Session 1, Introduction to Basic Java Web Application

Objectives

Introduction to java web application

+ The core and basic of Java web server technologies

+ Web design vs server technologies

Setup Environment :

JDK: 1.7 or higher

Servlet container: Tomcat 7, Glassfish 4.1, etc.

Intergrate Netbeans 8. with the web container.

Creating/Building the first application:

Learn to create Servlet

Learn to create JSP

Deploy the web application

What is HTML?

HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- HTML uses **markup tags** to describe web pages

HTML Tags

HTML markup tags are usually called HTML tags

- HTML tags are keywords surrounded by **angle brackets** like <html>
- HTML tags normally **come in pairs** like <b> and </b>
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- Start and end tags are also called **opening tags** and **closing tags**.

HTML Documents = Web Pages

- HTML documents **describe web pages**
- HTML documents **contain HTML tags** and plain text
- HTML documents are also **called web pages**

Web browser

- The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages.
- The browser does not display the HTML tags, but uses the tags to **interpret** the content of the page

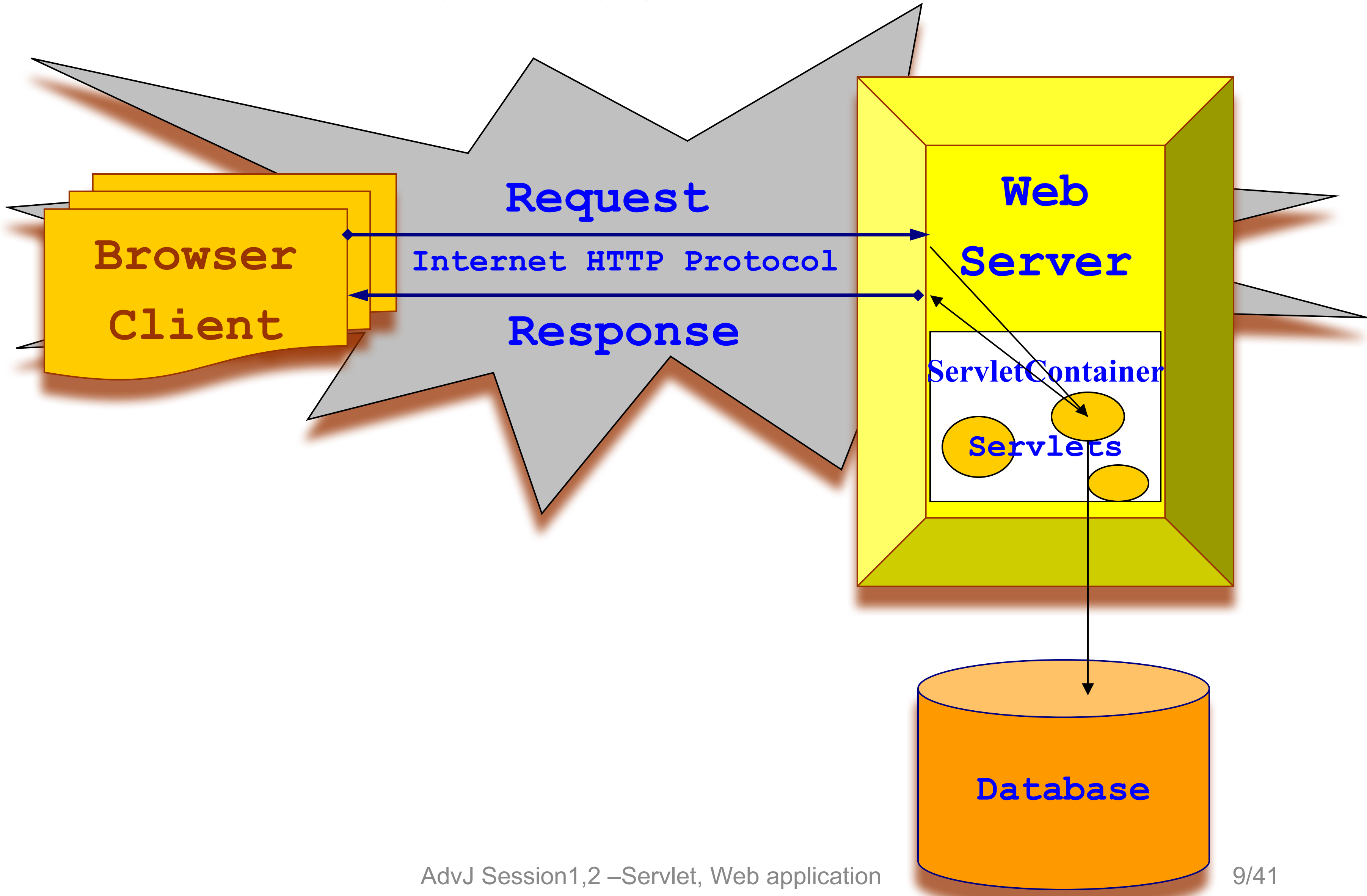
Web page example

```
<html>
	<body>

		<h1>My First Heading</h1>

		<p>My first paragraph</p>

		<a href=“http:\\cms.fpt.edu.vn”>Studying place</a>
	</body>
</html>
```

