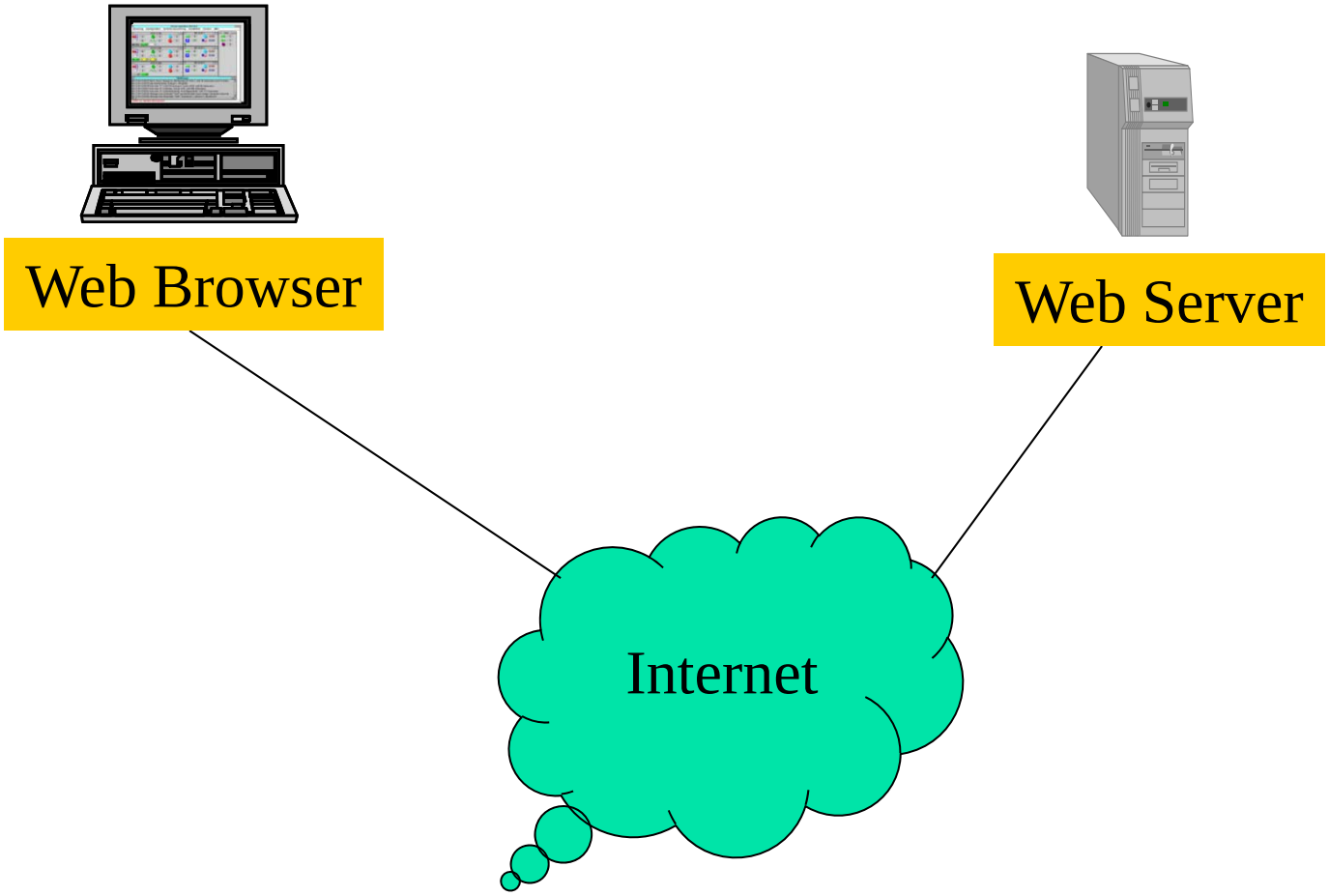
The Enterprise Technologies

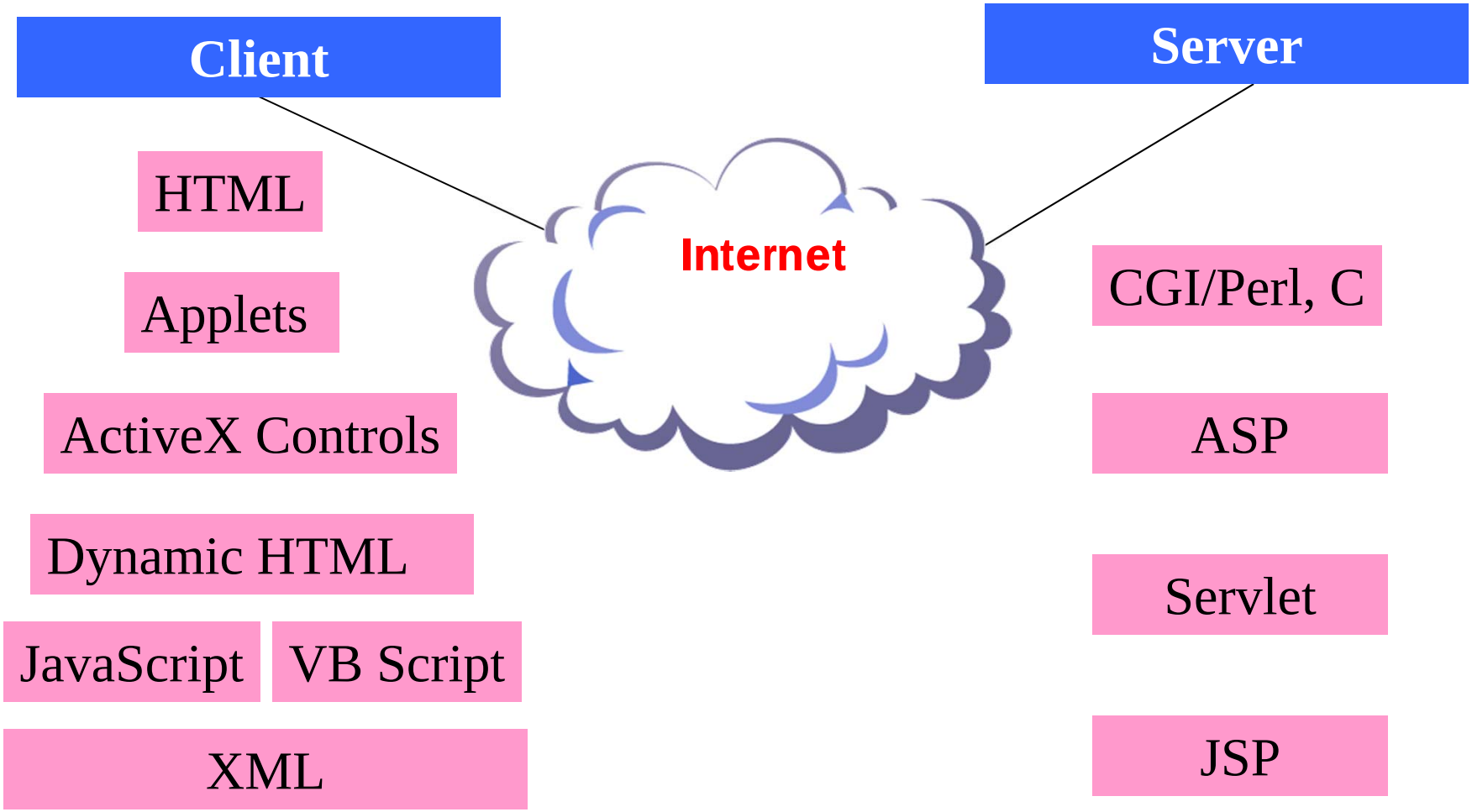
What is the average lifespan for the Enterprise Technologies?

- Better answered if we understood:
  - ☞ Programming Technologies grow fast and disappear within 10-30 years
  - ☞ Principles of programming last for generations not few decades
  - ☞ Did you hear about Algol68?
  - ☞ The enterprise architecture has a significant impact on the lifespan of the technology.

Internet Architecture: Simple Model=request+reply



Client Side vs. Server Side Scripting/Programming Languages, and Technologies



Evolution of Internet

How did it progress …

- Simple request-response model
  - Using Static HTML pages
- Dynamic Context and Personalization
  - Write CGI scripts in C, Perl
  - Problem: script maintenance, reusability, security
- Better Technologies: ASP, JSP/Servlet

Servlets and Java Server pages

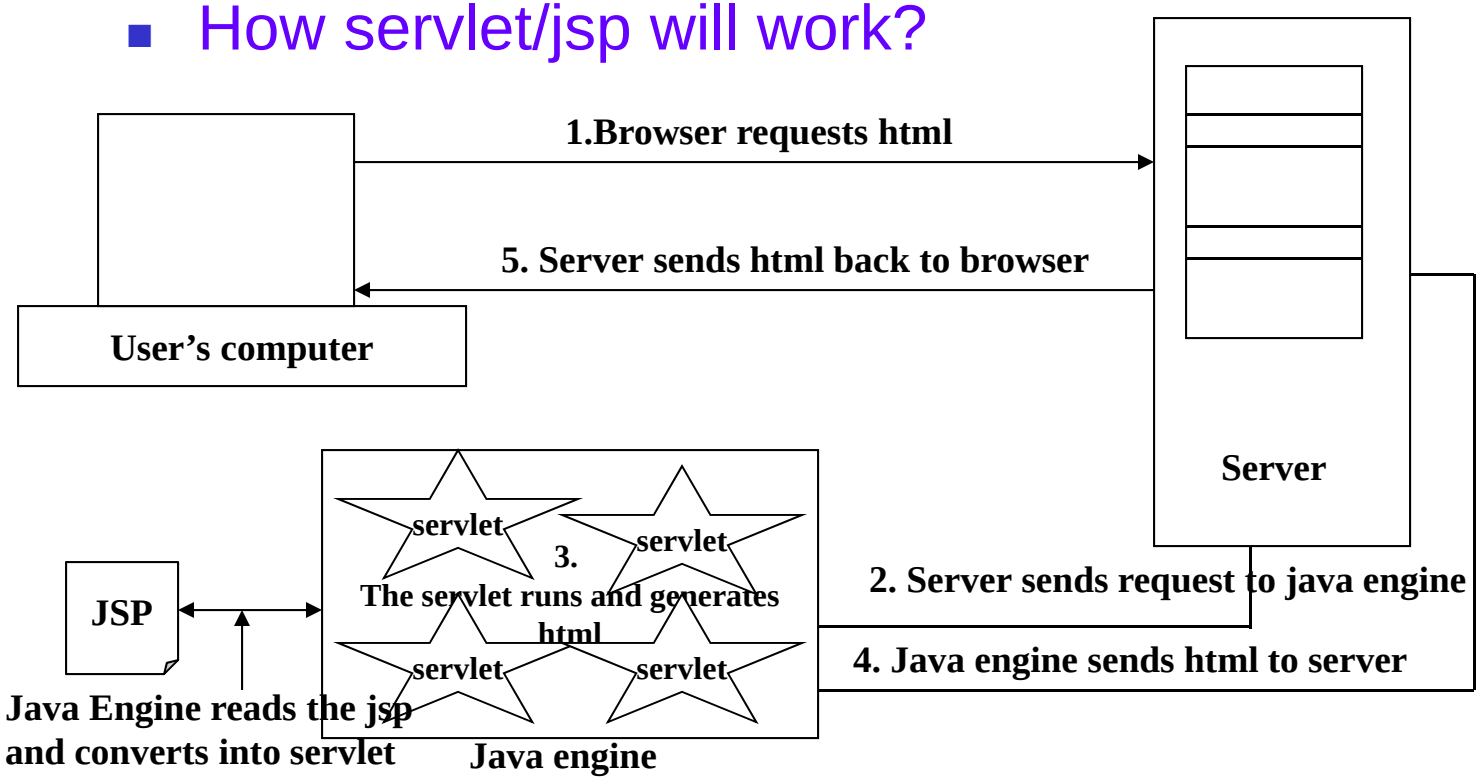
- A servlet is an object written in java that is equipped to receive a request and build and send back the response. Written in java, servlet inherits all the language strength, including speed since java is compiled.

Servlets and Java Server pages

- JSPs are built in java language so they are cross platform and inherits all the java's strength. They are fast and they can be changed easily. The servlet/jsp engine converts the jsp into servlet and it will compile and load it.
- Major application server vendors has announced support for jsp.

Servlets and Java Server pages

- How servlet/jsp will work?



JSP : JAVA Server Pages

- **JavaServer(TM) Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages.**

- **JSP is an HTML/JAVA hybrid language.**

- **Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable component model.**

JSP : JAVA Server Pages

- **JAVA Web Server and Apache/Tomcat support the implementation of JSP**
- **When the Web server processes a JSP, it generates a servlet that corresponds to that JSP;**
- **Think of JSP as a way to automate the design of servlets**
- **The Java Server Pages architecture enables the separation of content generation from content presentation.**

The Advantages of Servlets Over “Traditional” CGI

- Servlets are JAVA solutions for the problems of CGI scripts:
  - **cross-platform**
  - **Reusability**
  - **Efficient**
  - **Convenient : Lots of high-level utilities**
  - **Powerful: Sharing data, pooling, persistence**
  - **Secure**
  - **No shell escapes, no buffer overflows**
  - **Inexpensive: plenty of free and low-cost servers.**
- Servlets can be embedded in many different servers because the servlet API, which you use to write servlets, assumes nothing about the server's environment or protocol. Servlets have become most widely used within HTTP servers.

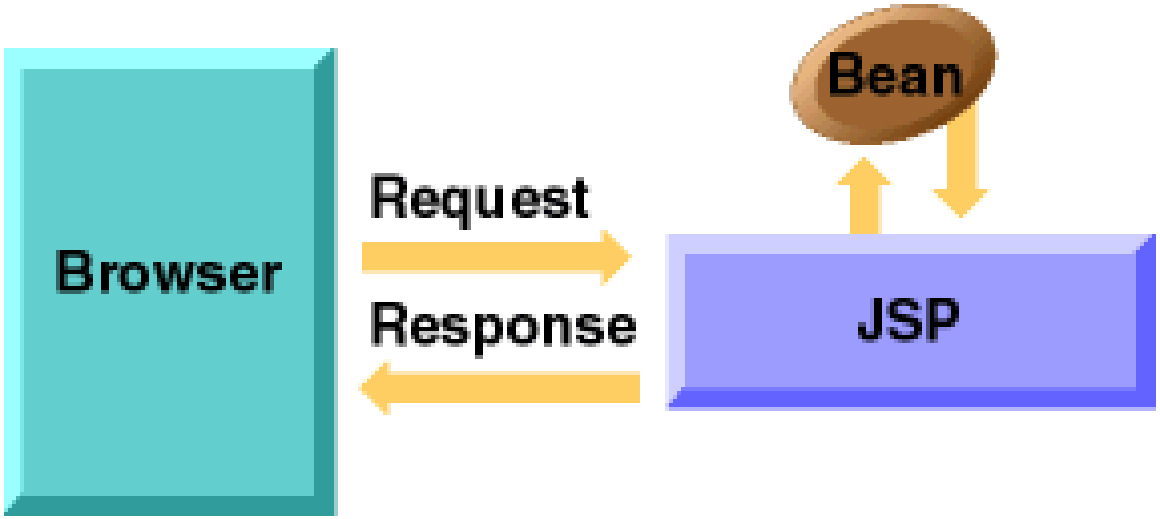
Few Good Reasons to Use JAVA Servlets

- A servlet can handle multiple requests concurrently, and can synchronize requests. This allows servlets to support systems such as ***on-line conferencing***.

