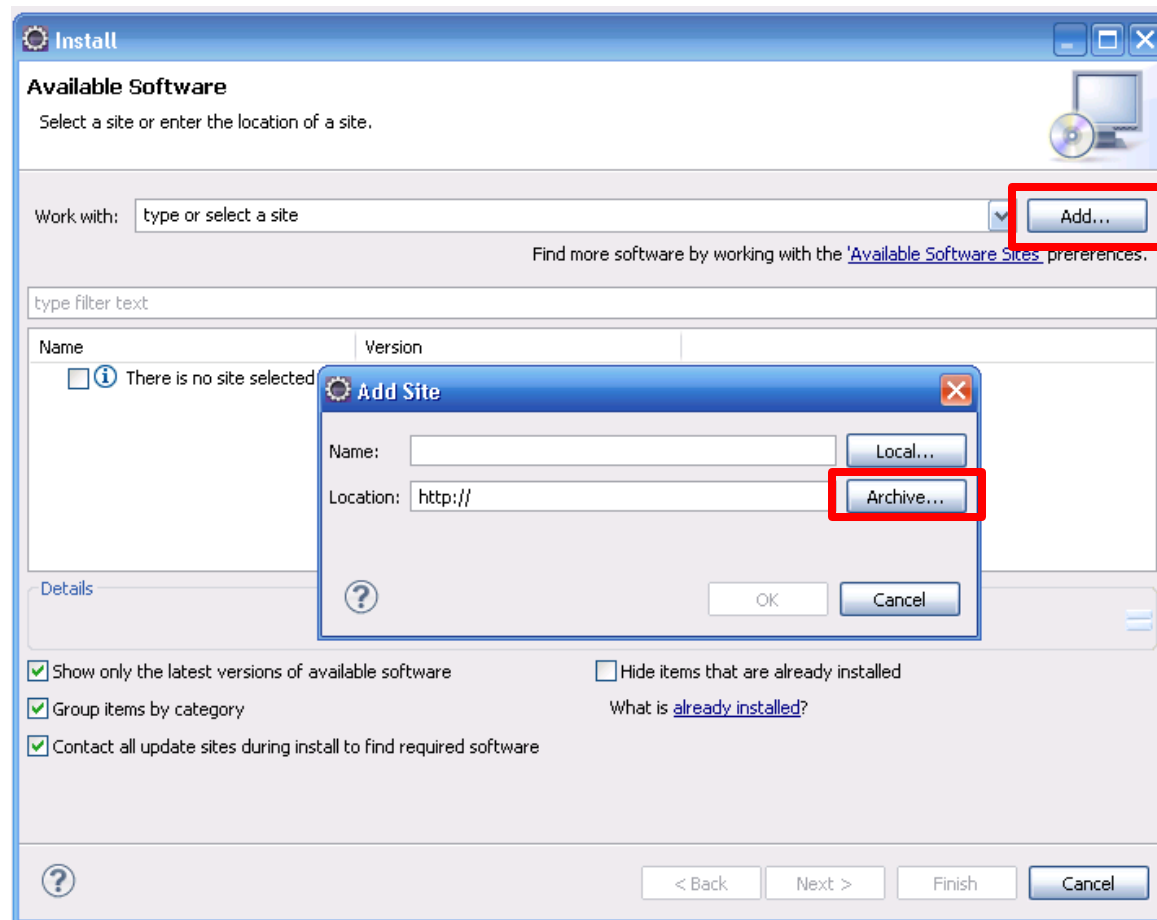
- In Eclipse: Help → Install New Software...





JBosstools installieren (3)

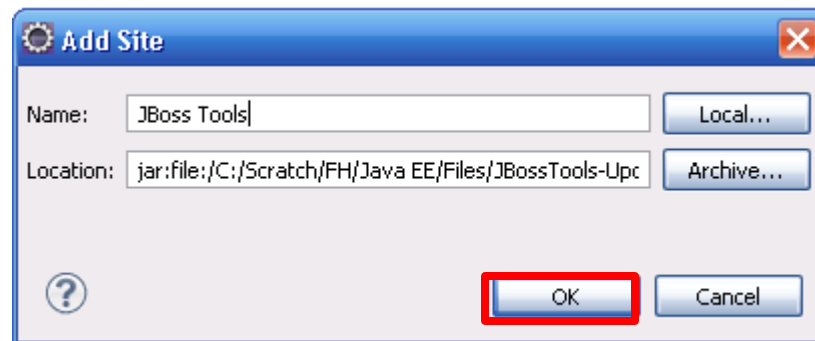
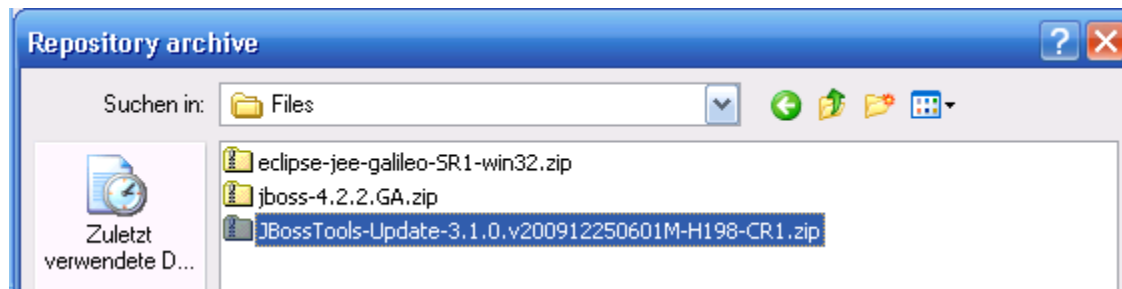
- Add → Archive





JBosstools installieren (4)

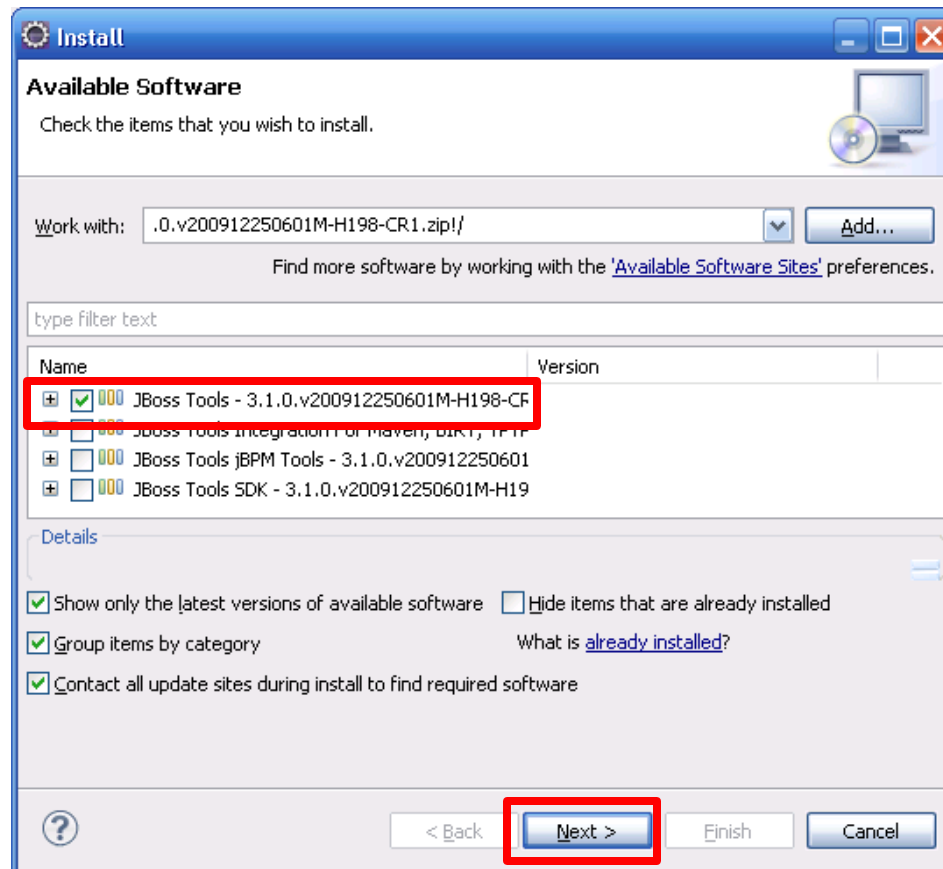
- Jetzt das JBoss Tools Archiv auswählen





JBosstools installieren (5)

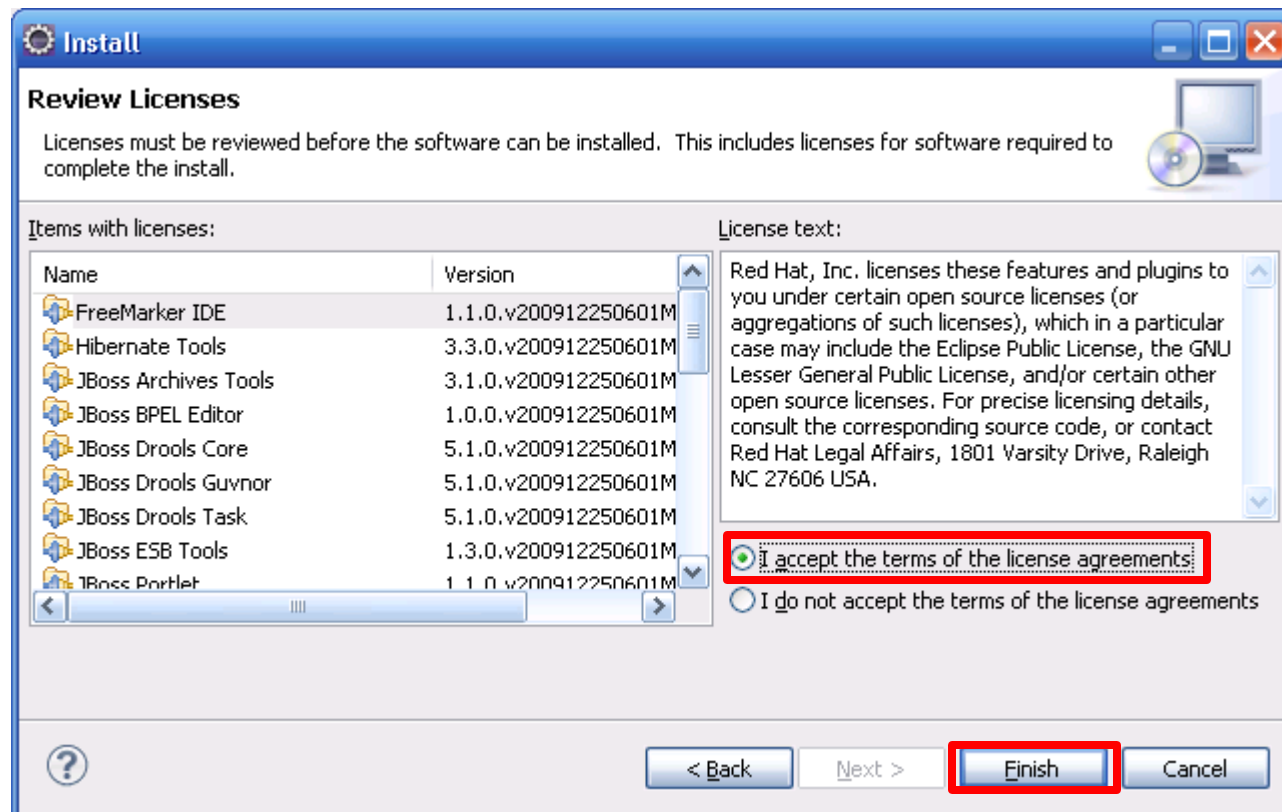
- Nur die Tools auswählen





JBosstools installieren (6)