What a Servlet is

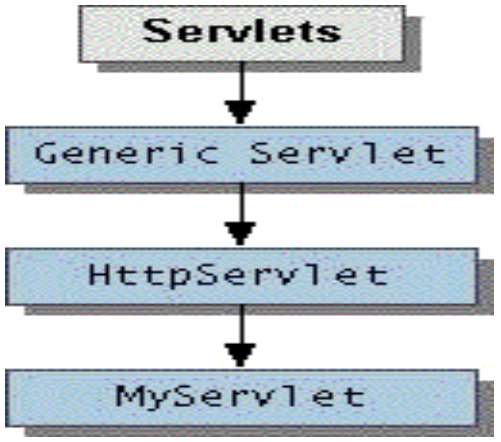
- Servlets are <u>small Java programs that run on a Web server</u> and help to build dynamic Web pages.
- Servlets <u>receive</u> and <u>respond</u> to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.
- Java Servlet technology was created as a portable way to provide <u>dynamic</u>, <u>user-oriented content</u>.

What a Servlet is…



Architecture of the Servlet Package

- The `javax.servlet` package provides interfaces and classes for writing servlets

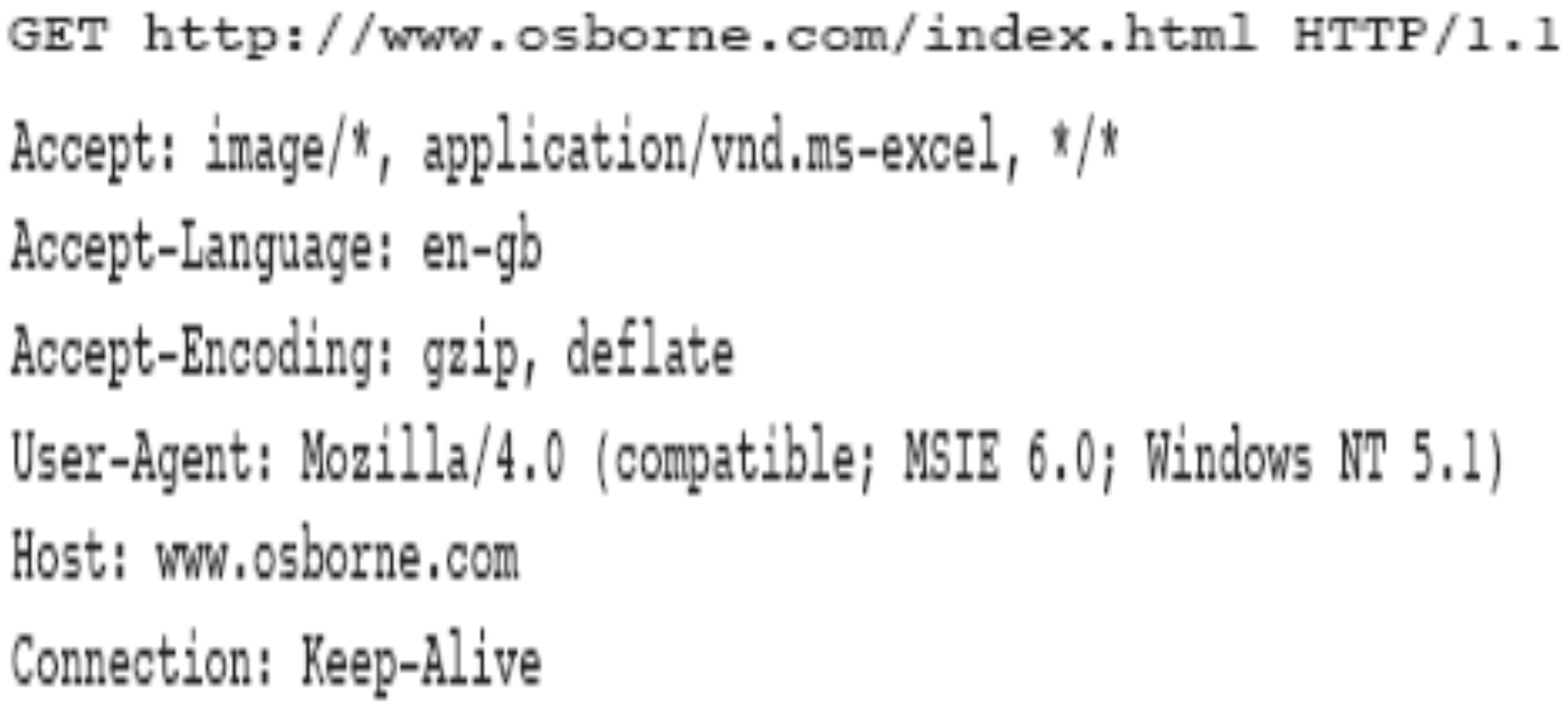


- **When a servlet accepts a call from a client, it receives two objects:**
  - ***ServletRequest*, which encapsulates the communication from the client to the server.**
