

Servlets





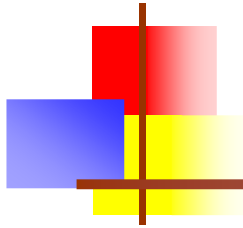
Servers

- A **server** is a computer that responds to requests from a **client**
 - Typical requests: provide a web page, upload or download a file, send email
- A **server** is also the software that responds to these requests; a **client** could be the browser or other software making these requests
- Typically, your little computer is the client, and someone else's big computer is the server
 - However, any computer can be a server
 - It is not unusual to have server software and client software running on the same computer

The Apache logo graphic consists of a red square, a blue square, and a yellow square arranged in a 2x2 grid, with a vertical line passing through the center of the squares.

Apache

- Apache is a *very* popular server
 - 66% of the web sites on the Internet use Apache
- Apache is:
 - Full-featured and extensible
 - Efficient
 - Robust
 - Secure (at least, more secure than other servers)
 - Up to date with current standards
 - Open source
 - Free



Ports

- A **port** is a connection between a server and a client
 - Ports are identified by positive integers
 - A port is a software notion, not a hardware notion, so there may be very many of them
- A service is associated with a specific port

CGI Scripts

- CGI stands for “Common Gateway Interface”

Client sends a request to server

Server starts a CGI script

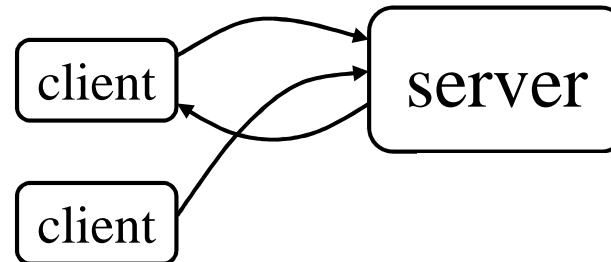
Script computes a result for server
and quits

Server returns response to client

Another client sends a request

Server starts the CGI script again

Etc.



Servlets

- A **servlet** is like an applet, but on the server side

Client sends a request to server

Server starts a servlet

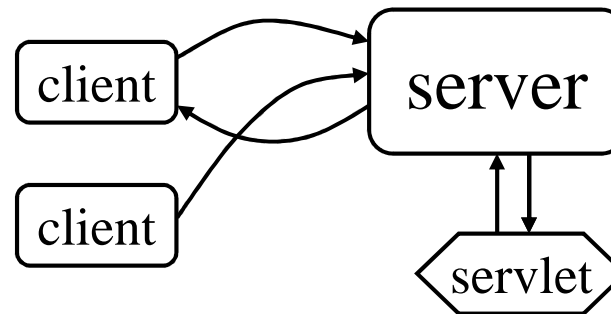
Servlet computes a result for server and *does not quit*

