

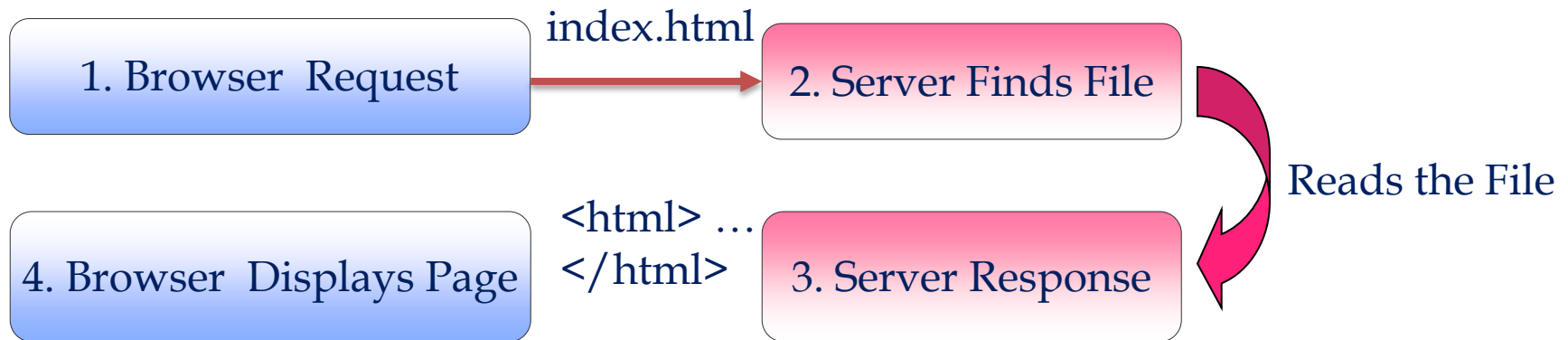
Servlets

Overview

- ◆ Introduction to Servlets
- ◆ Servlets Life Cycle
- ◆ Servlet Architecture
- ◆ Life Cycle of Servlet
- ◆ Deploying Servlets
- ◆ Web Container
- ◆ Servlet Context & Servlet Config
- ◆ Servlet Chaining
- ◆ Session Management

Introduction to Servlets

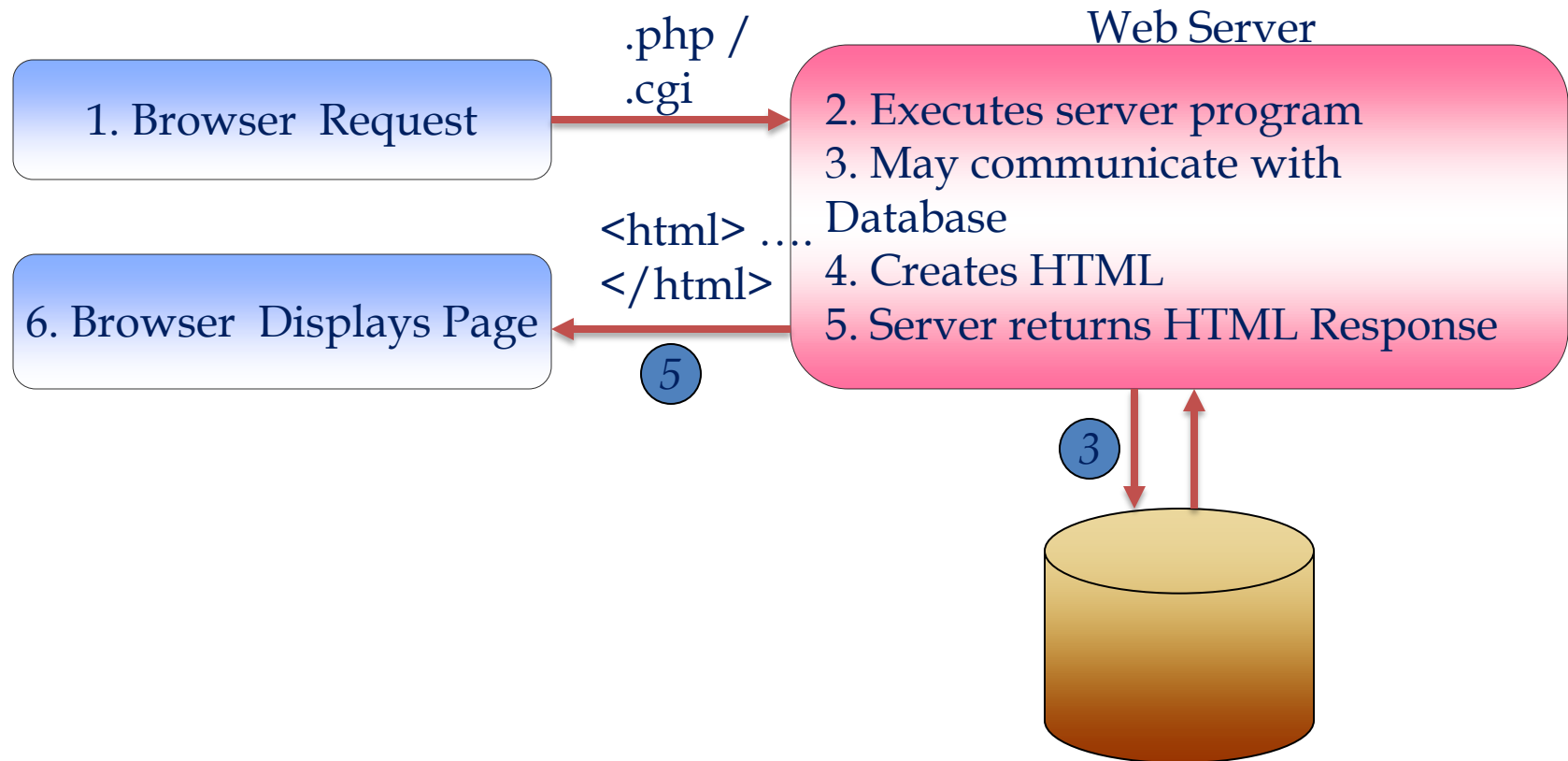
- ◆ Static HTTP transaction - Browser requests for index.html, and Server responds with HTML file



