



Distributed Computing

Lesson 12: Java Servlets

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 API and Examples
- 3 Installing GlassFish



website

- Is there an easy way to use the HTTP protocol?
- How to use the Java Servlet API to easily access HTTP?

- The “bare bones” of HTTP are simple and can easily be implemented
- However, to deal with all possible stuff that can be sent or received properly is not easy, e.g., there are lots of possible header lines with different meanings and formats
- We know: TCP and IP are complicated to implement, but sockets are an easy-to-use API for that
- We would like to have such an easy API also for HTTP, i.e., something that sits on top of sockets and parses/produces all protocol text for us, so we can focus on the application behavior
- Then we could also build dynamic applications that can receive and process data from forms

- Simple standardized Java API ^[1] for interaction with HTTP ^[2, 3] protocol

- Simple standardized Java API ^[1] for interaction with HTTP ^[2, 3] protocol
- HTTP knows different request methods, such as GET , POST , PUT , DELETE , . . .

- Simple standardized Java API ^[1] for interaction with HTTP ^[2, 3] protocol
- HTTP knows different request methods, such as GET , POST , PUT , DELETE , . . .
- A servlet is a software component that is able to receive, process, and answer requests

- Simple standardized Java API ^[1] for interaction with HTTP ^[2, 3] protocol
- HTTP knows different request methods, such as GET , POST , PUT , DELETE , . . .
- A servlet is a software component that is able to receive, process, and answer requests
- A (low level) way to process incoming data/requests and to dynamically generate output

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will
 - use sockets to access TCP streams

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will
 - use sockets to access TCP streams and
 - read the protocol text

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will
 - use sockets to access TCP streams and
 - read the protocol text and
 - translate the text to/from Java objects

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will
 - use sockets to access TCP streams and
 - read the protocol text and
 - translate the text to/from Java objects
 - that are handed to your implementation of the Java Servlet interfaces

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will
 - use sockets to access TCP streams and
 - read the protocol text and
 - translate the text to/from Java objects
 - that are handed to your implementation of the Java Servlet interfaces
 - Your implementation will then create a response by storing information in other Java objects

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will
 - use sockets to access TCP streams and
 - read the protocol text and
 - translate the text to/from Java objects
 - that are handed to your implementation of the Java Servlet interfaces
 - Your implementation will then create a response by storing information in other Java objects
 - which then are handed back to the Servlet Container

- Java Servlets ^[1, 4–6] are basically defined as interfaces that are called from a *Servlet Container*
- The Servlet Container will
 - use sockets to access TCP streams and
 - read the protocol text and
 - translate the text to/from Java objects
 - that are handed to your implementation of the Java Servlet interfaces
 - Your implementation will then create a response by storing information in other Java objects
 - which then are handed back to the Servlet Container
 - which will create the proper HTTP text and send it back over the TCP stream to the client who made the request

- There are different implementations of Servlet Containers

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]
 - Google App Engine (GAE) ^[12–14]

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]
 - Google App Engine (GAE) ^[12–14] \Rightarrow see Cloud Computing

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]
 - Google App Engine (GAE) ^[12–14] \Rightarrow see Cloud Computing
 - GlassFish ^[15–17]

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]
 - Google App Engine (GAE) ^[12–14] \Rightarrow see Cloud Computing
 - **GlassFish** ^[15–17] \Rightarrow reference implementation (by Sun) of Java EE

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]
 - Google App Engine (GAE) ^[12–14] \Rightarrow see Cloud Computing
 - **GlassFish** ^[15–17] \Rightarrow reference implementation (by Sun) of Java EE (but as of November 2013, no commercial support anymore ^[18])

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]
 - Google App Engine (GAE) ^[12–14] ⇒ see Cloud Computing
 - **GlassFish** ^[15–17] ⇒ reference implementation (by Sun) of Java EE (but as of November 2013, no commercial support anymore ^[18])
 - Oracle WebLogic Server ^[19] (commercially supported, both WebLogic and GlassFish belong to Oracle...)

- There are different implementations of Servlet Containers:
 - Apache Tomcat ^[7, 8]
 - Jetty ^[9]
 - JBoss/WildFly ^[10, 11]
 - Google App Engine (GAE) ^[12–14] \Rightarrow see Cloud Computing
 - **GlassFish** ^[15–17] \Rightarrow reference implementation (by Sun) of Java EE (but as of November 2013, no commercial support anymore ^[18])
 - Oracle WebLogic Server ^[19] (commercially supported, both WebLogic and GlassFish belong to Oracle...)
- Most of them use thread pools, similar to what we learned in the sockets lesson, to deal with parallel requests¹

¹(with improvements for asynchronous I/O ^[20])

- Java Servlets have one method for each HTTP request method

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method (specified in HTTP protocol ^[2, 3]), e.g.,
 - `void doGet(HttpServletRequest, HttpServletResponse)`

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method (specified in HTTP protocol ^[2, 3]), e.g.,
 - `void doGet(HttpServletRequest, HttpServletResponse)`
 - `void doPost(HttpServletRequest, HttpServletResponse)`

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method (specified in HTTP protocol ^[2, 3]), e.g.,
 - `void doGet(HttpServletRequest, HttpServletResponse)`
 - `void doPost(HttpServletRequest, HttpServletResponse)`
 - `void doPut(HttpServletRequest, HttpServletResponse)`

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method (specified in HTTP protocol ^[2, 3]), e.g.,
 - `void doGet(HttpServletRequest, HttpServletResponse)`
 - `void doPost(HttpServletRequest, HttpServletResponse)`
 - `void doPut(HttpServletRequest, HttpServletResponse)`
 - `void delete(HttpServletRequest, HttpServletResponse)`
 - ...

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- **Input:** `javax.servlet.http.HttpServletRequest` object that holds the data of the request

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- **Input:** `javax.servlet.http.HttpServletRequest` object that holds the data of the request, e.g.,
 - `Cookie[] getCookies()` : get the cookies attached to the request

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- **Input:** `javax.servlet.http.HttpServletRequest` object that holds the data of the request, e.g.,
 - `Cookie[] getCookies()`
 - `String getMethod()` : get the method (e.g., `PUT`, `POST`, `GET`, ...)