  - ***ServletResponse*, which encapsulates the communication from the servlet to the client.**

HTTP Requests

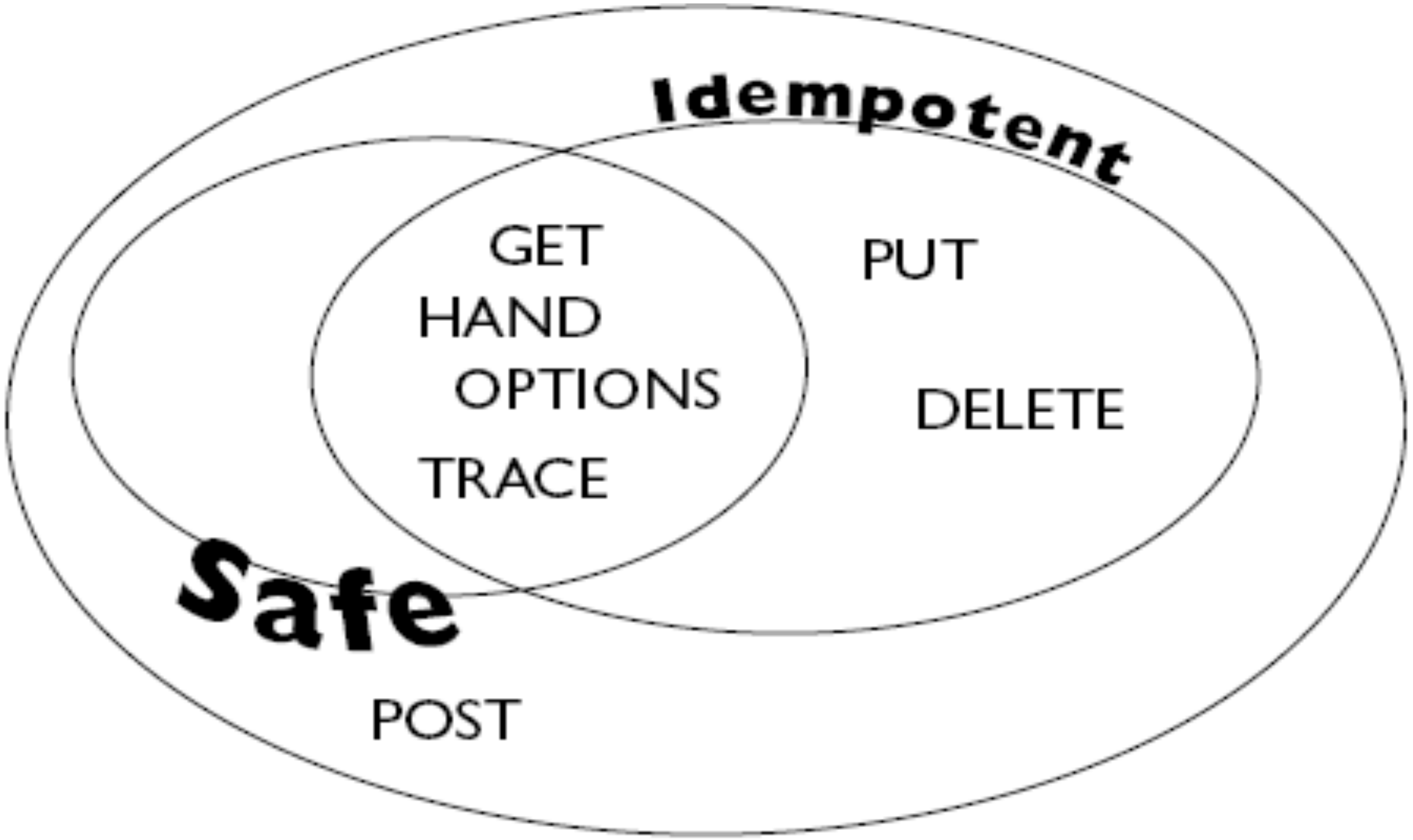
- An HTTP request consists of
  - ***a request method***
  - ***a request URL***
  - ***header fields***
  - ***body.***
- HTTP 1.1 defines the following request methods:
  - **GET** - retrieves the resource identified by the request URL.
  - **HEAD** - returns the headers identified by the request URL.
  - **POST** - sends data of unlimited length to the web server.
  - **PUT** - stores a resource under the request URL.
  - **DELETE** - removes the resource identified by the request URL.
  - **OPTIONS** - returns the HTTP methods the server supports.
  - **TRACE** - returns the header fields sent with the TRACE request.