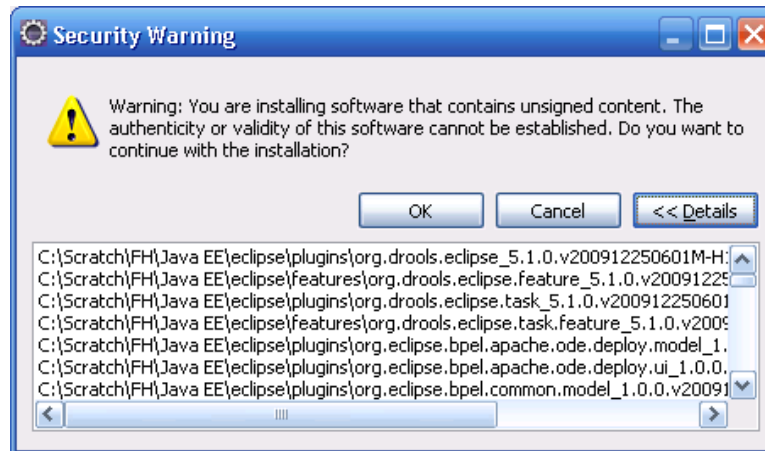
- Next → I accept... → Finish



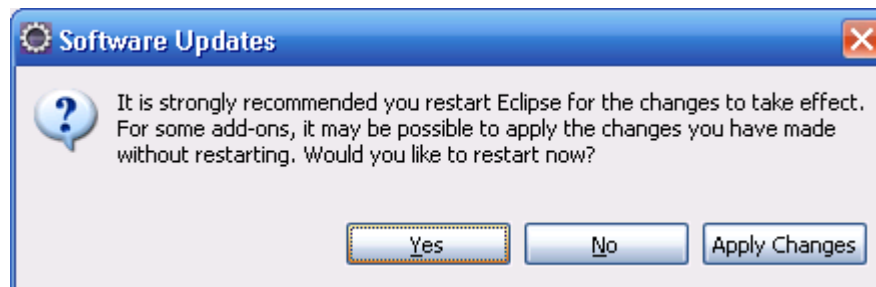


JBosstools installieren (7)

- Evtl. Security Warning bestätigen



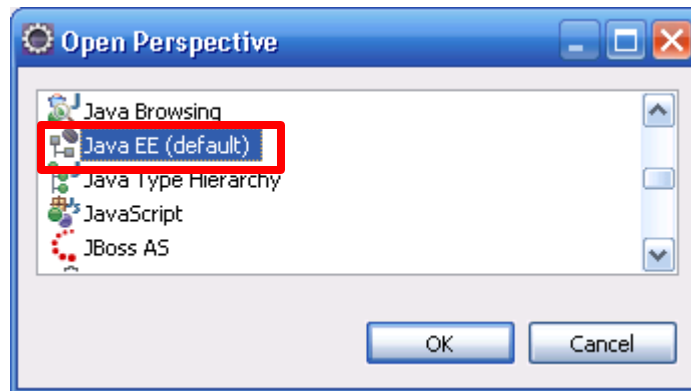
- Eclipse neu starten





JBoss in Eclipse steuern

- In Eclipse: Window → Open Perspective → Other → Java EE

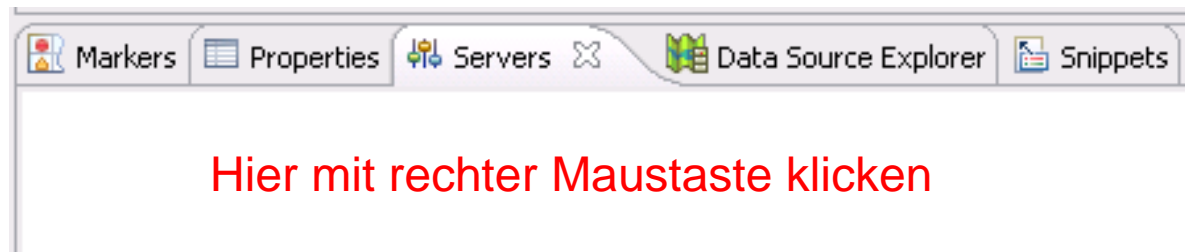


- Die Java EE Perspektive ist möglicherweise bereits eingestellt!



JBoss in Eclipse steuern (2)

- Im unteren Bereich erscheint die „Server View“

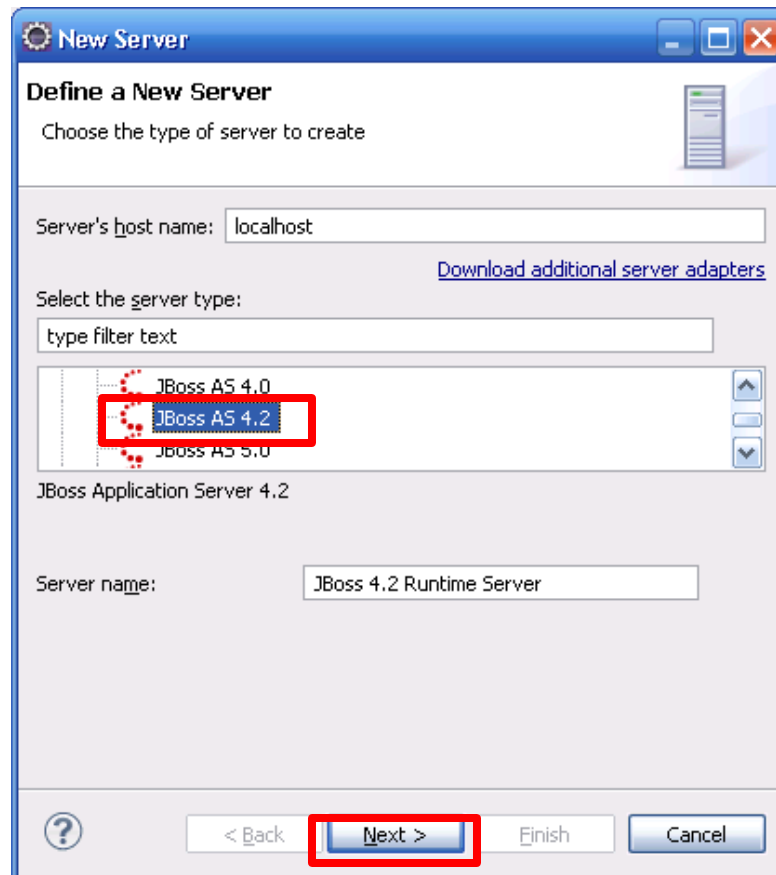


- Rechtsklick → New → Server



JBoss in Eclipse steuern (3)

- JBoss AS 4.2 auswählen





JBoss in Eclipse steuern (4)

New Server

JBoss Runtime
JBoss Application Server 4.2

A JBoss Server runtime references a JBoss installation directory. It can be used to set up classpaths for projects which depend on this runtime, as well as by a "server" which will be able to start and stop instances of JBoss.

Name
JBoss 4.2 Runtime

Home Directory
C:\Scratch\FH\Java EE\jboss-4.2.2.GA Browse...

JRE
Default JRE JRE

Configuration
Directory: server Browse...

- all
- custom**
- default
- minimal

Copy... Delete...

Finish Cancel

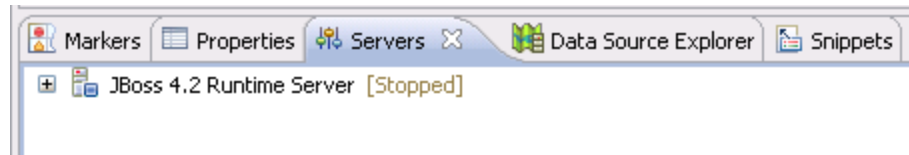
Hier Verzeichnis zum entpackten JBoss angeben

Hier erscheinen dann alle Serverkonfigurationen. Wir wählen „custom“ aus



JBoss in Eclipse steuern (5)

- JBoss erscheint jetzt in der „Servers“ View



- Zum Starten mit der rechten Maustaste anklicken und „Start“ wählen



JBoss in Eclipse steuern (6)

- Eine „Console“ View wird sich öffnen und die Ausgaben von JBoss anzeigen
- Falls nicht: Rechtsklick auf den Server → Show in → Console

```
JBoss 4.2 Runtime Server [JBoss Application Server Startup Configuration] C:\Programme\Java\jre6\bin\javaw.exe (17.02.2010 15:49:48)
15:50:37,757 INFO [DLQ] Bound to JNDI name: queue/DLQ
15:50:37,977 INFO [ConnectionFactoryBindingService] Bound ConnectionManager 'jboss.jca:service=ConnectionFactoryBinding,name=JmsXA'
15:50:38,208 INFO [TomcatDeployer] deploy, ctxPath=/jmx-console, warUrl=../deploy/jmx-console.war/
15:50:38,428 INFO [Http11Protocol] Starting Coyote HTTP/1.1 on http-localhost%2F127.0.0.1-8080
15:50:38,458 INFO [AjpProtocol] Starting Coyote AJP/1.3 on ajp-localhost%2F127.0.0.1-8009
15:50:38,468 INFO [Server] JBoss (MX MicroKernel) [4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200710221139)] Started in 46s:266ms
```