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- **Input:** `javax.servlet.http.HttpServletRequest` object that holds the data of the request, e.g.,
 - `Cookie[] getCookies()`
 - `String getMethod()`
 - `String getParameter(String name)` : get the value of a query parameter

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- **Input:** `javax.servlet.http.HttpServletRequest` object that holds the data of the request, e.g.,
 - `Cookie[] getCookies()`
 - `String getMethod()`
 - `String getParameter(String name)`
 - `String getContentType()` : get the content type of the request

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- **Input:** `javax.servlet.http.HttpServletRequest` object that holds the data of the request, e.g.,
 - `Cookie[] getCookies()`
 - `String getMethod()`
 - `String getParameter(String name)`
 - `String getContentType()`
 - `String getRequestedURI()` : get the requested URI

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- **Input:** `javax.servlet.http.HttpServletRequest` object that holds the data of the request, e.g.,
 - `Cookie[] getCookies()`
 - `String getMethod()`
 - `String getParameter(String name)`
 - `String getContentType()`
 - `String getRequestURI()`
 - `HttpSession getSession()` : get the session to which the request belongs
 - ...

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into, e.g.,
 - `void addCookie(Cookie cookie)` : store a cookie

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into, e.g.,
 - `void addCookie(Cookie cookie)`
 - `void setStatus(int sc)` : set the status code for the request

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into, e.g.,
 - `void addCookie(Cookie cookie)`
 - `void setStatus(int sc)`
 - `void sendError(int sc)` : return an error code

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into, e.g.,
 - `void addCookie(Cookie cookie)`
 - `void setStatus(int sc)`
 - `void sendError(int sc)`
 - `void setContentType(String type)` : set the response content type (e.g., `text/html`)

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into, e.g.,
 - `void addCookie(Cookie cookie)`
 - `void setStatus(int sc)`
 - `void sendError(int sc)`
 - `void setContentType(String type)`
 - `void setCharacterEncoding(String charset)` : set the output character encoding

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into, e.g.,
 - `void addCookie(Cookie cookie)`
 - `void setStatus(int sc)`
 - `void sendError(int sc)`
 - `void setContentType(String type)`
 - `void setCharacterEncoding(String charset)`
 - `ServletOutputStream getOutputStream()` and `java.io.PrintWriter getWriter()` : get streams to write the output to

- Java Servlets have one method for each HTTP request method
- Java Servlets extend class `javax.servlet.http.HttpServlet`
 - Provides one handler method for each HTTP method ^[2, 3]
- Input: `javax.servlet.http.HttpServletRequest`
- Instance of `javax.servlet.http.HttpServletResponse` : object to store the request **output**/response into, e.g.,
 - `void addCookie(Cookie cookie)`
 - `void setStatus(int sc)`
 - `void sendError(int sc)`
 - `void setContentType(String type)`
 - `void setCharacterEncoding(String charset)`
 - `ServletOutputStream getOutputStream()` and `java.io.PrintWriter getWriter()`
 - ...

- Let us make a small servlet that accepts a HTTP get request

- Let us make a small servlet that accepts a HTTP get request
- And prints the IP address of the sender of the request

- Let us make a small servlet that accepts a HTTP get request
- And prints the IP address of the sender of the request
- And sends a “Hello” back as text

- Let us make a small servlet that accepts a HTTP get request
- And prints the IP address of the sender of the request
- And sends a “Hello” back as text
- Access via `http://localhost:8080/myServlets/HelloWorld`

Listing: [HelloWorldServlet.java]: A simple servlet serving some text.

```
package myServlets;

import java.io.IOException;                import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;               import javax.servlet.http.HttpServletRequest;
import javax.servlet.ServletException;      import javax.servlet.http.HttpServlet;

public class HelloWorldServlet extends HttpServlet {//extend HTTP Servlet base class

    // this method is called by the servlet container for incoming GET requests
    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

        resp.setContentType("text/plain"); // state that we will send a normal text file
        PrintWriter out = resp.getWriter(); // get the writer to write our response

        out.write("Hello_ to_" + req.getRemoteHost() + ":" + req.getRemotePort()
            + "_ from_" + req.getLocalName() + ":" + req.getLocalPort() + ".");

    }
}
```

- Java Servlets are usually deployed in `.war` (Web application ARchive) files formats

- Java Servlets are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a `.jar` archive with a special file structure, which in turn is a `.zip` archive with a special file structure

- Java Servlets are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a `.jar` archive with a special file structure, which in turn is a `.zip` archive with a special file structure:
 - it contains a folder `WEB-INF`

- Java Servlets are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a `.jar` archive with a special file structure, which in turn is a `.zip` archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a file `web.xml`

- Java Servlets are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a `.jar` archive with a special file structure, which in turn is a `.zip` archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a 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

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

- Java Servlets are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a `.jar` archive with a special file structure, which in turn is a `.zip` archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a 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
- in `web.xml`, we specify the Servlets 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)

- Java Servlets are usually deployed in `.war` (Web application ARchive) files formats
- a `.war` is a `.jar` archive with a special file structure, which in turn is a `.zip` archive with a special file structure:
 - it contains a folder `WEB-INF`
 - the folder `WEB-INF` contains a 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
- in `web.xml`, we specify the Servlets 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]: A web.xml file for the HelloWorld Servlet

```
<?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">

  <servlet>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>myServlets.HelloWorldServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
  </servlet-mapping>

</web-app>
```

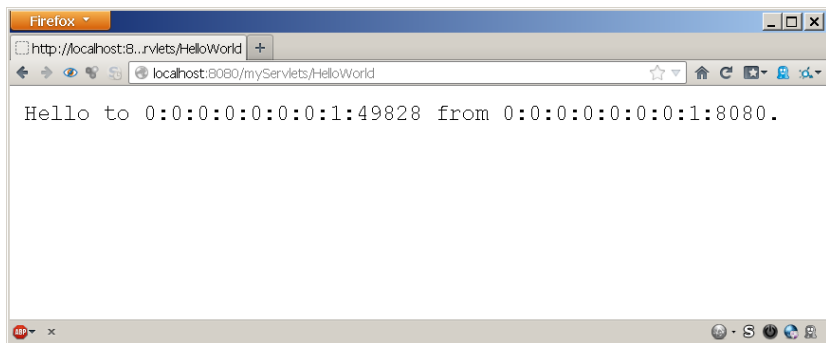

- Deploy servlet (see slides “Deploying a WAR”)

- Deploy servlet (see slides “Deploying a WAR”)
- Start servlet container (e.g., GlassFish, Tomcat, Jetty, GAE)

