



Distributed Computing

Lesson 13: JSPs and Beans

Thomas Weise · 汤卫思

tweise@hfu.edu.cn · <http://www.it-weise.de>

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

- 1 Introduction
- 2 JSP
- 3 Examples
- 4 Beans
- 5 Advanced Topics
- 6 Summary
- 7 Installing GlassFish



website

- What are JavaServer Pages (JSPs)?
- How can we create dynamic web sites with user interactions by using JavaServer Pages?
- How do JavaServer Pages combine with what we already know?
- What are Java Beans and how do they combine with JSPs?

- We want to create dynamic web sites

- We want to create dynamic web sites
- Actually, we can do that already

- We want to create dynamic web sites
- Actually, we can do that already:
 - We can implement HTTP over TCP with sockets, deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms

- We want to create dynamic web sites
- Actually, we can do that already:
 - We can implement HTTP over TCP with sockets, deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms
 - We can use Java Servlets ^[1–4], deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms

- We want to create dynamic web sites
- Actually, we can do that already:
 - We can implement HTTP over TCP with sockets, deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms
 - We can use Java Servlets ^[1–4], deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms
- Both is possible, but cumbersome and awful to maintain

- We want to create dynamic web sites
- Actually, we can do that already:
 - We can implement HTTP over TCP with sockets, deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms
 - We can use Java Servlets ^[1–4], deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms
- Both is possible, but cumbersome and awful to maintain
- Templates like PHP ^[5] where program code is mixed with HTML directives offer an alternative

- We want to create dynamic web sites
- Actually, we can do that already:
 - We can implement HTTP over TCP with sockets, deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms
 - We can use Java Servlets ^[1-4], deal with requests and responses with parameters in our code, and “programmatically” create HTML documents with forms
- Both is possible, but cumbersome and awful to maintain
- Templates like PHP ^[5] where program code is mixed with HTML directives offer an alternative
- JavaServer Pages offer the same functionality by extending our learned technology stack
(TCP socket ← HTTP Servlet (with Thread Pool) ← JavaServer Pages)

- JavaServer Pages (JSP) ^[6–10] are a template-based text-content generating technology

- JavaServer Pages (JSP) ^[6–10] are a template-based text-content generating technology
- Idea: Define a template file (usually `.jsp`) which contains some static (HTML) text and some Java code

- JavaServer Pages (JSP) ^[6–10] are a template-based text-content generating technology
- Idea: Define a template file (usually `.jsp`) which contains some static (HTML) text and some Java code
- This file is then translated to a Java Servlet

- JavaServer Pages (JSP) ^[6–10] are a template-based text-content generating technology
- Idea: Define a template file (usually `.jsp`) which contains some static (HTML) text and some Java code
- This file is then translated to a Java Servlet
- The servlet contains the static text as `String`s which are written to the servlet response object

- JavaServer Pages (JSP) ^[6–10] are a template-based text-content generating technology
- Idea: Define a template file (usually `.jsp`) which contains some static (HTML) text and some Java code
- This file is then translated to a Java Servlet
- The servlet contains the static text as `String`s which are written to the servlet response object
- The Java code is part of the servlet and may create additional output

- JavaServer Pages (JSP) ^[6–10] are a template-based text-content generating technology
- Idea: Define a template file (usually `.jsp`) which contains some static (HTML) text and some Java code
- This file is then translated to a Java Servlet
- The servlet contains the static text as `String`s which are written to the servlet response object
- The Java code is part of the servlet and may create additional output
- The servlet is compiled when the corresponding website is accessed the first time

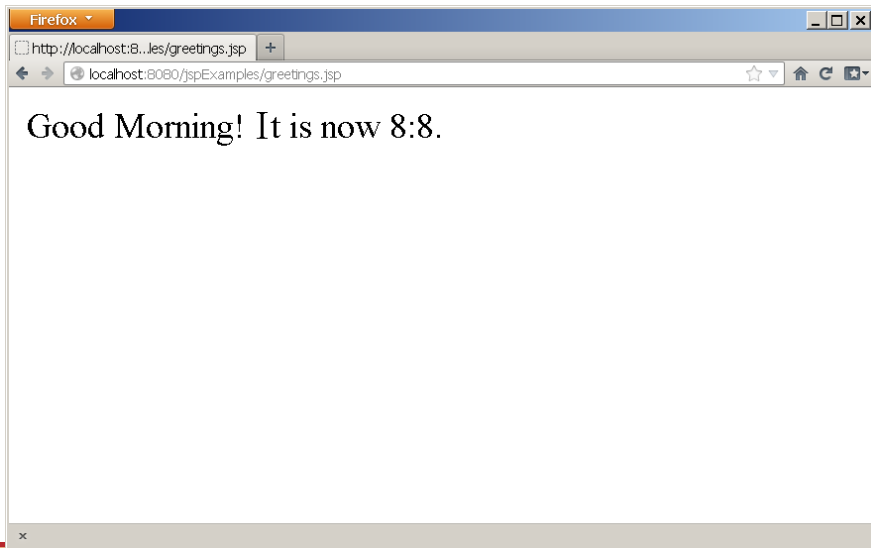
- JavaServer Pages (JSP) ^[6–10] are a template-based text-content generating technology
- Idea: Define a template file (usually `.jsp`) which contains some static (HTML) text and some Java code
- This file is then translated to a Java Servlet
- The servlet contains the static text as `String`s which are written to the servlet response object
- The Java code is part of the servlet and may create additional output
- The servlet is compiled when the corresponding website is accessed the first time
- Result: Elegant way to mix HTML and Java to build dynamic websites

Listing: [greetings.jsp]: A simple JSP greeting you.

```
<html>
  <head>
    <% java.util.Date clock = new java.util.Date(); %>
    <title>Hello! (<%= clock %>)</title>
  <body>

    <% if (clock.getHours() < 15) { %>
      Good Morning!
    <% } else { %>
      Good Evening!
    <% } %>

    It is now <%= clock.getHours() %> o'clock and <%=
      clock.getMinutes() %> minutes.
  </body>
</html>
```



Listing: [greetings.jsp.java]: The Java source code generated for greetings.jsp.

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class greetings_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

    private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();

    private static java.util.List<String> _jspx_dependants;

    private org.glassfish.jsp.api.ResourceInjector _jspx_resourceInjector;

    public java.util.List<String> getDependents() {
        return _jspx_dependants;
    }

    public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        JspWriter _jspx_out = null;
        PageContext _jspx_page_context = null;

        try {
            response.setContentType("text/html");
            response.setCharacterEncoding("UTF-8");
            pageContext = _jspxFactory.getPageContext(this, request, response,
                null, true, 8020, true);
            _jspx_page_context = pageContext;
            application = pageContext.getServletContext();
            config = pageContext.getServletConfig();
            session = pageContext.getSession();
            out = pageContext.getWriter();
            _jspx_out = out;
            _jspx_resourceInjector = (org.glassfish.jsp.api.ResourceInjector)
                application.getAttribute("com.sun.appserv.jsp.resource.injector");

            out.write("<html>");
            out.write("<head>");
            out.write("<title>");
            java.util.Date clock = new java.util.Date();
            out.write("</title>");
            out.write("<meta charset='utf-8'>");
            out.print(" ");
            out.write("</title>");
            out.write("</head>");
            out.write("<body>");
            out.write("<h1>");
            if (clock.getHour() < 12) {
                out.write("<h1>");
                out.write("<h2>Good Morning");
            }
            out.write("</h1>");
            out.write("<h2>");
            out.write("<h3>");
            out.write("<h4>");
            out.print(" ");
            out.write("</h4>");
            out.print(" ");
            out.write("</h4>");
            out.write("</body>");
            out.write("</html>");
        } catch (Throwable t) {
            if ((t instanceof ServletException) &&
                out != _jspx_out) {
                if (out != null && out.getBufferSize() != 0)
                    out.clearBuffer();
                if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
                else throw new ServletException(t);
            }
            finally {
                _jspxFactory.releasePageContext(_jspx_page_context);
            }
        }
    }
}
```

- In a JavaServer Page, we can access a set of implicit objects

- In a JavaServer Page, we can access a set of implicit objects:

request		the	HttpServletRequest	object of the request
---------	--	-----	--------------------	-----------------------

- In a JavaServer Page, we can access a set of implicit objects:

request		the <code>HttpServletRequest</code> object of the request
response		the <code>HttpServletResponse</code> object

- In a JavaServer Page, we can access a set of implicit objects:

request	the <code>HttpServletRequest</code> object of the request
response	the <code>HttpServletResponse</code> object
out	a <code>PrintWriter</code> object for sending output to the client

- In a JavaServer Page, we can access a set of implicit objects:

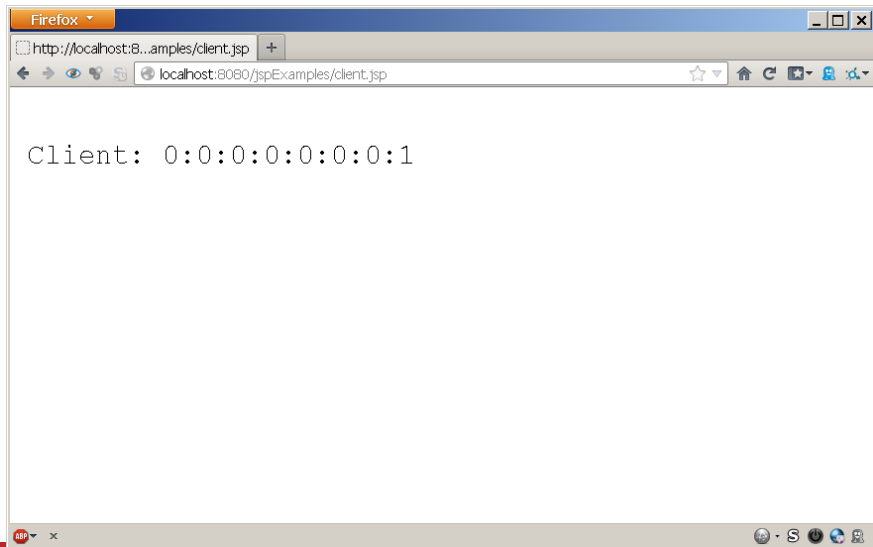
request	the <code>HttpServletRequest</code> object of the request
response	the <code>HttpServletResponse</code> object
out	a <code>PrintWriter</code> object for sending output to the client
session	the <code>HttpSession</code> object of the session

- In a JavaServer Page, we can access a set of implicit objects:

<code>request</code>	the <code>HttpServletRequest</code> object of the request
<code>response</code>	the <code>HttpServletResponse</code> object
<code>out</code>	a <code>PrintWriter</code> object for sending output to the client
<code>session</code>	the <code>HttpSession</code> object of the session
<code>application</code>	<code>ServletContext</code> object allowing you to store/read application-global data

Listing: [printClientAddress.jsp]: A very simple JSP with implicit directives.