HTTP Request Sample



```
GET http://www.osborne.com/index.html HTTP/1.1
Accept: image/*, application/vnd.ms-excel, */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: www.osborne.com
Connection: Keep-Alive
```

Idempotency and Safety Methods



HTTP Responses

- An HTTP response contains
  - ***a result code***
  - ***header fields***
  - ***a body***
- The HTTP protocol expects the result code and all header fields to be returned before any body content
- Some commonly used status codes include:
  - **404** ***- indicates that the requested resource is not available.***
  - **401** ***- indicates that the request requires HTTP authentication.***
  - **500** ***- indicates an error inside the HTTP server which prevented it from fulfilling the request.***
  - **503** ***- indicates that the HTTP server is temporarily overloaded, and unable to handle the request.***

FPT UNIVERSITY
DREAM OF INNOVATION

HTTP 1.1 Status Codes

101 *Switching Protocols*

Server will comply with Upgrade header and change to different protocol. (New in HTTP 1.1)

200 *OK*

Everything's fine; document follows for GET and POST requests. This is the default for servlets; if you don't use setStatus, you'll get this.

201 *Created*

Server created a document; the Location header indicates its URL.

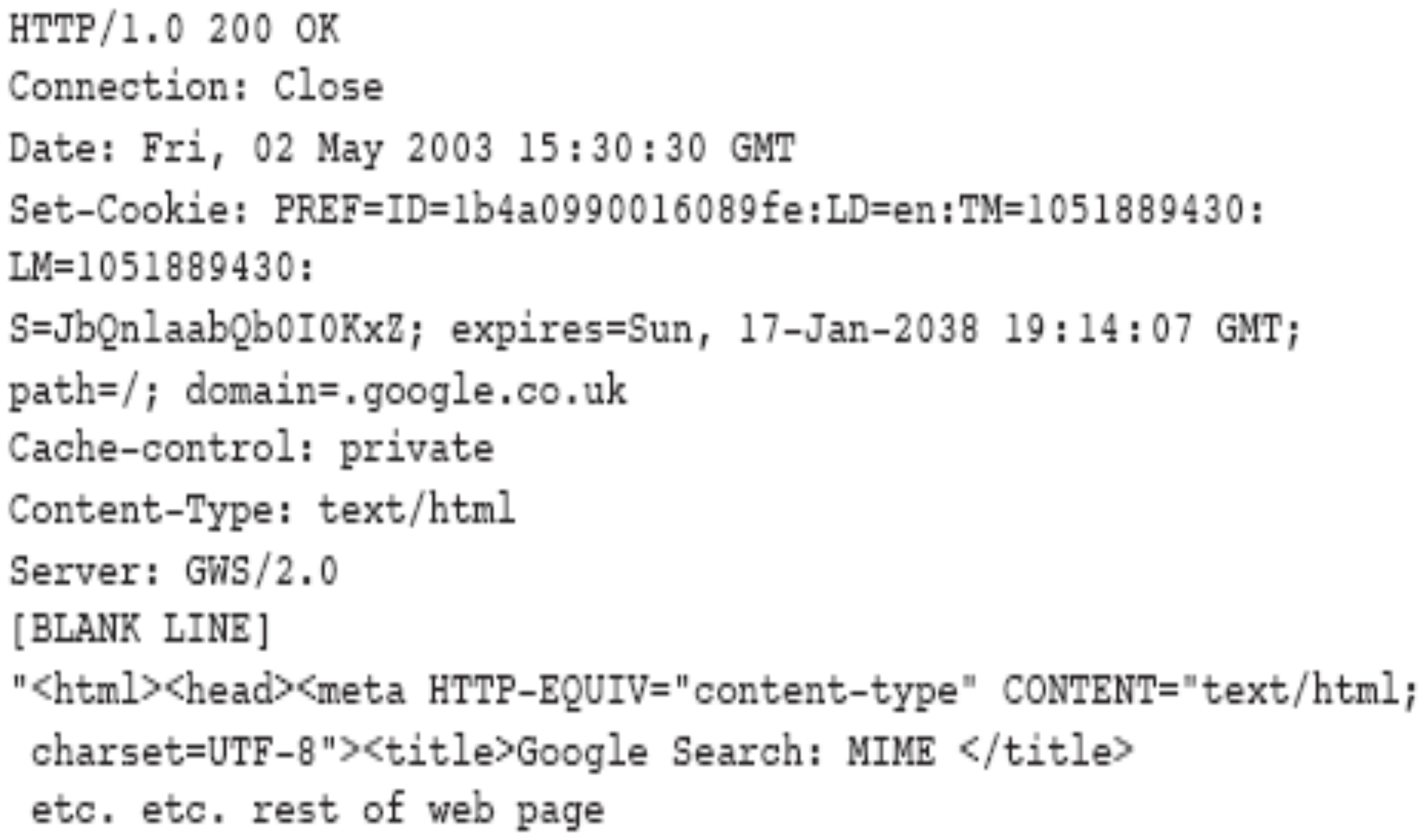
202 *Accepted*

Request is being acted upon, but processing is not completed.

203 *Non-Authoritative Information*

Document is being returned normally, but some of the response headers might be incorrect since a document copy is being used.

HTTP Response Sample



```
HTTP/1.0 200 OK
Connection: Close
Date: Fri, 02 May 2003 15:30:30 GMT
Set-Cookie: PREF=ID=1b4a0990016089fe:LD=en:TM=1051889430:
LM=1051889430:
S=JbQnlaabQb0I0KxZ; expires=Sun, 17-Jan-2038 19:14:07 GMT;
path=/; domain=.google.co.uk
Cache-control: private
Content-Type: text/html
Server: GWS/2.0
[BLANK LINE]
"<html><head><meta HTTP-EQUIV="content-type" CONTENT="text/html;
 charset=UTF-8"><title>Google Search: MIME </title>
 etc. etc. rest of web page
```