- Deploy servlet (see slides “Deploying a WAR”)
- Start servlet container (e.g., GlassFish, Tomcat, Jetty, GAE)
- Access the assigned URL, e.g.,
`http://localhost:8080/myServlets/HelloWorld`

- Deploy servlet (see slides “Deploying a WAR”)
- Start servlet container (e.g., GlassFish, Tomcat, Jetty, GAE)
- Access the assigned URL, e.g.,
`http://localhost:8080/myServlets/HelloWorld:`
 - where myServlets is the name of the war archive and

- Deploy servlet (see slides “Deploying a WAR”)
- Start servlet container (e.g., GlassFish, Tomcat, Jetty, GAE)
- Access the assigned URL, e.g.,
`http://localhost:8080/myServlets/HelloWorld:`
 - where `myServlets` is the name of the war archive and
 - `HelloWorld` is a servlet registered in the `web.xml` file inside `myServlets.war`



- Let us make a small servlet that accepts a HTTP get request

- Let us make a small servlet that accepts a HTTP get request
- And sends a “HTML” page back that contains all the data of the request

- Let us make a small servlet that accepts a HTTP get request
- And sends a “HTML” page back that contains all the data of the request
- Access via `http://localhost:8080/myServlets/RequestData`

Listing: [RequestDataServlet.java]: A simple servlet printing request data.

```
package myServlets;

import java.io.IOException; import javax.servlet.ServletException; import java.util.Enumeration;
import java.io.PrintWriter; import javax.servlet.http.HttpServlet; import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;

public class RequestDataServlet extends HttpServlet { //extend HTTP Servlet base class

    @Override //implement the HTTP GET request handler method
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        resp.setContentType("text/html"); // set answer format to HTML format

        PrintWriter out = resp.getWriter();
        out.println("<html><body><pre>"); // print HTML header
        out.println("Method=" + req.getMethod()); // print the used HTTP method
        out.println("URI=" + req.getRequestURI()); // print the requested URI
        out.println("RemoteAddr=" + req.getRemoteAddr()); //print client's address

        out.println("\nRequest_headers:"); // print all the request headers
        Enumeration e = req.getHeaderNames();
        while (e.hasMoreElements()) {
            String name = ((String) (e.nextElement()));
            out.println(name + "=" + req.getHeader(name));
        }

        out.println("\nForm_data:"); // print all form data/dynamic query components
        e = req.getParameterNames();
        while (e.hasMoreElements()) {
            String name = (String) (e.nextElement());
            out.println(name + "=" + req.getParameter(name));
        }

        out.println("\nCookies:"); // print all cookies
        Cookie[] cookies = req.getCookies();
        if (cookies != null) {
            for (int i = 0; i < cookies.length; i++) {
                Cookie c = cookies[i];
                out.println(c.getName() + "=" + c.getValue());
            }
        }

        Cookie cn = new Cookie("Customer", "0815"); // add a new cookie
        resp.addCookie(cn); // next time we open this page, it will be printed
        out.println("</pre></body></html>"); // print HTML footer
    }
}
```

Listing: [web.xml]: The web.xml file for the RequestDataServlet.

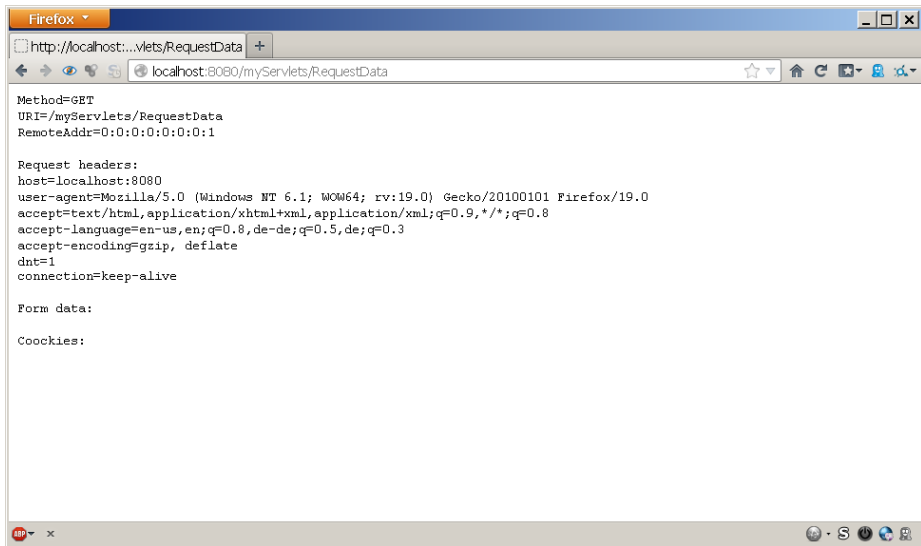
```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"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

  <servlet>
    <servlet-name>RequestData</servlet-name>
    <servlet-class>myServlets.RequestDataServlet</servlet-class>
  </servlet>

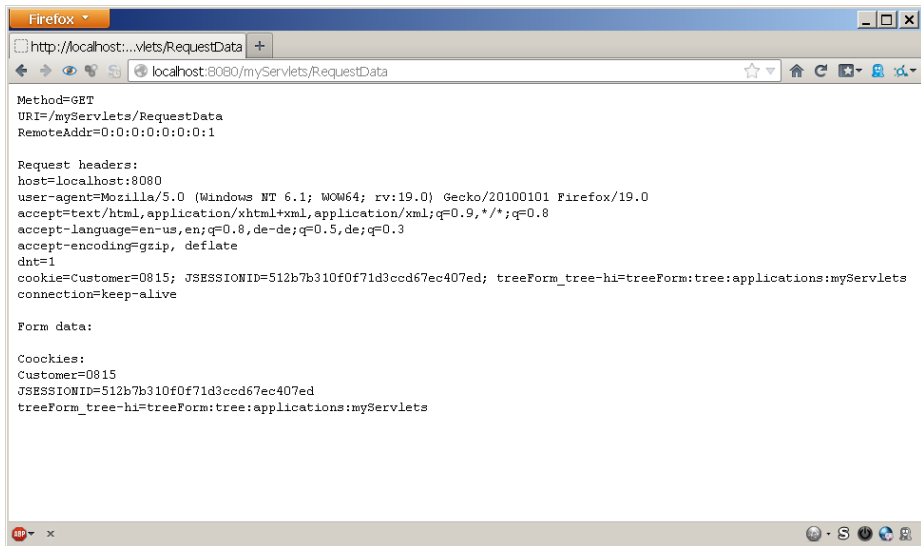
  <servlet-mapping>
    <servlet-name>RequestData</servlet-name>
    <url-pattern>/RequestData</url-pattern>
  </servlet-mapping>

</web-app>
```