- Forwarding requests. Servlets can forward requests to other servers and servlets. Thus servlets can be used to ***balance load*** among several servers that mirror the same content

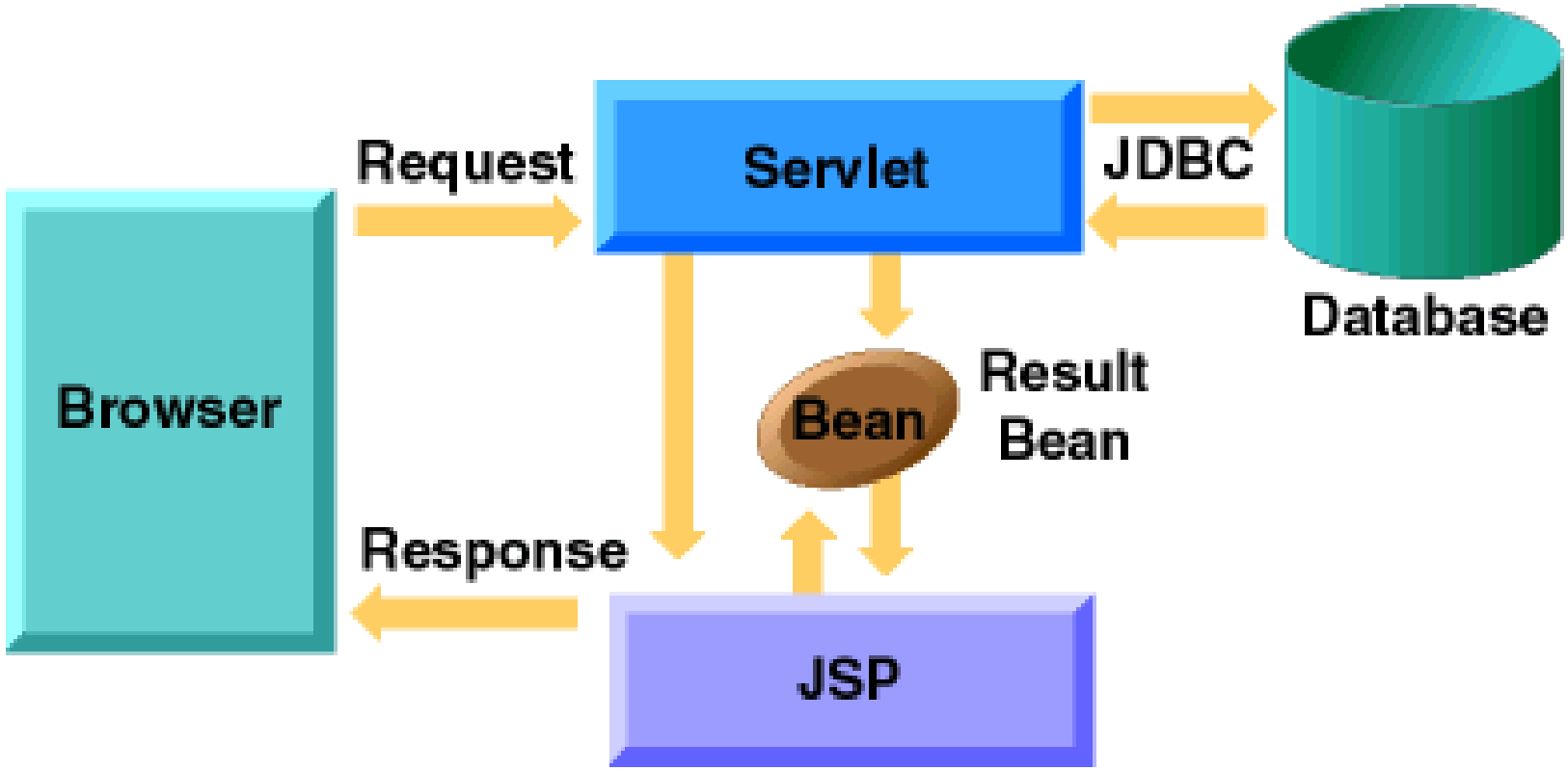
The Servlet/JSP Access Model

(1) Request comes directly to JSP

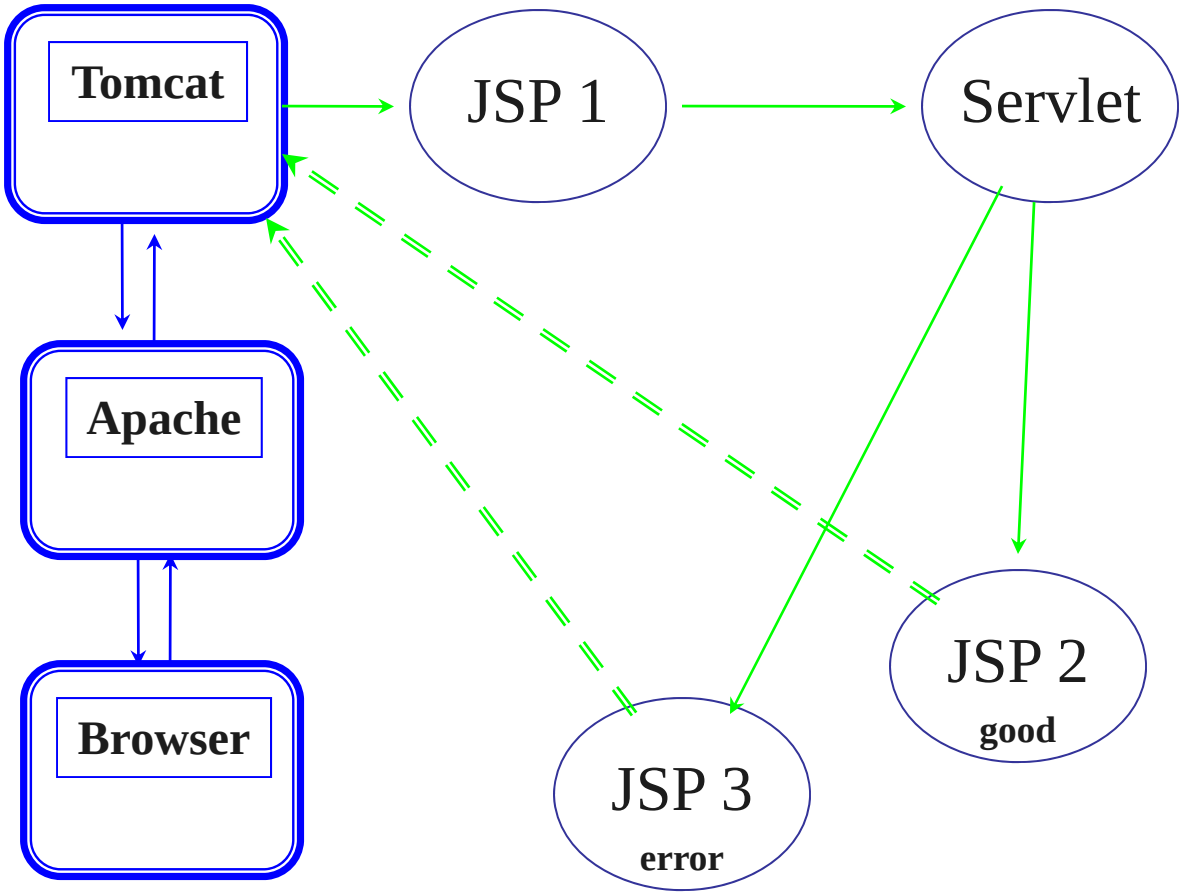


The Servlet/JSP Access Model

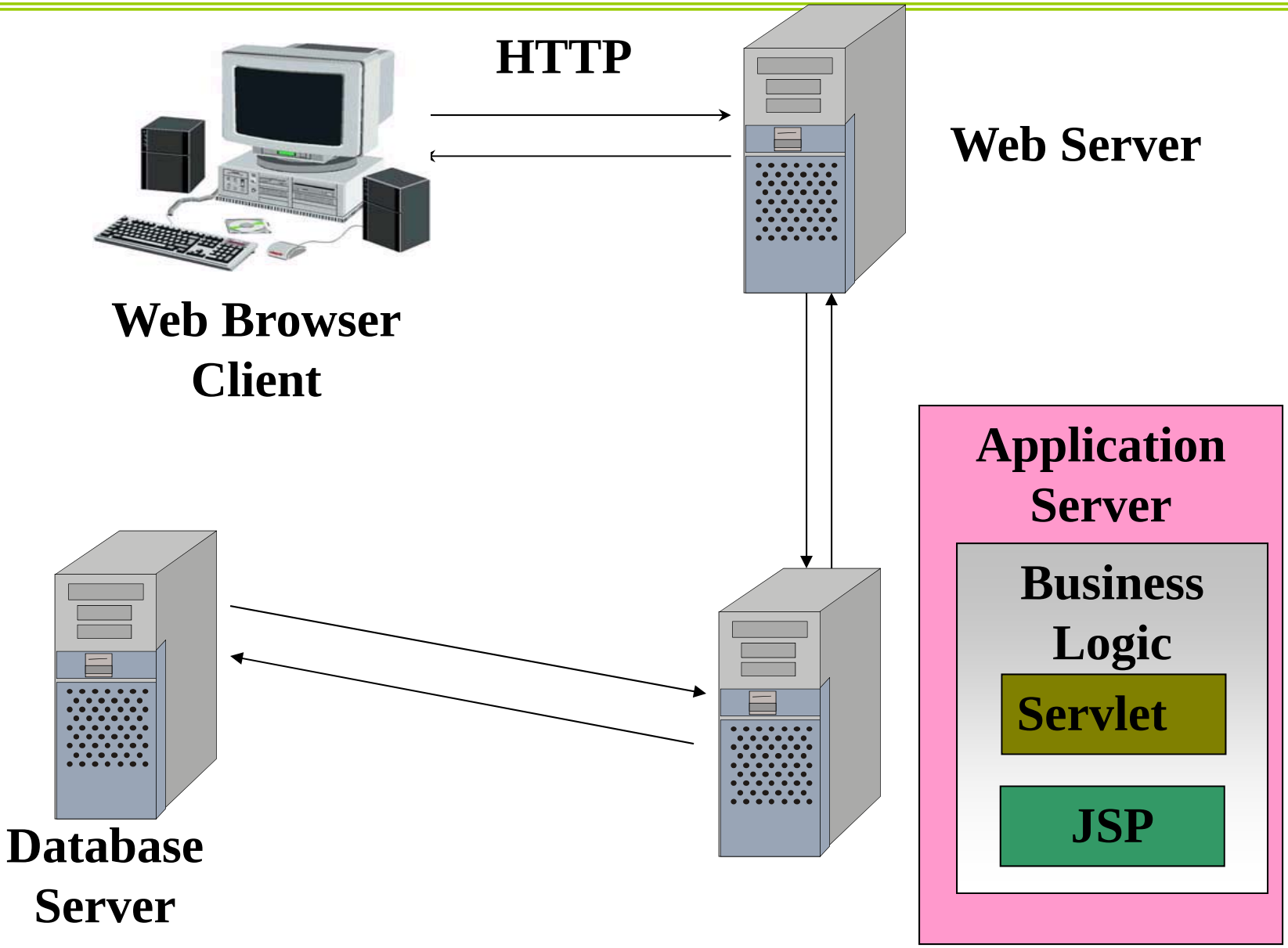
(2) Request comes through a servlet



An application with Servlets and JSP



The Enterprise Server Components



Multitier Applications:

- Many of today's application are three-tier distributed applications, consisting of
  - user interface
  - business logic
  - database access

Multitier Applications

- The user interface in such an application is often created using HTML, Dynamic HTML, VBScript, JAVA Script, or Java Applets.
- All browsers, designing the user interface to access through web browser guarantees portability across all platforms that have browsers, support HTML.
- Using the networking provided automatically by the browser, the user interfaces with middle-tier business logic.

Multitier Applications:

- The middle tier can then access the database to manipulate the data. All three tiers may reside on separate computers that are connected to a network

- In multitier architectures, Web servers are increasingly used to build the middle tier.

Multitier Applications:

- They provide the business logic that manipulates data from databases and that communicates with client web browsers.
- Servlet, through JDBC, can interact with a database systems.

Multitier Applications:

- In three tier distributed application –The user interface in a browser-using HTML.
- The middle tier is a Java Servlet that handles requests from the client browser and provides access to the third tier.
- 3rd tier like Microsoft Access database accessed via JDBC

Enterprise Application Architecture 1

- 1-Tier Architecture
- Mainframe with dumb terminals
  - All processing takes place on the mainframe
  - Presentation to user on dumb terminal
  - No distributed code or data
  - Business Logic on the Server

Enterprise Application Architecture 2

- 2-Tier Architecture
- Application or Database Server with a “fat” client
  - a.k.a. Client-Server
  - Code exists on the Client, Data/Business Logic on the server.
- Code is not centralized
- Business Logic on the Client

Enterprise Application Architecture 3

- 3-Tier Architecture (a.k.a. n-Tier)
- Client-Application Server-Database Server
- Could be more complicated
  - (ex.)Client-Web Server-EJB Server-Middleware Server-Application Server-Database Server
  - Code is distributed, but Business Logic is kept in the middle layers

Web Applications

- An application that is invoked via a web browser and whose functions are performed on a web server or application server
- N-Tier
  - Client issues request
  - Server sends the response
    - Web Server
    - App Server
    - Database Server

Web and Application Servers

- Web servers are designed to serve static content quickly and efficiently.
  - Examples: Apache, iPlanet iWS, IIS
  - Forward requests for dynamic content to an Application server
- Application servers are designed to create dynamic content
  - contains business logic and data
  - manages persistence and transaction
  - ensures data integrity and security

HTTP : Request-Response Model

- HTTP follows the Request-Response model
- HTTP 1.1 (RFC 2616)
  - http://www.w3.org/Protocols/
- HTTPS
  - HTTP over SSL (Secure Socket Layer)
  - Client establishes a connection to the server.
  - Client sends a request to the server.
  - Server sends a response to the client.
  - Server closes the connection.

HTTP Requests

- *Request-method Request-URI HTTP-version*
  (*Request- header* **) ***
  blank line
  [*Message- body* **]**

- Request methods: `GET POST PUT DELETE TRACE OPTIONS CONNECT HEAD`

- Request headers:
  key **:** *value*

HTTP Request

- GET and POST are the most common requests
  - GET means “get whatever information the URL points to”
  - POST is used to tell the server to perform some action using supplied data by the request
  - In reality, they can do the same thing, but GET query data is encoded in the URL, POST data is sent in the body of the request

HTTP Request

- GET is usually used when all data can be encoded in a query string and is not part of a form
- POST is usually used to submit form information.
- We will see the difference when we look into the Servlet specification and see some examples.