Server returns response to client

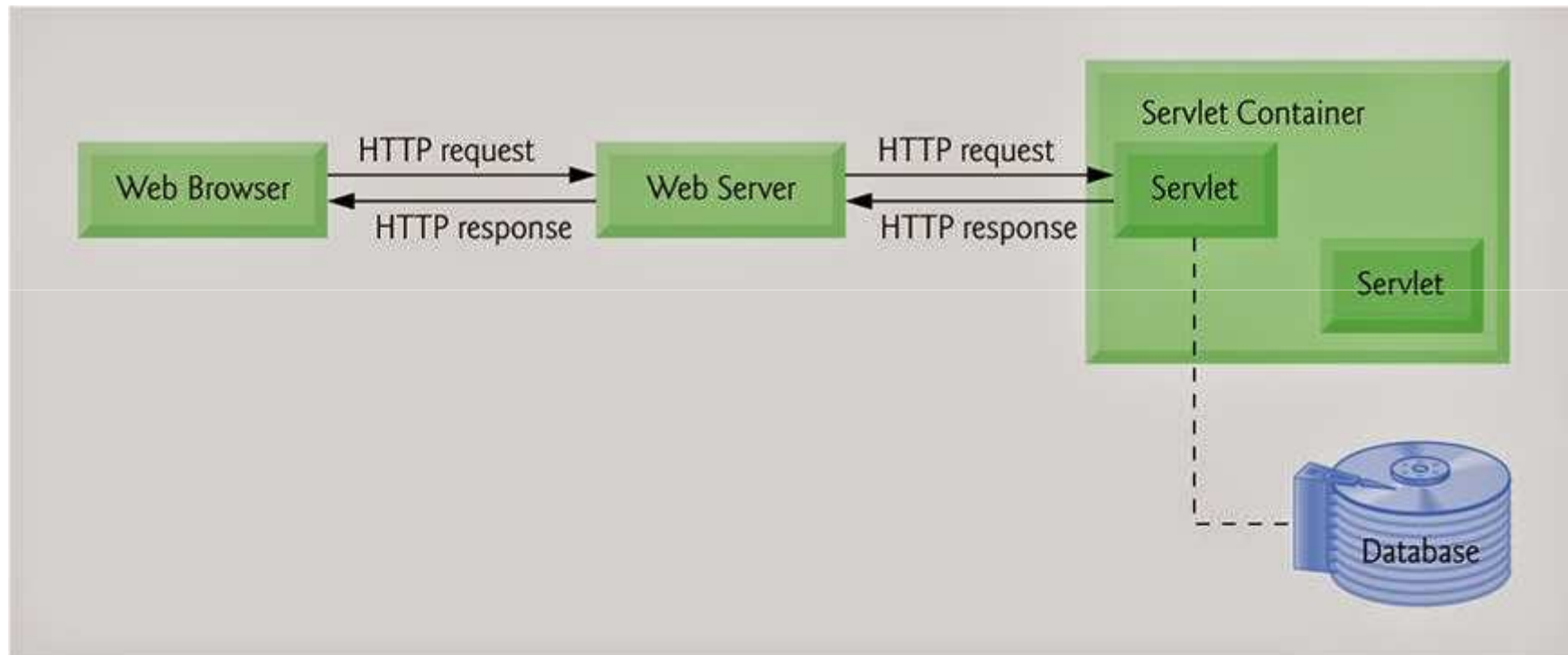
Another client sends a request

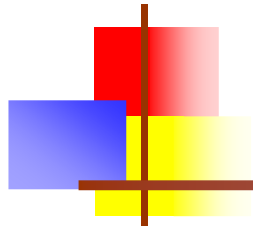
Server calls the servlet again

Etc.



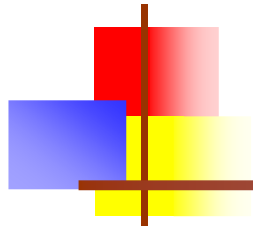
Servlet Architecture





Servlet Architecture

- Servlets read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.



Servlets vs. CGI scripts

- Advantages:
 - Running a servlet doesn't require creating a separate process each time
 - A servlet stays in memory, so it doesn't have to be reloaded each time
 - There is only one instance handling multiple requests, not a separate instance for every request
 - Untrusted servlets can be run in a “sandbox”
- Disadvantage:
 - Less choice of languages (CGI scripts can be in any language)

The logo consists of a red square, a blue square, and a yellow square arranged in a 2x2 grid, with a vertical line passing through the center of the squares.

Tomcat

- **Tomcat** is the Servlet Engine than handles servlet requests for Apache
 - Tomcat is a “helper application” for Apache
 - It’s best to think of Tomcat as a “servlet container”
- Apache can handle many types of web services
 - Apache can be installed without Tomcat
 - Tomcat can be installed without Apache
- It’s easier to install Tomcat standalone than as part of Apache
 - By itself, Tomcat can handle web pages, servlets, and JSP
- Apache and Tomcat are open source (and therefore free)



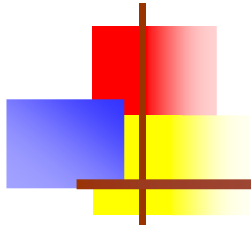
Servlets

- A **servlet** is any class that implements the `javax.servlet.Servlet` interface
 - In practice, most servlets extend the `javax.servlet.http.HttpServlet` class
 - Some servlets extend `javax.servlet.GenericServlet` instead
- Servlets, like applets, usually lack a **main** method, but must implement or override certain other methods