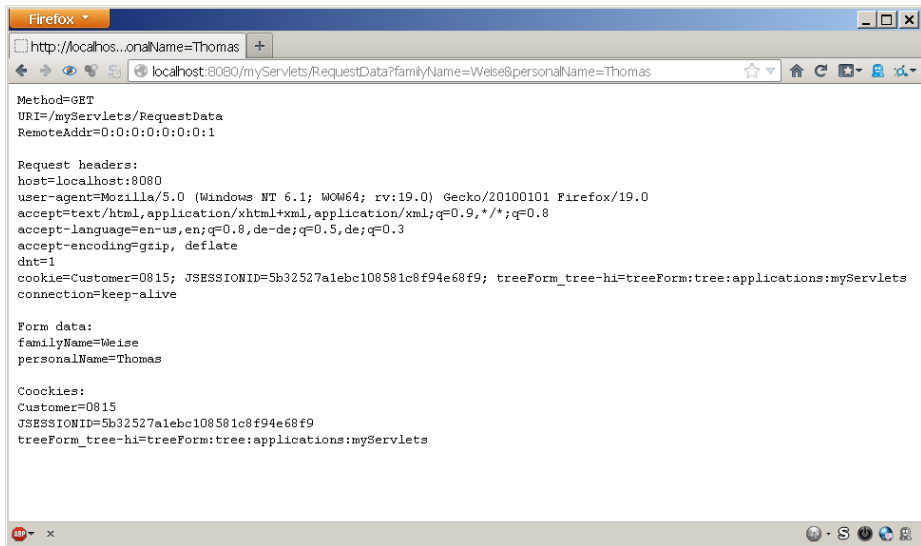
- OK, let's see what data we can get from a request the first time we access the website



- OK, let's see what data we can get from a request the first time we access the website
- ...and the second time.



- OK, let's see what data we can get from a request the first time we access the website
- ... and the second time.
- Does this also work with form data?



- OK, let's see what data we can get from a request the first time we access the website
- ... and the second time.
- Does this also work with form data?
- This means that Java Servlets are *one* way to create dynamic internet applications

- Let us make a small servlet that demonstrates session data

- Let us make a small servlet that demonstrates session data

- Let us make a small servlet that demonstrates session data
- Access via `http://localhost:8080/myServlets/SessionData`

Listing: [SessionDataServlet.java]: A simple servlet printing session data.

```
package myServlets;

import java.io.IOException; import java.util.Enumeration; import javax.servlet.http.HttpSession;
import java.io.PrintWriter; import java.util.Date; import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;

public class SessionDataServlet extends HttpServlet {//extend HTTP Servlet base class
    @Override //implement the HTTP GET request handler method
    public void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html"); // set answer format to HTML format
        PrintWriter out = res.getWriter();

        out.println("<html><body><pre>"); // print HTML header

        HttpSession s = req.getSession(true); // get/create session
        out.println("Session:␣" + s.getId()); // print session id
        out.println("created:␣" + new Date(s.getCreationTime())); //...and info
        out.println("last␣access:␣" + new Date(s.getLastAccessedTime()));

        Enumeration e = s.getAttributeNames(); //get attributes of sessions
        while (e.hasMoreElements()) { //and print them
            String name = (String) e.nextElement();
            String value = s.getAttribute(name).toString();
            out.println(name + " = " + value);
        }

        s.setAttribute("MyAttribute", "MyValue"); //set attribute "MyAttribute" of session
        s.setAttribute("MyAttribute2", s.getLastAccessedTime());

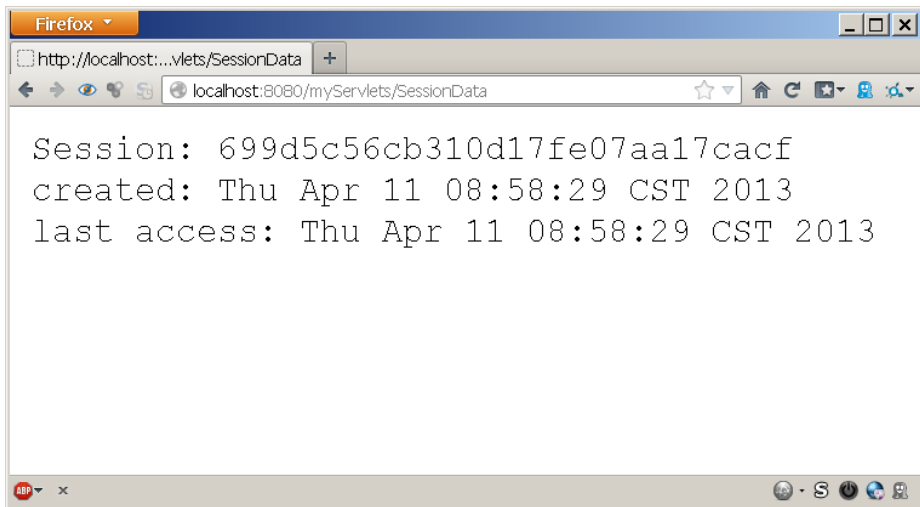
        out.println("</pre></body></html>"); // print HTML footer
    }
}
```

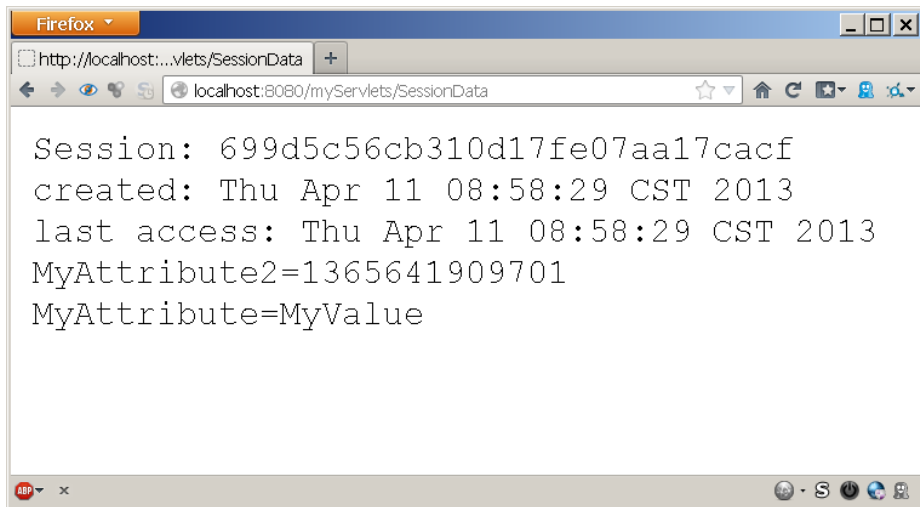
Listing: [web.xml]: The web.xml file for the SessionDataServlet.