HTTP Response

- *HTTP-version Status-code Reason-phrase*
  (*Response-header* **) ***
  blank line
  [*Message-body* **]**
  Response headers:
  key **:** *value*
  Content- Type: MIME Types
  type **/** *subtype*

Free Servlet and JSP Engines

- Apache Tomcat
  - http://tomcat.apache.org/
- Allaire/Macromedia JRun
  - http://www.adobe.com/products/jrun/
- New Atlanta ServletExec
  - http://www.servletexec.com/
- Gefion Software LiteWebServer
  - http://freecode.com/projects/lws
- Caucho's Resin
  - http://www.caucho.com/

Java Servlets

- Servlets provide extensions of functionalities of servers, most commonly HTTP servers.
- They are precompiled Java programs that are executed on the server side, inside a servlet container.
- Servlets provide the following benefits:
  - Efficiency - thread vs. process
  - Portability - platform independent
  - Robustness - runs in a JVM
  - Extensibility - can access any Java API
  - Security - not shell based, runs in a JVM

Servlet Container

- Also known as a *servlet engine*
- An extension to a web server
- The servlet engine is a key component to an application server
- Uses HTTP to communicate with the client
- Takes the request data from the web server, processes it, then responds back through the web server

Tomcat 7.0.X

- Tomcat can be used
  - As a standalone web server
  - As an add-on to a web server
    - In-process or out-of-process
    - Supports IIS, Apache, Netscape web servers

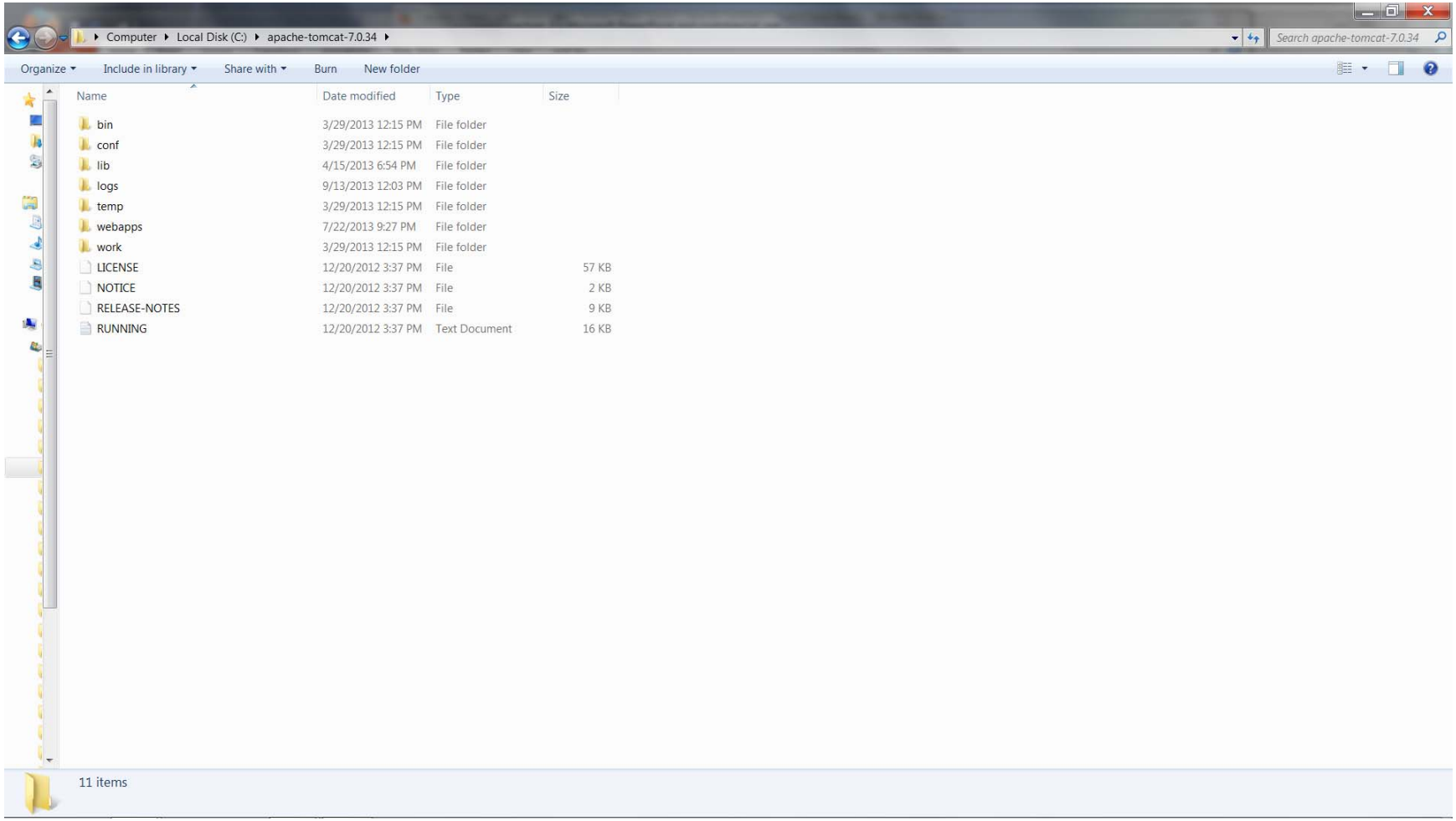
Installing Tomcat

- Download the zip file from http://tomcat.apache.org/
- Get the current version (7.0.34)
- Unzip it to a directory
- The top level directory for Tomcat is apache-tomcat-7.0.34
- This directory is known as *Tomcat root* or *Tomcat home*
- It will be referred to as <tomcat_home>

Subdirectories

- In <tomcat_home>:
  - bin - scripts to startup and shutdown Tomcat
  - lib - classes visible to Tomcat **and** webapps
    - lib - shared jars
  - conf - configuration files
  - logs - log files
  - server - classes and libraries for Tomcat itself
  - webapps - web applications
  - work - scratch directory for Tomcat

Subdirectories



Starting/Stopping

- To start Tomcat
  - startup.bat (Windows)
- To stop Tomcat
  - execute shutdown.bat (Windows)
- See <tomcat_home>/Running.txt for most common problems if Tomcat won’t start

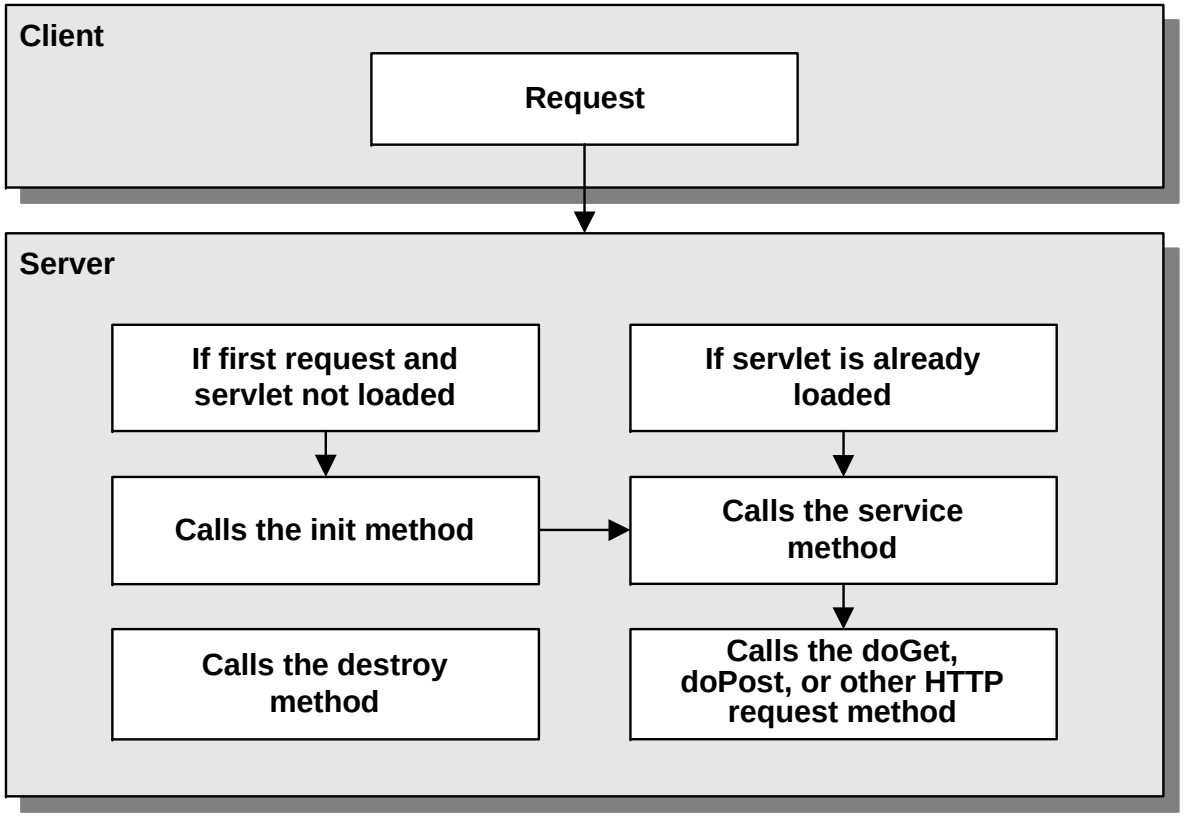
Configuration

- Default port: 8080
- The preconfigured version we downloaded is using port 80
- To access, point web browser to

  `http://localhost/`

  - The file you see is located at `<tomcat_home>\webapps\ROOT\index.jsp`
  - Each directory under webapps is a *web application (more on these later)*

How the server handles a request for a servlet



A simple Servlet

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


public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World ... !");
  }
}
```

Development vs. Deployment

- **Development**
  - Where you create and compile your servlets
- **Deployment**
  - Where you place your servlets byte code for execution

Running a simple Servlet

- Go to directory c:\csj and setup the environment by executing the **env-setup-for-tomcat**
- Pay careful addition to the classpath that it has been setup correctly

  - set JAVA_HOME=C:\jdk1.7
  - set PATH="C:\jdk1.7\bin";%PATH%
  - set CLASSPATH=.;C:\apache-tomcat-7.0.34\lib\servlet-api.jar;C:\apache-tomcat-7.0.34\lib\jsp-api.jar;C:\apache-tomcat-7.0.34\lib\el-api.jar;C:\apache-tomcat-7.0.34\lib\commons-beanutils-1.8.3.jar
  - set ANT_HOME=c:\apache-tomcat-7.0.34
  - set TOMCAT_HOME=C:\apache-tomcat-7.0.34
  - set CATALINA_HOME=C:\apache-tomcat-7.0.34

Running a simple Servlet

- Compile the servlet
  - C:\CSJ\S2>javac HelloWorld.java
- Copy the file Hello.class to `<tomcat_home>\webapps\csj\WEB-INF\classes`
- Start\Restart Tomcat
  - It takes sometime to reload the new bytecode; better if you restart while you are testing
- Point your browser to
  - http://localhost/csj/HelloWorld

Running a simple Servlet

- Compile the servlet
  - C:\CSJ\L2>javac HelloWorld.java
- Copy the file Hello.class to `<tomcat_home>\webapps\csj\WEB-INF\classes`
- Start\Restart Tomcat
  - It takes sometime to reload the new bytecode; better if you restart while you are testing
- Point your browser to
  - http://localhost/csj/HelloWorld

Running a simple Servlet (cont.)

- **Steps to do**
  1. **Setup the path**
  2. **Compile your sevlet**
  3. **Deploy the servlet bytecode into Tomcat/csj context**
  4. **Start the server**

Running a simple Servlet (cont.)

C:\Windows\system32\cmd.exe

```
c:\csj\S1>env-setup-for-tomcat

c:\csj\S1>set JAVA_HOME=C:\jdk1.7

c:\csj\S1>set PATH="C:\jdk1.7\bin";"C:\jdk1.7\bin";C:\Program Files (x86)\AMD APP\bin\x86_64;C:\Program Files (x86)\AMD APP\bin\x86;C:\Program Files\Common Files\Microsoft Shared\Windows Live
;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Intel\W
iFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;c:\Program File
ntec\VIP Access Client\;C:\Program Files (x86)\Sony\VAIO Startup Setting Too

c:\csj\S1>set CLASSPATH=.;C:\apache-tomcat-7.0.34\lib\servlet-api.jar;C:\apa
ils-1.8.3.jar

c:\csj\S1>set ANT_HOME=c:\apache-tomcat-7.0.34

c:\csj\S1>set TOMCAT_HOME=C:\apache-tomcat-7.0.34

c:\csj\S1>set CATALINA_HOME=C:\apache-tomcat-7.0.34

c:\csj\S1>
c:\csj\S1>start

c:\csj\S1>
```

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

c:\csj\S1>cd c:\x34\bin

c:\apache-tomcat-7.0.34\bin>startup
Using CATALINA_BASE:   "C:\apache-tomcat-7.0.34"
Using CATALINA_HOME:   "C:\apache-tomcat-7.0.34"
Using CATALINA_TMPDIR: "C:\apache-tomcat-7.0.34\temp"
Using JRE_HOME:        "C:\jdk1.7"
Using CLASSPATH:       "C:\apache-tomcat-7.0.34\bin\bootstrap.jar;C:\apache-tomcat-7.0.34\bin\tomcat-juli.jar"
c:\apache-tomcat-7.0.34\bin>
```

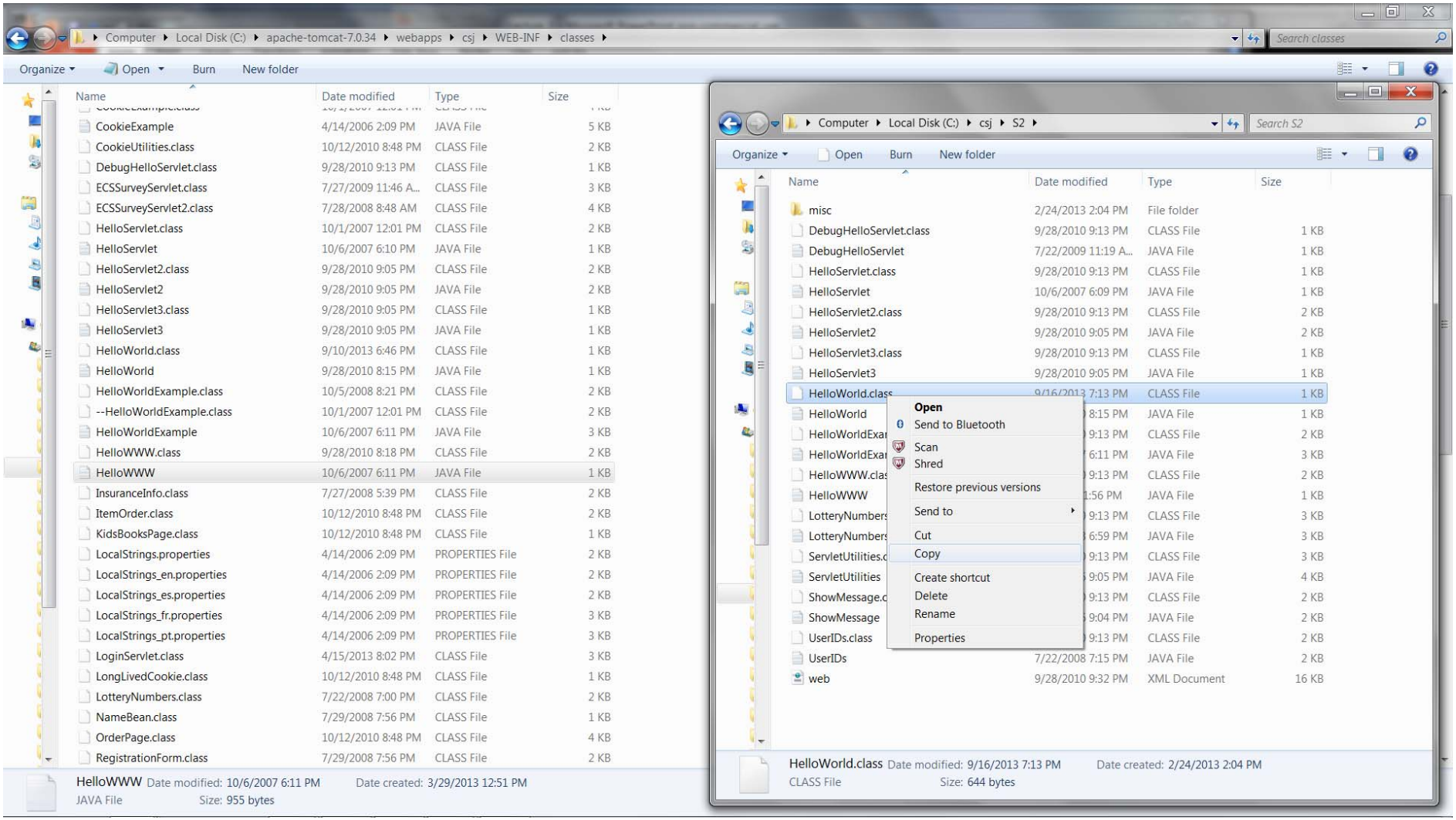
Tomcat

```
INFO: Deploying web application directory C:\apache-tomcat-7.0.34\webapps\exampl
es
Sep 16, 2013 7:10:10 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\apache-tomcat-7.0.34\webapps\host-m
anager
Sep 16, 2013 7:10:10 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\apache-tomcat-7.0.34\webapps\jsp-ex
amples
Sep 16, 2013 7:10:10 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\apache-tomcat-7.0.34\webapps\jsp-to
-servlet-to-jsp
Sep 16, 2013 7:10:10 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\apache-tomcat-7.0.34\webapps\manage
r
Sep 16, 2013 7:10:10 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\apache-tomcat-7.0.34\webapps\mvc
Sep 16, 2013 7:10:10 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\apache-tomcat-7.0.34\webapps\ROOT
Sep 16, 2013 7:10:10 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-80"]
Sep 16, 2013 7:10:10 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Sep 16, 2013 7:10:10 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 1533 ms
```