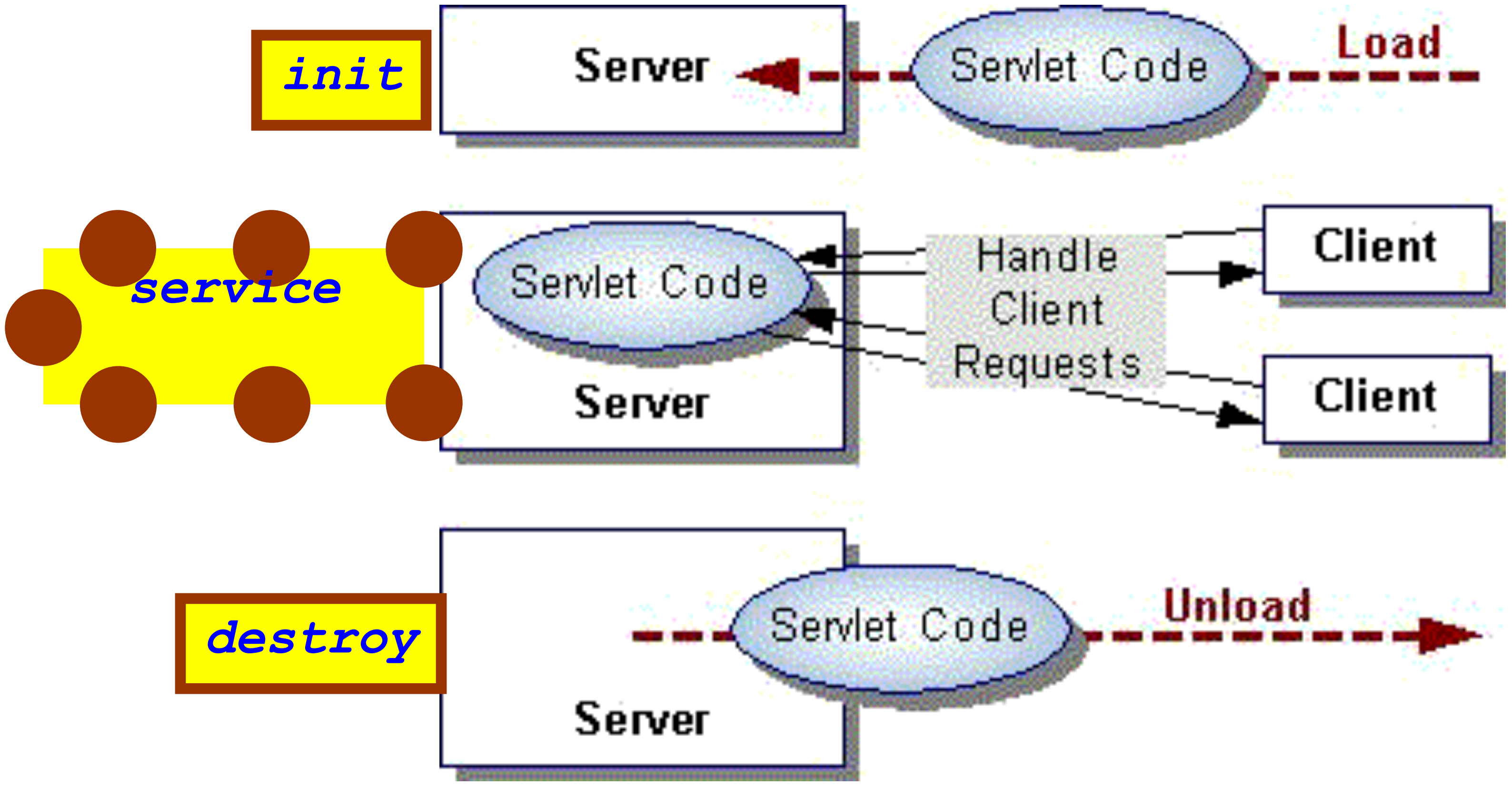
A Simple Servlet

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

public class FirstServlet extends HttpServlet {

  public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException
   {
     PrintWriter out = response.getWriter();
     out.println(“<h1>First Servlet</h1>");
   }
}
```

First Sevlet Demo

Demo\FirstServlet\WEB-INF\classes\FirstServlet.java

The Servlet Life Cycle



HttpServlet Class

- The protocol defines a set of text-based request messages called HTTP ‘methods’ implemented in *HttpServlet* class:
  - ***doGet*** Called by the server (via the service method)to allow a servlet to handle a GET request
  - **doHead** Receives an HTTP HEAD request from the protected service method and handles the request
  - **doPost** called by the server to allow a servlet to handle post request
  - **doPut** Called by the server (via the service method) to allow a servlet to handle a PUT request
  - **doDelete** Called by the server (via the service method) to allow a servlet to handle a DELETE request
  - **doTrace** Called by the server (via the service method) to allow a servlet to handle a TRACE request
  - **doOptions** Called by the server (via the service method) to allow a servlet to handle a OPTIONS request

Request Headers

- It can also send a number of headers:
  - Accept The MIME types the browser prefers.
  - Accept-Charset The character set the browser expects.
  - Content-Length (for POST messages, how much data is attached)
  - Connection Use persistent connection? If a servlet gets a Keep-Alive.
  - Cookie (one of the most important headers)
  - Host (host and port as listed in the original URL)
  - If-Modified-Since (only return documents newer than this)
  - Referer (the URL of the page containing the link the user followed to get to current page)

Request Headers…

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

public class ShowRequestHeaders extends HttpServlet {
  public void doGet(HttpServletRequest request,HttpServletResponse response) throws
      ServletException, IOException
  {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            Enumeration headerNames = request.getHeaderNames();
            out.println("<TABLE>");
            while(headerNames.hasMoreElements()) {
                String headerName = (String)headerNames.nextElement();
                out.println("<TR><TD>" + headerName+"</TD>");
                out.println("<TD>" + request.getHeader(headerName)+"</TD></TR>");
            }
            out.println("</TABLE>");
      }
}
```

Show Request Header Demo

Demo\FirstServlet\WEB-INF\classes\ShowRequestHeaders.java

HTTP Response

- When a Web server responds to a request from Web client, the response typically consists of a status line, some response headers, a blank line, and the document:

HTTP/1.1 200 OK	status line
Content-Type: text/plain	response header
	blank line
Welcome to Servlets World	the document

Status line:

– **HTTP version**
– **An integer that is interpreted as a status code**
– **A very short message corresponding to the status code.**

– **In most cases, all of the headers are optional except for Content-Type, which specifies the MIME type of the document that follows**

HTML Form

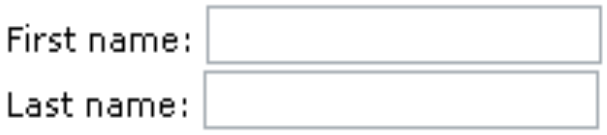
- A form is an area that can contain form elements.
- Form elements are elements that allow the user to enter information (like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.) in a form.

HTML Form Inputs

- Text fileds

```
<html>
<body>
    <form>
        First name: <input type="text" name="firstname" />
        <br />
        Last name: <input type="text" name="lastname" />
    </form>
</body>
<html>
```

How it looks in a browser:


First name:
Last name:

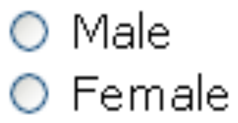
DREAM OF INNOVATION

HTML Form Inputs..