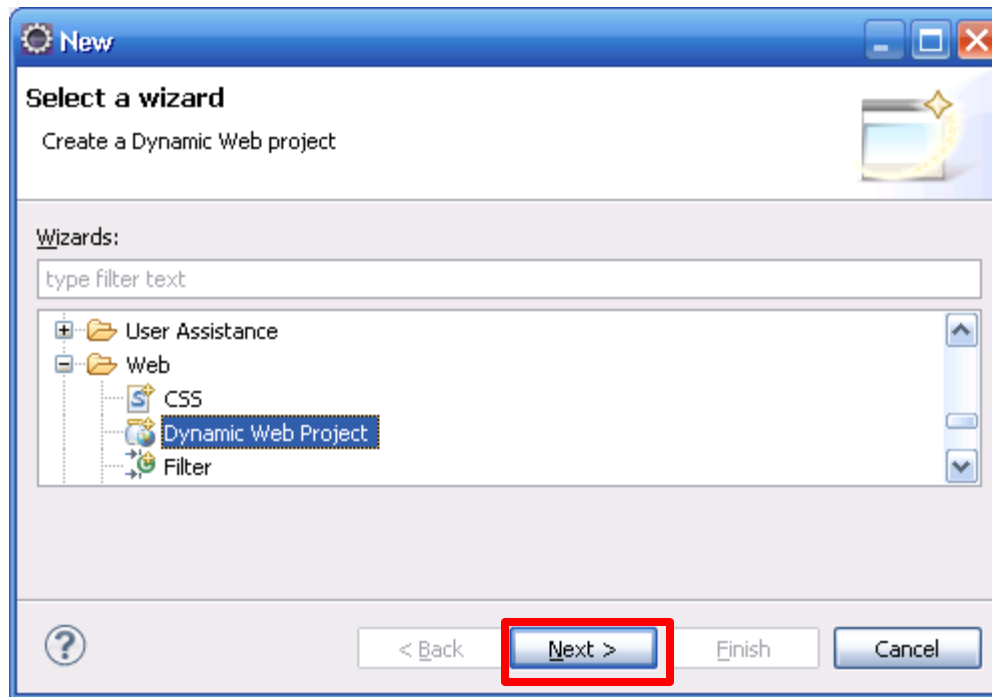
Aufgaben

- Hello World in einem Servlet ausgeben
 - An den Browser eines beliebigen Clients



Projekt anlegen

- In Eclipse: File → New → Other
- Web → Dynamic Web Project





Projekt anlegen (2)

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: HelloServlet

Project contents
☒ Use default
Directory: C:\Scratch\FH\Java EE\workspace\HelloServlet Browse...

Target runtime
JBoss v4.2 New...

Dynamic web module version
2.5

Configuration
Default Configuration for JBoss v4.2 Modify...
A good starting point for working with JBoss v4.2 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name: EAR New...

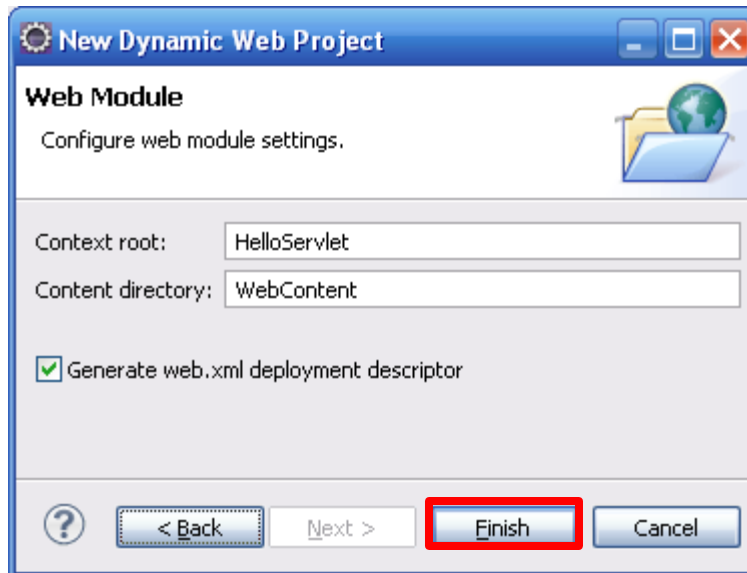
Working sets
☐ Add project to working sets
Working sets: Select...

< Back Next > Finish Cancel

Projektname festlegen,
ansonsten stimmen
meist schon die
Standardeinstellungen!



Projekt anlegen (3)

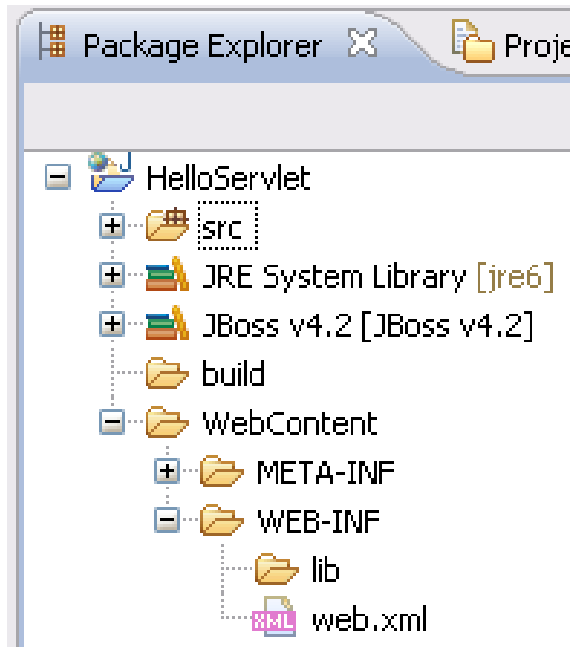


Immer „Next >“ klicken,
bis dieser letzte Dialog
erscheint, dann „Finish“



Projekt anlegen (4)

- Automatisch generierte Projektstruktur:



Wichtige Verzeichnisse:

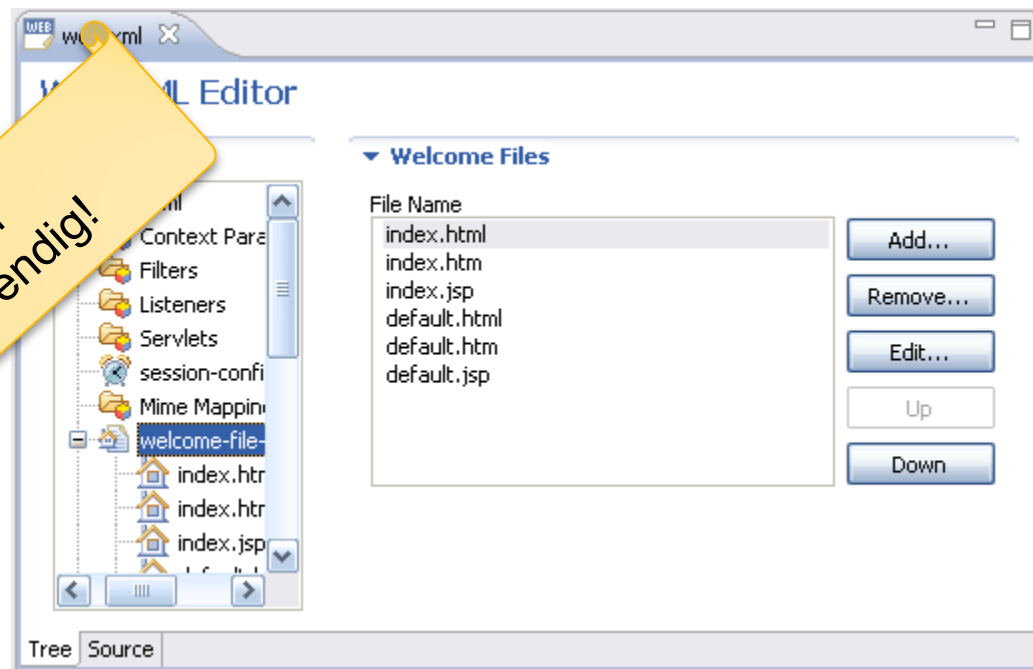
- src
→ enthält Java Quellcode
- WebContent
→ Enthält JSPs
- WebContent\WEB-INF
→ web.xml
- WebContent\WEB-INF\lib
→ enthält alle Bibliotheken (werden automatisch eingebunden)



Die Datei web.xml

- „Deployment Descriptor“
- Legt fest, welche Servlets/JSPs über welche Namen erreichbar sind

Keine manuellen
Änderungen notwendig!





Servlet erzeugen

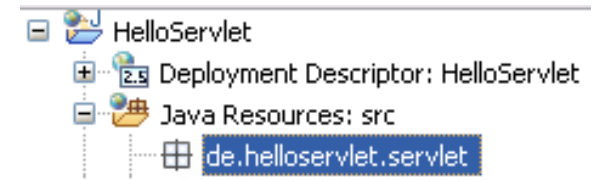
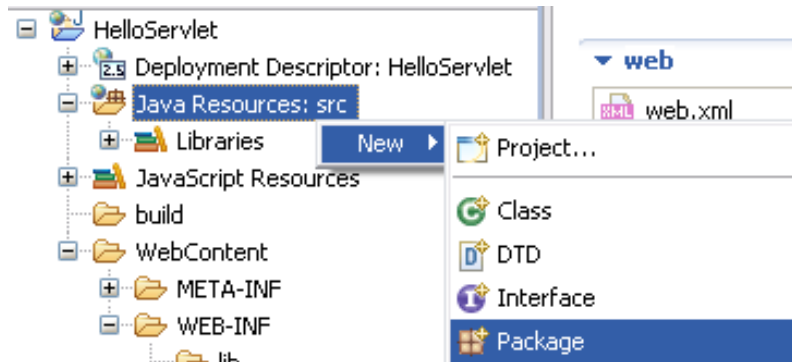
- Zuerst neues Package anlegen:
 - Rechtsklick auf „src“, dann New → Package
 - Name für das Package:
`de.helloservlet.servlet`





Servlet erzeugen (2)

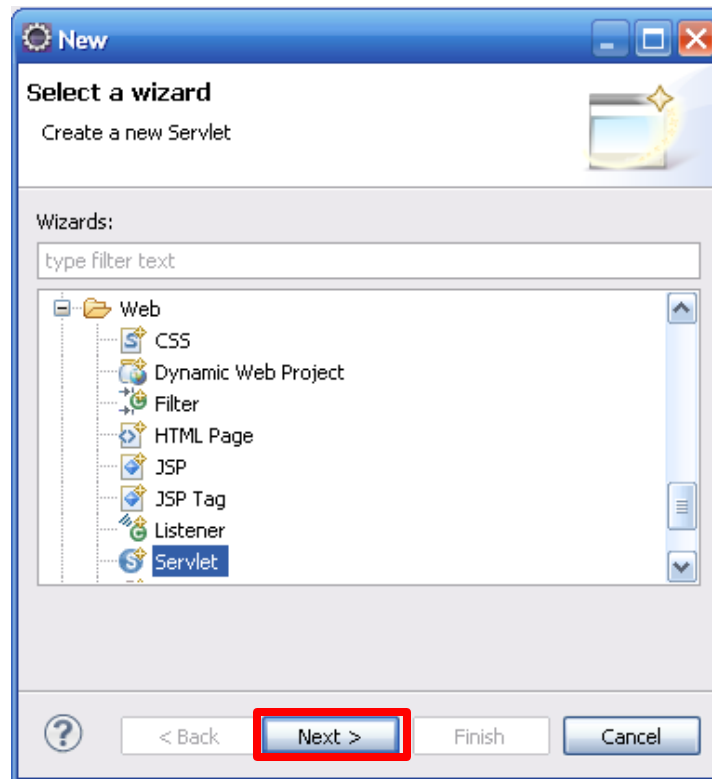
- Rechtsklick auf das neue Package, dann New → Other...