```
<% response.setContentType("text/plain"); %>  
Client: <%= request.getRemoteAddr() %>
```



Listing: [printClientAddress.jsp.java]: The Java source code generated for printClientAddress.jsp.

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class printClientAddress_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

    private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();

    private static java.util.List<String> _jspx_dependants;

    private org.glassfish.jsp.api.ResourceInjector _jspx_resourceInjector;

    public java.util.List<String> getDependants() {
        return _jspx_dependants;
    }

    public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        JspWriter _jspx_out = null;
        PageContext _jspx_page_context = null;

        try {
            response.setContentType("text/html");
            response.setHeader("X-Powered-By", "JSP/2.3");
            pageContext = _jspxFactory.getPageContext(this, request, response,
                null, true, 8192, true);
            _jspx_page_context = pageContext;
            application = pageContext.getServletContext();
            config = pageContext.getServletConfig();
            session = pageContext.getSession();
            out = pageContext.getOut();
            _jspx_out = out;
            _jspx_resourceInjector = (org.glassfish.jsp.api.ResourceInjector)
                application.getAttribute("com.sun.appserv.jsp.resource.injector");

            response.setContentType("text/plain");
            out.write("\n");
            out.write("<Client:>");
            out.print( request.getRemoteAddr() );
        } catch (Throwable t) {
            if (!(t instanceof SkipPageException)){
                out = _jspx_out;
                if (out != null && out.getBufferSize() != 0)
                    out.clearBuffer();
                if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
                else throw new ServletException(t);
            }
        } finally {
            _jspxFactory.releasePageContext(_jspx_page_context);
        }
    }
}
```

- JavaServer Pages (JSP) introduce some special tags for embedding Java code and output directives into HTML/text ^[6–8, 11]

- JavaServer Pages (JSP) introduce some special tags for embedding Java code and output directives into HTML/text ^[6–8, 11]

Element	Syntax	Meaning
Expression	<code><%= expression %></code>	expression is evaluated and placed in output

- JavaServer Pages (JSP) introduce some special tags for embedding Java code and output directives into HTML/text ^[6–8, 11]

Element	Syntax	Meaning
Expression	<code><%= expression %></code>	expression is evaluated and placed in output
Scriptlet	<code><% code %></code>	code inserted in <code>service</code> method of compiled Servlet

- JavaServer Pages (JSP) introduce some special tags for embedding Java code and output directives into HTML/text ^[6–8, 11]

Element	Syntax	Meaning
Expression	<code><%= expression %></code>	expression is evaluated and placed in output
Scriptlet	<code><% code %></code>	code inserted in <code>service</code> method of compiled Servlet
Declaration	<code><%! code %></code>	code inserted in body of Servlet class, outside of service method

- JavaServer Pages (JSP) introduce some special tags for embedding Java code and output directives into HTML/text ^[6–8, 11]

Element	Syntax	Meaning
Expression	<code><%= expression %></code>	expression is evaluated and placed in output
Scriptlet	<code><% code %></code>	code inserted in <code>service</code> method of compiled Servlet
Declaration	<code><%! code %></code>	code inserted in body of Servlet class, outside of service method
Directive	<code><%@ page att="val" %></code>	directives to the servlet about general setup e.g., <code>import="package.class", contentType="MIMEtype", session="true false", extends="package.class"</code>

- JavaServer Pages (JSP) introduce some special tags for embedding Java code and output directives into HTML/text ^[6–8, 11]

Element	Syntax	Meaning
Expression	<code><%= expression %></code>	expression is evaluated and placed in output
Scriptlet	<code><% code %></code>	code inserted in <code>service</code> method of compiled Servlet
Declaration	<code><%! code %></code>	code inserted in body of Servlet class, outside of service method
Directive	<code><%@ page att="val" %></code>	directives to the servlet about general setup e.g., <code>import="package.class"</code> , <code>contentType="MIMEtype"</code> , <code>session="true false"</code> , <code>extends="package.class"</code>
Directive	<code><%@ include file="url" %></code>	file on local system to include when page translated to Servlet

- JavaServer Pages (JSP) introduce some special tags for embedding Java code and output directives into HTML/text ^[6–8, 11]

Element	Syntax	Meaning
Expression	<code><%= expression %></code>	expression is evaluated and placed in output
Scriptlet	<code><% code %></code>	code inserted in <code>service</code> method of compiled Servlet
Declaration	<code><%! code %></code>	code inserted in body of Servlet class, outside of service method
Directive	<code><%@ page att="val" %></code>	directives to the servlet about general setup e.g., <code>import="package.class"</code> , <code>contentType="MIMEtype"</code> , <code>session="true false"</code> , <code>extends="package.class"</code>
Directive	<code><%@ include file="url" %></code>	file on local system to include when page translated to Servlet
Comment	<code><%-- comment --%></code>	comment – ignored when JSP page is translated to Servlet

- Create a JSP which counts its visitors

- Create a JSP which counts its visitors
- Accessed via
(<http://localhost:8080/myJSPs/counterUnsynchronized.jsp>)

Listing: [counterUnsynchronized.jsp]: A JSP counting visitors.

```
<html>
  <head>
    <title>Visitor Counter</title>
  </head>

  <body>
    <%! long numVisitors = 0; %>

    <p>Hello!
      You are the <%= ++numVisitors%><sup>th</sup> visitor!</p>

    <p>
      <% for(int i = 0; i<= 10; i++) { %>
        <p>Modifying a member variable in an
          unsynchronized fashion is dangerous.</p>
      <% } %>
    </p>
  </body>
</html>
```

Listing: [counterUnsynchronized.jsp.java]: The Java source code generated for counterUnsynchronized.jsp.

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class counterUnsynchronized_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

    long numVisitors = 0;
    private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();

    private static java.util.List<String> _jspx_dependants;

    private org.glassfish.jsp.api.ResourceInjector _jspx_resourceInjector;

    public java.util.List<String> getDependents() {
        return _jspx_dependants;
    }

    public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        JspWriter _jspx_out = null;
        PageContext _jspx_page_context = null;

        try {
            response.setContentType("text/html");
            response.setHeader("Cache-Control", "no-cache");
            pageContext = _jspxFactory.getPageContext(this, request, response,
                null, true, 8192, true);
            _jspx_page_context = pageContext;
            application = pageContext.getServletContext();
            config = pageContext.getServletConfig();
            session = pageContext.getSession();
            out = pageContext.getWriter();
            _jspx_out = out;
            _jspx_resourceInjector = (org.glassfish.jsp.api.ResourceInjector)
                application.getResource("com.sun.jsp.jsp_resource.injector");

            out.write("<html>\r\n");
            out.write("<...<head>\r\n");
            out.write("<...<title>Visitor_Counters</title>\r\n");
            out.write("<...</head>\r\n");
            out.write("<...>\r\n");
            out.write("<...<body>\r\n");
            out.write("<...");
            out.write("<...");
            out.write("<...");
            out.write("<...<h1>\r\n");
            out.write("<...</h1>\r\n");
            out.write("<...<p>Hello\r\n");
            out.write("<...<script>function show_time()");
            out.print(" { numVisitors);");
            out.write("<script>{if page> Visitor</p>\r\n");
            out.write("<...</h1>\r\n");
            out.write("<...<p>\r\n");
            out.write("<...");
            for(int i = 0; i<= 10; i++) {
                out.write("<...");
                out.write("<...<script>{if modifying_a_member_variable, is, is\r\n");
                out.write("<...<script>{if unsynchronized, fashion, is, dangerous. </p>\r\n");
                out.write("<...");
            }
            out.write("<...");
            out.write("<...<p>\r\n");
            out.write("<...</body>\r\n");
            out.write("<...</html>\r\n");
        } catch (Throwable t) {
            if (!t instanceof SkipPageException){
                out = _jspx_out;
                if (out != null && out.getBufferSize() != 0)
                    out.clearBuffer();
                if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
                else throw new ServletException(t);
            }
        } finally {
            _jspxFactory.releasePageContext(_jspx_page_context);
        }
    }
}
```


- Many users may access your website at the same time

- Many users may access your website at the same time
- Servlet containers use thread pools to process requests in *parallel*

- Many users may access your website at the same time
- Servlet containers use thread pools to process requests in *parallel*
- Even simple operations like `++numVisitors` are usually not *atomic*

- Many users may access your website at the same time
- Servlet containers use thread pools to process requests in *parallel*
- Even simple operations like `++numVisitors` are usually not *atomic*
- Situations like *lost update* may occur, or data structures might be destroyed entirely

- Many users may access your website at the same time
- Servlet containers use thread pools to process requests in *parallel*
- Even simple operations like `++numVisitors` are usually not *atomic*
- Situations like *lost update* may occur, or data structures might be destroyed entirely
- These situations will happen rarely.

- Many users may access your website at the same time
- Servlet containers use thread pools to process requests in *parallel*
- Even simple operations like `++numVisitors` are usually not *atomic*
- Situations like *lost update* may occur, or data structures might be destroyed entirely
- These situations will happen rarely.
- They might not be reproducible in tests and never be encountered when debugging.

- Many users may access your website at the same time
- Servlet containers use thread pools to process requests in *parallel*
- Even simple operations like `++numVisitors` are usually not *atomic*
- Situations like *lost update* may occur, or data structures might be destroyed entirely
- These situations will happen rarely.
- They might not be reproducible in tests and never be encountered when debugging.
- *Synchronization* and protection of *critical sections* (see lesson 15) are thus very, very important!

Listing: [counterSynchronized.jsp]: A JSP counting visitors properly.

```
<html>
  <head>
    <title>Visitor Counter</title>
  </head>

  <body>
    <%= private long numVisitors = 0;
      private synchronized final long __inc() {
        return ++numVisitors;
      } %>

    <p>Hello!
      You are the <%= __inc()%><sup>th</sup> visitor!</p>

    <p>Now we are using a synchronized method.</p>
  </body>
</html>
```


Listing: [counterSynchronized.jsp.java]: The Java source code generated for counterSynchronized.jsp.

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class counterSynchronized_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

    private long numVisitors = 0;
    private synchronized final long _inc() {
        return ++numVisitors;
    }

    private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();
    private static java.util.List<String> _jspx_dependants;
    private org.glassfish.jsp.api.ResourceInjector _jspx_resourceInjector;

    public java.util.List<String> getDependants() {
        return _jspx_dependants;
    }

    public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        JspWriter _jspx_out = null;
        PageContext _jspx_page_context = null;

        try {
            response.setContentType("text/html");
            response.setHeader("X-Powered-By", "JSP/2.3");
            pageContext = _jspxFactory.getPageContext(this, request, response,
                null, true, 8192, true);
            _jspx_page_context = pageContext;
            application = pageContext.getServletContext();
            config = pageContext.getServletConfig();
            session = pageContext.getSession();
            out = pageContext.getOut();
            _jspx_out = out;
            _jspx_resourceInjector = (org.glassfish.jsp.api.ResourceInjector)
                application.getAttribute("com.sun.appserv.jsp.resource.injector");

            out.write("<html>\r\n");
            out.write("<...head>\r\n");
            out.write("<...title>Visitor_Counter</title>\r\n");
            out.write("<.../head>\r\n");
            out.write("<...r\n");
            out.write("<...tbody>\r\n");
            out.write("<r\n");
            out.write("<...r\n");
            out.write("<...p>Hello \r\n");
            out.write("<...%>You are, the");
            out.print(" _inc());
            out.write("<emp>th</emp> visitor</p>\r\n");
            out.write("<...r\n");
            out.write("<...p>We use, using a_synchronized_method.</p>\r\n");
            out.write("<.../body>\r\n");
            out.write("</html>");
        } catch (Throwable t) {
            if (!(t instanceof SkipPageException)){
                out = _jspx_out;
                if (out != null && out.getBufferSize() != 0)
                    out.clearBuffer();
                if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
                else throw new ServletException(t);
            }
        } finally {
            _jspxFactory.releasePageContext(_jspx_page_context);
        }
    }
}
```

- JSPs are usually deployed in `.war` (Web application ARchive) files formats (see last lecture)

- JSPs are usually deployed in `.war` (Web application ARchive) files formats (see last lecture)
- a `.war` is a ZIP archive with a special file structure

- JSPs are usually deployed in `.war` (Web application ARchive) files formats (see last lecture)
- a `.war` is a ZIP archive with a special file structure:
 - it contains a folder `WEB-INF`

- JSPs are usually deployed in `.war` (Web application ARchive) files formats (see last lecture)
- a `.war` is a ZIP archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a file `web.xml`

- JSPs are usually deployed in `.war` (Web application ARchive) files formats (see last lecture)
- a `.war` is a ZIP archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a file `web.xml`
 - and a file `sun-web.xml`

- JSPs are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a ZIP archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a file `web.xml`
 - and a file `sun-web.xml`
- in `web.xml` and `sun-web.xml`, we specify the JSPs provided in the archive and how they can be accessed

- JSPs are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a ZIP archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a file `web.xml`
 - and a file `sun-web.xml`
- in `web.xml` and `sun-web.xml`, we specify the JSPs provided in the archive and how they can be accessed
- (we will later have a lesson just on `xml`, but the syntax here is straightforward)