Radio Buttons:

```
<html>
<body>
      <form>
            <input type="radio" name="sex" value="male" /> Male
            <br />
            <input type="radio" name="sex" value="female" /> Female
</form>
</body>
<html>
```

How it looks in a browser:



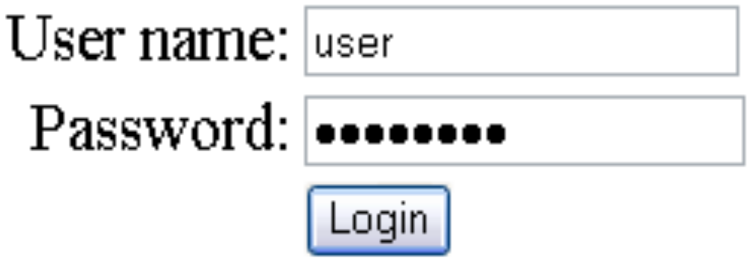
The Form's Action Attribute and the Submit Button

- When the user clicks on the "Submit" button, the content of the form is sent to the server. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

The Form's Action Attribute and the Submit Button…

```
<html>
<body>
    <form name="input" action="/LoginServlet" method="post">
    <table>
          <tr>
            <td>User name:</td><td><input type="text" name="user"/></td>
          </tr>
          <tr>
            <td>Password:</td><td><input type="password" name="pass"/></td>
          </tr>
          <tr>
            <td></td><td><input type="submit" value="Login"/></td>
          </tr>
    </table>
    </form>
</body>
<html>
```

The Form's Action Attribute and the Submit Button…


User name: user

Password: ●●●●●●●●●

Form Data

- Call `getParameter` method of the `HttpServletRequest`, supplying the parameter name as an argument

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                                        throws ServletException, IOException
 {
            //get form data
             String u =request.getParameter("user");
             String p=request.getParameter("pass");