Servlet erzeugen (3)

- Beim Unterpunkt „Web“ den Punkt „Servlet“ auswählen





Servlet erzeugen (4)

- Klassename „MyFirstServlet“ wählen

Create Servlet
Specify class file destination.

Web project: HelloServlet

Source folder: /HelloServlet/src Browse...

Java package: de.helloservlet.servlet Browse...

Class name: MyFirstServlet

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

☐ Use an existing Servlet class or JSP

Class name: MyFirstServlet Browse...

? < Back **Next >** Finish Cancel



Servlet erzeugen (5)

- Gewünschte URL Mappings angeben

Create Servlet

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization Parameters:

Name	Value	Description
<input type="text"/>		

Buttons: Add..., Edit..., Remove

URL Mappings:

<input type="text" value="/MyFirstServlet"/>
--

Buttons: Add..., Edit..., Remove

Navigation: < Back, **Next >**, Finish, Cancel

web.xml wird
automatisch
angepasst!



Servlet erzeugen (6)

- Methoden auswählen

Create Servlet

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

Modifiers: ☒ Public ☐ Abstract ☐ Final

Interfaces: Add... Remove

Which method stubs would you like to create?

☒ Constructors from superclass
☒ Inherited abstract methods

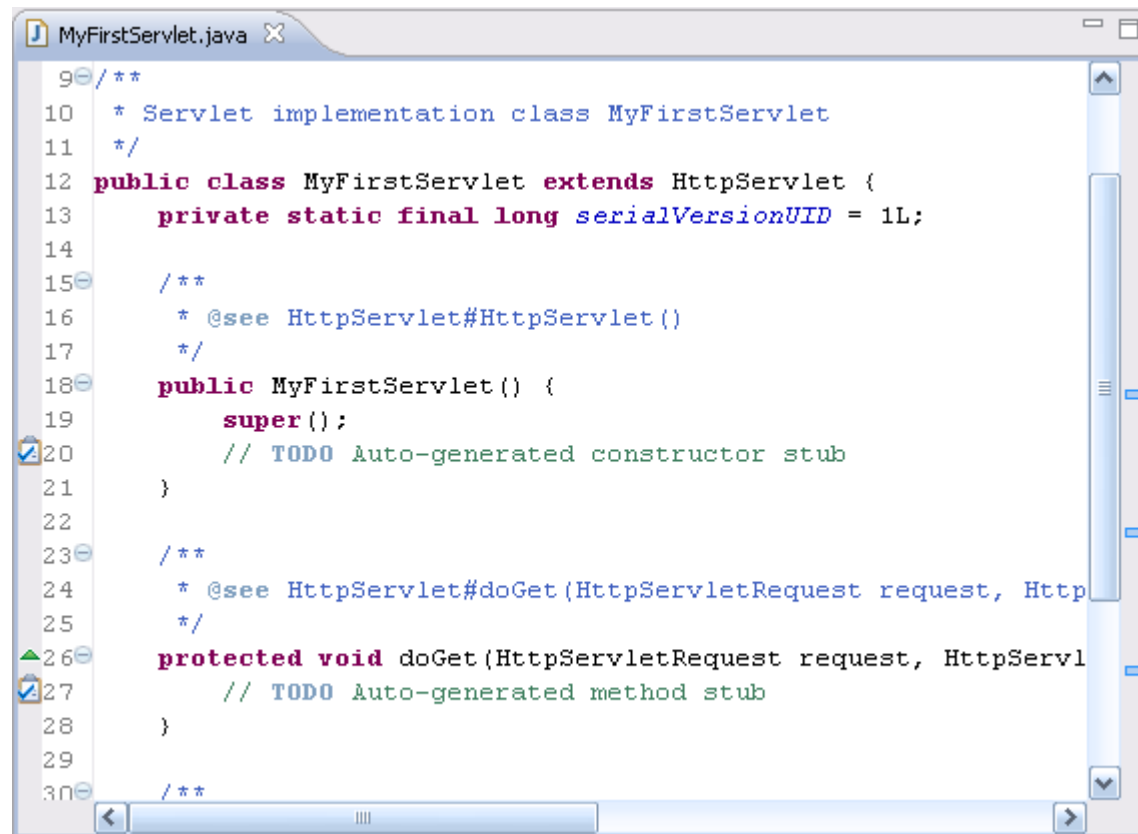
<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input checked="" type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

? < Back Next > Finish Cancel



Servlet erzeugen (7)

- Eclipse generiert nun das Servlet



```
MyFirstServlet.java
9 /**
10  * Servlet implementation class MyFirstServlet
11  */
12 public class MyFirstServlet extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14
15     /**
16      * @see HttpServlet#HttpServlet()
17      */
18     public MyFirstServlet() {
19         super();
20         // TODO Auto-generated constructor stub
21     }
22
23     /**
24      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
25      */
26     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27         // TODO Auto-generated method stub
28     }
29
30     /**
```

Intermezzo: post vs. get

- Post
 - Alle Parameter, welche mittels HTML Formular übertragen wurden
 - Dieses muss das Attribut „method“ auf „post“ gesetzt haben
 - Beliebig viele/lange Parameter möglich

```
<form action="zielseite" method="post">  
    <input type="text" name="param1" />  
</form>
```

Intermezzo: post vs. get (2)

- Get
 - Alle Parameter werden mit in die URL codiert
 - URL wird sehr lang!
 - Erster Parameter wird mit ? Eingeleitet, alle weiteren mit &

`http://localhost/MyApp/index.php?param1=hallo¶m2=welt`



Servlet ausfüllen

- Wir implementieren sowohl doPost() als auch doGet()
- Unterschiedliches Verhalten je nach post/get nicht notwendig
 - Beide sollen die Methode handleRequest aufrufen, welche wir selber hinzufügen



Servlet ausfüllen (2)

- Parameter request und response einfach weiterreichen
- `handleRequest` gibt „Hello Servlet!“ auf die Konsole aus
- Code: `Beispiele\Servlets\Hello Servlet` auf Konsole



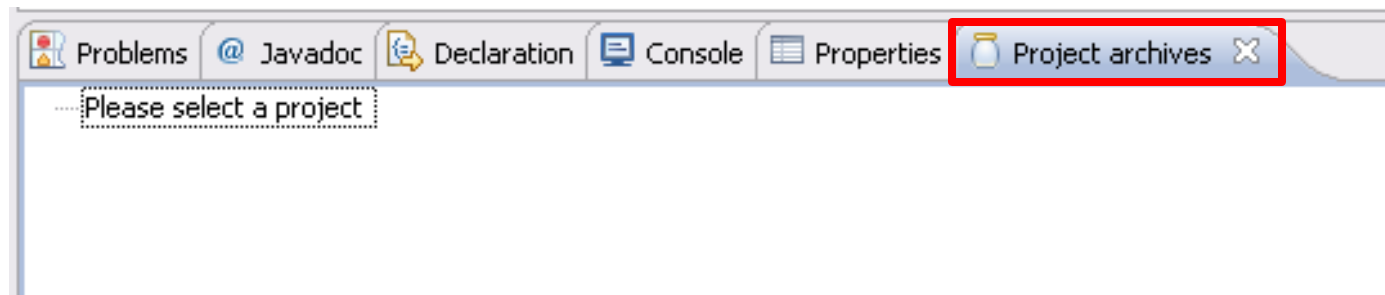
Servlet ausfüllen (3)

```
23-  /**
24   * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
25   */
26-  protected void doGet(HttpServletRequest request, HttpServletResponse response)
27      throws ServletException, IOException {
28      handleRequest(request, response);
29  }
30
31-  /**
32   * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
33   */
34-  protected void doPost(HttpServletRequest request, HttpServletResponse response)
35      throws ServletException, IOException {
36      handleRequest(request, response);
37  }
38
39-  protected void handleRequest(HttpServletRequest request, HttpServletResponse response)
40      throws ServletException, IOException {
41      System.out.println( "Hello Servlet!" );
42  }
```



Webanwendung deployen

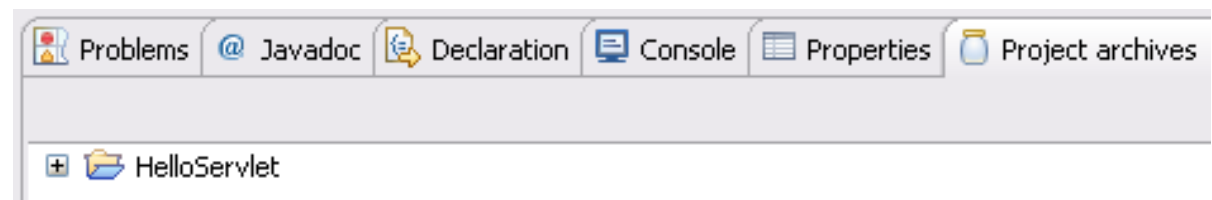
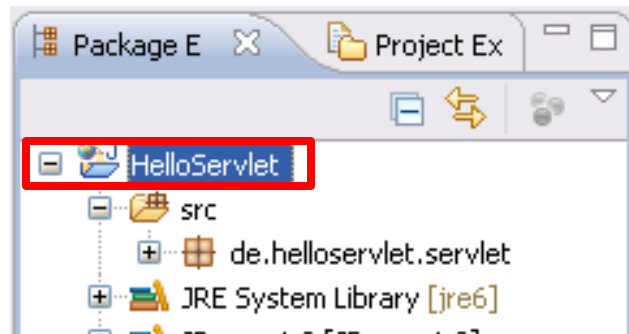
- Anwendung packen und auf Server kopieren
- Eclipse hilft dabei: in der JBoss AS Ansicht unten auf „Project Archives“ klicken





Webanwendung deployen (2)