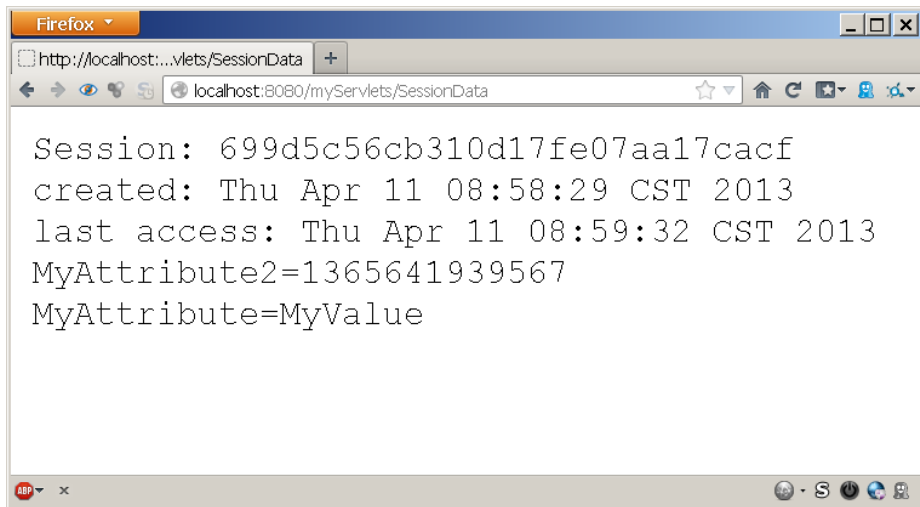
```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"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

  <servlet>
    <servlet-name>SessionData</servlet-name>
    <servlet-class>myServlets.SessionDataServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>SessionData</servlet-name>
    <url-pattern>/SessionData</url-pattern>
  </servlet-mapping>
</web-app>
```







- OK, now we know how to use Java Servlets ... but what is this good for?
- Java Servlets and similar techniques are the backbone of enterprise computing and dynamic websites
- All web servers are based on HTTP and servlets provide an easy way to access this protocol
- We can use servlets to dynamically create websites (this is how JavaServer Pages work, see next lecture)
- Web Services use mainly HTTP as well and most Web Service implementations are, actually, servlets
- Platform-as-a-Service cloud structures (such as the Google App Engine ^[12–14]) often allow you to deploy servlets

- We already knew. . .
 - HTML is the basic language in which web pages are developed.
 - URLs are the “addresses” of web pages in the internet.
 - HTTP protocol is the backbone of the WWW and many enterprise software architectures.
 - We learned how to implement HTTP via TCP sockets in a parallel way.
 - HTML forms: send data from browser (client) to web server.

- We already knew. . .
 - HTML is the basic language in which web pages are developed.
 - URLs are the “addresses” of web pages in the internet.
 - HTTP protocol is the backbone of the WWW and many enterprise software architectures.
 - We learned how to implement HTTP via TCP sockets in a parallel way.
 - HTML forms: send data from browser (client) to web server.
- Now we learned. . .
 - Java Servlets as API for accessing HTTP under Java
 - Inherently dynamic way to create information/answer requests
 - GlassFish as example/reference architecture for Enterprise Edition implementation of Java Servlet container.

- 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^[15–17]

- 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^[15–17]
- 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^[15–17]
 - 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."*

²For our later lesson on 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**²

²For our later lesson on 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**²
- Open the file
`{GLASSFISH_DIR}\glassfish4\glassfish\config\asenv.bat`

²For our later lesson on 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**²
- Open the file
`{GLASSFISH_DIR}\glassfish4\glassfish\config\asenv.bat`
- Find the entry "set AS_JAVA=..."

²For our later lesson on 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**²
- 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"`)

²For our later lesson on 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**²
- 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"

²For our later lesson on JavaServer Pages, a **JRE** (Java Runtime Environment) is not enough – it must be a **IDE** (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

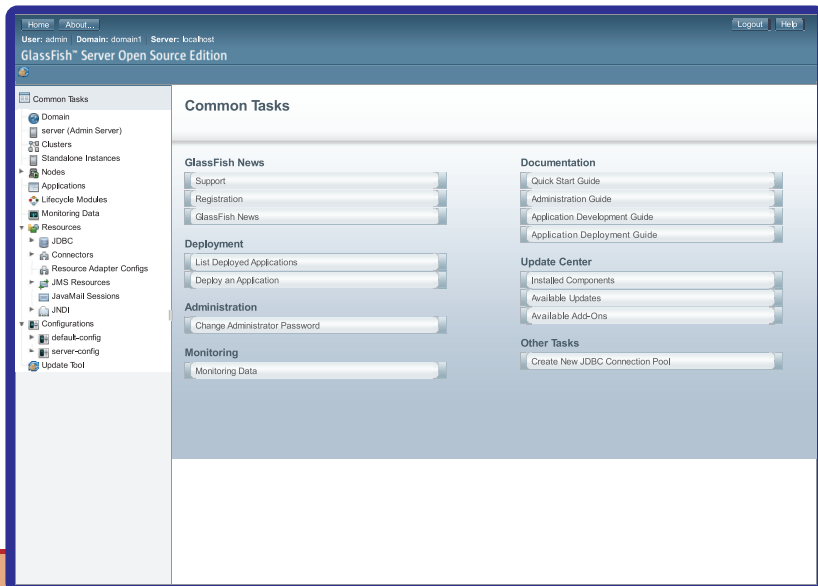
Welcome to GlassFish Server Open Source Edition 3.1.2.2 (build 5).