- JSPs are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a ZIP archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a file `web.xml`
 - and a file `sun-web.xml`
- in `web.xml` and `sun-web.xml`, we specify the JSPs provided in the archive and how they can be accessed
- (we will later have a lesson just on `xml`, but the syntax here is straightforward)
- (on the slides “Create a WAR” and “Deploying a WAR”, we give a tutorial on how to package and deploy `.war` archives)

Listing: [web.xml]: The web.xml specification for our examples.

```
<?xml version="1.0" encoding="utf-8"?>
<web-app
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">

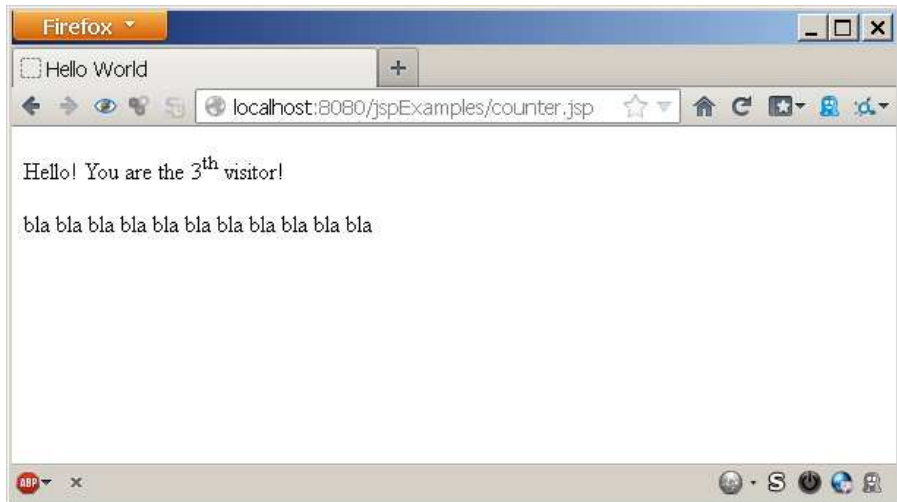
  <display-name>myJSPs</display-name>
  <distributable />

</web-app>
```

Listing: [sun-web.xml]: The sun-web.xml specification for our examples.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server 9.0_
        Servlet 2.5//EN"
        'http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd'>

<sun-web-app>
    <context-root>/myJSPs</context-root>
</sun-web-app>
```



- Create a JSP with a form that allows you to rate this course

- Create a JSP with a form that allows you to rate this course
- (uses the same `web.xml` and `sun-web.xml`)

- Create a JSP with a form that allows you to rate this course
- (uses the same `web.xml` and `sun-web.xml`)
- Accessed via (`http://localhost:8080/myJSPs/rating.jsp`)

Listing: [rating.jsp]: A JSP allowing you to rate this course.

```
<html>
<head>
<title>Rate this Course</title>
</head>

<% int ratingsCount = 0;
int[] ratings = new int[6]; %>

<body>
<h1>Course Evaluation</h1>
<h2>Rate this Course</h2>
<form method="get">
<p>Give a rating how you like this course:</p>
<table border="1">
<tr>
<td bgcolor="#ff0000">rubbish, makes no sense<input type="radio" name="rating"
value="1" /></td>
<td bgcolor="#ff8080">pretty boring <input type="radio" name="rating"
value="2" /></td>
<td bgcolor="#ffff00">so-so <input type="radio" name="rating"
value="3" /></td>
<td bgcolor="#808080">not bad <input type="radio" name="rating"
value="4" /></td>
<td bgcolor="#00ff00">OK, this is great <input type="radio" name="rating"
value="5" /></td>
<td bgcolor="#008080">Honestly, I don't care <input type="radio" name="rating"
value="6" /></td>
</tr>
<tr>
<td colspan="6">
<input type="submit" value="submit_my_rating" />
</td></tr>
</table>
</form>

<% String rating = request.getParameter("rating"); //get the "GET" parameter, i.e., the
form result
int ratingInt = -1;
if (rating != null) { //if the page is not loaded because of form submit, rating will be
null
try { //only if submit was clicked, we get here
ratingInt = Integer.parseInt(rating); //check if rating is int (as it should be)
} catch (Exception e) {
ratingInt = -1;
}

if ((ratingInt >= 1) && (ratingInt <= ratings.length)) { //a valid rating was cast
synchronized(this) { //synchronized update and read of member variables
ratings[ratingInt - 1]++;
ratingsCount++;
}

<h2>Ratings</h2>
<p>So far <% = ratingsCount%> valid ratings have been issued.</p>
<table border="1">
<tr>
<td bgcolor="#ff0000">rubbish, makes no sense</td>
<td bgcolor="#ff8080">pretty boring</td>
<td bgcolor="#ffff00">so-so</td>
<td bgcolor="#808080">not bad</td>
<td bgcolor="#00ff00">OK, this is great</td>
<td bgcolor="#008080">Honestly, I don't care</td>
</tr>
<tr>
<% for(int i = 0; i < ratings.length; i++) { %>
<td><% = ratings[i]></td>
<% } // for %>
</tr>
</table>

<% } } // if %>
</body>
</html>
```


Firefox

Voter!

localhost:8080/jspExamples/voter.jsp?rb1=5

Course Evaluation

This website has been loaded 10 times.

What's your opinion?

If vote that this course is:

rubbish, makes no sense	pretty boring	so-so	not bad	OMG, this is great	Honestly, I don't care
-------------------------	---------------	-------	---------	--------------------	------------------------

submit my vote

Vote Results

So far 7 valid votes have been issued.

rubbish, makes no sense	pretty boring	so-so	not bad	OMG, this is great	Honestly, I don't care
3	0	0	1	2	1

- Goal: Separate application logic and presentation

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`, i.e., which can be stored to a stream

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`, i.e., which can be stored to a stream
 - which can have *properties*

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`, i.e., which can be stored to a stream
 - which can have *properties*
- Property

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`, i.e., which can be stored to a stream
 - which can have *properties*
- Property:
 - “emulated” variable of type `Type` (e.g., `String`) with the name `myVariable`

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`, i.e., which can be stored to a stream
 - which can have *properties*
- Property:
 - “emulated” variable of type `Type` (e.g., `String`) with the name `myVariable`
 - Read access via method `Type getMyVariable()`

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`, i.e., which can be stored to a stream
 - which can have *properties*
- Property:
 - “emulated” variable of type `Type` (e.g., `String`) with the name `myVariable`
 - Read access via method `Type getMyVariable()`
 - Write access via method `void setMyVariable(Type value)`

- Goal: Separate application logic (into a class) and presentation (into the JavaServer Page)
- Java Bean ^[12, 13]: Simple Java class:
 - which have a 0-argument constructor
 - which implement `java.io.Serializable`, i.e., which can be stored to a stream
 - which can have *properties*
- Property:
 - “emulated” variable of type `Type` (e.g., `String`) with the name `myVariable`
 - Read access via method `Type getMyVariable()`
 - Write access via method `void setMyVariable(Type value)`
 - Does not be a real variable

- Beans are created and maintained by servlet container for different scopes

- Beans are created and maintained by servlet container for different scopes:
 - Application: One bean instance for all requests and servlets of the application

- Beans are created and maintained by servlet container for different scopes:
 - Application: One bean instance for all requests and servlets of the application
 - Session: One bean instance for each session

- Beans are created and maintained by servlet container for different scopes:
 - Application: One bean instance for all requests and servlets of the application
 - Session: One bean instance for each session
 - Request: One bean instance per request

- Special tags for accessing beans

- Special tags for accessing beans:
 - `jsp:useBean` -tag: access bean of a given class, a name, and a scope (bean is created if it does not exist yet)

- Special tags for accessing beans:
 - `jsp:useBean` -tag: access bean of a given class, a name, and a scope (bean is created if it does not exist yet)
 - `jsp:getProperty` -tag: get/print value of a given property of a given bean

- Special tags for accessing beans:
 - `jsp:useBean` -tag: access bean of a given class, a name, and a scope (bean is created if it does not exist yet)
 - `jsp:getProperty` -tag: get/print value of a given property of a given bean
 - `jsp:setProperty` -tag: set value of a property (`*` for all, use request parameters if no value is specified)

- Create a simple shopping cart via JavaServer Pages

- Create a simple shopping cart via JavaServer Pages
- (uses the same `web.xml` and `sun-web.xml`)

- Create a simple shopping cart via JavaServer Pages
- (uses the same `web.xml` and `sun-web.xml`)
- Access via `http://localhost:8080/myJSPs/shoppingCart.jsp`

Listing: [shoppingCart.jsp]: A JSP resembling a shopping cart.

```
<html>
<head><title>Shopping Cart</title></head>
<body>
<jsp:useBean id="cart" scope="session" class="shopping.Cart" />
<% String command = request.getParameter("submit");
    if("add".equals(command))    {
        cart.setAdd(request.getParameter("item"));    }
    else if("remove".equals(command)) {
        cart.setRemove(request.getParameter("item")); } %>

<p>Shopping Card:</p>
<ol>
    <% for (String element : cart.getItems()) { %>
        <li><%= element %></li>
    <% } %>
</ol>

<hr>
<%@ include file="shoppingCartForm.jsp" %>
</body>
</html>
```

Listing: [shoppingCartForm.jsp]: The add/remove item form of the cart.

```
<form method="post">
  Add or remove Items from the shopping cart:<br/>

  <select name="item">
    <option>iPad</option>
    <option>&#x9999;&#x8549;</option>
    <option>&#x897F;&#x7EA2;&#x67FF;</option>
    <option>&#x571F;&#x8C46;</option>
    <option>Jacket</option>
  </select>

  <input type=submit name="submit" value="add">
  <input type=submit name="submit" value="remove">
</form>
```



Listing: [Cart.java]: The shopping cart bean.

```
package shopping;

import java.io.Serializable; import java.util.ArrayList;

public class Cart implements Serializable {
    private ArrayList<String> contents; // the item list

    public Cart() {
        this.contents = new ArrayList<>();
    }

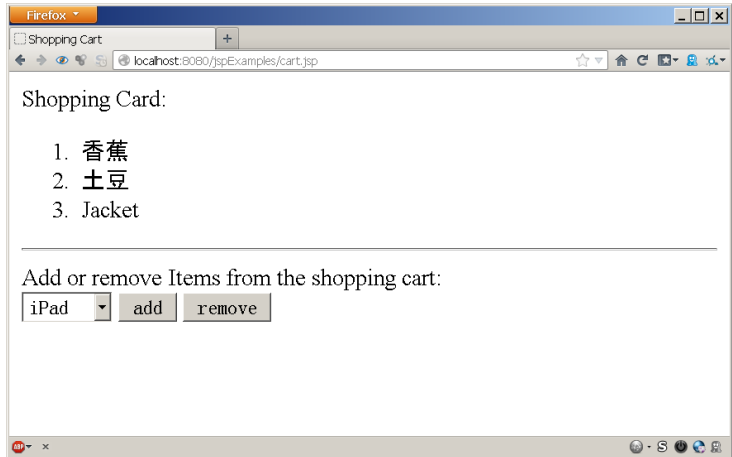
    /** write-only property "add" */
    public void setAdd(String name) {
        this.contents.add(name);
    }

    /** write-only property "remove" */
    public void setRemove(String name) {
        this.contents.remove(name);
    }

    /** create read-only property "items" */
    public String[] getItems() {
        return this.contents.toArray(new String[this.contents.size()]);
    }
}
```

Listing: [shoppingCart.jsp.java]: The Java source code generated for shoppingCart.jsp.

[illegible]



- Including Java code into JavaServer Pages is very powerful

- Including Java code into JavaServer Pages is very powerful . . . but also a bit dangerous

- Including Java code into JavaServer Pages is very powerful . . . but also a bit dangerous
 - source code of bigger pages quickly gets hard to read and understand

- Including Java code into JavaServer Pages is very powerful . . . but also a bit dangerous
 - source code of bigger pages quickly gets hard to read and understand
 - dilutes the separation of presentation layer and logic

- Including Java code into JavaServer Pages is very powerful . . . but also a bit dangerous
 - source code of bigger pages quickly gets hard to read and understand
 - dilutes the separation of presentation layer and logic
- **Idea:** Add another level of abstraction, replace Java code with simpler expressions (EL) and tags (JSTL)

- Including Java code into JavaServer Pages is very powerful . . . but also a bit dangerous
 - source code of bigger pages quickly gets hard to read and understand
 - dilutes the separation of presentation layer and logic
- **Idea:** Add another level of abstraction, replace Java code with simpler expressions (EL) and tags (JSTL)
- JSP Expression Language

- Including Java code into JavaServer Pages is very powerful ... but also a bit dangerous
 - source code of bigger pages quickly gets hard to read and understand
 - dilutes the separation of presentation layer and logic
- **Idea:** Add another level of abstraction, replace Java code with simpler expressions (EL) and tags (JSTL)
- JSP Expression Language: mainly designed for accessing page-level Beans

- Including Java code into JavaServer Pages is very powerful . . . but also a bit dangerous
 - source code of bigger pages quickly gets hard to read and understand
 - dilutes the separation of presentation layer and logic
- **Idea:** Add another level of abstraction, replace Java code with simpler expressions (EL) and tags (JSTL)
- JSP Expression Language: mainly designed for accessing page-level Beans, but also provides simple mathematical and Boolean expressions

- Including Java code into JavaServer Pages is very powerful ... but also a bit dangerous
 - source code of bigger pages quickly gets hard to read and understand
 - dilutes the separation of presentation layer and logic
- **Idea:** Add another level of abstraction, replace Java code with simpler expressions (EL) and tags (JSTL)
- JSP Expression Language: mainly designed for accessing page-level Beans, but also provides simple mathematical and Boolean expressions
- Expressions take the form $\${expression}$

Listing: [expressionLanguageExamples.jsp]: Some examples for the EL.