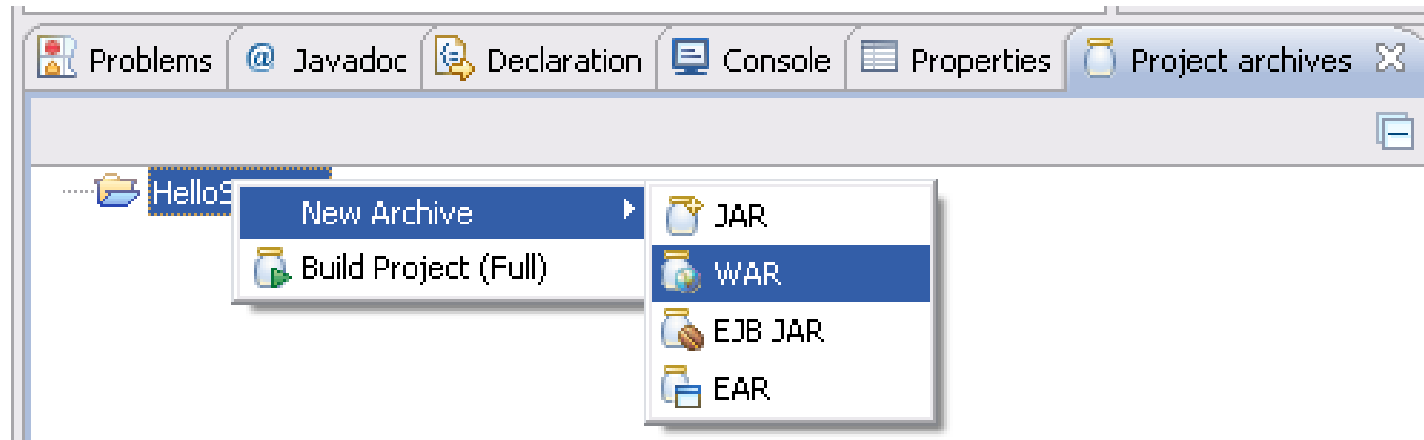
- Links auf das Projekt klicken
 - Es erscheint dann bei Project Archives





Webanwendung deployen (3)

- Bei „Project Archives“ Rechtsklick auf das Projekt, dann New Archive → WAR





Webanwendung deployen (4)

New WAR

Create a new archive

Archive information

Archive name:

Destination:

Relative to ☒ workspace ☐ file system

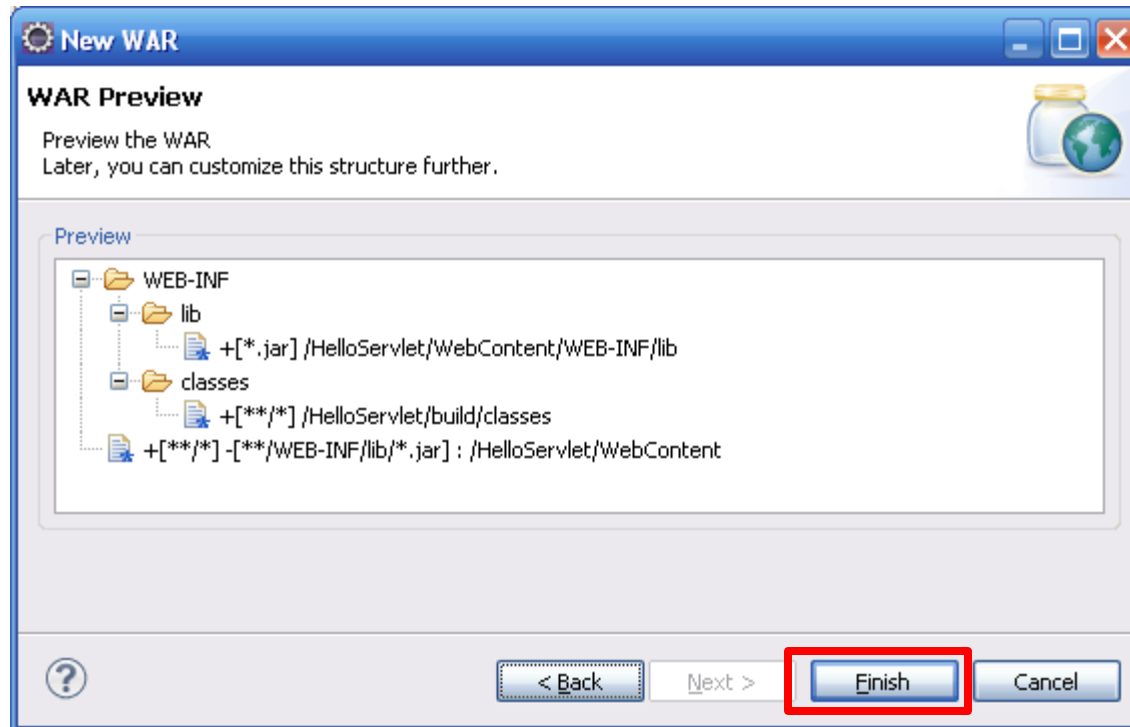
Archive type

☒ Standard archive using zip compression

☐ Exploded archive resulting in a folder (no compression)



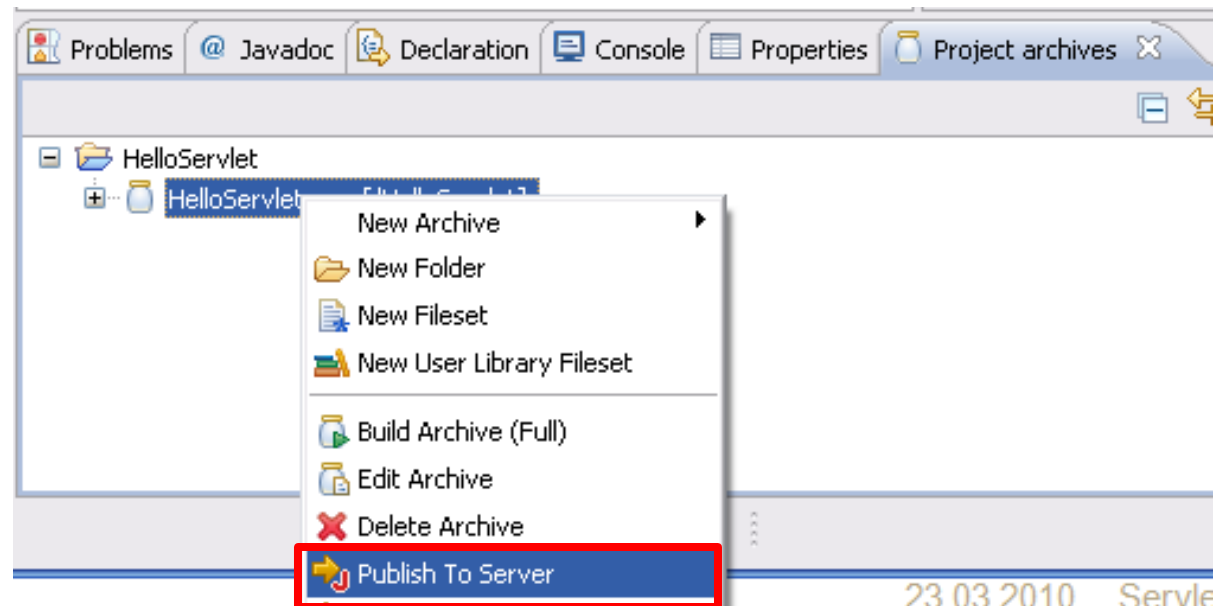
Webanwendung deployen (5)



Standardmässig werden alle Bibliotheken und .class-Dateien mit gepackt,
Sowie der komplette Inhalt des Verzeichnisses „WebContent“ !



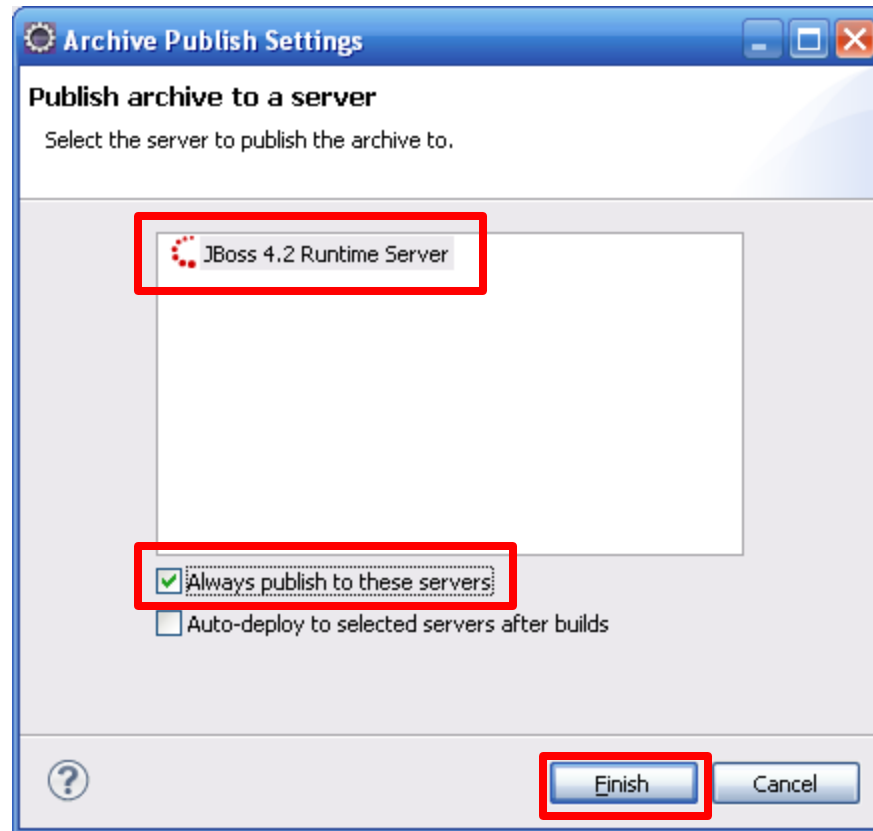
Webanwendung deployen (6)



Rechtsklick auf die neu entstandene WAR-Datei → Publish To Server



Webanwendung deployen (7)



JBoss kann auch nachträglich noch gestartet werden!