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

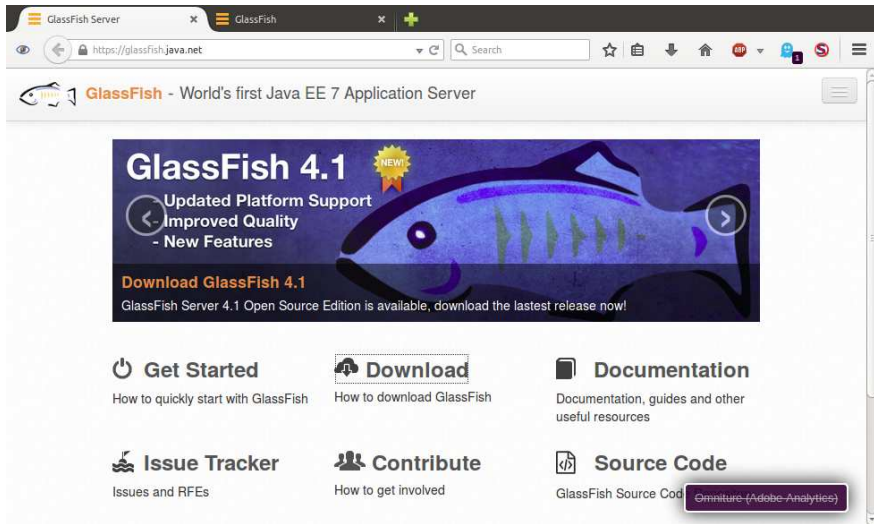
Use thread pools to limit a service to a specific number of concurrent threads.

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 shows "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 text stating "GlassFish Server 4.1 Open Source Edition is available, download the latest release now!". The page is organized into a grid of six sections: "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

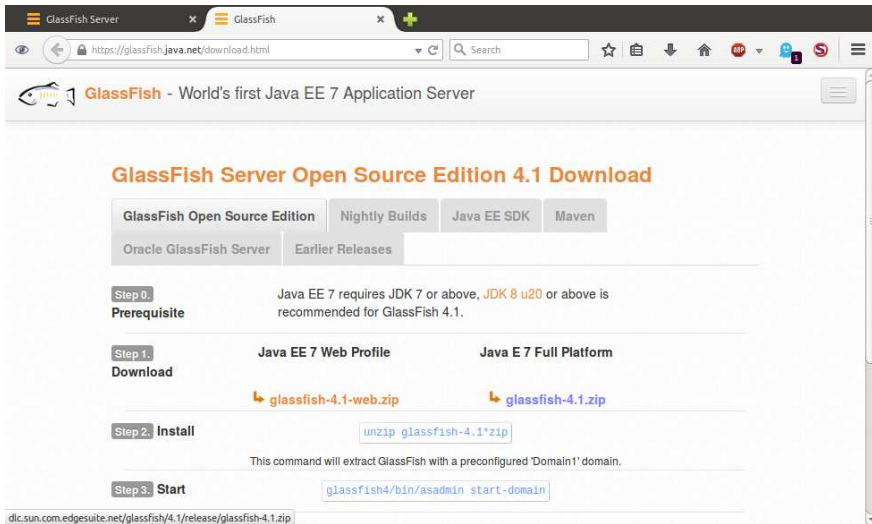
Documentation
Documentation, guides and other useful resources

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 the address bar displaying `https://glassfish.java.net/download.html`. The page title is "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".

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

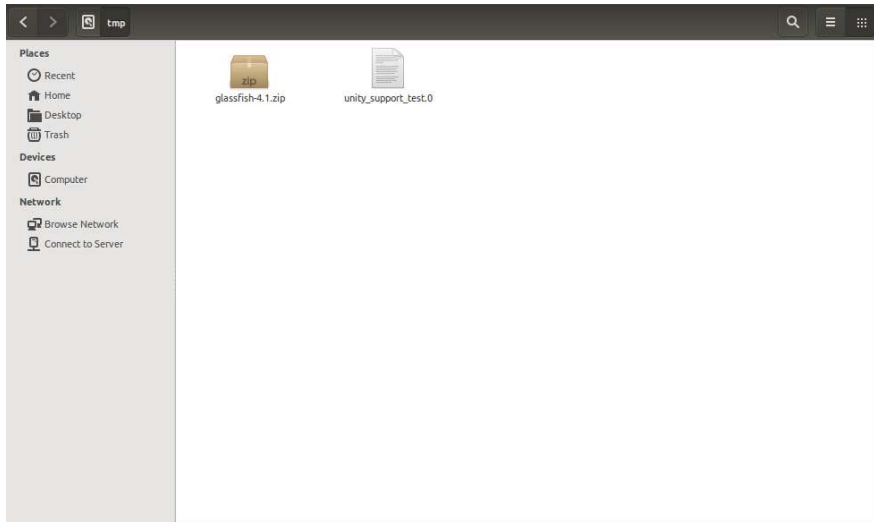
Step 1. Download

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`

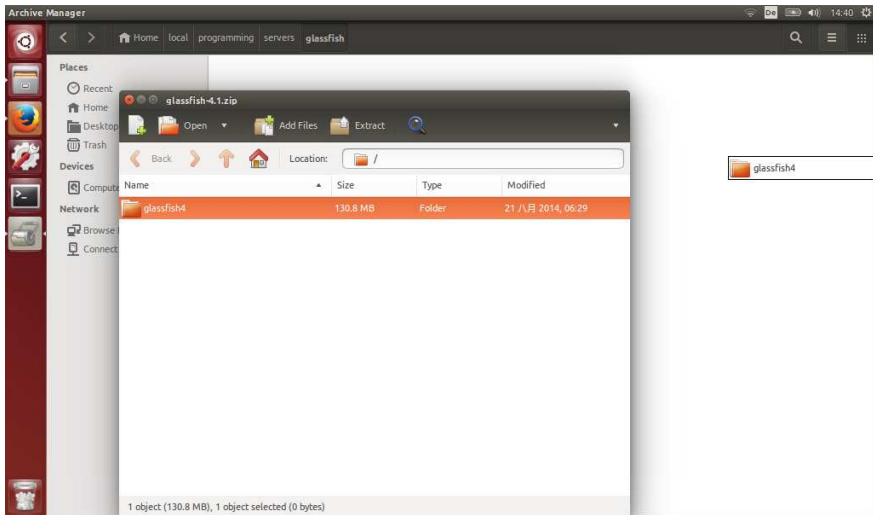
dlc.sun.com.edgesuite.net/glassfish/4.1/release/glassfish-4.1.zip