```
<h1>Examples for using the Expression Language (EL)</h1>

<ul>
<li>20 modulo 7 is written as <code>&#x24;{20 mod 7}</code> and evaluates to
    <span>${20 mod 7}</span>.</li>

<li>Is 20*10 different from 200? This question can be written as
    <code>&#x24;{((20*10) ne 200) ? "yes" : "no"}</code>.
    The answer is "<span>${((20*10) ne 200) ? "yes" : "no"}</span>".</li>

<li>The context path below which you can find the server pages of this example
    is
    accessed via <code>&#x24;{pageContext.request.contextPath}</code>, which
    evaluates to "<span>${pageContext.request.contextPath}</span>".</li>

<li>You have accessed this page from address
    "<span>${pageContext.request.remoteAddr}</span>",
    which I can obtain via
    <code>&#x24;{pageContext.request.remoteAddr}</code>.</li>

<li>The computer and port where this server is listening is
    "<span>${header.host}</span>",
    accessible via <code>&#x24;{header.host}</code>.</li>
</ul>
```


Listing: [expressionLanguageExamples.jsp.java]: The Java source code generated for expressionLanguageExamples.jsp.

```

import org.apache.spark.sql.*
import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

// Create a SparkSession
val spark = SparkSession.builder().appName("PageRank").getOrCreate()

// Load the graph data
val graph = GraphLoader.fromEdgeList(spark, "edges.csv")

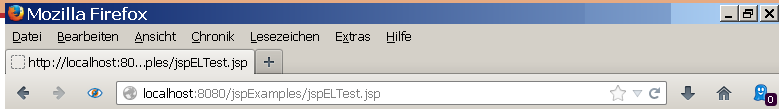
// Compute PageRank
val pagerank = PageRank.compute(graph, 0.1, 0.001, 100)

// Get the PageRank values for each node
val pagerankValues = pagerank.vertices.map { v, r, c, d, s, t, l, e, f, i, d, o, u, t, s, i, z, e } => {
  (v.getId, r)
}

// Save the PageRank values to a file
val pagerankFile = "pagerank.txt"
val pagerankWriter = new BufferedWriter(new FileWriter(pagerankFile))
pagerankValues.foreach { (id, rank) => {
  pagerankWriter.write(id + "\t" + rank + "\n")
}
}
pagerankWriter.close()

// Print the PageRank values
val pagerankList = pagerankValues.toList
pagerankList.foreach { (id, rank) => {
  println(s"Node $id: PageRank = $rank")
}
}

```



20 modulo 7 is 6

Is 20*10 different from 200? false

Is this class boring? false

/jspExamples

127.0.0.1

localhost:8080

jspExamples

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions, e.g., `pageContext`

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions, e.g., `pageContext`, `request`

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions, e.g., pageContext, request, header

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions, e.g., `pageContext`, `request`, `header`, `cookie`

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions, e.g., `pageContext`, `request`, `header`, `cookie`, `pageContext.Session`

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions, e.g., `pageContext`, `request`, `header`, `cookie`, `pageContext.Session`, `pageContext.Request`

+	addition	-	subtraction
*	multiplication	/, div	division
%, mod	modulo division	==, eq	check for equality
!=, ne	check for inequality	<, lt	less then?
>, gt	greater then?	<=, le	less or equal then?
>=, ge	greater or equal then?	&&, and	logical and
, or	logical or	!, not	logical ot
a?b:c	logical conditional	empty	test for null

- Also, many of the objects provided by the web container can be accessed in expressions, e.g., `pageContext`, `request`, `header`, `cookie`, `pageContext.Session`, `pageContext.Request`, `pageContext.Response`...

- Motivation: reduce amount of pure Java code in JSPs

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks
- JSTL offers five sets of standard tags to ease your work

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks
- JSTL offers five sets of standard tags to ease your work:

Name	Prefix	Description

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks
- JSTL offers five sets of standard tags to ease your work:

Name	Prefix	Description
core	c	URI: http://java.sun.com/jsp/jstl/core basic tags, include control flow primitives like loops and alternatives

|

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks
- JSTL offers five sets of standard tags to ease your work:

Name	Prefix	Description
core	c	URI: http://java.sun.com/jsp/jstl/core basic tags, include control flow primitives like loops and alternatives
fmt	fmt	URI: http://java.sun.com/jsp/jstl/fmt tags for formatting of date, currency, and number values, as well as for i18n

|

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks
- JSTL offers five sets of standard tags to ease your work:

Name	Prefix	Description
core	c	URI: http://java.sun.com/jsp/jstl/core basic tags, include control flow primitives like loops and alternatives
fmt	fmt	URI: http://java.sun.com/jsp/jstl/fmt tags for formatting of date, currency, and number values, as well as for i18n
xml	x	URI: http://java.sun.com/jsp/jstl/xml processing of XML data, offers loops and alternatives, but conditions defined as XPath

|

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks
- JSTL offers five sets of standard tags to ease your work:

Name	Prefix	Description
core	c	URI: http://java.sun.com/jsp/jstl/core basic tags, include control flow primitives like loops and alternatives
fmt	fmt	URI: http://java.sun.com/jsp/jstl/fmt tags for formatting of date, currency, and number values, as well as for i18n
xml	x	URI: http://java.sun.com/jsp/jstl/xml processing of XML data, offers loops and alternatives, but conditions defined as XPath
sql	sql	URI: http://java.sun.com/jsp/jstl/sql tags for accessing data bases from a JSP

- Motivation: reduce amount of pure Java code in JSPs
- Create “tags” (which are translated to Java code) for repetitively occurring tasks
- JSTL offers five sets of standard tags to ease your work:

Name	Prefix	Description
core	c	URI: http://java.sun.com/jsp/jstl/core basic tags, include control flow primitives like loops and alternatives
fmt	fmt	URI: http://java.sun.com/jsp/jstl/fmt tags for formatting of date, currency, and number values, as well as for i18n
xml	x	URI: http://java.sun.com/jsp/jstl/xml processing of XML data, offers loops and alternatives, but conditions defined as XPath
sql	sql	URI: http://java.sun.com/jsp/jstl/sql tags for accessing data bases from a JSP
functions	fn	URI: http://java.sun.com/jsp/jstl/functions mainly string manipulation functions

Listing: [jstlExamples.jsp]: Some examples for the JSTL.

```
<h1>Examples for using the JSP Standard Tag Library</h1>

<jsp:useBean id="ourDate" class="java.util.Date" />
<jsp:useBean id="format" class="java.text.SimpleDateFormat" />

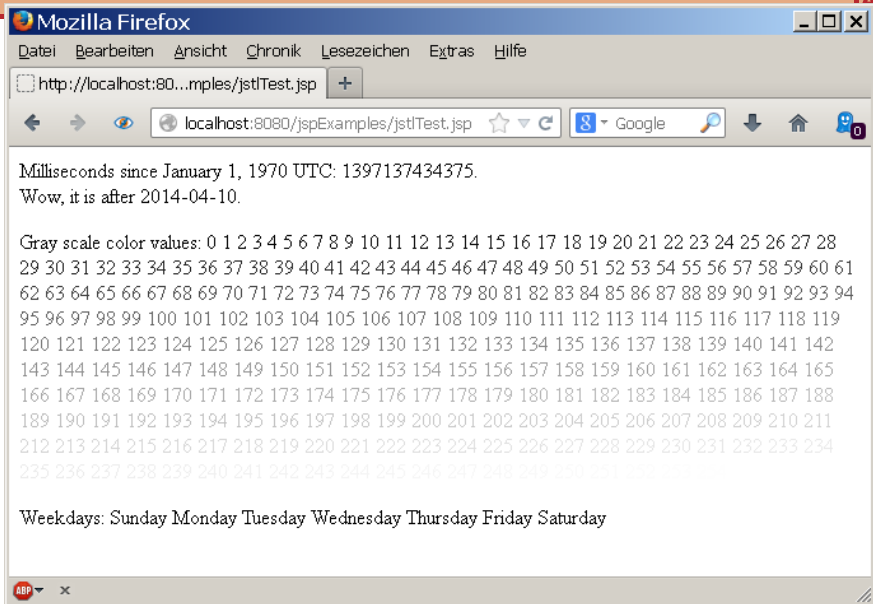
<ul>
<li>
<code>&lt;c:if test="#{x24;{ourDate.time}>1460217600000}">Wow, it is after 2016-04-10.&lt;/c:if></code>
evaluates to "<span><c:if test="#{ourDate.time > 1460217600000}">Wow,&lt;it is after 2016-04-10.</c:if></span>"
</li>

<li>
if-then-else style expressions can be written as:
<pre>
&lt;c:choose><br/>
&lt;c:when test="#{x24;{ourDate.time}>1460217600000}">Wow, it is after 2016-04-10.&lt;/c:when><br/>
&lt;c:otherwise>No, it is before 2016-04-10.&lt;/c:otherwise><br/>
&lt;/c:choose></pre>
which evaluates to
"<span><c:choose>
<c:when test="#{ourDate.time > 1460217600000}">Wow,&lt;it is after 2016-04-10.</c:when>
<c:otherwise>No,&lt;it is before 2016-04-10.</c:otherwise>
</c:choose></span>"
</li>

<li>
Loops over numerical variables can be implemented as follows:
<pre>
Gray scale color values:<br/>
&lt;c:forEach var="col" begin="0" end="255"><br/>
&nbsp;&nbsp;&lt;span style="color:rgb(&#{x24;{col}},&#{x24;{col}},&#{x24;{col}})">&#{x24;{col}}&lt;/span><br/>
&lt;/c:forEach><br/>
</pre> becomes<br/>
Gray scale color values:
  <c:forEach var="col" begin="0" end="255">
    <span style="color:rgb(&#{col},&#{col},&#{col})">&#{col}</span>
  </c:forEach>
</li>

<li>
We can also loop over collections, such as the week day names:
<code>&lt;c:forEach var="w" items="#{x24;{format.dateFormatSymbols.weekdays}}">&#{x24;{w}}&lt;/c:forEach></code>
becomes
"<span><c:forEach var="w" items="#{format.dateFormatSymbols.weekdays}">&#{w}</c:forEach></span>"
</li>