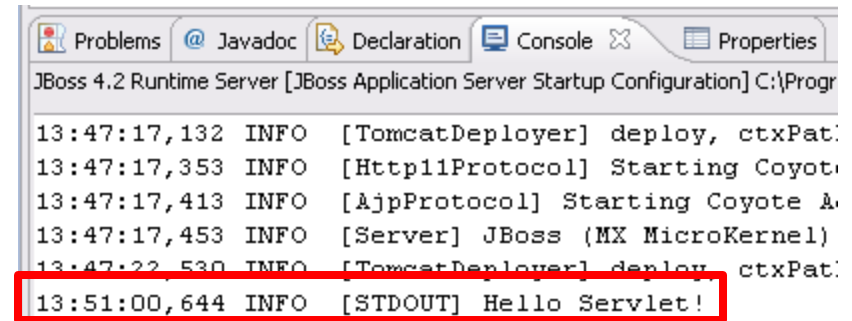
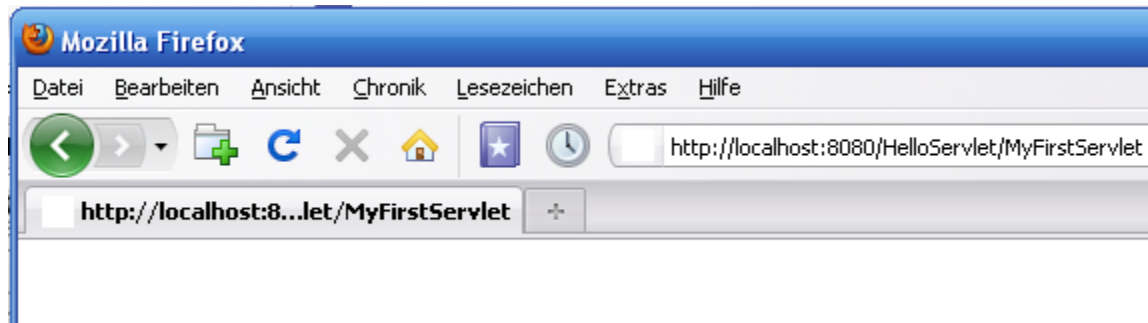
Webanwendung deployen (8)

```
JBoss 4.2 Runtime Server [JBoss Application Server Startup Configuration] C:\Programme\Java\jre6\bin\javaw.  
13:47:16,672 INFO [DLQ] Bound to JNDI name: queue/DLQ  
13:47:16,902 INFO [ConnectionFactoryBindingService] Bound ConnectionManager 'jboss.jca:service=Cor  
13:47:17,132 INFO [TomcatDeployer] deploy, ctxPath=/jmx-console, warUrl=.../deploy/jmx-console.war  
13:47:17,353 INFO [Http11Protocol] Starting Coyote HTTP/1.1 on http-localhost%2F127.0.0.1-8080  
13:47:17,413 INFO [AjpProtocol] Starting Coyote AJP/1.3 on ajp-localhost%2F127.0.0.1-8009  
13:47:17,453 INFO [Server] JBoss (MX MicroKernel) [4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200  
13:47:22,530 INFO [TomcatDeployer] deploy, ctxPath=/HelloServlet, warUrl=.../tmp/deploy/tmp2936155
```

Anwendung kann jetzt vom Browser aus aufgerufen werden!



Webanwendung aufrufen



→ <http://localhost:8080/HelloServlet/MyFirstServlet>



HTML Seite anzeigen

- Möglichkeit 1:
 - HTML Code im Servlet an den Browser schicken
- Nachteile
 - Unflexibel
 - Präsentationsschicht steckt im Controller
→ keine 3-Schichten-Architektur!



HTML Seite anzeigen (2)

- Servlet anpassen:

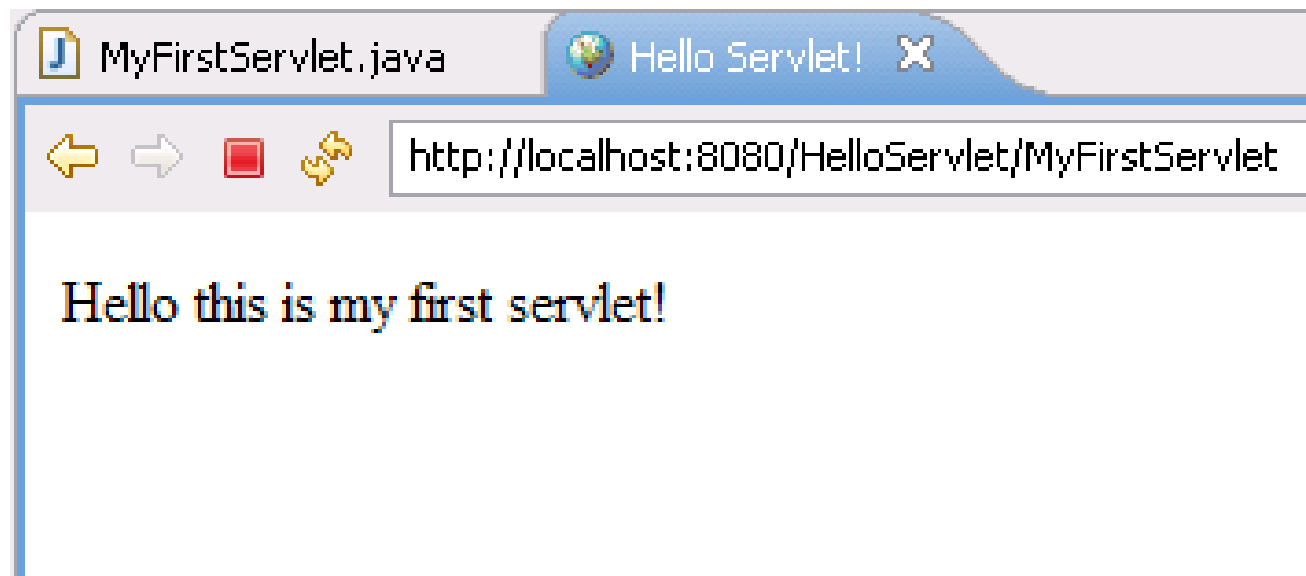
```
39-   protected void handleRequest(HttpServletRequest request, HttpServlet
40       throws ServletException, IOException {
41       System.out.println( "Hello Servlet!" );
42       response.getWriter().write( "<html>"
43           + "<head><title>Hello Servlet!</title></head>"
44           + "<body>Hello this is my first servlet!</body>"
45           + "</html>" );
46   }
```

- Writer-Objekt kann über das Response-Objekt bezogen werden
 - write() sendet Text zum Browser



HTML Seite anzeigen (3)

- Anzeige im Browser (hier: Eclipse-Intern)



Fazit

- HTML Seiten können im Servlet generiert und an den Browser gesendet werden
- Unsauber, schlecht wartbar
- Gibt es eine bessere Lösung??



JSP

Java Server Pages

Was sind Java Server Pages?

- Ähnlich zu HTML
- Enthalten spezielle Tags zur Vereinfachung der Datenpräsentation
- Jetzt: Trennung in reine Präsentationsschicht möglich!

JSP Erweiterungen

- Standard Taglib (JSTL) (beliebte Implementierung: Apache Standard Taglibs)
 - Iterieren über Listen
 - Kontrollstrukturen
 - Etc...
- Eigene Tags
 - Können frei in Java implementiert werden



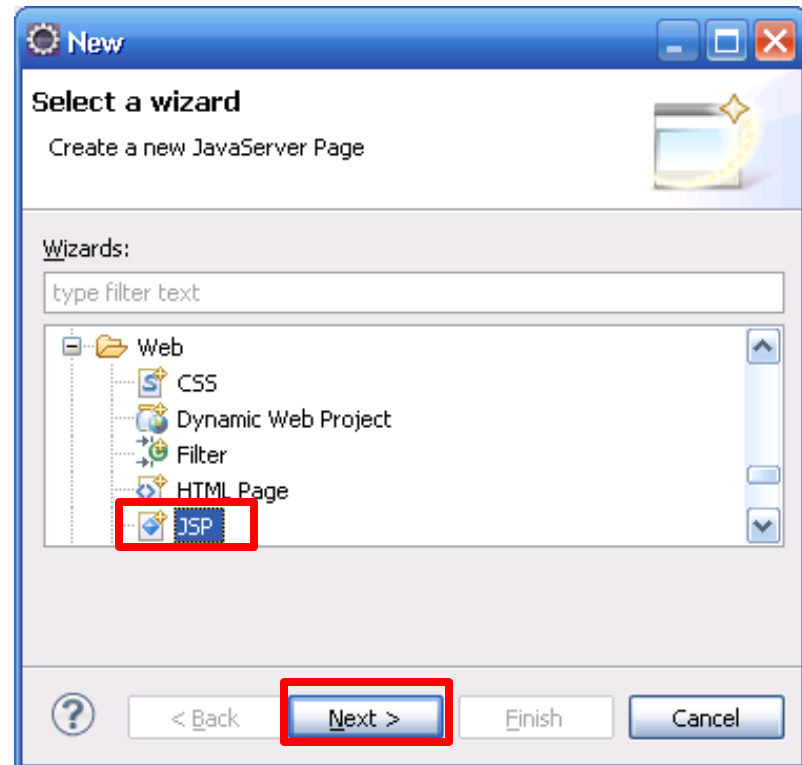
Aufgaben

- HTML Code aus dem Servlet entfernen und in eine JSP auslagern
- Einen Parameter per URL übergeben und in der JSP anzeigen
- Einen Wert in der Benutzersession speichern (im Servlet) und anzeigen (in der JSP!)