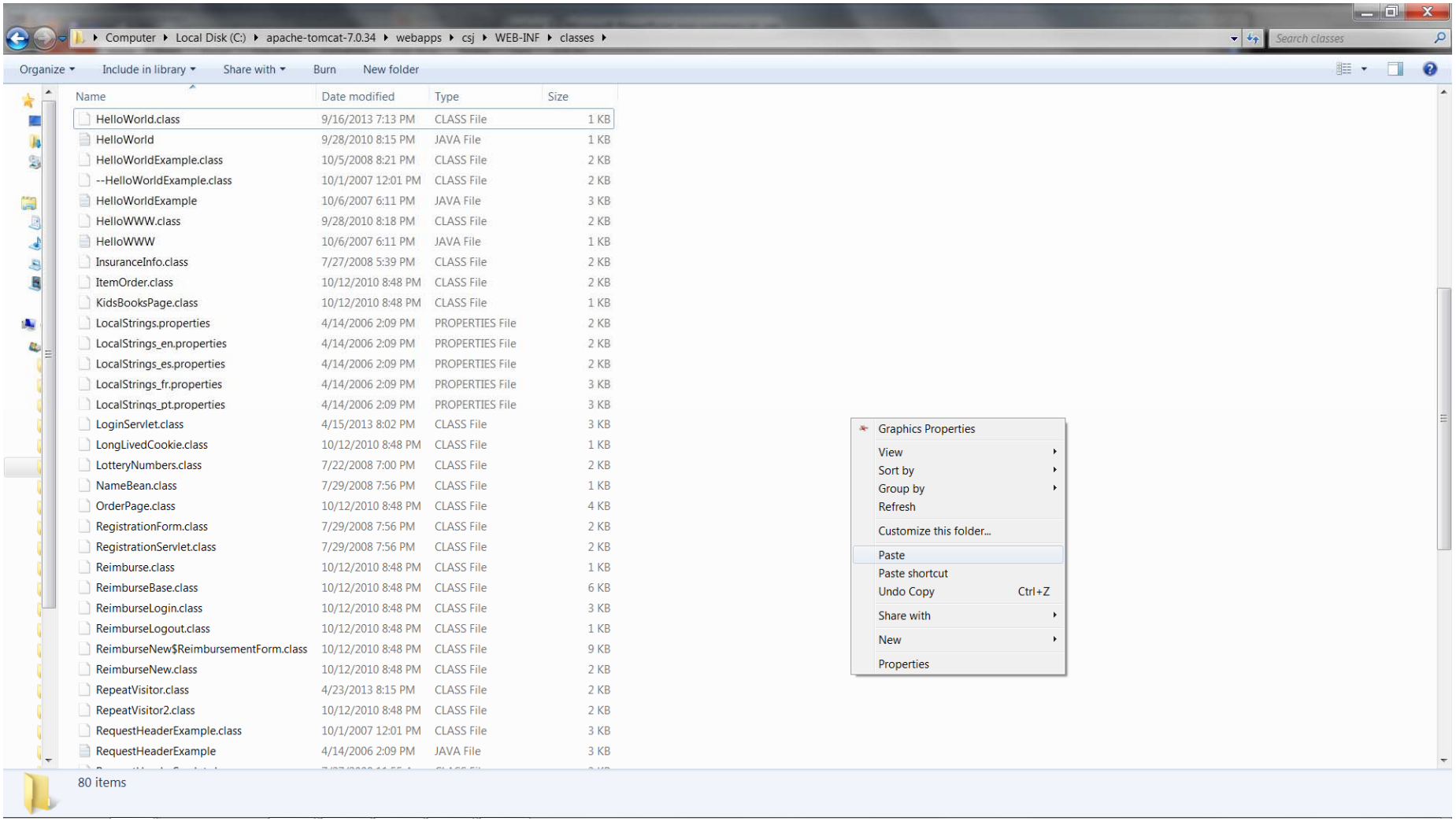
Running a simple Servlet (cont.)

```
C:\Windows\system32\cmd.exe

c:\csj\S2>
c:\csj\S2>
c:\csj\S2>
c:\csj\S2>
c:\csj\S2>
c:\csj\S2>
c:\csj\S2>dir HelloWorld.*
 Volume in drive C has no label.
 Volume Serial Number is 289B-5690

 Directory of c:\csj\S2

09/28/2010  08:15 PM               647 HelloWorld.java
               1 File(s)            647 bytes
               0 Dir(s)  470,491,037,696 bytes free

c:\csj\S2>
c:\csj\S2>
c:\csj\S2>javac HelloWorld.java

c:\csj\S2>
c:\csj\S2>
c:\csj\S2>dir HelloWorld.*
 Volume in drive C has no label.
 Volume Serial Number is 289B-5690

 Directory of c:\csj\S2

09/16/2013  07:13 PM               644 HelloWorld.class
09/28/2010  08:15 PM               647 HelloWorld.java
               2 File(s)          1,291 bytes
               0 Dir(s)  470,491,033,600 bytes free

c:\csj\S2>
```

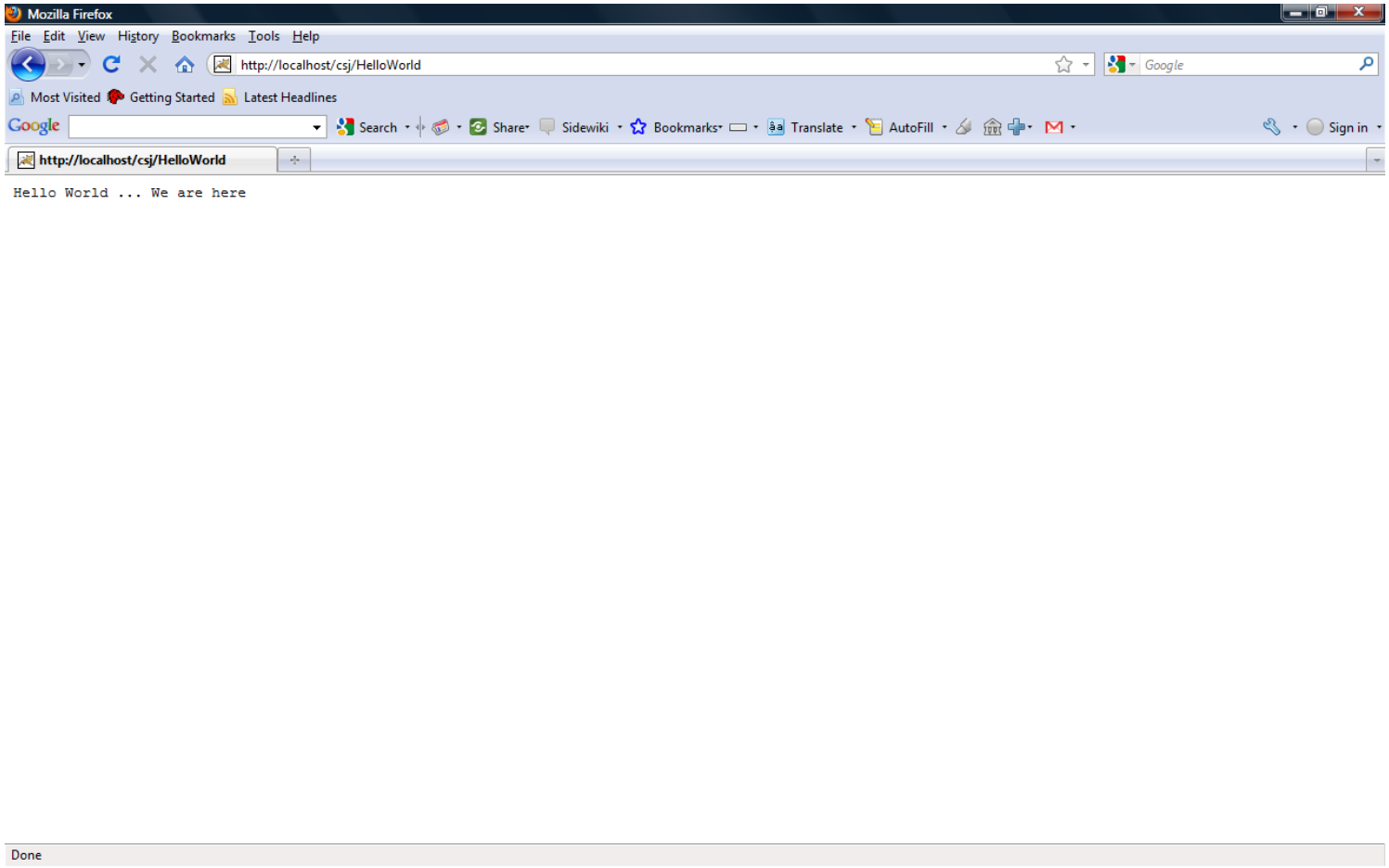
Running a simple Servlet (cont.)



Running a simple Servlet (cont.)



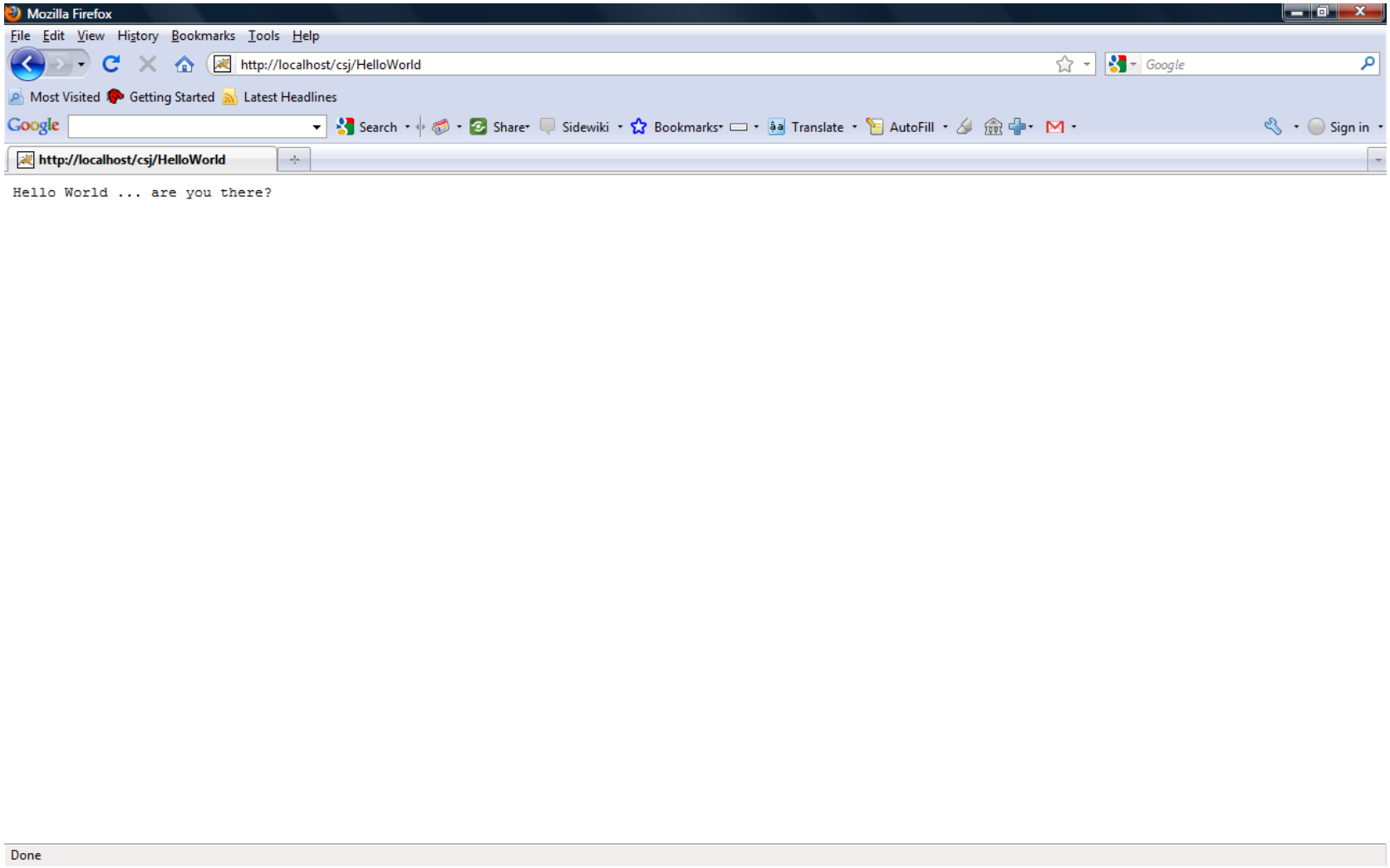
Running a simple Servlet (cont.)



Running a simple Servlet

- Modify HelloWorld.java and restart your tomcat server
- Refresh the browser webpage
- Point your browser to
  - http://localhost/csj/HelloWorld

Running a simple Servlet (cont.)