             response.setContentType("text/html");
             PrintWriter out = response.getWriter();
             out.println("<html><body>");
             out.println("<h1>You sent me:</h1>");
             out.println(u+"<br>"+p);
             out.println("</body></html>");
 }
```

Web application

- A **web application** or **webapp** is an application that is accessed via web browser over a network such as the Internet or an intranet. It is also a computer software application that is coded in a browser-supported language (such as HTML, JavaScript, Java, etc.) and reliant on a common web browser to render the application executable.
- Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client.

File and Directory Structure

- A Place for Everything and Everything in Its Place.
- *Construct the file and directory structure of a Web Application that may contain:*
  - *static content,*
  - *JSP pages,*
  - *servlet classes,*
  - *the deployment descriptor,*
  - *tag libraries,*
  - *JAR files and Java class files;*
  - *and describe how to protect resource fi les from HTTP access.*

Special Directories Beneath the Context Root

- /WEB-INF/classes—for classes that exist as separate Java classes (*not* packaged within JAR files). These might be servlets or other support classes.
- /WEB-INF/ lib—for JAR fi les. These can contain anything at all—the main servlets for your application, supporting classes that connect to databases—whatever.
- / WEB-INF itself is the home for an absolutely crucial file called web.xml, the deployment descriptor file.

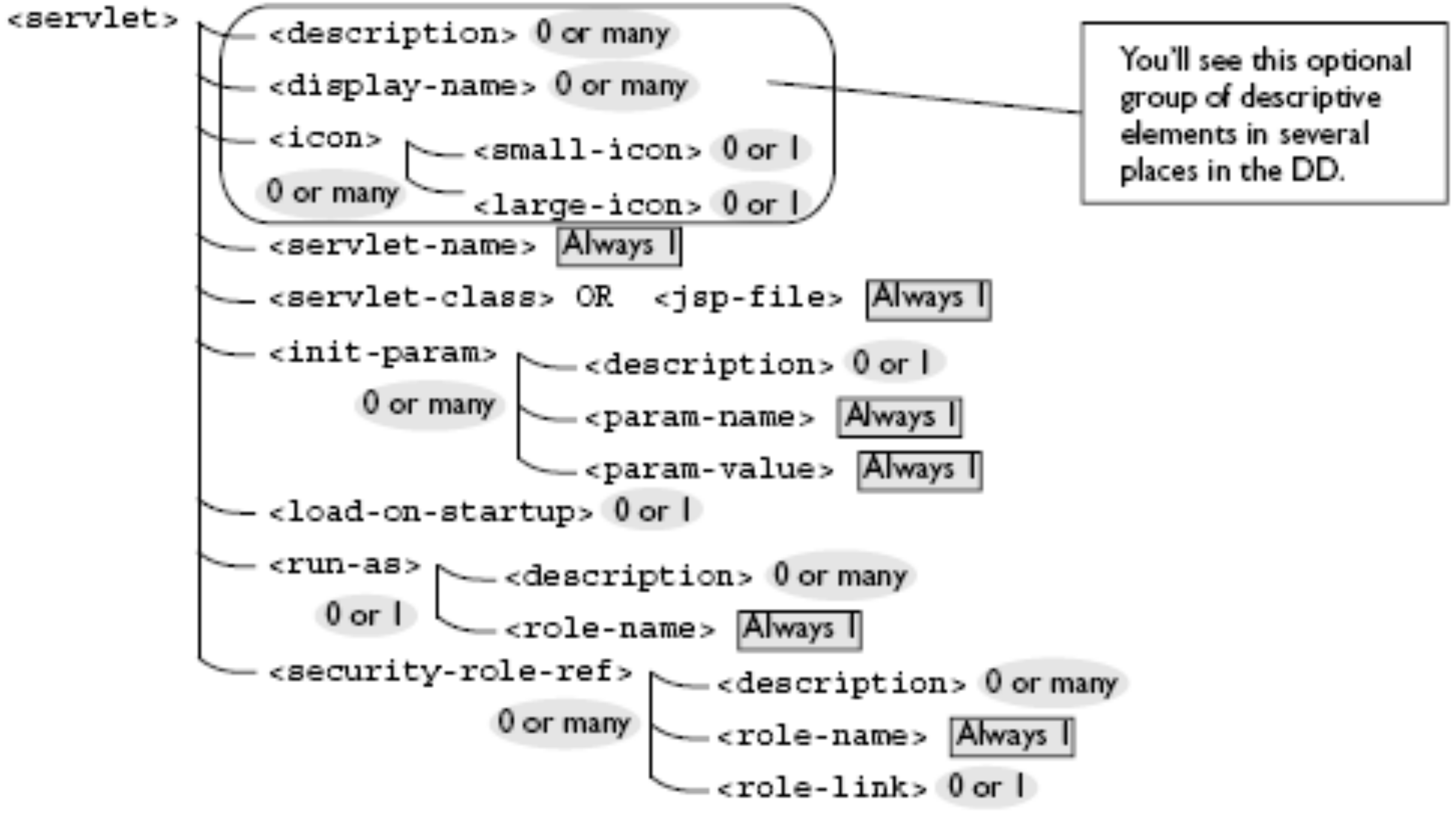
Deployment Descriptor Elements

- The first thing to note about the deployment descriptor file is that it’s an XML file. Given that the name is web.xml.

Overall Structure of the Deployment Descriptor.

- <web-app>
  - <description>
  - <display-name>
  - <icon>
  - <distributable>
  - <context-param>
  - <filter>
  - <filter-mapping>
  - <listener>
  - **<servlet>**
  - **<servlet-mapping>**
  - <session-config>
  - **<mime-mapping>**
  - **<welcome-file-list>**
  - **<error-page>**
  - <jsp-config>