- ◆ HTTP(Hyper Text Transport Protocol)is the protocol that clients and servers use on the webto communicate

Introduction to Servlets

- ◆ Dynamic HTTP transaction - Browser requests OrderServlet.class, server runs program that creates HTML, server returns HTML to browser

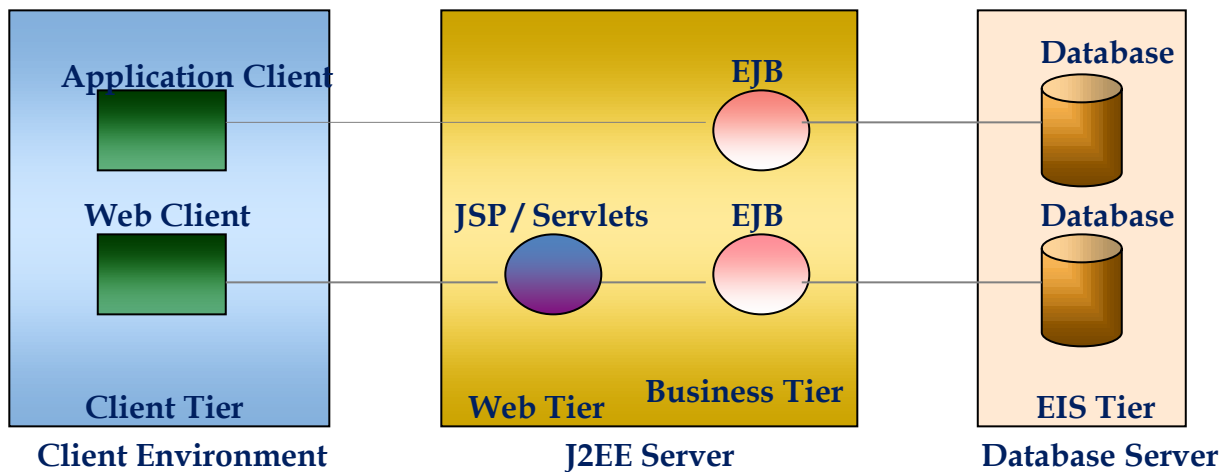


Webserver

- ◆ A computer having server software installed within it which serves web pages
- ◆ A program that uses client/ server model and World Wide Web's Hypertext Transfer Protocol (HTTP)
- ◆ Responsible for accepting HTTP requests from clients (web browsers) and serving HTTP responses
- ◆ Popular web servers
 - Apache HTTP Server (Apache)
 - Microsoft Internet Information Server (IIS)
 - Sun Java System Web Server