JSP erzeugen

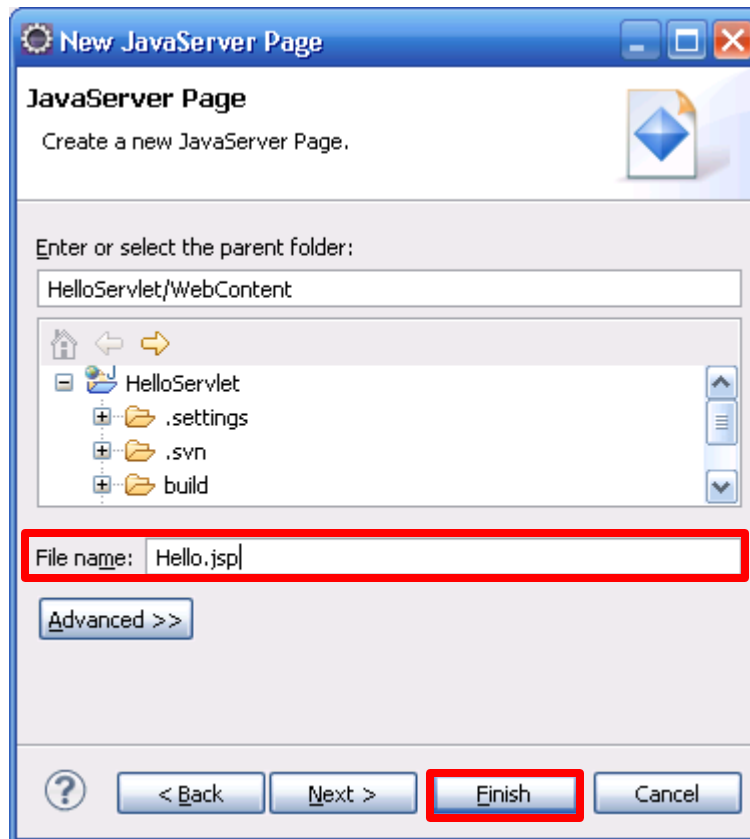
- Rechtsklick auf das Verzeichnis WebContent → New → Other → Web → JSP





JSP erzeugen (2)

- Name festlegen, „Finish“ anklicken





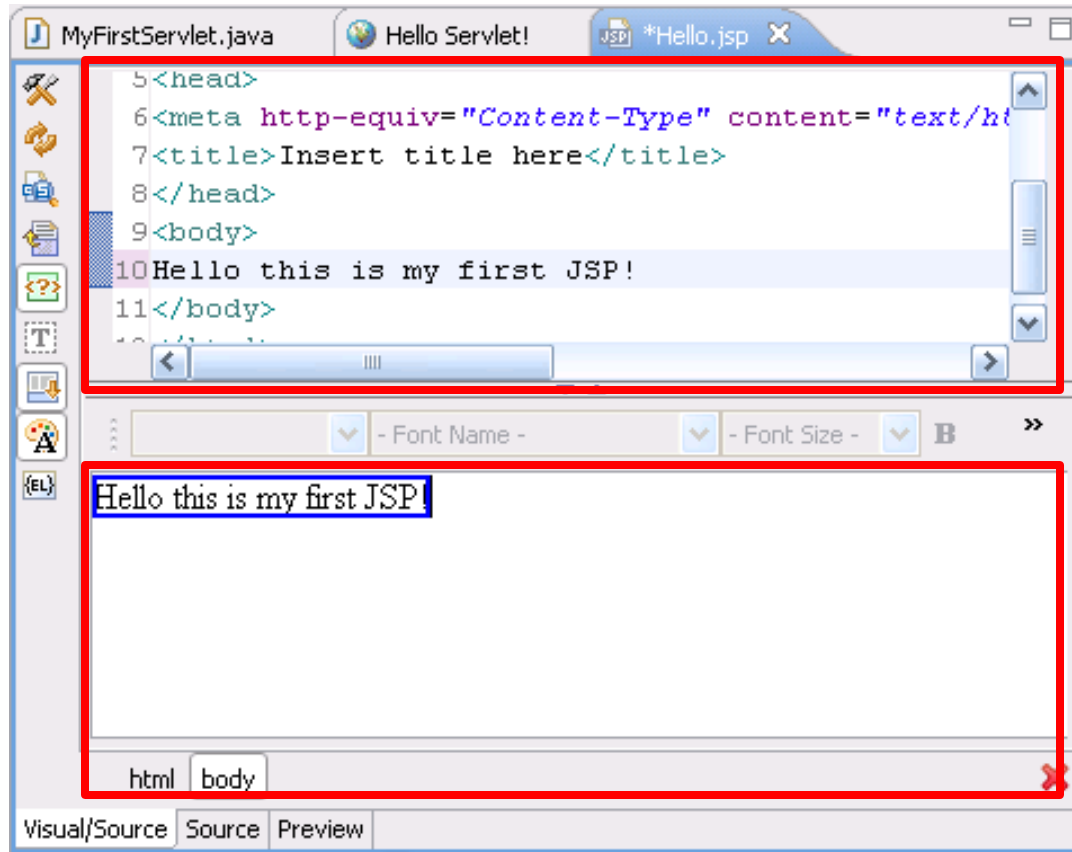
JSP erzeugen (3)

- Eclipse füllt die neue JSP automatisch mit Code für ein Grundgerüst
- Der JSP Editor besteht aus einer Codeansicht und einer *WYS/WYG** Ansicht
- Vorschau der Seite möglich

* What You See Is What You Get (grafische Ansicht)



Der JSP Editor



Quellcode

WYSIWYG

Vorschau der Seite



Hello.jsp

- Titel und Text ergänzen

```
MyFirstServlet.java  Hello Servlet!  JSP Hello.jsp X
1 <%@ page language="java" contentType="text/html; ci
2     pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Trans:
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html.
7 <title>Hello JSP!</title>
8 </head>
9 <body>
10 Hello this is my first JSP!
11 </body>
12
13 </html>
```



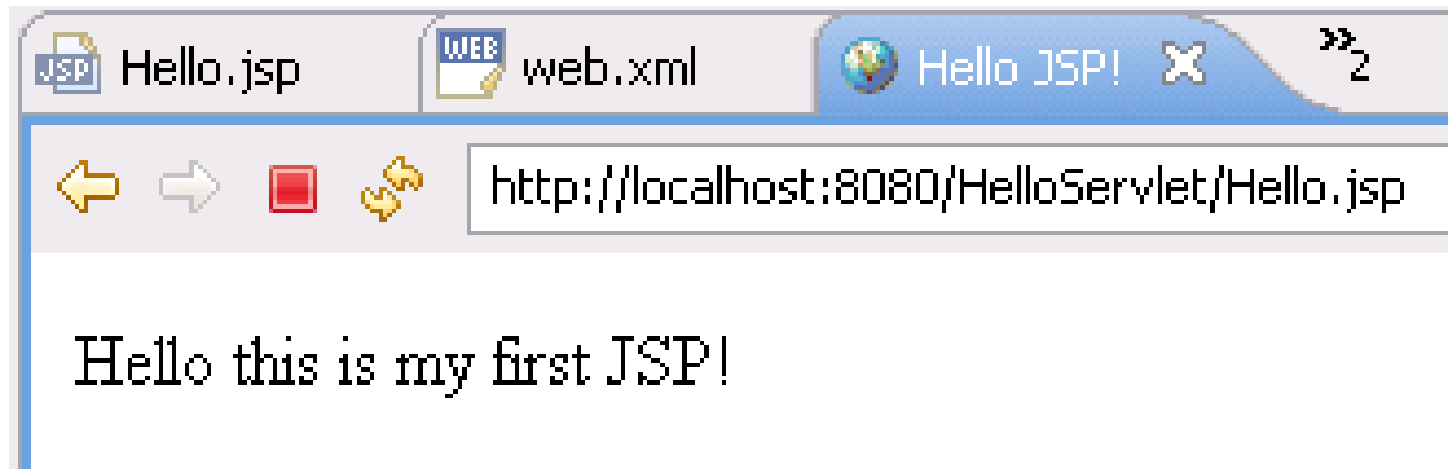
JSP aufrufen

- Da die JSP im Verzeichnis WebContent liegt, wird sie beim nächsten Packen und Deployen automatisch hochgeladen!
- Hierzu: Rechtsklick auf das Projektverzeichnis → Run as → Run on Server



JSP aufrufen (2)

- JSP ist direkt per URL aufrufbar



Servlet vs JSP

- JSP werden serverseitig wieder in Servlets umgewandelt! Wozu also?
- Bessere Trennung von Aufgaben
 - Webentwickler kennen HTML, Java dagegen nicht unbedingt!
 - Einsatz von JSP für die Präsentationsschicht vereinfacht alles

Servlet als Schnittstelle

- Wenn alle Requests an ein Servlet gehen, wie werden dann JSP angezeigt?
- → Servlet kann auf eine JSP „umleiten“



Zu JSP umleiten

- Anpassen der Servletmethode „handleRequest“:

```
protected void handleRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    RequestDispatcher dispatcher = this.getServletContext()
        .getRequestDispatcher( "/Hello.jsp" );
    dispatcher.forward( request, response );

} //handleRequest
```

Request- und Response Objekt werden ebenfalls durchgereicht




JSTL installieren

- JSTL auch für einfache Aufgaben notwendig, z.B. Auslesen und Anzeigen von Variablen
- Download:
 - <http://tomcat.apache.org/taglibs/standard/>

Standard Taglib

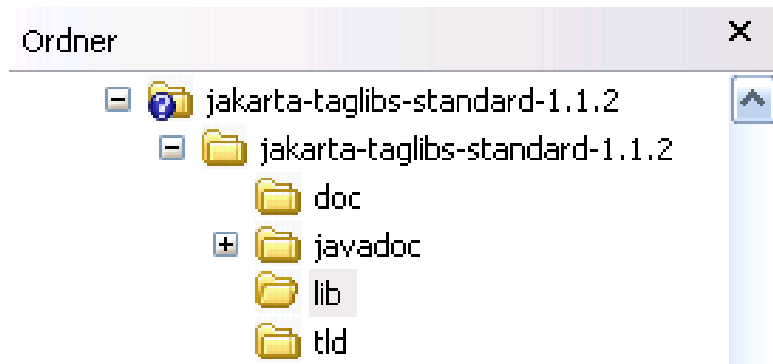
JSP(tm) Standard Tag Library implementations

Apache hosts the Apache Standard Taglib, an implementation of the [JSP Standard Tag Library \(JSTL\)](#) specification. Various versions are available.