</ul>
```

- JavaServer Pages as a technology for developing dynamic web applications with user interaction
- Produce HTML documents
- Are Java Servlets ^[1-4]
- Run in a Servlet Container that performs the HTTP interactions for us by using sockets and which achieves good performance by using thread pools
- JSPs feature JavaBeans, JSP EE, and JSTL techniques

- See the documentation of the Java Servlets example in the GitHub Repository.

- See the documentation of the Java Servlets example in the GitHub Repository.
- Download the newest open source edition from <http://javaee.github.io/glassfish/download/>

- See the documentation of the Java Servlets example in the GitHub Repository.
- Download the newest open source edition from <http://javaee.github.io/glassfish/download/>, at the time of this writing, this is:
 - GlassFish Server 4.1.2. Java EE 7 Web Profile^[14–16]

- See the documentation of the Java Servlets example in the GitHub Repository.
- Download the newest open source edition from <http://javaee.github.io/glassfish/download/>, at the time of this writing, this is:
 - GlassFish Server 4.1.2. Java EE 7 Web Profile^[14–16]
- Unzip the archive and choose a directory, say, `{GLASSFISH_DIR}` as target folder

- See the documentation of the Java Servlets example in the GitHub Repository.
- Download the newest open source edition from <http://javaee.github.io/glassfish/download/>, at the time of this writing, this is:
 - GlassFish Server 4.1.2. Java EE 7 Web Profile ^[14–16]
 - glassfish-4.1.2-web.zip
- Unzip the archive and choose a directory, say, `{GLASSFISH_DIR}` as target folder

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)
- If everything goes well, a lot of log information will come

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)
- If everything goes well, a lot of log information will come:
 - some JVM initialization blabla

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)
- If everything goes well, a lot of log information will come:
 - some JVM initialization blabla
 - Launching GlassFish on Felix platform (whatever that means)

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)
- If everything goes well, a lot of log information will come:
 - some JVM initialization blabla
 - Launching GlassFish on Felix platform (whatever that means)
 - a lot of INFO log entries

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)
- If everything goes well, a lot of log information will come:
 - some JVM initialization blabla
 - Launching GlassFish on Felix platform (whatever that means)
 - a lot of INFO log entries
 - If that works, go to slide “GlassFish Administration”

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)
- If everything goes well, a lot of log information will come:
 - some JVM initialization blabla
 - Launching GlassFish on Felix platform (whatever that means)
 - a lot of INFO log entries
 - If that works, go to slide “GlassFish Administration”
- Under Windows, a window may pop up asking you for allowing the program internet access permission, which you should OK.

- Open the command prompt or terminal (under Windows: Windows-key+R, type “cmd”, hit enter)
- Change the directory to `{GLASSFISH_DIR}\glassfish4\bin`
- Type `asadmin start-domain --verbose`, hit enter (under Linux, put a `./` before the command)
- If everything goes well, a lot of log information will come:
 - some JVM initialization blabla
 - Launching GlassFish on Felix platform (whatever that means)
 - a lot of INFO log entries
 - If that works, go to slide “GlassFish Administration”
- Under Windows, a window may pop up asking you for allowing the program internet access permission, which you should OK.
- However, instead you may also get some error messages, which we discuss on the following two slides.

- You may get a message like *"The system cannot find the path."*

¹For JavaServer Pages, a **JRE** (Java Runtime Environment) is not enough – it must be a **JDK** (Java Developer Kit).

- You may get a message like *"The system cannot find the path."*
- This means GlassFish cannot find the path to the right **JDK**¹

¹For JavaServer Pages, a **JRE** (Java Runtime Environment) is not enough – it must be a **JDK** (Java Developer Kit).

- You may get a message like *"The system cannot find the path."*
- This means GlassFish cannot find the path to the right **JDK**¹
- Open the file
`{GLASSFISH_DIR}\glassfish4\glassfish\config\asenv.bat`

¹For JavaServer Pages, a **JRE** (Java Runtime Environment) is not enough – it must be a **SDK** (Java Developer Kit).

- You may get a message like *"The system cannot find the path."*
- This means GlassFish cannot find the path to the right **JDK**¹
- Open the file
`{GLASSFISH_DIR}\glassfish4\glassfish\config\asenv.bat`
- Find the entry `"set AS_JAVA=..."`

¹For JavaServer Pages, a **JRE** (Java Runtime Environment) is not enough – it must be a **SDK** (Java Developer Kit).

- You may get a message like *"The system cannot find the path."*
- This means GlassFish cannot find the path to the right **JDK**¹
- Open the file
`{GLASSFISH_DIR}\glassfish4\glassfish\config\asenv.bat`
- Find the entry `"set AS_JAVA=..."`
- Make sure that it points to an existing **JDK** (in my case:
`"set AS_JAVA=C:\Program Files\Java\jdk1.7.0_01"`

¹For JavaServer Pages, a **JRE** (Java Runtime Environment) is not enough – it must be a **SDK** (Java Developer Kit).

- You may get a message like *"The system cannot find the path."*
- This means GlassFish cannot find the path to the right **JDK**¹
- Open the file
`{GLASSFISH_DIR}\glassfish4\glassfish\config\asenv.bat`
- Find the entry `"set AS_JAVA=..."`
- Make sure that it points to an existing **JDK** (in my case:
`"set AS_JAVA=C:\Program Files\Java\jdk1.7.0_01"`)
- Store your changes, go back to slide "Getting it to run"

¹For JavaServer Pages, a **JRE** (Java Runtime Environment) is not enough – it must be a **JDK** (Java Developer Kit).

- You may get a message like *“There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains.”*

- You may get a message like *“There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains.”*
- For some reason, no domain was created during the installation process

- You may get a message like *“There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains.”*
- For some reason, no domain was created during the installation process \Rightarrow we can do this now
- open the command prompt (windows-key+R, type “cmd”, hit enter)

- You may get a message like *“There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains.”*
- For some reason, no domain was created during the installation process \Rightarrow we can do this now
- open the command prompt (windows-key+R, type “cmd”, hit enter)
- change the directory to {GLASSFISH_DIR}\glassfish4\bin

- You may get a message like *“There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains.”*
- For some reason, no domain was created during the installation process \Rightarrow we can do this now
- open the command prompt (windows-key+R, type “cmd”, hit enter)
- change the directory to {GLASSFISH_DIR}\glassfish4\bin
- type `asadmin create-domain --adminport 4848 domain1`, hit enter

- You may get a message like *“There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains.”*
- For some reason, no domain was created during the installation process \Rightarrow we can do this now
- open the command prompt (windows-key+R, type “cmd”, hit enter)
- change the directory to {GLASSFISH_DIR}\glassfish4\bin
- type `asadmin create-domain --adminport 4848 domain1`, hit enter

- You may get a message like *“There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains.”*
- For some reason, no domain was created during the installation process \Rightarrow we can do this now