Java server-side web components

- ◆ A web component is a software entity that runs on a web server
- ◆ The J2EE specification defines two types of web components
 - Servlets , Java Server Pages(JSPs)



J2EE Application N-Tier Architecture

Servlets

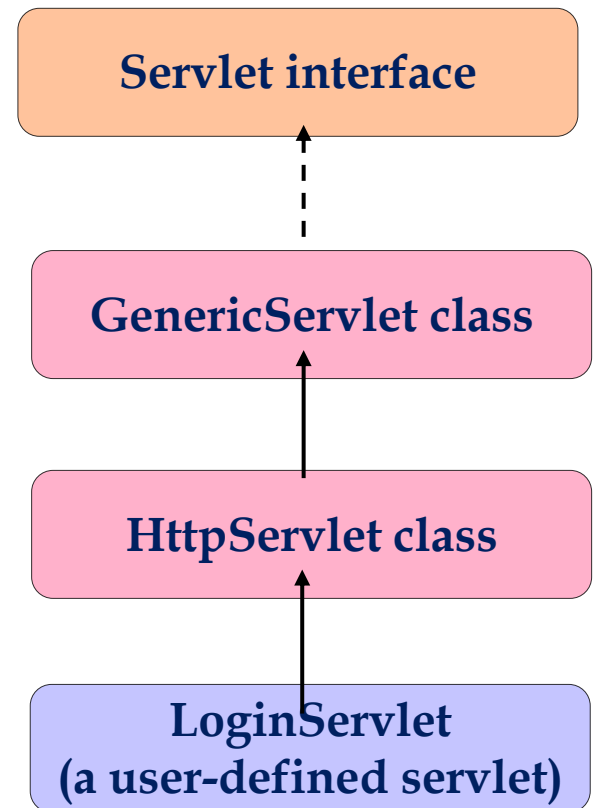
- ◆ A Java class that runs on a web server and dynamically handles client requests
- ◆ Extends the functionality of a web server by receiving client requests and dynamically generating a response
- ◆ Servlets are Java-based, hence platform independent

Uses of Servlets

- ◆ Processing and/or storing data submitted by a HTML form
 - Example: Processing data of a login form
- ◆ Providing dynamic content
 - Example: Returning results of a database query to the client
- ◆ Managing state information
 - Example: Online shopping managing shopping carts for multiple customers