URI

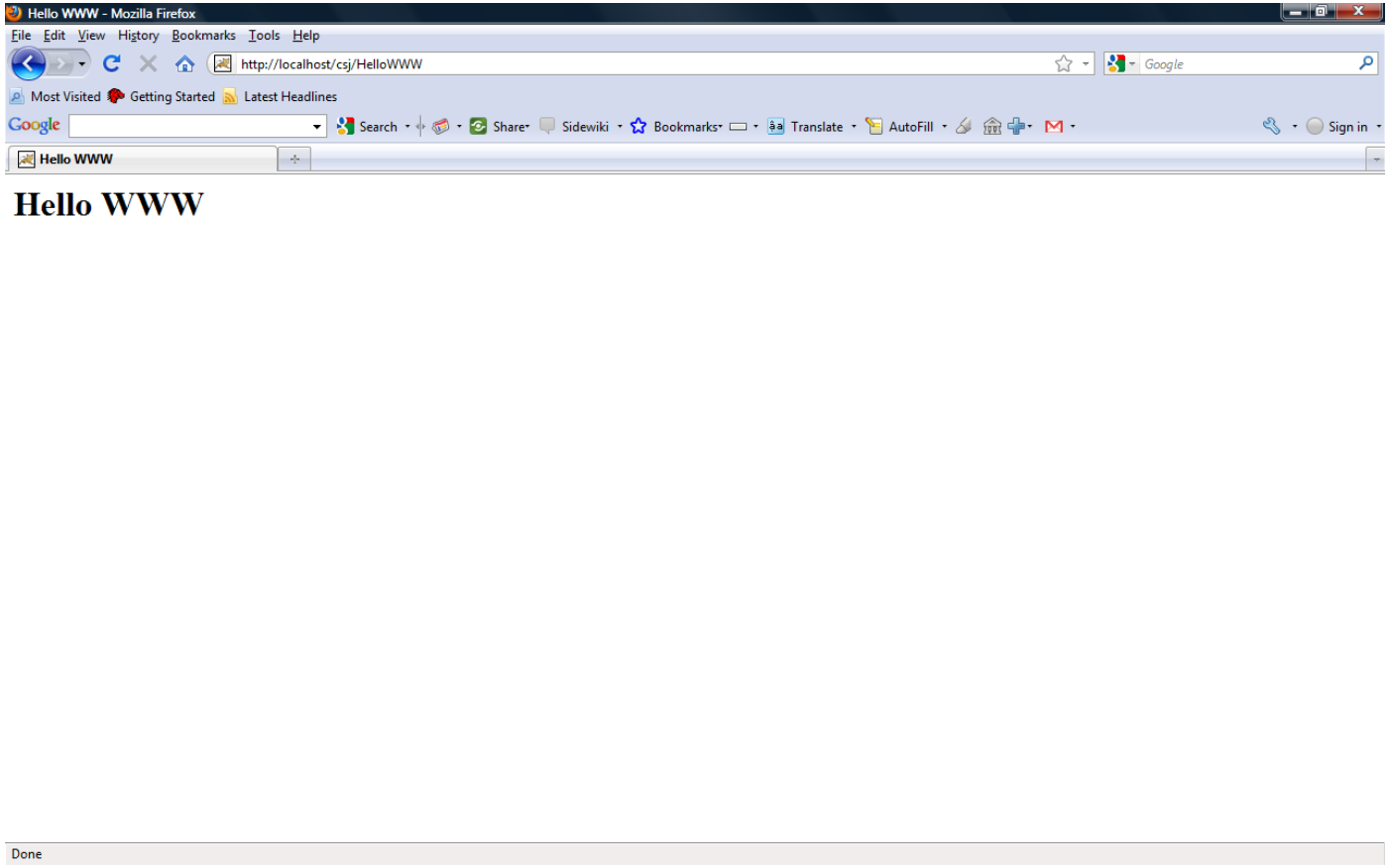
- Uniform Resource Identifier
- RFC 2396
- Uniform Resource Locator (URL) is a subset of URI
- http://localhost/csj/HelloWorld
  - http - protocol
  - localhost - host
  - 80 - port
  - /csj/HelloWorld - resource on server

Printing HTML

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

public class HelloWWW extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
      "Transitional//EN\">\n";
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>Hello WWW</TITLE></HEAD>\n" +
                "<BODY>\n" +
                "<H1>Hello WWW</H1>\n" +
                "</BODY></HTML>");
  }
}
```

Printing HTML (cont.)

- Run the program : http://localhost/csj/HelloWWW

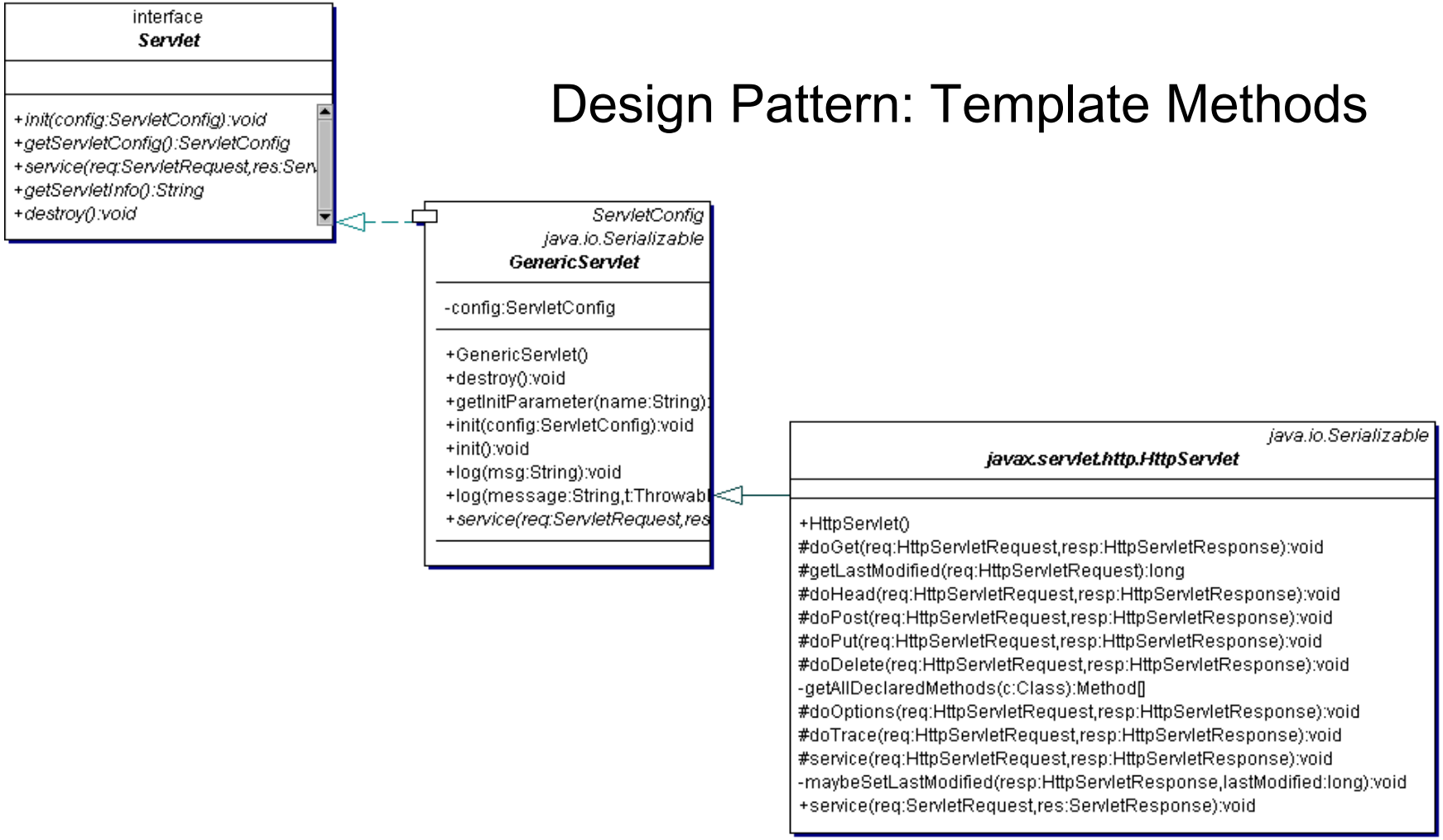


Java Servlet API

- Package `javax.servlet` contains interfaces and abstract classes for protocol independent generic servlets.
- Package `javax.servlet.http` contains interfaces and abstract classes for servlets using the HTTP protocol.
- extends the interfaces and classes in `javax.servlet`
- The servlet engine provides implementation of these two packages:
  - `<tomcat_home>/lib/servlet-api.jar`

Servlet Framework

Design Pattern: Template Methods



Servlet Framework (cont.)

- interface `javax.servlet.Servlet`
  - Defines methods that all servlets must implement.
  - `init()`, `service()`, `destroy()`
- abstract class `javax.servlet.GenericServlet`
  - Defines a generic, protocol-independent servlet.
  - `init()`, `service()`, `destroy()`
- abstract class `javax.servlet.http.HttpServlet`
  - Defines a servlet using the HTTP protocol.
  - `doGet()`, `doPost()`, etc.
- interface `javax.servlet.SingleThreadModel`
  - Ensures that servlets handle only one request at a time.
  - Marker interface. Defines no methods.

Servlet Framework (cont.)

- interface `javax.servlet.ServletRequest`
  - provides client request information to a servlet.
- interface `javax.servlet.ServletResponse`
  - assists a servlet in sending a response to the client.
- interface `javax.servlet.http.HttpServletRequest`
  - extends `javax.servlet.ServletRequest`
  - provides request information for HTTP servlets.
- interface `javax.servlet.http.HttpServletResponse`
  - extends `javax.servlet.ServletResponse`
  - provides HTTP-specific functionality in sending a response.

Servlet Lifecycle

- The life cycle of a servlet is controlled by the servlet engine.
- Servlets are instantiated (loaded) and destroyed (unloaded) by the servlet engine.
- For a generic servlet, the servlet engine calls the following methods:
  - `init()` when the servlet is first loaded.
  - `service()` when each request is received.
  - `destroy()` just before the servlet is unloaded.

Servlet Lifecycle: `init()`

- `void init()`
- Called once when the servlet is first loaded.
- Not called for each request.
- To perform one-time initialization or configuration of the servlet, e. g.
  - reading initialization parameters
  - reading configuration parameters from property files or other resources
  - setting up databases
- To be overridden by subclasses. No need to call `super.init()`.

Servlet Lifecycle: `service()`

- `abstract void service(ServletRequest request,ServletResponse response)`
- Called when each request is received.
- To be overridden by subclasses to handle requests.
- Executed in a new thread for each request.
- Multiple threads may execute this method at the same time.
- However, you may choose the *single thread model* to prevent multiple threads to execute this method at the same time.
  - `implements interface SingleThreadModel`

Servlet Lifecycle: `destroy()`

- `void destroy()`
- Called just before the servlet is unloaded.
- Not called after each request.
- To release resources.
- Servlets may be unloaded by the servlet container at any time.

Servlet Lifecycle for HttpServlet

- The `service()` method is overridden to dispatch requests to `doGet()`, `doPost()`, etc.
  - allows services to be added incrementally.
  - Do not override the `service()` method.
  - Override `doGet()`, `doPost()` etc. to handle requests `GET`, `POST`, etc.

HttpServlet Methods

- `void doGet( HttpServletRequest request,HttpServletResponse response)`
- `void doPost( HttpServletRequest request, HttpServletResponse response)`
- `void doPut( HttpServletRequest request, HttpServletResponse response)`
- `void doDelete( HttpServletRequest request, HttpServletResponse response)`
- `void doOptions( HttpServletRequest request, HttpServletResponse response)`
- `void doTrace( HttpServletRequest request, HttpServletResponse response)`

Web Applications

- A Web Application is a collection of servlets, html files, JSPs, classes, and resources used by a servlet container
- There is a one-to-one relationship between webapps and a `ServletContext` object
- Web applications are a way to package an application for the servlet container and separate multiple applications from one another inside the same container.

Web Applications (cont.)

- To create a web application context, the servlet container must be configured.
- In Tomcat, the web application is placed in a directory under `<tomcat_root>/webapps`
- The context must be configured for the application server to use it. Requests will be dispatched to that webapp based on the URI used to access it.

The WEB-INF Directory

`<WebApp>/WEB-INF/web.xml`

- The configuration file of the web application, known as the *deployment descriptor*

`<WebApp>/WEB-INF/classes/`

- Contains the class files of servlets.
- The subdirectory structure of this directory must coincide with the package structure of the servlets

`<WebApp>/WEB-INF/lib/*.jar`

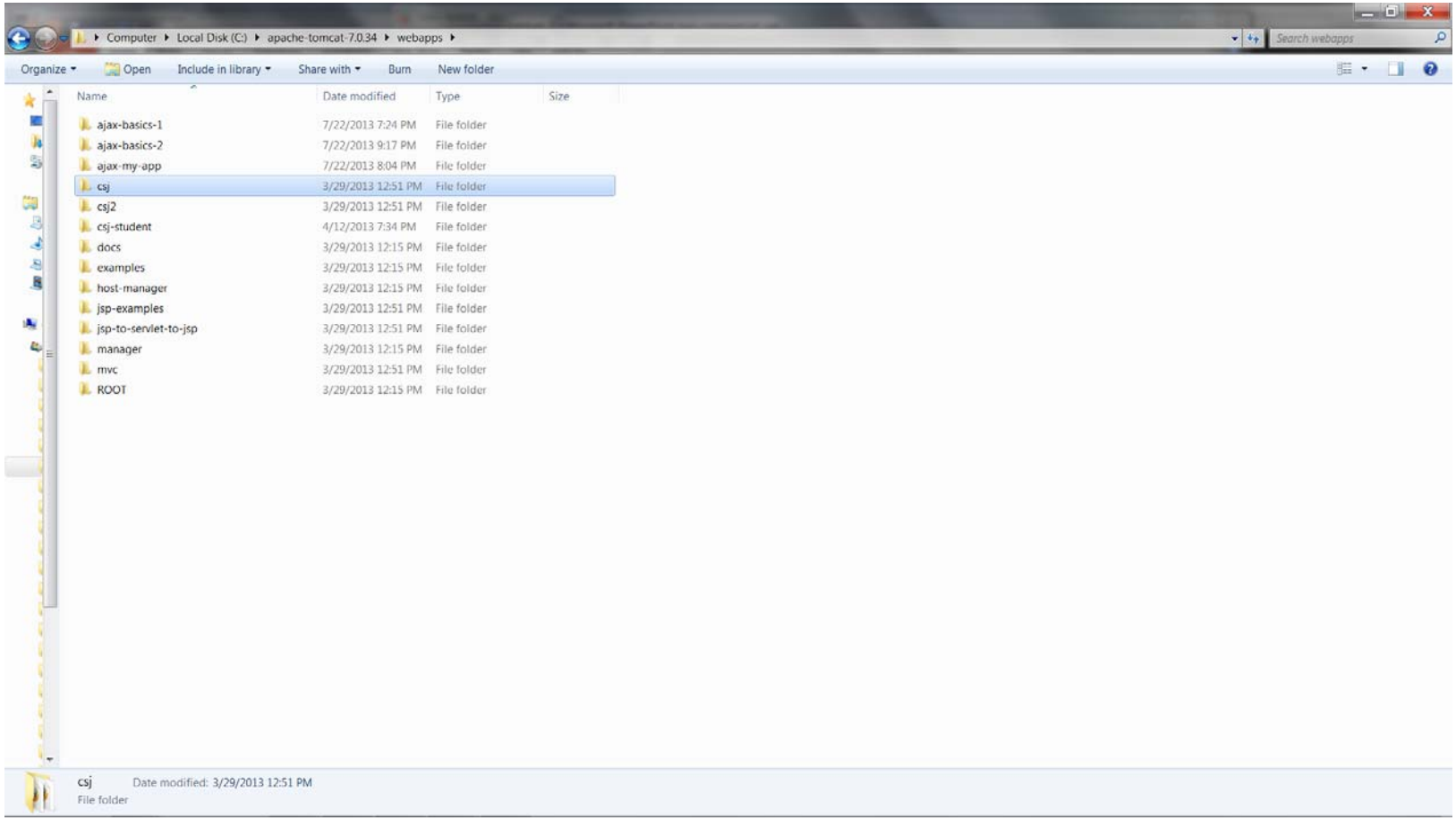
- Contains the jar files of other Java classes used by the web application, such as third-party packages.

Creating the Document Root

- To test the new contexts
  - Create the document root directory and subdirectories under the document root:
- For context `csj`
  - `<tomcat_root>/webapps/csj`
  - `<tomcat_root>/webapps/csj/WEB-INF`
  - `<tomcat_root>/webapps/csj/WEB-INF/classes`

Creating the Document Root

- You can create as many contexts as you want …



Creating web.xml

- Create a deployment descriptor.
- A minimum web.xml :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
</web-app>
```

Deploying a Web App

- Copy web.xml to `<WebApp>/WEB-INF`
  - Context csj
    - `<tomcat_home>/webapps/csj/WEB-INF/`
- Copy servlet class files (`HelloWWW.class`) to `<WebApp>/WEB-INF/classes`
  - Context csj
    - `<tomcat home>/webapps/csj/WEB-INF/classes/`