- open the command prompt (windows-key+R, type “cmd”, hit enter)
- change the directory to {GLASSFISH_DIR}\glassfish4\bin
- type `asadmin create-domain --adminport 4848 domain1`, hit enter

- You may get a message like *"There are no domains in {GLASSFISH_DIR}\glassfish\glassfish4\domains."*
- For some reason, no domain was created during the installation process \Rightarrow we can do this now
- open the command prompt (windows-key+R, type "cmd", hit enter)
- change the directory to {GLASSFISH_DIR}\glassfish4\bin
- type `asadmin create-domain --adminport 4848 domain1`, hit enter
- If everything succeeds, go back to slide "Getting it to run"

- Open the web browser

- Open the web browser
- Type `http://localhost:4848` in your address bar, hit enter

ORACLE

GlassFish Server Administration Console

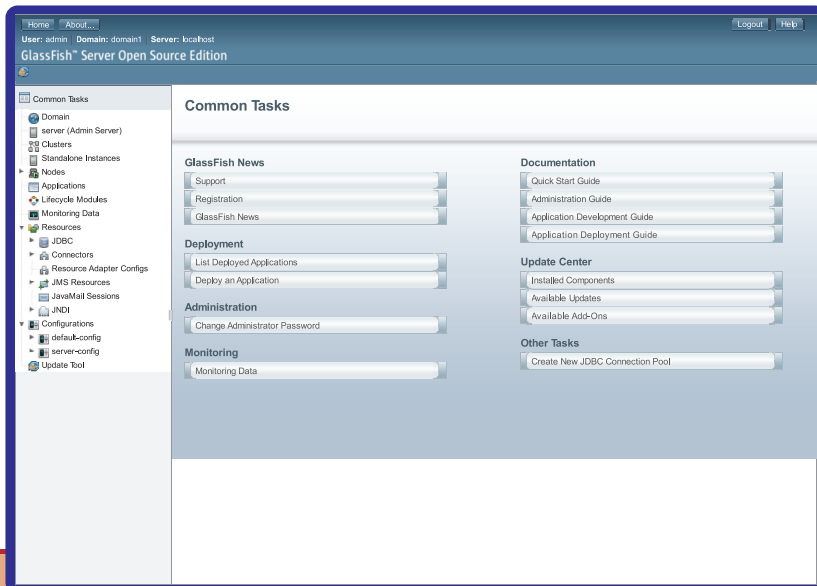
Welcome to GlassFish Server Open Source Edition 3.1.2.2 (build 5).



Status: The Admin Console is starting. Please wait.

If the browser does not refresh the page automatically please reload the page.

- Open the web browser
- Type `http://localhost:4848` in your address bar, hit enter
- If everything goes well, you should come to the administration form
- If you arrive at the administration screen, then everything is fine

A screenshot of the GlassFish Administration Console web interface. The interface has a blue header bar with navigation links and user information. A left sidebar contains a tree view of the system's configuration. The main content area is titled "Common Tasks" and is divided into several sections: "GlassFish News", "Deployment", "Administration", "Monitoring", "Documentation", "Update Center", and "Other Tasks". Each section contains a list of links to various administrative functions and resources.

Home About...

User: admin Domain: domain1 Server: localhost

Logout Help

GlassFish™ Server Open Source Edition

Common Tasks

- Domain
 - server (Admin Server)
- Clusters
 - Standalone Instances
- Nodes
 - Applications
 - Lifecycle Modules
- Monitoring Data
- Resources
 - JDBC
 - Connectors
 - Resource Adapter Configs
 - JMS Resources
 - JavaMail Sessions
 - JNDI
- Configurations
 - default-config
 - server-config
- Update Tool

Common Tasks

GlassFish News

- Support
- Registration
- GlassFish News

Deployment

- List Deployed Applications
- Deploy an Application

Administration

- Change Administrator Password

Monitoring

- Monitoring Data

Documentation

- Quick Start Guide
- Administration Guide
- Application Development Guide
- Application Deployment Guide

Update Center

- Installed Components
- Available Updates
- Available Add-Ons

Other Tasks

- Create New JDBC Connection Pool

- Open the web browser
- Type `http://localhost:4848` in your address bar, hit enter
- If everything goes well, you should come to the administration form
- If you arrive at the administration screen, then everything is fine
- By the way, you can even see that GlassFish is using thread pools, exactly like we described in the sockets lesson...

[Home](#) [About...](#)

User: admin Domain: domain1 Server: localhost

Logout Help

GlassFish™ Server Open Source Edition

Tree

Common Tasks

Domain

- server (Admin Server)

Clusters

Standalone Instances

Nodes

- Applications
- LifeCycle Modules
- Monitoring Data

Resources

- JDBC
- Connectors
- Resource Adapter Configs
- JMS Resources
- JavaMail Sessions
- JNDI

Configurations

- default-config
- server-config
 - JVM Settings
 - Logger Settings
 - Web Container
 - EJB Container
 - Java Message Service
 - Security
 - Transaction Service
 - HTTP Service
 - Virtual Servers
 - Network Config
 - Thread Pools
 - ORB
 - Admin Service
 - Connector Service
 - Monitoring
 - Custom Properties

Thread Pools

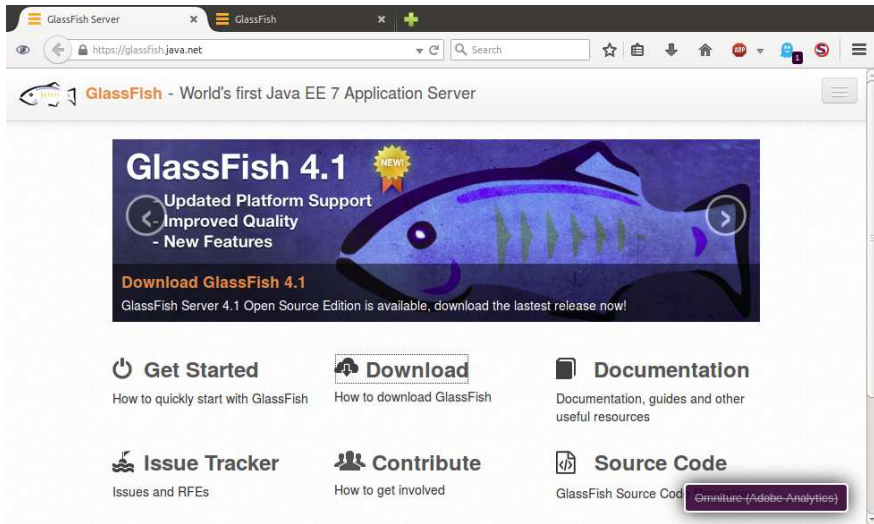
Use thread pools to limit a service to a specific number of concurrent threads.

Configuration Name: server-config

Thread Pools (3)

New... Delete

Thread PoolID	Max Thread Pool Size	Min Thread Pool Size	Max Queue Size	Idle Thread Timeout
<input type="checkbox"/> admin-thread-pool	50	2	256	900
<input type="checkbox"/> http-thread-pool	5	2	4096	900
<input type="checkbox"/> thread-pool-1	200	2	4096	900



The screenshot shows a web browser window with two tabs: "GlassFish Server" and "GlassFish". The address bar displays "https://glassfish.java.net". The page header features the GlassFish logo and the text "GlassFish - World's first Java EE 7 Application Server". The main content area has a large banner for "GlassFish 4.1" with a "NEW!" badge. The banner lists "Updated Platform Support", "Improved Quality", and "New Features". Below the banner is a "Download GlassFish 4.1" button and a note that the "GlassFish Server 4.1 Open Source Edition is available, download the latest release now!". The page is organized into a grid of six links: "Get Started" (How to quickly start with GlassFish), "Download" (How to download GlassFish), "Documentation" (Documentation, guides and other useful resources), "Issue Tracker" (Issues and RFEs), "Contribute" (How to get involved), and "Source Code" (GlassFish Source Code). A small "Omniure (Adobe Analytics)" tag is visible in the bottom right corner of the page.

GlassFish 4.1 NEW!

Updated Platform Support
Improved Quality
New Features

Download GlassFish 4.1

GlassFish Server 4.1 Open Source Edition is available, download the latest release now!

Get Started
How to quickly start with GlassFish

Download
How to download GlassFish

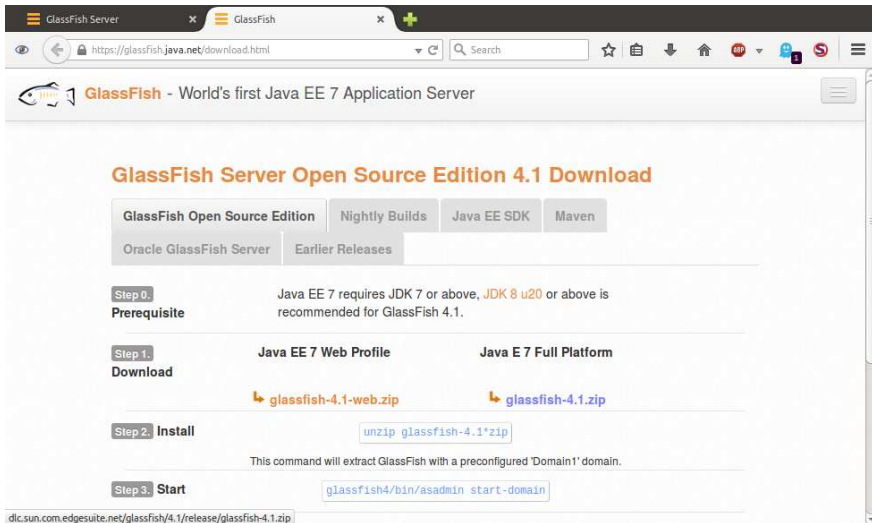
Documentation
Documentation, guides and other useful resources

Issue Tracker
Issues and RFEs

Contribute
How to get involved

Source Code
GlassFish Source Code

Omniure (Adobe Analytics)



The screenshot shows a web browser window with two tabs: 'GlassFish Server' and 'GlassFish'. The address bar shows 'https://glassfish.java.net/download.html'. The page header features the GlassFish logo and the text 'GlassFish - World's first Java EE 7 Application Server'. The main heading is 'GlassFish Server Open Source Edition 4.1 Download'. Below this, there are tabs for 'GlassFish Open Source Edition' (selected), 'Nightly Builds', 'Java EE SDK', 'Maven', 'Oracle GlassFish Server', and 'Earlier Releases'. The page is divided into four steps: Step 0: Prerequisite, Step 1: Download, Step 2: Install, and Step 3: Start. Step 0 mentions that Java EE 7 requires JDK 7 or above, with JDK 8 u20 or above recommended. Step 1 shows two download links: 'glassfish-4.1-web.zip' for the Java EE 7 Web Profile and 'glassfish-4.1.zip' for the Java E 7 Full Platform. Step 2 shows the command 'unzip glassfish-4.1*.zip' and a note that this command will extract GlassFish with a preconfigured 'Domain1' domain. Step 3 shows the command 'glassfish4/bin/asadmin start-domain'. At the bottom left, there is a URL: 'dlc.sun.com.edgesuite.net/glassfish/4.1/release/glassfish-4.1.zip'.

GlassFish Server Open Source Edition 4.1 Download

GlassFish Open Source Edition | Nightly Builds | Java EE SDK | Maven | Oracle GlassFish Server | Earlier Releases

Step 0. Prerequisite | Java EE 7 requires JDK 7 or above, **JDK 8 u20** or above is recommended for GlassFish 4.1.

Step 1. Download

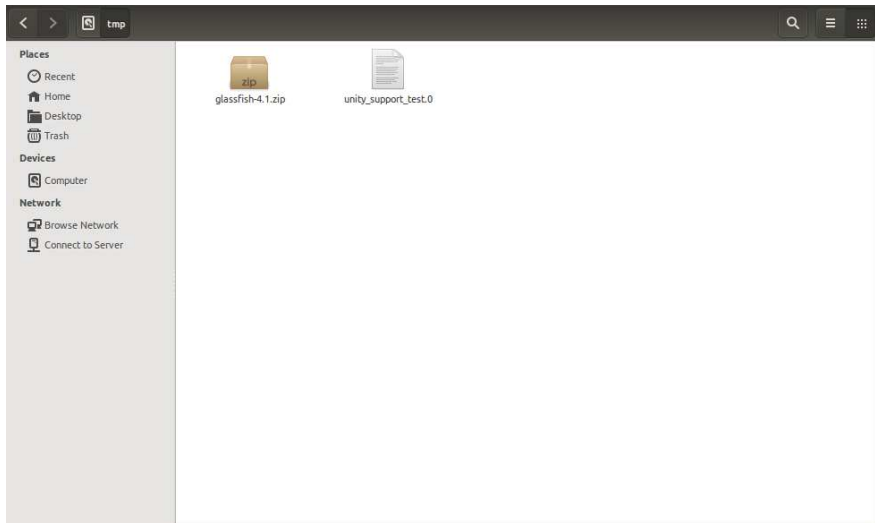
Java EE 7 Web Profile	Java E 7 Full Platform
glassfish-4.1-web.zip	glassfish-4.1.zip

Step 2. Install | `unzip glassfish-4.1*.zip`

This command will extract GlassFish with a preconfigured 'Domain1' domain.

Step 3. Start | `glassfish4/bin/asadmin start-domain`

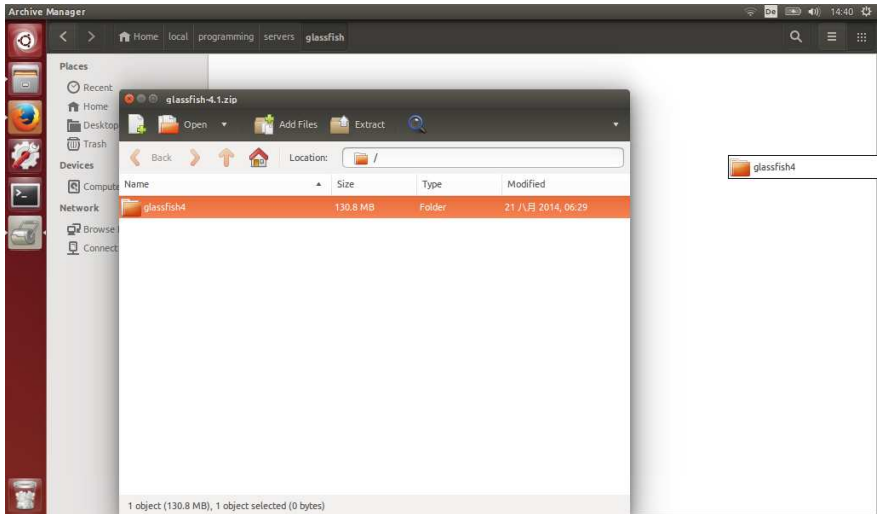
dlc.sun.com.edgesuite.net/glassfish/4.1/release/glassfish-4.1.zip



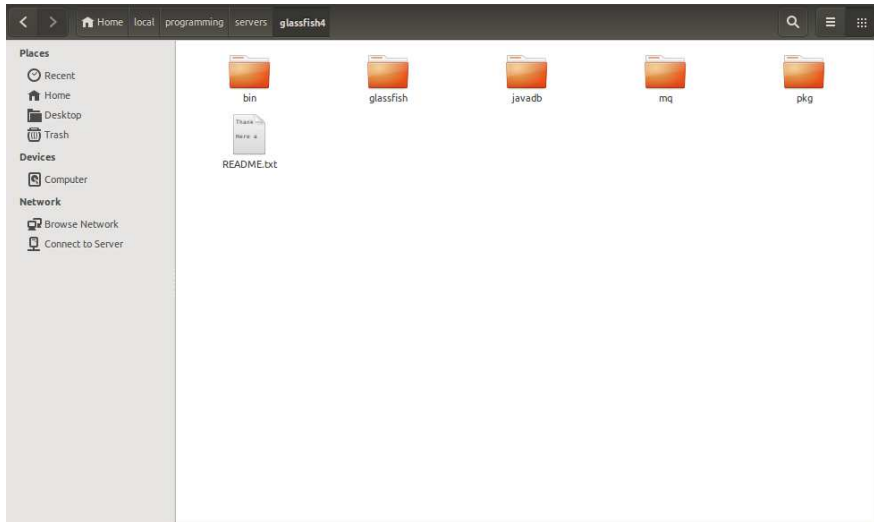
Installing GlassFish (Linux)



Installing GlassFish (Linux)

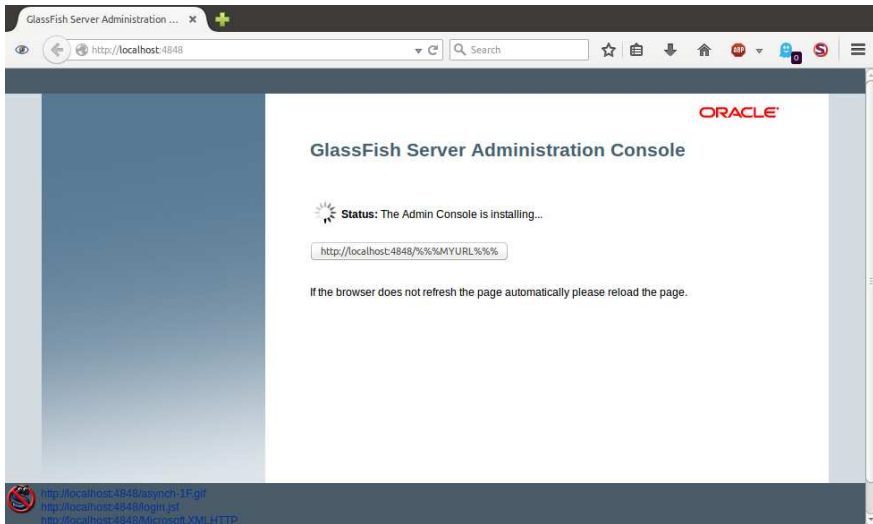


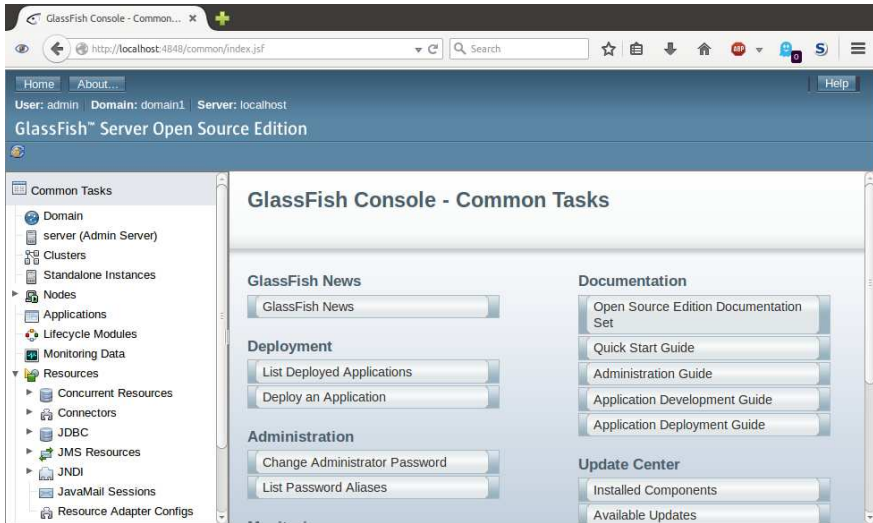
Installing GlassFish (Linux)