Important servlet methods, I

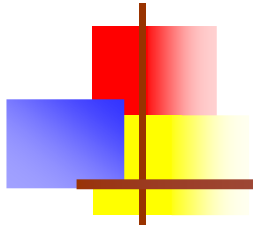
- When a servlet is first started up, its **init(ServletConfig *config*)** method is called
 - **init** should perform any necessary initializations
 - **init** is called only once, and does not need to be thread-safe
- Every servlet request results in a call to **service(ServletRequest *request*, ServletResponse *response*)**
 - **service** calls another method depending on the type of service requested
 - Usually you would override the called methods of interest, not **service** itself
 - **service** handles multiple simultaneous requests, so it and the methods it calls *must be thread safe*
- When the servlet is shut down, **destroy()** is called
 - **destroy** is called only once, but must be thread safe (because other threads may still be running)



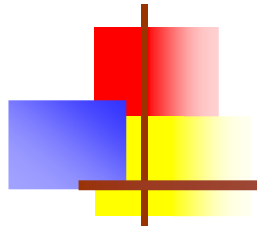
Life Cycle of Servlet

- Three methods are central to the life cycle of a servlet.
- They include:
 - **init()**
 - **service()**
 - **destroy()**
- **Steps:**
 - First, assume that a user enters a **Uniform Resource Locator (URL)** to a web browser.
 - Second, this HTTP request is received by the web server. The server maps this request to a particular servlet.
 - Third, the server invokes the **init() method of the servlet. This method is invoked only** when the servlet is first loaded into memory.
 - Fourth, the server invokes the **service() method of the servlet. This method is called** to process the HTTP request.
 - Finally, the server may decide to unload the servlet from its memory..The server calls the **destroy() method to relinquish any resources such as file handles that are allocated for the servlet.**

Servlet Terminology

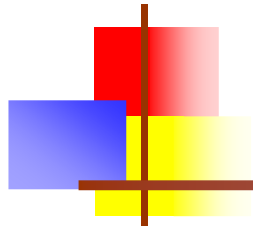


- HTTP
- HTTP Request Types
- Difference between Get and Post method
- Container
- Server and Difference between web server and application server
- Content Type
- Introduction of XML
- Deployment



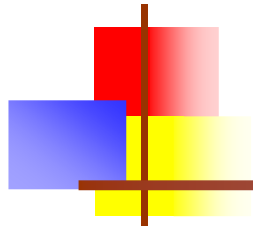
HTTP (Hyper Text Transfer Protocol)

- Http is the protocol that allows web servers and browsers to exchange data over the web.
- It is a request response protocol.
- Http uses reliable TCP connections by default on TCP port 80.
- It is stateless means each request is considered as the new request. In other words, server doesn't recognize the user by default.



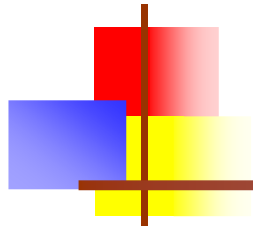
Http Request method

HTTP Request	Description
GET	Asks to get the resource at the requested URL.
POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
PUT	Says to put the enclosed info (the body) at the requested URL.
DELETE	Says to delete the resource at the requested URL.
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond



Difference between get and post

GET	POST
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
3) Get request can be bookmarked	Post request cannot be bookmarked
4) Get request is idempotent . It means second request will be ignored until response of first request is delivered.	Post request is non-idempotent
5) Get request is more efficient and used more than Post	Post request is less efficient and used less than get.



Container

- It provides runtime environment for JavaEE (j2ee) applications.
- It performs many operations that are given below:
 - Life Cycle Management
 - Multithreaded support
 - Object Pooling
 - Security etc.



Server

- There are two types of servers:
 - Web Server
 - Application Server

Web Server

- Web server contains only web or servlet container. It can be used for servlet, jsp, struts, jsf etc. It can't be used for EJB.
 - Example of Web Servers are: **Apache Tomcat** and **Resin**.