Testing the New Context

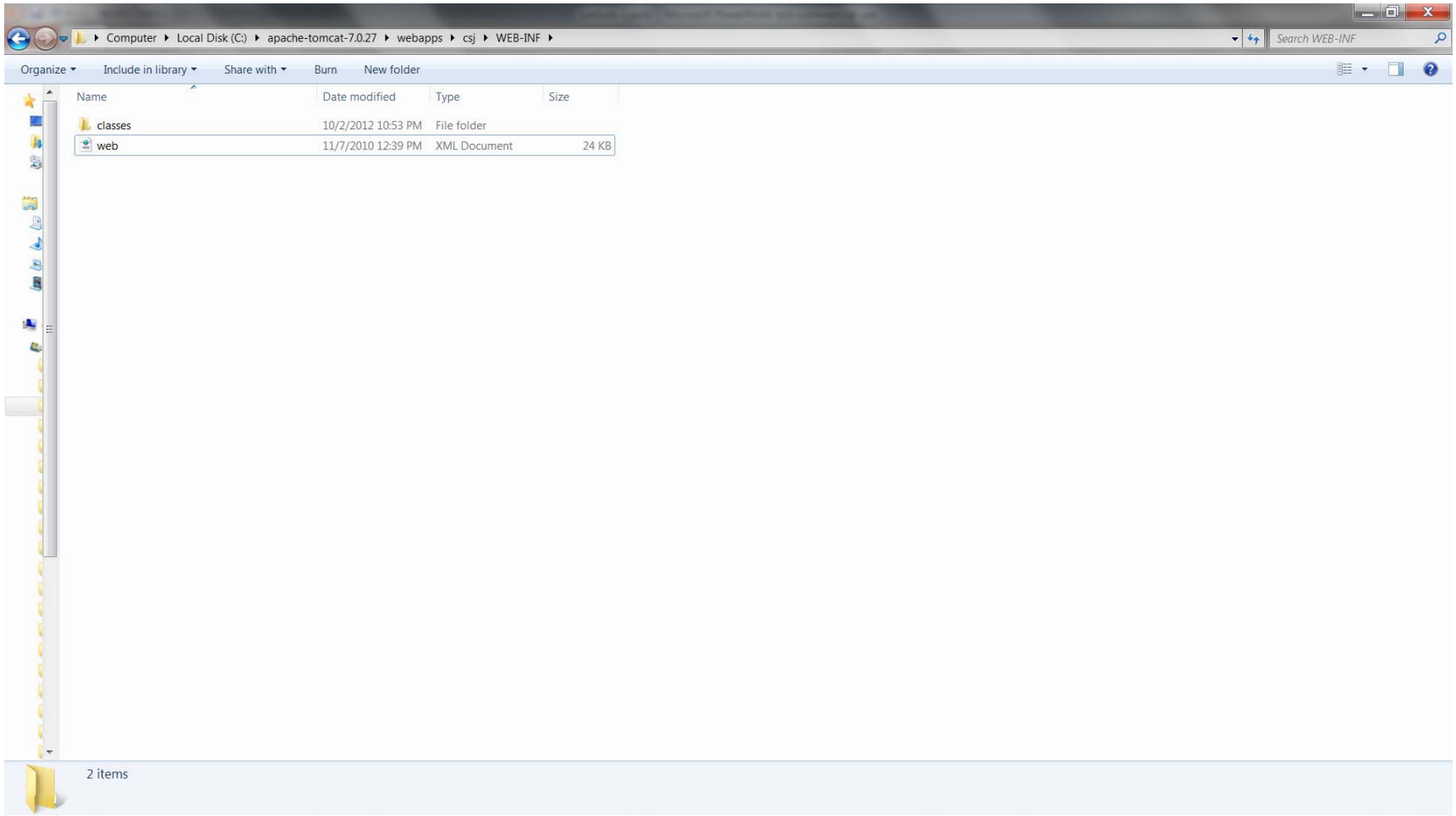
- Invoke the web apps in the new contexts as follows:
  - `http://localhost/csj/HelloWWW`
  - `http://localhost` protocol, host, and port
    - `/csj` the context prefix
    - `HelloWWW` name of the servlet class

Configuring Web Apps

- The `servlet` and a `servlet-mapping` element in deployment descriptor (web.xml).
- `servlet` element: *servlet definition*
  - It defines a name for a servlet by mapping the name to the class name of the servlet.
  - It sets initialization parameters.
  - Different servlet names can be mapped to the same servlet class.
- `servlet-mapping` element: *servlet mapping*
  - It maps a URL or a URL pattern to a servlet name.
  - Different URL can be mapped to the same servlet name.

Configuring Web Apps

- Make sure you have deployment descriptor (web.xml) for csj web-app.



Servlet Definition and Mapping

- An example of servlet definition and a servlet mapping:

```
<web-app>
  <servlet>
   <servlet-name>ShowMsg</servlet-name>
   <servlet-class>ShowMessage</servlet-class>
  </servlet>
  <servlet-mapping>
   <servlet-name>ShowMsg</servlet-name>
   <url-pattern>/ShowMsg</url-pattern>
  </servlet-mapping>
</web-app>
```

Using a Servlet Definition

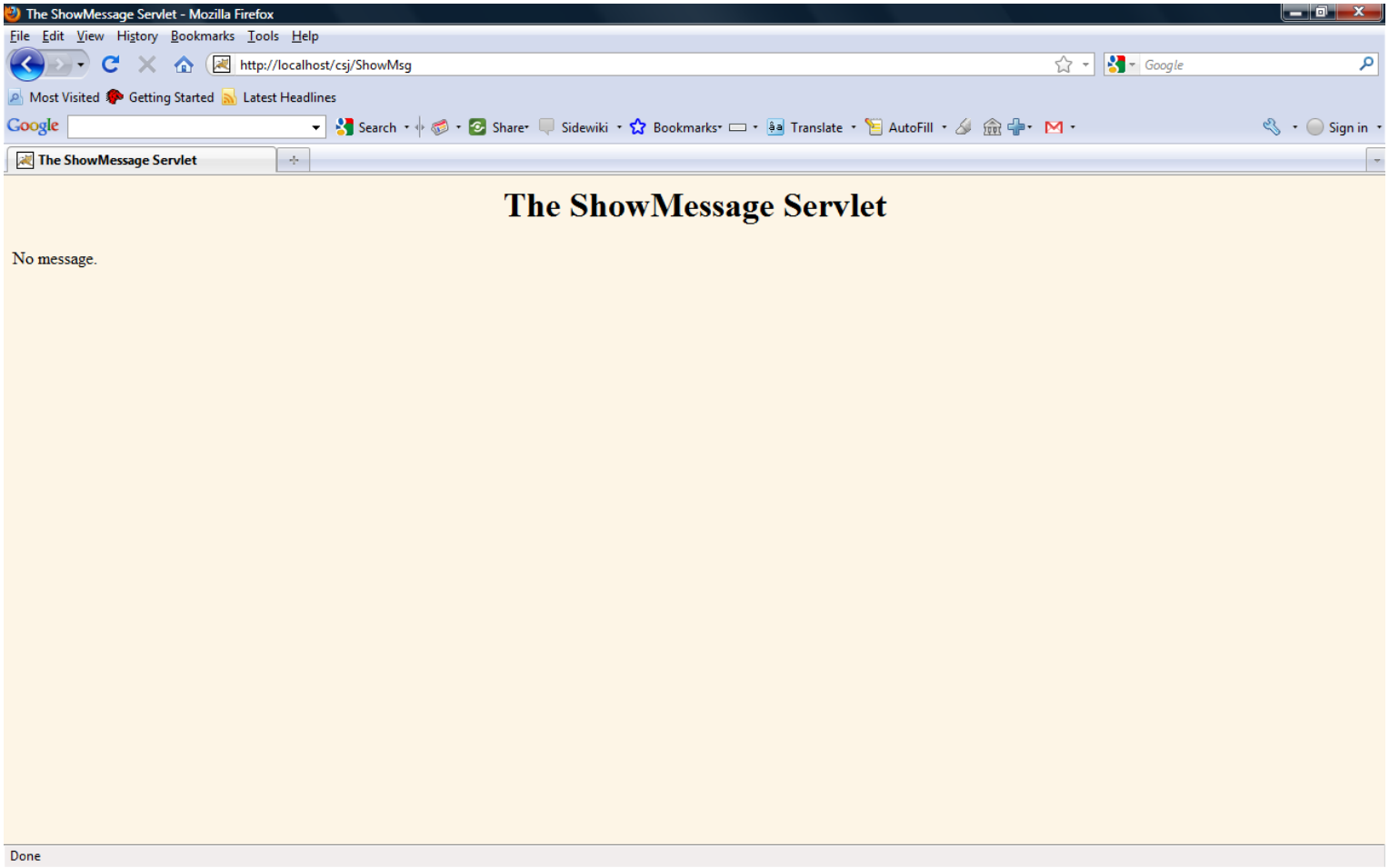
- A servlet can be invoked with the name specified in the servlet definition:

- `Compile and deploy ShowMessage.java`
  - `It uses` ServletUtilities.java; make sure it is in the same directory when you compile
- `Run the two`
  - `http://localhost/csj/ShowMessage`
  - `http://localhost/csj/ShowMsg`

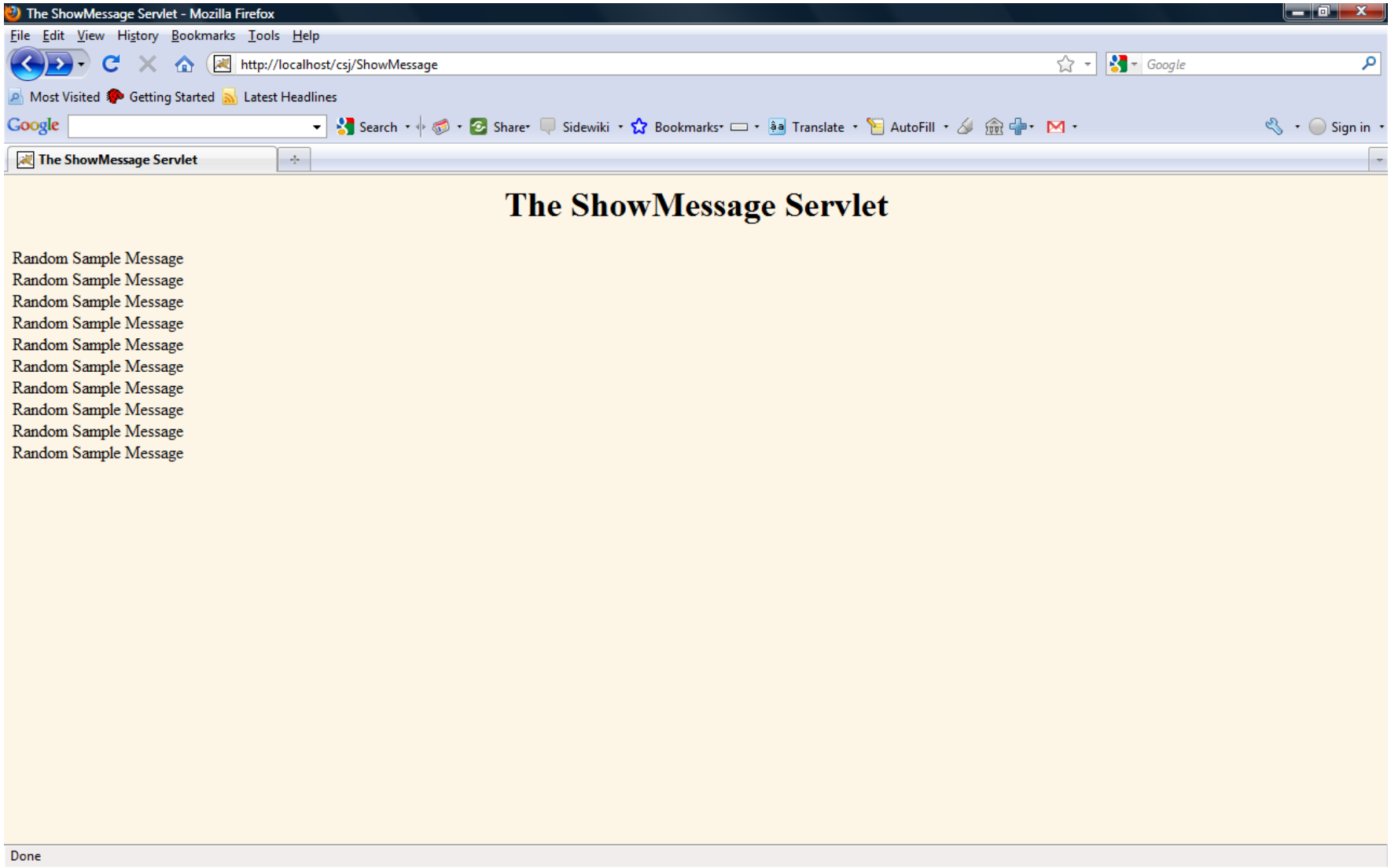
- `http://localhost/csj/ShowMsg`
  - `http://localhost` protocol, host, and port
  - `/csj` the context prefix
  - `ShowMsg` the name of the servlet

- Notice the difference between ShowMessage and ShowMsg

Using a Servlet Definition



Using a Servlet Definition



Lets Review Web.xml

- csj web-app Context has the web.xml, lets review the XML file for it
  - See the source code for Web.xml

Lets Review Web.xml

web - Notepad

File Edit Format View Help

```
<servlet>
   <servlet-name>ShowMessage</servlet-name>
   <servlet-class>ShowMessage</servlet-class>

        <init-param>
                <param-name>
                message
                 </param-name>
                <param-value>
                Random Sample Message
                </param-value>
        </init-param>

        <init-param>
                <param-name>
                repeats
                </param-name>
                <param-value>
                10
                </param-value>
        </init-param>

</servlet>


<servlet>
   <servlet-name>ShowMsg</servlet-name>
   <servlet-class>ShowMessage</servlet-class>
</servlet>
```

Ln 154, Col 21

Initialization Parameters

- Sometimes we want to initialize a servlet when it is first loaded by the servlet engine.
  - configuration
  - specialization
  - customization
- Initialization parameters are usually handled in the
  - `init()` method.

We will see examples for both methods

The init() methods

- `void init()`
  - Override this one to initialize a servlet
- `void init( ServletConfig config)`
  - Do not override this one
  - If you do override, you must call `super.init(config)`
  - You can access the `ServletConfig`'s methods through the servlet itself, since the interface is implemented by `GenericServlet`

HelloServlet

- We want to modify `HelloServlet` servlet.
- Takes two initialization parameters
  - `bgcolor, fgcolor`

- http://localhost/csj/HelloServlet2

HelloServlet revisited (cont.)