  - <security-constraint>
  - <login-config>
  - <security-role>

<servlet> and Its Important Subelements



Deployment Descriptor simple web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd
      version="2.4">

   <display-name>Servlet 2.4 Examples</display-name>
   <description>
    Servlet 2.4 Examples.
   </description>
   <!-- Define servlets that are included in the example application -->
   <servlet>
      <servlet-name>FirstServlet</servlet-name>
      <servlet-class>FirstServlet</servlet-class>
   </servlet>

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

Welcome Files

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>mainlibrary/catalog.jsp</welcome-file>
</welcome-file-list>
```

Packaging Your Web Application

- **A WAR Is Not a JAR**
  - Although a WAR fi le can be produced in the same way as a JAR fi le, and has the same underlying fi le format, it is different. The most obvious difference is the file extension naming convention: .jar for Java ARchive, and .war for Web (Application) ARchive.
  - WARs are packaged for a different purpose: to make it as easy as possible for a web container to deploy an application.

WAR file

- Several web containers have automatic deployment mechanisms.
- The server recommended for this course—Tomcat 7.x or Glassfish 4.x—has a “**webapps**” directory.
    - Place a WAR file in this directory, and Tomcat (by default) will un-jar the contents into the file system under the **webapps** directory.
    - a **context root** directory is the same **name as the WAR file** (but without the .war extension)— then makes the application available for use.

War file demo

Demo\WarFile\FirstServlet.war

Summary

HTML Introduction

- What is HTML?
- HTML Tags
- HTML Documents = Web Pages
- Web browser
- Example

Servlets

- What a Servlet is and how you can use one.
- How to define and write servlets.
- Basic Servlet Structure.
- Request / Response Headers.
- Handling Form Data.
- Java Servlet Specification
- Jakarta-tomcat-7.x or Glassfish 4.x

Web application

- File and Directory Structure
- Deployment Descriptor Elements
- WAR Files