Version	JSTL version	Requirements	Getting the Taglib
Standard 1.2	JSTL 1.2 (not yet JCP approved)	Servlet 2.5, JavaServer Pages 2.1	svn 
Standard 1.1	JSTL 1.1	Servlet 2.4, JavaServer Pages 2.0	download 
Standard 1.0	JSTL 1.0	Servlet 2.3, JavaServer Pages 1.2	download 

JSTL installieren (2)

- Taglib entpacken
- Im Unterverzeichnis „lib“ befinden sich die Dateien jstl.jar sowie standard.jar
- Diese müssen ins „lib“-Verzeichnis von JBoss kopiert werden



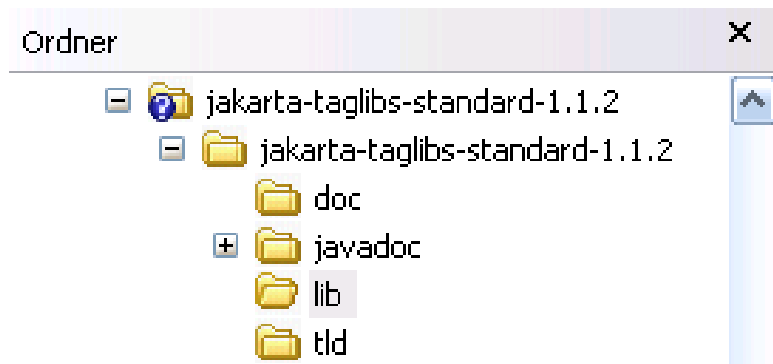
jstl.jar



standard.jar

JSTL installieren (3)

- Taglib entpacken
- Im Unterverzeichnis „lib“ befinden sich die Dateien jstl.jar sowie standard.jar
- Diese müssen ins „lib“-Verzeichnis der JBoss-Serverkonfiguration kopiert werden



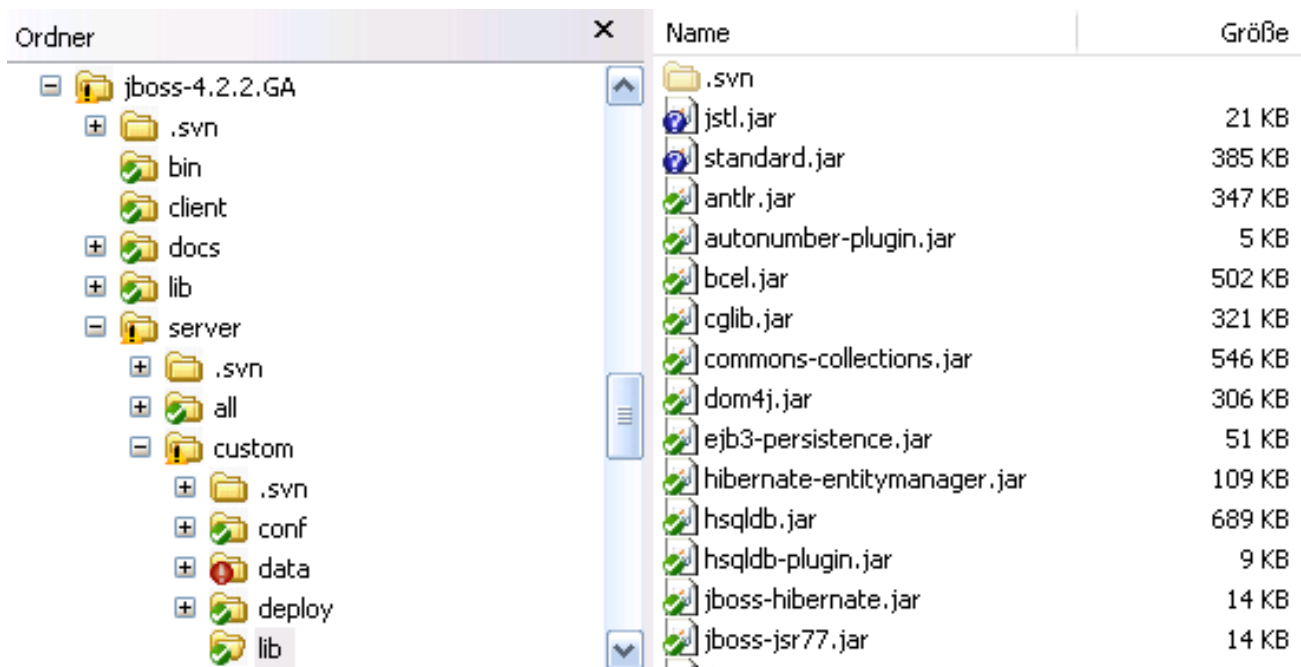
jstl.jar



standard.jar

JSTL installieren (4)

- Hier:
 - \jboss-4.2.2.GA\server\custom\lib



Ordner	Name	Größe
jboss-4.2.2.GA	.svn	
+ .svn	jstl.jar	21 KB
+ bin	standard.jar	385 KB
+ client	antlr.jar	347 KB
+ docs	autonumber-plugin.jar	5 KB
+ lib	bcel.jar	502 KB
- server	cglib.jar	321 KB
+ .svn	commons-collections.jar	546 KB
+ all	dom4j.jar	306 KB
- custom	ejb3-persistence.jar	51 KB
+ .svn	hibernate-entitymanager.jar	109 KB
+ conf	hsqldb.jar	689 KB
+ data	hsqldb-plugin.jar	9 KB
+ deploy	jboss-hibernate.jar	14 KB
+ lib	jboss-jsr77.jar	14 KB

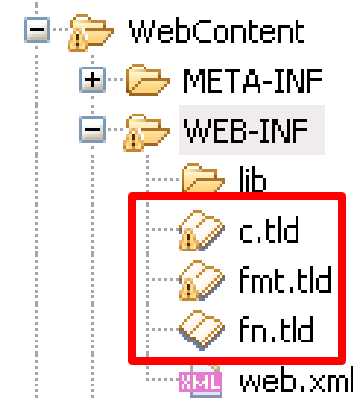
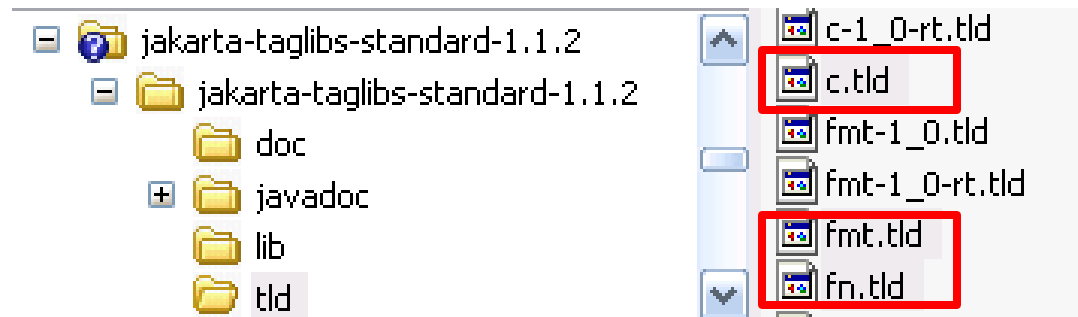
JSTL installieren (5)

- TLD Dateien kopieren (Taglib Deskriptoren)
- Aus dem Unterverzeichnis „tld“ der entpackten JSTL
- Ins Projektverzeichnis in den Unterordner „WebContent\WEB-INF“

JSTL installieren (5)

- TLD Dateien kopieren (Taglib Deskriptoren)
- Aus dem Unterverzeichnis „tld“ der entpackten JSTL
 - Dateien „c.tld“, „fmt.tld“ und „fn.tld“
- Ins Projektverzeichnis in den Unterordner „WebContent\WEB-INF“ kopieren

JSTL installieren (6)





Taglibs einbinden

- Werden ganz zu Beginn der JSP eingebunden
- Hello.jsp anpassen:

```
1<%-- Core with EL --%>
2<%@ taglib uri="/WEB-INF/c.tld" prefix="c" %>
3<%-- I18N Formatting with EL --%>
4<%@ taglib uri="/WEB-INF/fmt.tld" prefix="fmt" %>
5<%-- Functions --%>
6<%@ taglib uri="/WEB-INF/fn.tld" prefix="fn" %>
7
```

Parameter abrufen

- Alle vom Browser mitgeschickten Parameter befinden sich im Request-Objekt
- Zugriff im Servlet:
 - `Request.getParameter(„Parametername“)`

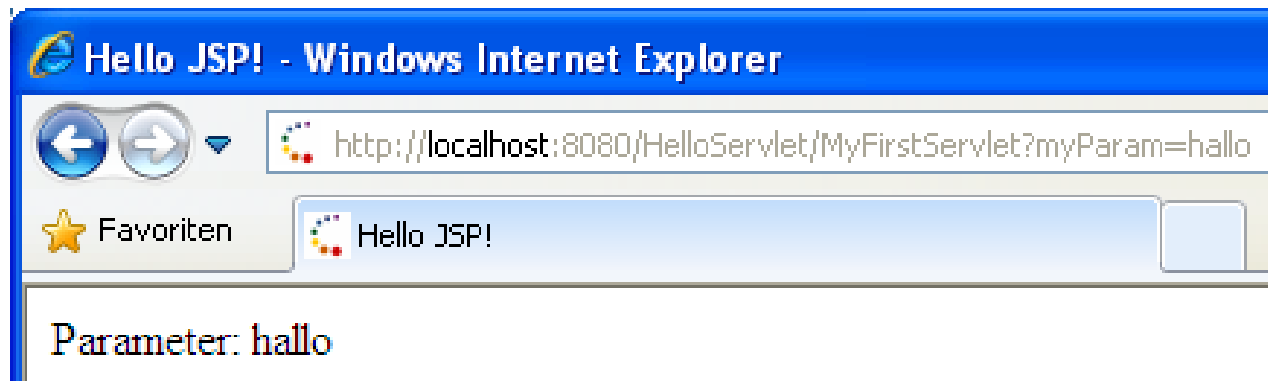
```
//Parameter einlesen
String param = request.getParameter( "myParam" );

System.out.println( "Parameter empfangen: " + param );
```

Parameter abrufen (2)

- In der JSP:
 - Es existiert automatisch eine Map namens „param“, die alle Parameter enthält!

```
<%-- Empfangenen Parameter ausgeben --%>  
Parameter: <c:out value="${param.myParam}" /><br />
```





JSTL

JavaServer Pages Standard Tag Library

JSTL

- Umfangreiche Funktionsbibliothek
- Ermöglicht Kontrollstrukturen innerhalb von JSP
- Eigene Tags können implementiert werden
- Ermöglicht erweiterte Funktionalität innerhalb einer JSP – *ohne* eingebetteten Java Code!

Ein paar Funktionen...

- Aus der Core Bibliothek:
 - out → Text ausgeben
 - set → Variable in Gültigkeitsbereich setzen
 - ...
 - Kontrollstrukturen: if, when, otherwise, ...
- Mehr unter:
 - <http://www.jsptutorial.org/content/jstl>

Sichtbarkeitsbereiche

- Beim Zugriff auf Variablen:
 - `${requestScope.var}`

Name der Variablen

„Scope“ → Sichtbarkeitsbereich

- pageScope (innerhalb der Bearbeitung der einen Seite)
- requestScope (innerhalb aller Seiten während einer Request-Bearbeitung)
- sessionScope (innerhalb aller Seiten, für einen Benutzer)
- applicationScope (innerhalb aller Seiten, für alle Benutzer)



Session Management

Session

- Eine Session beginnt, sobald ein Benutzer eine Webseite betritt
- Die Session endet, wenn der Benutzer eine Weile inaktiv ist (Timeout)
- Entwickler muss sich nicht um Cookies oder Session IDs kümmern

Ein Attribut in der Session speichern

- Z.B. im Servlet:

```
//Dieses Attribut bleibt die gesamte Sitzung über erhalten  
request.getSession().setAttribute( "userName", "Randall Flagg" );
```

- Oder in einer JSP:

```
<c:set var="userName" scope="session" value="Randall Flagg" />
```