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

public class HelloServlet2 extends HttpServlet {

  protected String bgcolor;
  protected String fgcolor;

  public void init() {
    bgcolor = getInitParameter("bgcolor");
    fgcolor = getInitParameter("fgcolor");
  }
   <doGet() method>
}
```

HelloServlet revisited (cont.)

Complete source code file for HelloServlet2

HelloServlet2 - Notepad

File Edit Format View Help

```
public class HelloServlet2 extends HttpServlet {

   protected String bgcolor = null;
   protected String fgcolor = null;


   /**
    * init gets initialization parameters from web.xml file
    */
   public void init() {
      bgcolor = getInitParameter("bgcolor");
      fgcolor = getInitParameter("fgcolor");
   }

   /** Processes requests for HTTP <code>GET</code> method.
   * @param request servlet request
   * @param response servlet response
   */
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, java.io.IOException {
      response.setContentType("text/html");
      java.io.PrintWriter out = response.getWriter();

      out.println("<html>");
      out.println("<head>");
      out.println("<title>Hello Servlet</title>");
      out.println("</head>");

      if (bgcolor != null) {
         out.println("<body bgcolor=\"" + bgcolor + "\">");
      } else {
         out.println("<body>");
      }

      if (fgcolor != null) {
         out.println("<h1><font color=\"" + fgcolor
            + "\"> Hello World!</font></h1>");
      } else {
         out.println("<h1>Hello World!</h1>");
      }
      out.println("</body>");
      out.println("</html>");
```

Ln 1, Col 1

Deploy HelloServlet2

- Copy the new HelloServlet2.class file to csj directory
- Edit the `web.xml` file to refer to the new servlet.
- Notice the difference between:
  - http://localhost/csj/HelloServlet2
  - http://localhost/csj/hello-green

Servlet Definition

New, in `web.xml`:

```
<servlet>
    <servlet-name>hello-green</servlet-name>
    <servlet-class>HelloServlet2</servlet-class>
    <init-param>
        <param-name>bgcolor</param-name>
        <param-value>white</param-value>
    </init-param>
    <init-param>
        <param-name>fgcolor</param-name>
        <param-value>green</param-value>
    </init-param>
</servlet>
```

Servlet Definition – Web.xml file

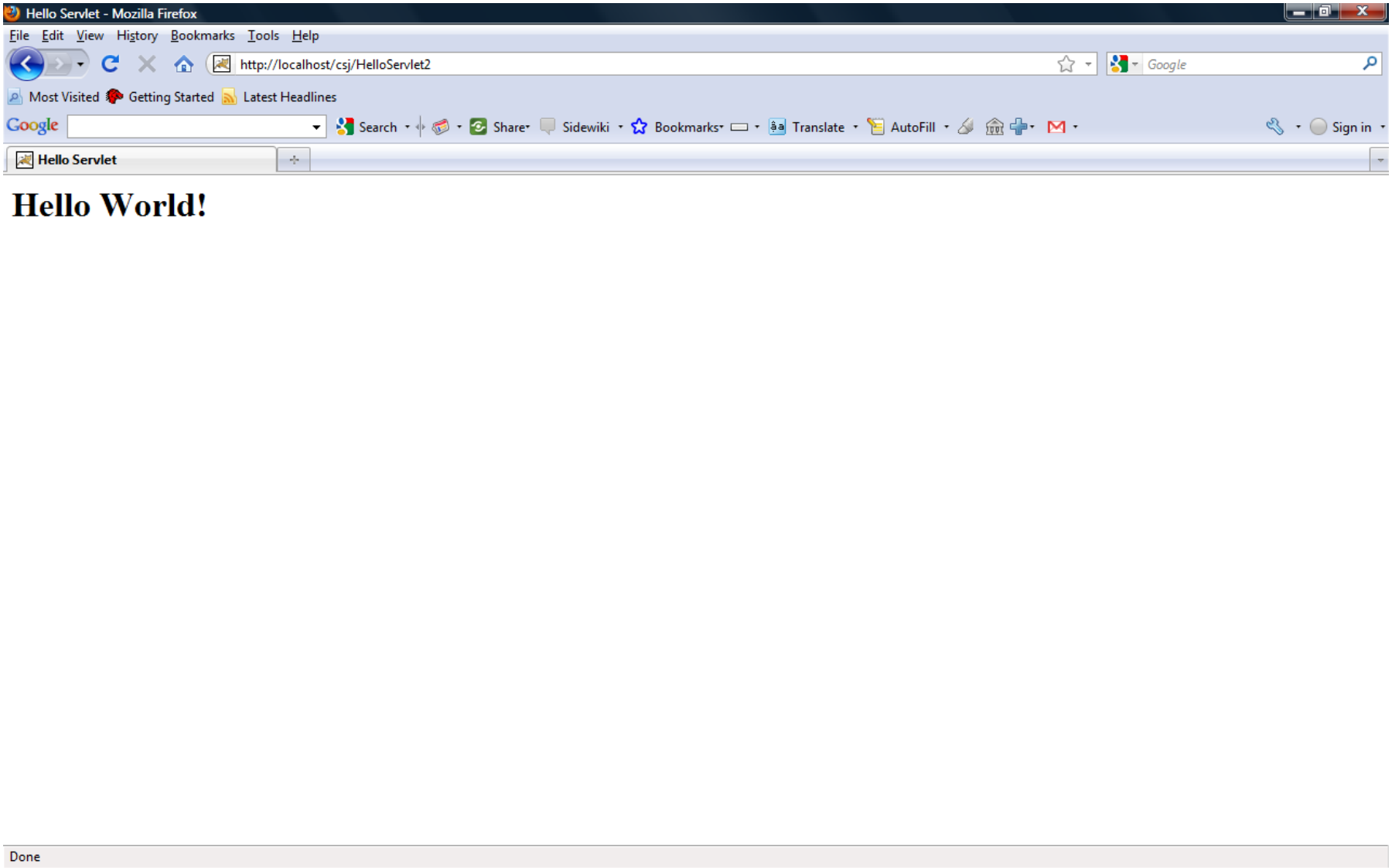
web - Notepad

File Edit Format View Help

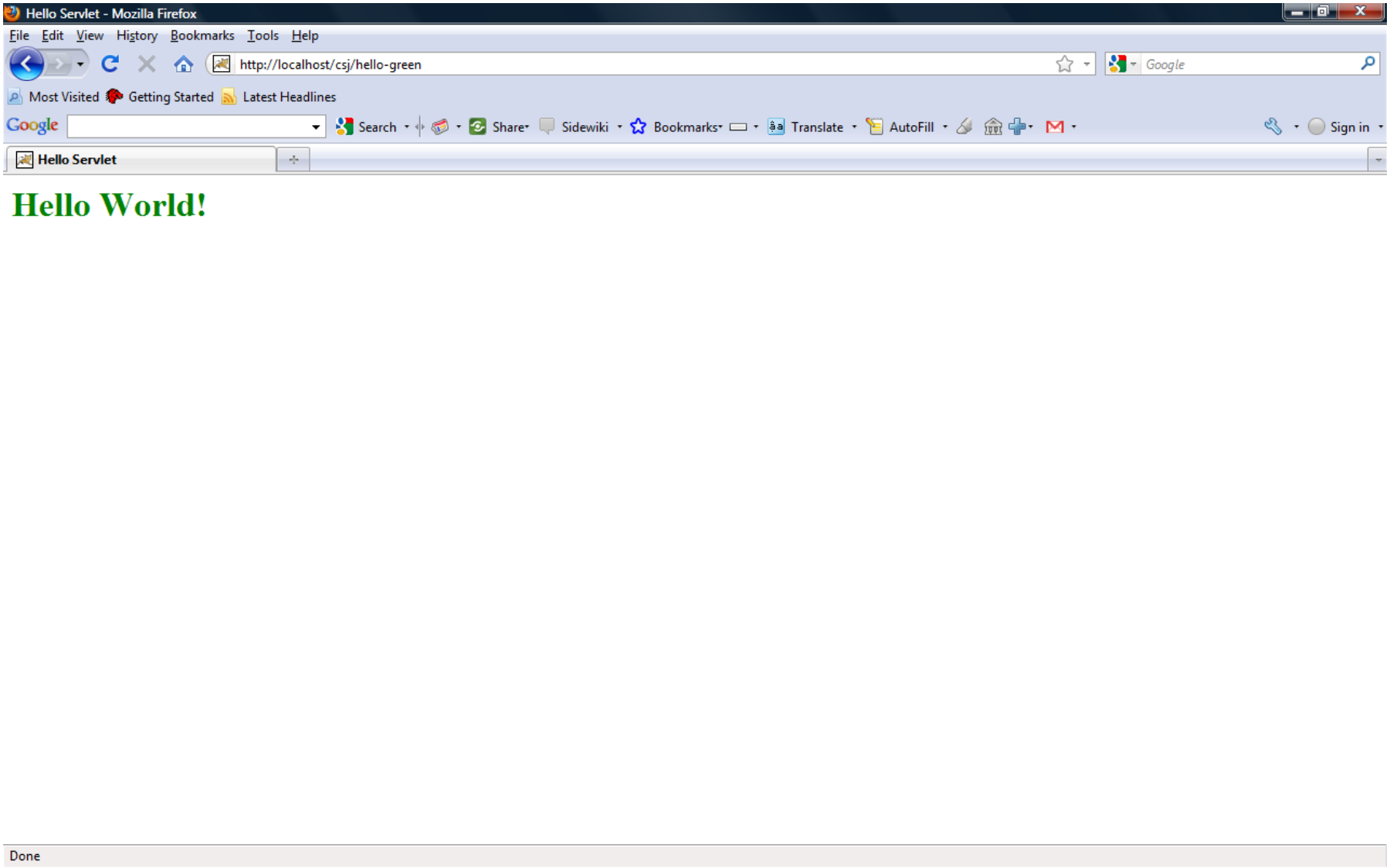
```
<servlet>
   <servlet-name>HelloServlet2</servlet-name>
   <servlet-class>HelloServlet2</servlet-class>
</servlet>


<servlet>
   <servlet-name>hello-green</servlet-name>
   <servlet-class>HelloServlet2</servlet-class>
   <init-param>
      <param-name>bgcolor</param-name>
      <param-value>white</param-value>
   </init-param>
   <init-param>
      <param-name>fgcolor</param-name>
      <param-value>green</param-value>
   </init-param>