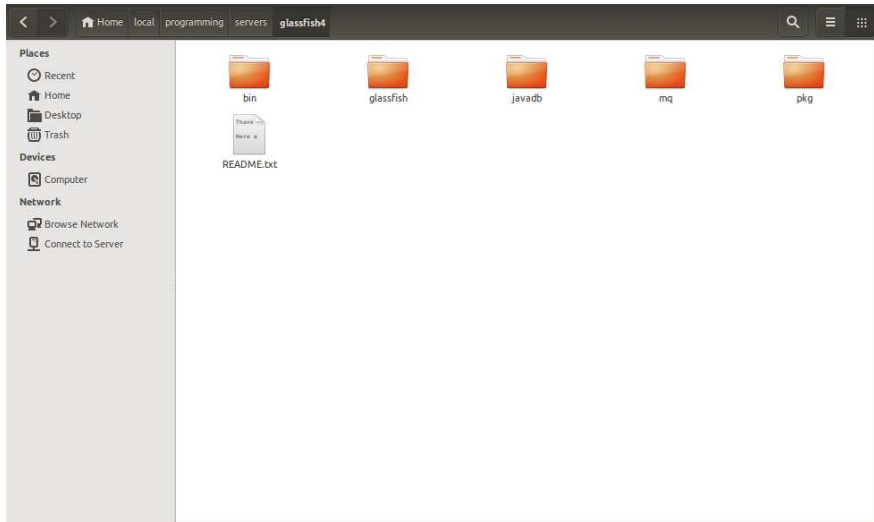
Installing GlassFish (Linux)



Installing GlassFish (Linux)



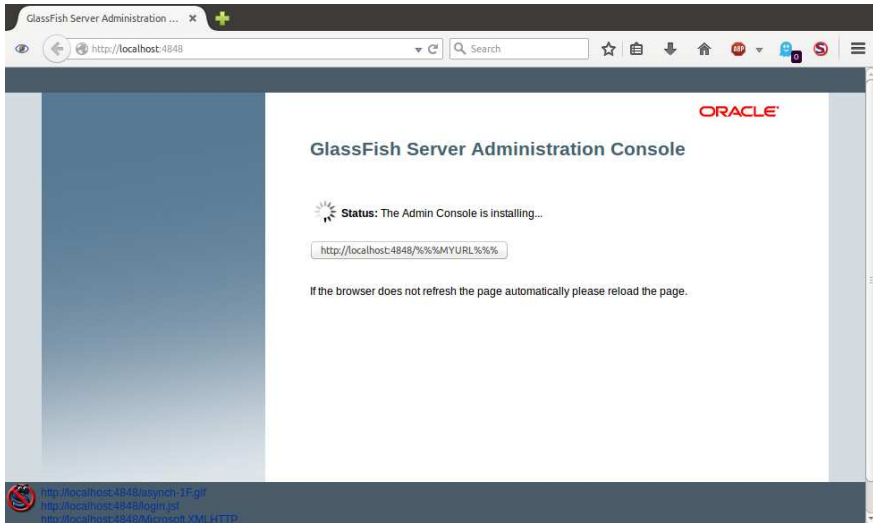
Installing GlassFish (Linux)

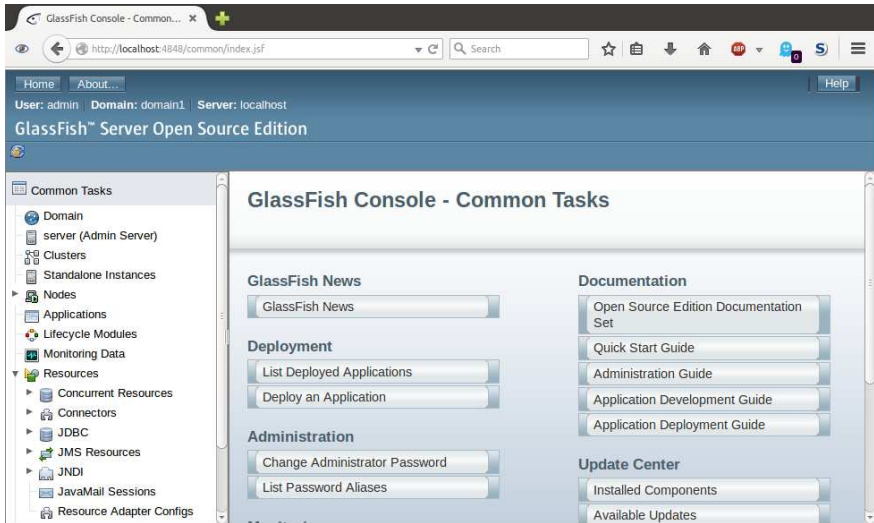


```
tweise@xiao: ~/local/programming/servers/glassfish4/bin
tweise@xiao:~/local/programming/servers/glassfish4/bin$ ./asadmin start-domain
Waiting for domain1 to start .....
```



```
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 displays `http://localhost:4848/common/index.jsf`. The page header includes navigation links for **Home** and **About...**, and a **Help** button. Below the header, the user information is shown: **User: admin**, **Domain: domain1**, and **Server: localhost**. The main title is **GlassFish™ Server Open Source Edition**.

The interface is divided into a left sidebar and a main content area. The sidebar, titled **Common Tasks**, contains a tree view of the system structure:

- 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

The main content area, titled **GlassFish Console - Common Tasks**, features several sections with buttons for various actions:

- 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>javaServlets</artifactId>
  <version>0.8.0</version>
  <packaging>war</packaging>
  <name>Java Servlets Examples</name>
  <description>Examples for using Java Servlets (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>
</dependencies>
```