```
tweise@xiao: ~/local/programming/servers/glassfish4/bin
tweise@xiao:~/local/programming/servers/glassfish4/bin$ ./asadmin start-domain
Waiting for domain1 to start .....
```

```
tweise@xiao: ~/local/programming/servers/glassfish4/bin
twiese@xiao:~/local/programming/servers/glassfish4/bin$ ./asadmin start-domain
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain Location: /home/tweise/local/programming/servers/glassfish4/glassfish/domains/domain1
Log File: /home/tweise/local/programming/servers/glassfish4/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
twiese@xiao:~/local/programming/servers/glassfish4/bin$
```





The screenshot shows the GlassFish Console web interface in a browser. The address bar shows `http://localhost:4848/common/index.jsf`. The page title is "GlassFish™ Server Open Source Edition". The user is logged in as "admin" on "domain1" at "localhost".

Common Tasks

- Domain
 - server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs

GlassFish Console - Common Tasks

GlassFish News

- GlassFish News

Deployment

- List Deployed Applications
- Deploy an Application

Administration

- Change Administrator Password
- List Password Aliases

Documentation

- Open Source Edition Documentation Set
- Quick Start Guide
- Administration Guide
- Application Development Guide
- Application Deployment Guide

Update Center

- Installed Components
- Available Updates

```
tweise@xiao: ~/local/programming/servers/glassfish4/bin
twiese@xiao:~/local/programming/servers/glassfish4/bin$ ./asadmin stop-domain
Waiting for the domain to stop ..
Command stop-domain executed successfully.
twiese@xiao:~/local/programming/servers/glassfish4/bin$
```

- Maven is maybe the most widely-used project build and dependency management tool in Java

- Maven is maybe the most widely-used project build and dependency management tool in Java
- It allows you to specify which other software your project depends on

- Maven is maybe the most widely-used project build and dependency management tool in Java
- It allows you to specify which other software your project depends on, which is then automatically downloaded and installed during the build process

- Maven is maybe the most widely-used project build and dependency management tool in Java
- It allows you to specify which other software your project depends on, which is then automatically downloaded and installed during the build process
- Maven can build your project and generate archives, documentation, a project website, and other artifacts

- Maven is maybe the most widely-used project build and dependency management tool in Java
- It allows you to specify which other software your project depends on, which is then automatically downloaded and installed during the build process
- Maven can build your project and generate archives, documentation, a project website, and other artifacts
- Maven supports unit testing, i.e., allows you to automatically check whether your code meets certain requirements

- Maven is maybe the most widely-used project build and dependency management tool in Java
- It allows you to specify which other software your project depends on, which is then automatically downloaded and installed during the build process
- Maven can build your project and generate archives, documentation, a project website, and other artifacts
- Maven supports unit testing, i.e., allows you to automatically check whether your code meets certain requirements
- Maven allows for automatic deployment (which we will not do here)

- Maven is maybe the most widely-used project build and dependency management tool in Java
- It allows you to specify which other software your project depends on, which is then automatically downloaded and installed during the build process
- Maven can build your project and generate archives, documentation, a project website, and other artifacts
- Maven supports unit testing, i.e., allows you to automatically check whether your code meets certain requirements
- Maven allows for automatic deployment (which we will not do here)
- Maven is integrated into Eclipse (good support in Eclipse Luna, very well integrated in Eclipse Mars)

- Maven is maybe the most widely-used project build and dependency management tool in Java
- It allows you to specify which other software your project depends on, which is then automatically downloaded and installed during the build process
- Maven can build your project and generate archives, documentation, a project website, and other artifacts
- Maven supports unit testing, i.e., allows you to automatically check whether your code meets certain requirements
- Maven allows for automatic deployment (which we will not do here)
- Maven is integrated into Eclipse (good support in Eclipse Luna, very well integrated in Eclipse Mars)
- The example project is provided with a Maven `pom` file, a file in the XML format in which Maven projects are specified.

Listing: [pom.xml] – Part 1: Basic Project Information

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>thomasWeise</groupId>
  <artifactId>javaServerPages</artifactId>
  <version>0.8.0</version>
  <packaging>war</packaging>
  <name>JavaServer Pages</name>
  <description>Examples for using Java Server Pages (in
    Java).</description>
```


Listing: [pom.xml] – Part 2: Information about Organization

```
<url>http://www.it-weise.de/</url>  
<organization>  
  <url>http://www.it-weise.de/</url>  
  <name>thomasWeise</name>  
</organization>
```

Listing: [pom.xml] – Part 3: Information about Developer

```
<developers>
  <developer>
    <id>thomasWeise</id>
    <name>Thomas Weise</name>
    <email>tweise@ustc.edu.cn</email>
    <url>http://www.it-weise.de/</url>
    <organization>University of Science and Technology of
      China (USTC)</organization>
    <organizationUrl>http://www.ustc.edu.cn/</organizationUrl>
    <roles>
      <role>architect</role>
      <role>developer</role>
    </roles>
    <timezone>China Time Zone</timezone>
  </developer>
</developers>
```

Listing: [pom.xml] – Part 4: Properties for Rest of pom

```
<properties>
  <encoding>UTF-8</encoding>
  <project.build.sourceEncoding>${encoding}</project.build.sourceEncoding>
  <project.reporting.outputEncoding>${encoding}</project.reporting.outputEncoding>
  <jdk.version>1.7</jdk.version>
</properties>
```

Listing: [pom.xml] – Part 5: Licensing

```
<licenses>
  <license>
    <name>GNU GENERAL PUBLIC LICENSE Version 3, 29 June
      2007</name>
    <url>http://www.gnu.org/licenses/gpl-3.0-standalone.html</url>
    <distribution>repo</distribution>
  </license>
</licenses>
```

Listing: [pom.xml] – Part 6: SCM, Issue Management, and Inception Year

```
<issueManagement>
  <url>https://github.com/thomasWeise/distributedComputingExamples/issues</url>
  <system>GitHub</system>
</issueManagement>

<scm>
  <connection>scm:git:git@github.com:thomasWeise/distributedComputingExamples.git</connection>
  <developerConnection>scm:git:git@github.com:thomasWeise/distributedComputingExamples.git</developerConnection>
  <url>git@github.com:thomasWeise/distributedComputingExamples.git</url>
</scm>

<inceptionYear>2016</inceptionYear>
```



Listing: [pom.xml] – Part 7: Dependencies

```
<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope> <!-- provided by servlet container -->
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.1.2</version>
  </dependency>
</dependencies>
```



Listing: [pom.xml] – Part 8: Build

```
<build>
  <finalName>myJSPs</finalName>

  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.1</version>
      <configuration>
        <source>${jdk.version}</source>
        <target>${jdk.version}</target>
        <encoding>${encoding}</encoding>
        <showWarnings>true</showWarnings>
        <showDeprecation>true</showDeprecation>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>
```

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .
- In the opening sub-menu, left-click on `Update Project...` .

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .
- In the opening sub-menu, left-click on `Update Project...` .
- In the opening window. . .

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .
- In the opening sub-menu, left-click on `Update Project...` .
- In the opening window...
 - Make sure the project (`JavaServlets`) is selected.

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .
- In the opening sub-menu, left-click on `Update Project...` .
- In the opening window...
 - Make sure the project (`JavaServlets`) is selected.
 - Make sure that `Update project configuration from pom.xml` is selected.

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .
- In the opening sub-menu, left-click on `Update Project...` .
- In the opening window...
 - Make sure the project (`JavaServlets`) is selected.
 - Make sure that `Update project configuration from pom.xml` is selected.
 - You can also select `Clean projects` .

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .
- In the opening sub-menu, left-click on `Update Project...` .
- In the opening window...
 - Make sure the project (`JavaServlets`) is selected.
 - Make sure that `Update project configuration from pom.xml` is selected.
 - You can also select `Clean projects` .
 - Click 'OK'.

- I assume that you have downloaded the examples ZIP or checked them out from GitHub.
- If you import this project in Eclipse, it may first show you a lot of errors.
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, left-click on `Maven` .
- In the opening sub-menu, left-click on `Update Project...` .
- In the opening window. . .
- Now the structure of the project in the 'package view' should slightly change, the project will be re-compiled, and the errors should disappear.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container:
 - it contains a folder WEB-INF

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container:
 - it contains a folder WEB-INF
 - the folder WEB-INF contains the file `web.xml`

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container:
 - it contains a folder WEB-INF
 - the folder WEB-INF contains the file `web.xml`
 - the folder WEB-INF contains the folder `classes` which contains all Java classes and packages that are part of the web application

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains the file `web.xml`
 - the folder `WEB-INF` contains the folder `classes` which contains all Java classes and packages that are part of the web application
 - the folder `WEB-INF` may contain the folder `libs` which contains additional required libraries.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Maven builds the `war` for us.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Maven builds the `war` for us.
- In Eclipse, when building for the first time, you do the following.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Make sure that you can see the package view on the left-hand side of the Eclipse window.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, choose `Run As` .

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, choose `Run As` .
- In the opening sub-menu choose `Run Configurations...` .

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, choose `Run As` .
- In the opening sub-menu choose `Run Configurations...` .
- In the opening window, choose `Maven Build`

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Make sure that you can see the package view on the left-hand side of the Eclipse window.
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, choose `Run As` .
- In the opening sub-menu choose `Run Configurations...` .
- In the opening window, choose `Maven Build`
- In the new window `Run Configurations` /
`Create, manage, and run configurations` , choose `Maven Build` in the small white pane on the left side.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Right-click on the project (`JavaServlets`) in the package view.
- In the opening pop-up menu, choose `Run As` .
- In the opening sub-menu choose `Run Configurations...` .
- In the opening window, choose `Maven Build`
- In the new window `Run Configurations` /
Create, manage, and run configurations , choose `Maven Build` in the small white pane on the left side.
- Click `New launch configuration` (the first symbol from the left on top of the small white pane).