</servlet>


<servlet>
   <servlet-name>hello-red</servlet-name>
   <servlet-class>HelloServlet2</servlet-class>
   <init-param>
      <param-name>bgcolor</param-name>
      <param-value>red</param-value>
```

Ln 158, Col 13

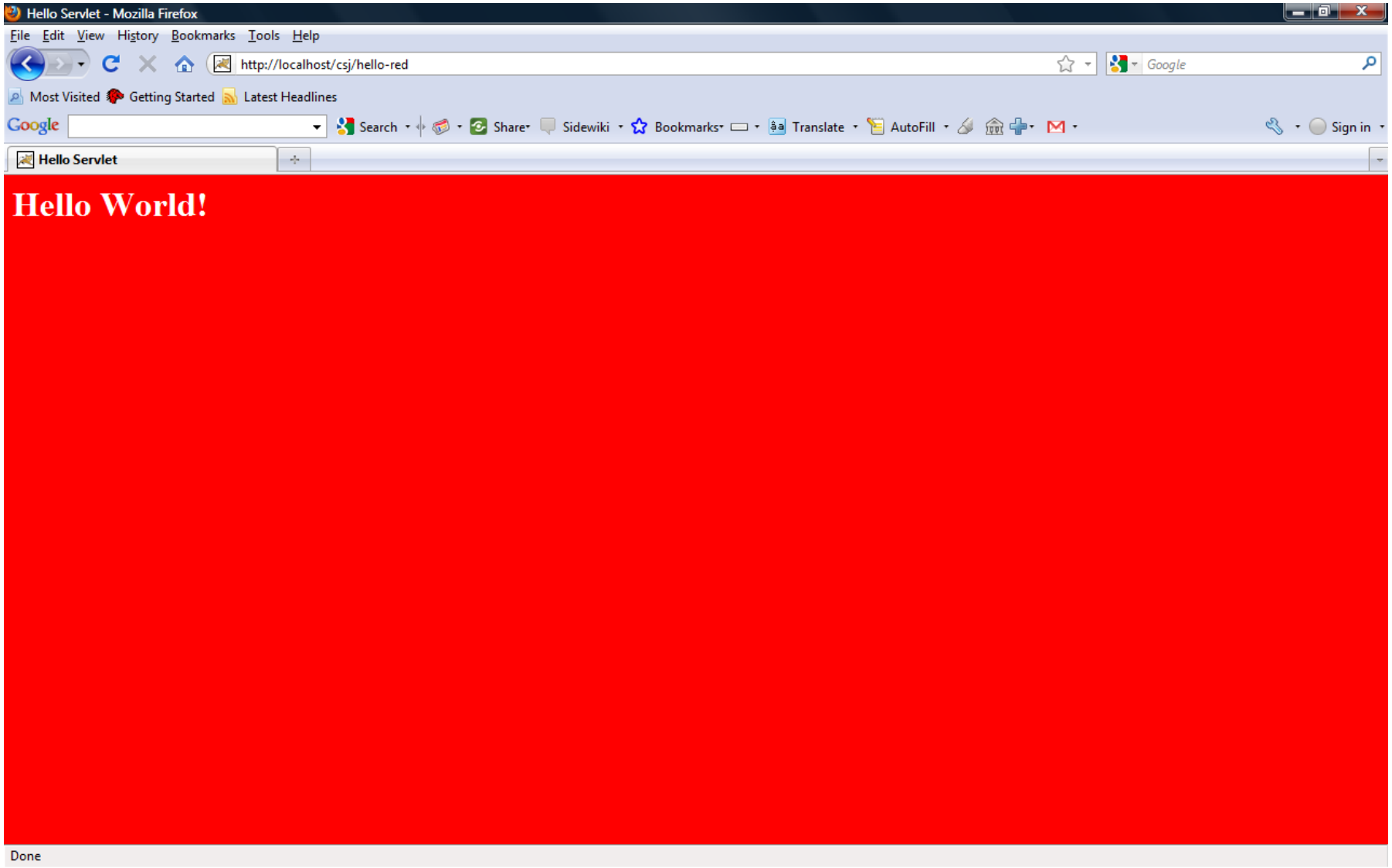
Invoking with a Servlet Mapping



Invoking with a Servlet Mapping



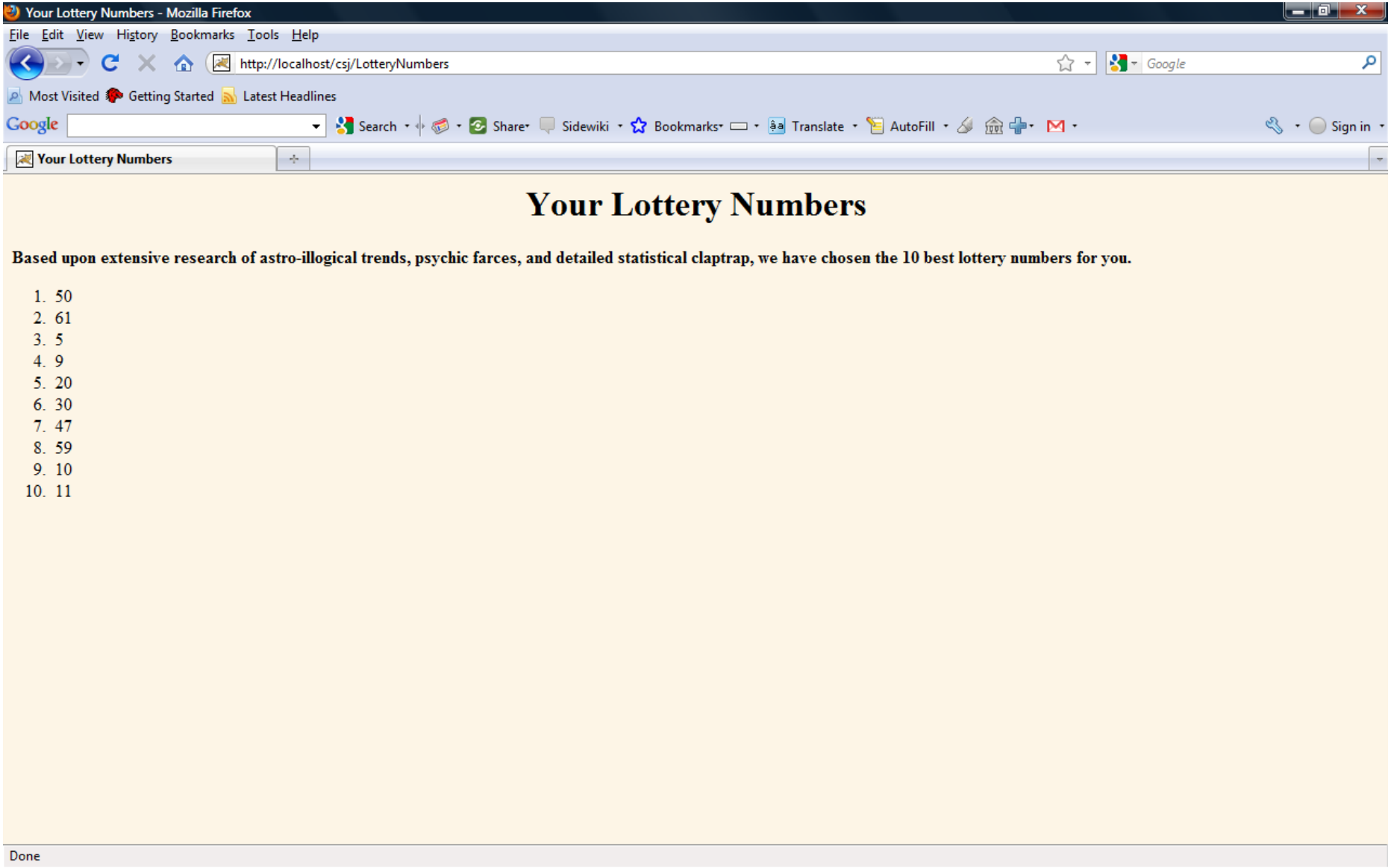
Invoking with a Servlet Mapping



LotteryNumbers Servlet

- How can we use If-Modified-Since request header in generating lottery numbers Servlet?
- The standard service method compares the date against any date specified in the If-Modified-Since request header. If the getLastModified date is later or if there is no If-Modified-Since header, the doGet method is called normally. But if the getLastModified date is the same or earlier, the service method sends back a 304 (Not Modified) response and does not call doGet. The browser should use its cached version of the page in such a case.
- Lets look at the code for LotteryNumbers.java

LotteryNumbers Servlet



Your Lottery Numbers

Based upon extensive research of astro-illogical trends, psychic farces, and detailed statistical claptrap, we have chosen the 10 best lottery numbers for you.

1. 50
2. 61
3. 5
4. 9
5. 20
6. 30
7. 47
8. 59
9. 10
10. 11

SingleThreadModel To Prevent Race Comditions

- In principle, you can prevent multithreaded access by having your servlet implement the SingleThreadModel interface, as below.
  - public class YourServlet extends HttpServlet
  - **implements SingleThreadModel** {
  - ...
  - }
- If you implement this interface, the system guarantees that there is never more than one request thread accessing a single instance of your servlet. In most cases, it does so by queuing all the requests and passing them one at a time to a single servlet instance.

- Lets look at the code for UserIDs.java

SingleThreadModel To Prevent Race Comditions

- UserIDs.java Servlet suffers from Race Conditions, Why?
  - Servlet that attempts to give each user a unique user ID. However, because it fails to synchronize access to the nextID field, it suffers from race conditions: two users could get the same ID.
- See the source code for UserIDs.java Servlet in the next slide …

SingleThreadModel To Prevent Race Comditions

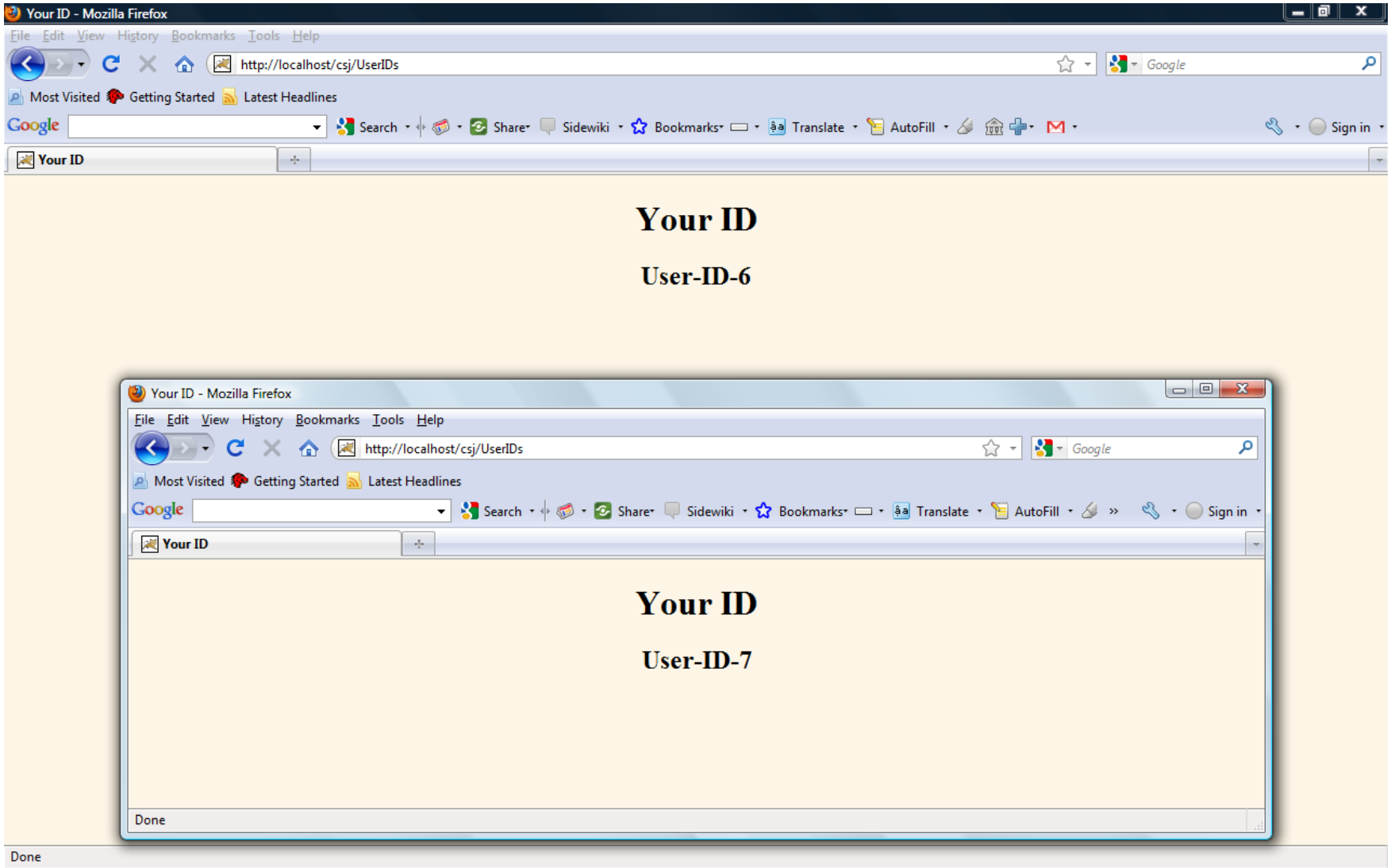
UserIDs - Notepad

File Edit Format View Help

```
 *  Taken from Core Servlets and JavaServer Pages 2nd Edition
 *  from Prentice Hall and Sun Microsystems Press,
 *  http://www.coreservlets.com/.
 *  &copy; 2003 Marty Hall; may be freely used or adapted.
 */

public class UserIDs extends HttpServlet {
  private int nextID = 0;

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Your ID";
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
      "Transitional//EN\">\n";
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
                "<CENTER>\n" +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>" + title + "</H1>\n");
    String id = "User-ID-" + nextID;
    out.println("<H2>" + id + "</H2>");
    nextID = nextID + 1;
    out.println("</BODY></HTML>");
  }
}
```

Ln 28, Col 1

UserIDs Servlet



Code that adds an instance variable to the EmailServlet class

```
package email;

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

public class AddToEmailListServlet2 extends HttpServlet
{
    // declare an instance variable for the page
    int globalCount;
        // instance variables are not thread-safe

    public void init() throws ServletException
    {
        globalCount = 0; // initialize the instance variable
    }
```

Code that adds an instance variable to the EmailServlet class (cont.)

```
protected void doPost(
    HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException
{
    // update global count variable
    globalCount++;   // this is not thread-safe

    // send response to browser
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println(
      "<!doctype html public "
    + "\"-//W3C//DTD HTML 4.0 Transitional//EN\">\n"
    + "<html>\n"
    + "<head>\n"
    + "  <title>Murach's Java Servlets and "
    + "JSP</title>\n"
    + "</head>\n"
    + "<body>\n"
```

Code that adds an instance variable to the EmailServlet class (cont.)

```
        + "<h1>Thanks for joining our email list</h1>\n"
        + "<p>This page has been accessed "
        +  globalCount + " times.</p>"
        + "</body>\n"
        + "</html>\n");

        out.close();
    }
}
```

How to code instance variables

- An *instance variable* of a servlet belongs to the one instance of the servlet and is shared by any threads that request the servlet.

- Instance variables are not *thread-safe*. In other words, two threads may conflict when they try to read, modify, and update the same instance variable at the same time, which can result in lost updates or other problems.

Common servlet problems

Problem	Possible solutions
The servlet won't compile	Make sure the compiler has access to the JAR files for all necessary APIs.
	Make sure the Java classes that you code are stored in the correct directories with the correct package statements.
The servlet won't run	Make sure the web server is running.
	Make sure you're using the correct URL.
Changes to the servlet aren't showing up	Make sure servlet reloading is on.
	Redeploy the application.
	Restart the server so it reloads all applications.
The page doesn't display correctly	Select the Source or Page Source command from your browser's View menu to view the HTML code and identify the problem. Then, you can fix the problem in the servlet.

How to print debugging data to the console

- When you're testing an application on your local system, you can use the println method of the System.out or System.err objects to display debugging messages on the console for the servlet engine.
- When you use debugging messages to display variable values, it's a good practice to include the servlet name and the variable name so the messages are easy to interpret.
- When you use an IDE like NetBeans, debugging data that is written to a server console is also available in an output window.

Two methods of the HttpServlet class used to log errors

Method	Description
`log(String message)`	Writes the specified message to the server's log file.
`log(String message, Throwable t)`	Writes the specified message to the server's log file, followed by the stack trace for the exception.

Computer ▸ Local Disk (C:) ▸ apache-tomcat-7.0.34 ▸ logs

Search logs

Organize ▾ Open ▾ Print Burn New folder

Name	Date modified	Type	Size
localhost_access_log.2013-09-18	9/18/2013 11:23 A...	Text Document	3 KB
catalina.2013-09-18	9/18/2013 11:23 A...	Text Document	0 KB
host-manager.2013-09-18	9/18/2013 11:23 A...	Text Document	0 KB
localhost.2013-09-18	9/18/2013 11:23 A...	Text Document	2 KB
manager.2013-09-18	9/18/2013 11:23 A...	Text Document	0 KB
localhost_access_log.2013-09-16	9/16/2013 7:53 PM	Text Document	2 KB
catalina.2013-09-16	9/16/2013 7:10 PM	Text Document	41 KB
localhost.2013-09-16	9/16/2013 7:10 PM	Text Document	4 KB
host-manager.2013-09-16	9/16/2013 7:02 PM	Text Document	0 KB
manager.2013-09-16	9/16/2013 7:02 PM	Text Document	0 KB
catalina.2013-09-13	9/13/2013 12:17 PM	Text Document	20 KB
localhost.2013-09-13	9/13/2013 12:03 PM	Text Document	2 KB
localhost_access_log.2013-09-13	9/13/2013 12:03 PM	Text Document	0 KB
host-manager.2013-09-13	9/13/2013 12:03 PM	Text Document	0 KB
manager.2013-09-13	9/13/2013 12:03 PM	Text Document	0 KB
catalina.2013-09-10	9/10/2013 7:24 PM	Text Document	180 KB
localhost.2013-09-10	9/10/2013 7:24 PM	Text Document	17 KB
localhost_access_log.2013-09-10	9/10/2013 7:23 PM	Text Document	5 KB
manager.2013-09-10	9/10/2013 6:51 PM	Text Document	1 KB
host-manager.2013-09-10	9/10/2013 6:27 PM	Text Document	0 KB
localhost_access_log.2013-07-22	7/22/2013 9:28 PM	Text Document	28 KB
catalina.2013-07-22	7/22/2013 9:27 PM	Text Document	63 KB
localhost.2013-07-22	7/22/2013 9:27 PM	Text Document	14 KB
host-manager.2013-07-22	7/22/2013 7:10 PM	Text Document	0 KB
manager.2013-07-22	7/22/2013 7:10 PM	Text Document	0 KB
catalina.2013-05-14	5/14/2013 10:00 PM	Text Document	20 KB
localhost_access_log.2013-05-14	5/14/2013 10:00 PM	Text Document	8 KB
localhost.2013-05-14	5/14/2013 8:12 PM	Text Document	2 KB
host-manager.2013-05-14	5/14/2013 8:12 PM	Text Document	0 KB
manager.2013-05-14	5/14/2013 8:12 PM	Text Document	0 KB
localhost_access_log.2013-05-01	5/1/2013 8:39 PM	Text Document	8 KB
catalina.2013-05-01	5/1/2013 6:43 PM	Text Document	59 KB

localhost_access_log.2013-09-18 Date modified: 9/18/2013 11:23 AM Date created: 9/18/2013 11:23 AM
Text Document Size: 2.81 KB

Mozilla Firefox

http://localhost/csj/DebugHelloServlet

localhost/csj/DebugHelloServlet

```
Hello World
```

Tomcat

```
es
Sep 18, 2013 11:23:13 AM org.apache.catalin
INFO: Deploying web application directory (
anager
Sep 18, 2013 11:23:13 AM org.apache.catalin
INFO: Deploying web application directory (
amples
Sep 18, 2013 11:23:13 AM org.apache.catalin
INFO: Deploying web application directory (
-servlet-to-jsp
Sep 18, 2013 11:23:13 AM org.apache.catalin
INFO: Deploying web application directory (
r
Sep 18, 2013 11:23:13 AM org.apache.catalin
INFO: Deploying web application directory (
Sep 18, 2013 11:23:13 AM org.apache.catalin
INFO: Deploying web application directory (
Sep 18, 2013 11:23:14 AM org.apache.coyote
INFO: Starting ProtocolHandler ["http-bio-8
Sep 18, 2013 11:23:14 AM org.apache.coyote
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Sep 18, 2013 11:23:14 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 2999 ms
DebugHelloServlet : This is the trace to stdout in doget
```

localhost_access_log.2013-09-18 - Notepad

File Edit Format View Help

```
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:22 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:23 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:24 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:25 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:26 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:34 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:35 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:36 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:37 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:24:51 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:02 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:10 -0500] "GET /csj/LotteryNumbers?x HTTP/1.1" 200 479
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:16 -0500] "GET /csj/LotteryNumbers?x=10 HTTP/1.1" 200 479
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:24 -0500] "GET /csj/LotteryNumbers?x=10 HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:24 -0500] "GET /csj/LotteryNumbers?x=10 HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:50 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:53 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:54 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:25:55 -0500] "GET /csj/LotteryNumbers HTTP/1.1" 304 -
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:40:06 -0500] "GET / HTTP/1.1" 200 12834
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:40:22 -0500] "GET /ServletThatISNotThere HTTP/1.1" 404 993
0:0:0:0:0:0:0:1 - - [18/Sep/2013:11:42:27 -0500] "GET /DebugHelloServlet HTTP/1.1" 404 985
```

Ln 1, Col 1

Secure Search

McAfee

DebugHelloServlet - Notepad

File Edit Format View Help

```
/*
 * DebugHelloServlet.java
 *
 */



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


public class DebugHelloServlet extends HttpServlet {

    /** Processes requests for HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, java.io.IOException {
        java.io.PrintWriter out = response.getWriter();
        out.println("Hello World");
        out.close();

                // debug info
                System.out.println("DebugHelloServlet : This is the trace to stdout in doget");
                log("trace inside the doget in DebugHelloServlet");
    }


}
```