Application Server

- Application server contains Web and EJB containers. It can be used for servlet, jsp, struts, jsf, ejb etc.
- Example of Application Servers are:
 - **JBoss** Open-source server from JBoss community.
 - **Glassfish** provided by Sun Microsystems. Now acquired by Oracle.
 - **Weblogic** provided by Oracle. It more secured.
 - **Websphere** provided by IBM.



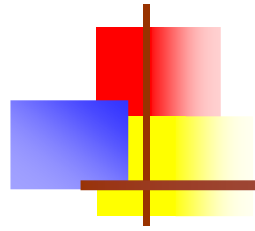
Content Type

- Content Type is also known as MIME (Multipurpose internet Mail Extension) Type. It is a **HTTP header** that provides the description about what are you sending to the browser.
- There are many content types:
 - text/html
 - text/plain
 - application/msword
 - application/vnd.ms-excel
 - application/jar
 - application/pdf
 - application/octet-stream
 - application/x-zip
 - images/jpeg
 - video/quicktime etc.

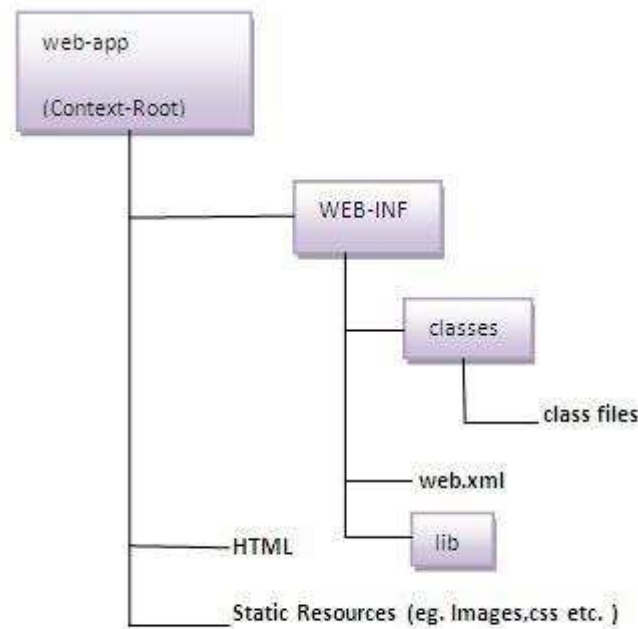


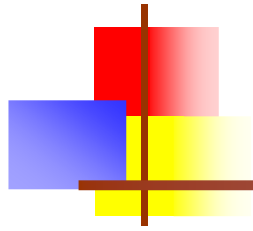
Steps to create a servlet using Apache Tomcat

- Create a directory structure
- Create a Servlet
- Compile the Servlet
- Create a deployment descriptor
- Start the server and deploy the project
- Access the servlet



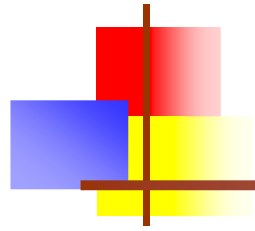
Create a directory structure





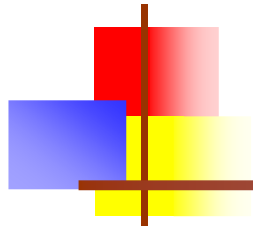
Create a Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FirstServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException
    {
        String n1=req.getParameter("uname");
        PrintWriter out=res.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>This a response page</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Hello:"+n1);
        out.println("</body>");
        out.println("</html>");
    }
}
```



Compile servlet

Jar file	Server
1) servlet-api.jar	Apache Tomcat
2) weblogic.jar	Weblogic
3) javaee.jar	Glassfish
4) javaee.jar	JBoss



Create Deployment Descriptor

- **<web-app>** represents the whole application.
- **<servlet>** is sub element of **<web-app>** and represents the servlet.
- **<servlet-name>** is sub element of **<servlet>** represents the name of the servlet.
- **<servlet-class>** is sub element of **<servlet>** represents the class of the servlet
- **<servlet-mapping>** is sub element of **<web-app>**. It is used to map the servlet
- **<url-pattern>** is sub element of **<servlet-mapping>**. This pattern is used at client side to invoke the servlet.