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- In the opening pop-up menu, choose `Run As`.
- In the opening sub-menu choose `Run Configurations...`
- In the opening window, choose `Maven Build`
- In the new window `Run Configurations` / `Create, manage, and run configurations`, choose `Maven Build` in the small white pane on the left side.
- Click `New launch configuration` (the first symbol from the left on top of the small white pane).
- Write a useful name for this configuration in the `Name` field. You can use this configuration again later.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- In the opening sub-menu choose `Run Configurations...`
- In the opening window, choose `Maven Build`
- In the new window `Run Configurations` /
`Create, manage, and run configurations`, choose `Maven Build` in the small white pane on the left side.
- Click `New launch configuration` (the first symbol from the left on top of the small white pane).
- Write a useful name for this configuration in the `Name` field. You can use this configuration again later.
- In the tab `Main` enter the `Base directory` of the project, this is the folder called `javaServlets` containing the Eclipse/Maven project.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- In the opening window, choose `Maven Build`
- In the new window `Run Configurations` /
`Create, manage, and run configurations`, choose `Maven Build` in the small white pane on the left side.
- Click `New launch configuration` (the first symbol from the left on top of the small white pane).
- Write a useful name for this configuration in the `Name` field. You can use this configuration again later.
- In the tab `Main` enter the `Base directory` of the project, this is the folder called `javaServlets` containing the Eclipse/Maven project.
- Under `Goals`, enter `clean compile war:war`. This will build a war archive.

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- In the new window `Run Configurations` / `Create, manage, and run configurations`, choose `Maven Build` in the small white pane on the left side.
- Click `New launch configuration` (the first symbol from the left on top of the small white pane).
- Write a useful name for this configuration in the `Name` field. You can use this configuration again later.
- In the tab `Main` enter the `Base directory` of the project, this is the folder called `javaServlets` containing the Eclipse/Maven project.
- Under `Goals`, enter `clean compile war:war`. This will build a war archive.
- Click `Apply`

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Click **New launch configuration** (the first symbol from the left on top of the small white pane).
- Write a useful name for this configuration in the **Name** field. You can use this configuration again later.
- In the tab **Main** enter the **Base directory** of the project, this is the folder called **javaServlets** containing the Eclipse/Maven project.
- Under **Goals**, enter **clean compile war:war**. This will build a war archive.
- Click **Apply**
- Click **Run**

- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- In the tab `Main` enter the `Base directory` of the project, this is the folder called `javaServlets` containing the Eclipse/Maven project.
- Under `Goals`, enter `clean compile war:war`. This will build a war archive.
- Click `Apply`
- Click `Run`
- The build will start, you will see its status output in the console window.

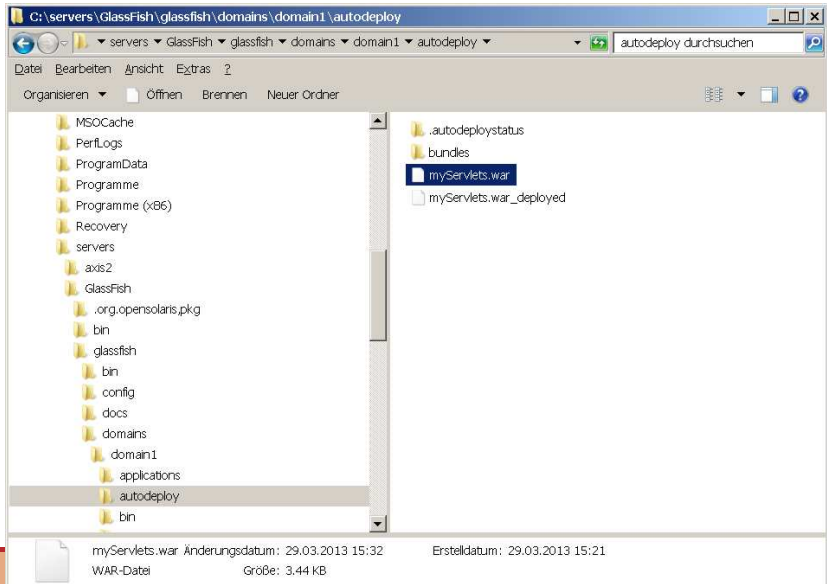
- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Under `Goals`, enter `clean compile war:war`. This will build a war archive.
- Click `Apply`
- Click `Run`
- The build will start, you will see its status output in the console window.
- The folder `target` will contain a file `myServlets.war` after the build. This is the deployable archive with our application.

- Deploying a WAR archive is easy

- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`

- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`
 - The running GlassFish server will automatically load and start it.

Deploying a WAR



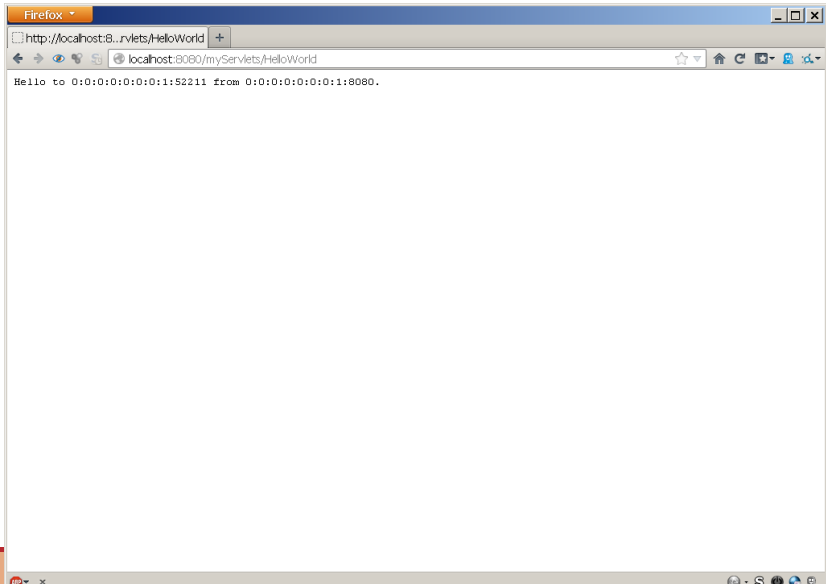
- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`
 - The running GlassFish server will automatically load and start it.
- You can now access the application under
`http://localhost:8080/myServlets/HelloWorld`

- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`
 - The running GlassFish server will automatically load and start it.
- You can now access the application under
`http://localhost:8080/myServlets/HelloWorld`, if
`myServlets` is the name you chose for your war file

- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`
 - The running GlassFish server will automatically load and start it.
- You can now access the application under
`http://localhost:8080/myServlets/HelloWorld`, if
`myServlets` is the name you chose for your war file and `HelloWorld` is a servlet that you have registered in the `web.xml` file

- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`
 - The running GlassFish server will automatically load and start it.
- You can now access the application under
`http://localhost:8080/myServlets/HelloWorld`, if
`myServlets` is the name you chose for your war file and `HelloWorld` is a servlet that you have registered in the `web.xml` file (which now is in folder `WEB-INF` of `myServlets.war`)

Deploying a WAR



- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`
 - The running GlassFish server will automatically load and start it.
- You can now access the application under
`http://localhost:8080/myServlets/HelloWorld`, if
`myServlets` is the name you chose for your war file and `HelloWorld` is a servlet that you have registered in the `web.xml` file (which now is in folder `WEB-INF` of `myServlets.war`)
- You can now also find the servlet in the administration console (see slide “GlassFish Administration”)

Deploying a WAR



[Home](#) [About...](#)

User: admin Domain: domain1 Server: localhost

GlassFish™ Server Open Source Edition

Common Tasks

Domain

- server (Admin Server)
- Clusters
- Standalone Instances

Nodes

Applications

- myServlets

Lifecycle Modules

Monitoring Data

Resources

- JDBC
- Connectors
- Resource Adapter Configs

JMS Resources

JavaMail Sessions

JNDI

Configurations

- default-config
- server-config

Update Tool

General

Descriptor

Edit Application

Save Cancel

Modify an existing application or module.

Name: myServlets

Status: ☒ Enabled

Virtual Servers:

server

Associates an Internet domain name with a physical server.

Context Root: /myServlets
Path relative to server's base URL.

Description:

Location: \${com.sun.aas.instanceRootURL}/applications/myServlets/

Libraries:

Modules and Components (5)

Module Name	Engines	Component Name	Type	Action
myServlets	[web]	---	---	Launch
myServlets		RequestData	Servlet	
myServlets		default	Servlet	
myServlets		jsp	Servlet	
myServlets		HelloWorld	Servlet	

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfu.edu.cn
<http://www.it-weise.de>

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China



Caspar David Friedrich, "Der Wanderer über dem Nebelmeer", 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog



1. Rajiv Mordani. *JSR 315: Java™ Servlet 3.0 Specification Version 3.0 Rev a (Maintenance Release)*, volume 315 of *Java Specification Requests (JSR)*. Maintenance release edition, December 2010. URL <http://download.oracle.com/otndocs/jcp/servlet-3.0-mrel-eval-oth-JSpec>.
2. Karl Moss. *Java Servlets*. McGraw-Hill Java Masters. Maidenhead, England, UK: McGraw-Hill Ltd., 1999. ISBN 0071351884 and 9780071351881. URL <http://books.google.de/books?id=ToBGAAAYAAJ>.
3. Jason Hunter and William Crawford. *Java Servlet Programming*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2010. ISBN 1449390676 and 9781449390679. URL <http://books.google.de/books?id=dsU4Lk-Gwk0C>.
4. Jeff M. Genender. *Enterprise Java Servlets*, volume 1. Reading, MA, USA: Addison-Wesley Publishing Co. Inc., 2002. ISBN 020170921X and 9780201709216. URL <http://books.google.de/books?id=MbhQAAAAAAAJ>.
5. Steven Holzner. *PHP: The Complete Reference*. Maidenhead, England, UK: McGraw-Hill Ltd., 2007. ISBN 0071508546 and 9780071508544. URL http://books.google.de/books?id=Ee_22ndP6gIC.
6. Pierre Delisle, Jan Luehe, and Mark Roth. *JSR 245: JavaServer Pages™ 2.1 Final Release*, volume 245 of *Java Specification Requests (JSR)*. May 8, 2006. URL <http://jcp.org/aboutJava/communityprocess/final/jsr245/index.html>.
7. Hans Bergsten. *JavaServer Pages*. The Java Series. Upper Saddle River, NJ, USA: Prentice Hall International Inc., Santa Clara, CA, USA: Sun Microsystems Press (SMP), and Reading, MA, USA: Addison-Wesley Professional, 2003. ISBN 0596005636 and 9780596005634. URL <http://books.google.de/books?id=JNE8sqfQNAUC>.
8. Vivek Chopra, Jon Eaves, Rupert Jones, Sing Li, and John T. Bell. *Beginning JavaServer Pages*. Wrox Beginning Guides. New York, NY, USA: John Wiley & Sons Ltd., 2005. ISBN 0764589520 and 9780764589522. URL <http://books.google.de/books?id=du6Y1dR0mAUC>.
9. Javaserer pages technology. In *The Java EE 5 Tutorial: For Sun Java System Application Server 9.1*, chapter 5. Redwood Shores, CA, USA: Oracle Corporation, 2010. URL <http://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html>.
10. Mukesh Prasad. Jsp tutorial, November 11, 2012. URL <http://www.jsptut.com/>.
11. Marty Hall. Javaserer pages (jsp) 1.0, September 23, 2010. URL <http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-JSP.html>.
12. Robert Englander. *Developing Java Beans*. Java Series. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1997. ISBN 1565922891 and 9781565922891. URL <http://books.google.de/books?id=TcnMkzTp6R4C>.
13. Elliotte Rusty Harold. *JavaBeans*. Foster City, CA, USA: Idg Books Worldwide, Inc., 1998. ISBN 0764580523 and 9780764580529. URL <http://books.google.de/books?id=ByOKQAAACAAJ>.
14. *Glassfish*. Redwood Shores, CA, USA: Oracle Corporation, revision 20130208.5d7f765 edition, 2013. URL <http://glassfish.java.net/>.

15. David Heffelfinger. *Java EE 6 with GlassFish 3 Application Server*. Birmingham, UK: Packt Publishing Limited, 2010. ISBN 1849510377 and 9781849510370. URL <http://books.google.de/books?id=GCFdQ5SxPnQC>.
16. Antonio Goncalves. *Beginning Java EE 6 with GlassFish 3*. Expert's Voice in Java Technology. New York, NY, USA: Apress, Inc., August 24, 2010. ISBN 143022889X and 9781430228899. URL <http://books.google.de/books?id=8pQbMr5X-yMC>.