Listing: [pom.xml] – Part 8: Build

```
<build>
  <finalName>myServlets</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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 (`Java Servlets`) 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 `Java Servlets` 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 `Java Servlets` 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 `Java Servlets` 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 `Java Servlets` 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 `Java Servlets` 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.

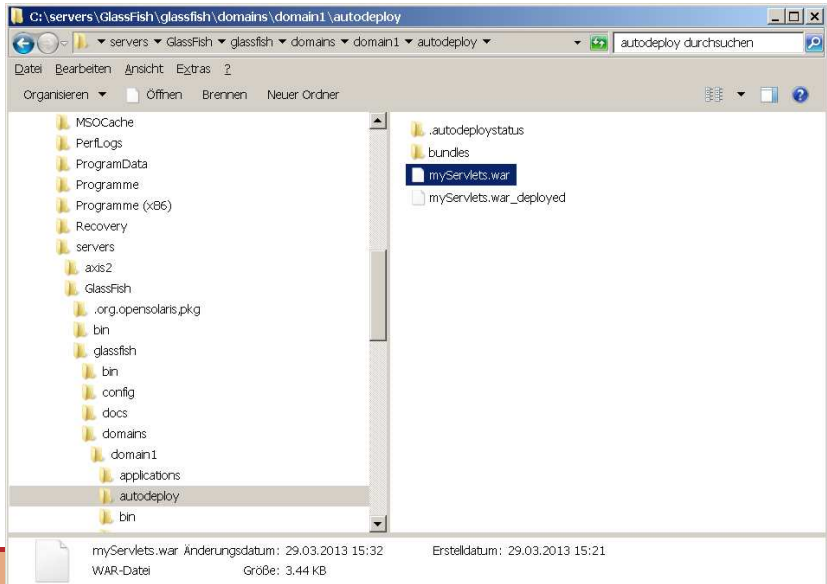
- A WAR (Web application ARchive) is a ZIP archive with a special file structure that can be deployed to a servlet container
- Under `Goals`, enter `clean compile war:war`. This will build a war archive.
- Click `Apply`
- Click `Run`
- The build will start, you will see its status output in the console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



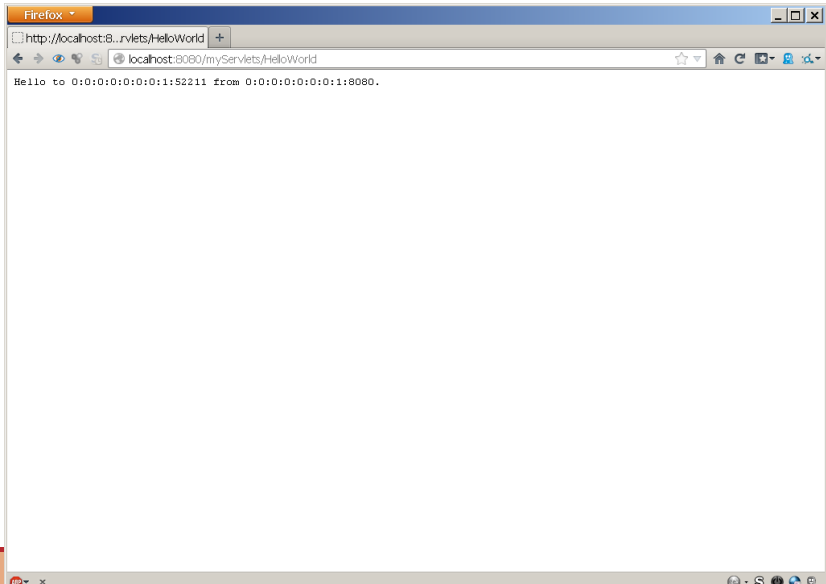
- Deploying a WAR archive is easy:
 - Copy it into the folder
`{GLASSFISH_DIR}\glassfish4\glassfish\domains\domain1\autodep`
 - The running GlassFish server will automatically load and start it.
- You can now access the application under
`http://localhost:8080/myServlets/HelloWorld`

- 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. Timothy John Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer Protocol – HTTP/1.0*, volume 1945 of *Request for Comments (RFC)*. Network Working Group, May 1996. URL <http://tools.ietf.org/html/rfc1945>.
3. R. Fielding, J. Gettys, Jeffrey Mogul, H. Frystyk, L. Masinter, P. Leach, and Timothy John Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, volume 2616 of *Request for Comments (RFC)*. Network Working Group, June 1999. URL <http://tools.ietf.org/html/rfc2616>.
4. 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>.
5. 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>.
6. 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>.
7. Jason Brittain and Ian F. Darwin. *Tomcat: The Definitive Guide*. Definitive Guide. Sebastopol, CA, USA: O'Reilly Media, Inc., 2007. ISBN 0596101066 and 9780596101060. URL <http://books.google.de/books?id=vJttHyVF0SUC>.
8. *Apache Tomcat*. Forest Hill, MD, USA: Apache Software Foundation, 1999. URL <https://tomcat.apache.org/>.
9. *Jetty WebServer*. Fortitude Valley BC, QLD, Australia: codehaus foundation and Riverview, NSW, Australia: Mort Bay Consulting Pty. Ltd, 1995. URL <http://jetty.codehaus.org/jetty/>.
10. Francesco Marchioni. *Jboss as 7 Configuration, Deployment and Administration*. Community Experience Distilled. Birmingham, UK: Packt Publishing Limited, 2011. ISBN 1849516790 and 9781849516792. URL http://books.google.de/books?id=av0NaXtoV_QC.
11. Tom Marrs and Scott Davis. *JBoss at Work: A Practical Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2009. ISBN 0596552939 and 9780596552930. URL <http://books.google.de/books?id=NiI3Jd8p4mcC>.
12. Google app engine, 2012. URL <https://developers.google.com/appengine/>.
13. Dan Sanderson. *Programming Google App Engine*. Animal Guide. Sebastopol, CA, USA: O'Reilly Media, Inc., 2009. ISBN 059652272X and 9780596522728. URL http://books.google.de/books?id=6cL_kCZ4NJ4C.
14. Eugene Ciurana. *Developing With Google App Engine*. Berlin, Germany: Springer-Verlag GmbH, 2009. ISBN 1-4302-1831-2 and 978-1-4302-1831-9. URL <http://books.google.de/books?id=ks9HQLVxa>.

15. *Glassfish*. Redwood Shores, CA, USA: Oracle Corporation, revision 20130208.5d7f765 edition, 2013. URL <http://glassfish.java.net/>.
16. 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>.
17. 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>.
18. John Clingan. Java ee and glassfish server roadmap update. *The Aquarium: News from the GlassFish Community*, November 4, 2013. URL https://blogs.oracle.com/theaquarium/entry/java_ee_and_glassfish_server.
19. *Oracle WebLogic Server*. Redwood Shores, CA, USA: Oracle Corporation, 2014. URL <http://www.oracle.com/us/products/middleware/cloud-app-foundation/weblogic/overview/index.html>.
20. Xinyu Liu. Asynchronous processing support in servlet 3.0: Why asynchronous processing is the new foundation of web 2.0. *JavaWorld: Solutions for Java Developers*, February 19, 2009. URL <http://www.javaworld.com/javaworld/jw-02-2009/jw-02-servlet3.html>.