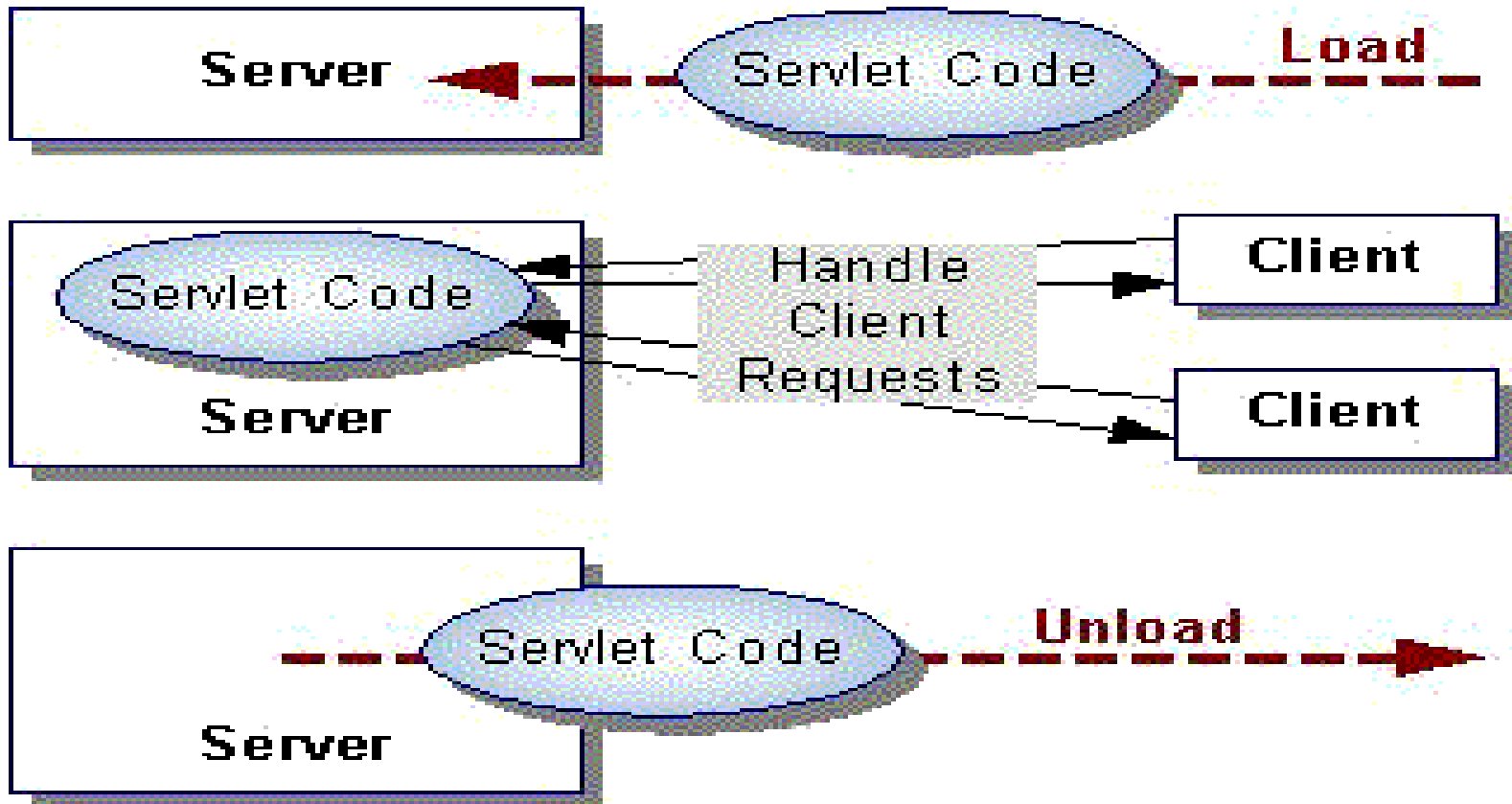
Servlet Architecture

- ◆ Java Servlet API are in packages `javax.servlet` and `javax.servlet.http`
 - Provide interfaces and classes for writing servlets
- ◆ All servlets must implement Servlet interface
 - Defines life-cycle methods
- ◆ Extend `GenericServlet` class
 - To implement generic services
- ◆ Extend `HttpServlet` class
 - To implement HTTP-specific services



Life Cycle of a Servlet

- ◆ An HTTP servlet's life cycle is controlled by the web container where it is deployed



Servlet Interface

- ◆ Provides methods that manage the servlet and its communications with clients
 - `init(ServletConfig)`
 - `service(ServletRequest, ServletResponse)`
 - `destroy()`
 - `getServletInfo()`

Servlet Life Cycle

- ◆ Interaction between a web server, servlet and a client is controlled using the life-cycle methods
- ◆ A servlet's life cycle methods are
 - `init()`
 - `service()`
 - `destroy()`
- ◆ The `init()` and `destroy()` methods will be called only once during the life time of a Servlet
- ◆ The `service()` , `doGet()` , `doPost()` will be called as many times as requests are received by the web container

Servlet LifeCycle

- ◆ Web container initializes servlet after web container loads and instantiates servlet class
- ◆ Override init method of Servlet interface if you want the servlet to
 - Read persistent configuration data
 - Initialize resources
 - Perform any other one-time activities

Servlet LifeCycle

- ◆ Once servlet is loaded and initialized, the servlet is able to handle client requests
- ◆ Web container processes the requests in servlet's service method
- ◆ Every time the server receives an incoming request for a servlet, it generates a new thread and calls the service method
- ◆ The service method then checks the HTTP request type and calls the appropriate doXXX method
 - Syntax: `public void service(ServletRequest req, ServletResponse res)`

Servlet LifeCycle

- ◆ Server may unload a servlet instance if
 - The servlet is idle for some time (default is 30 minutes)
 - The web application is undeployed
 - The server shuts down
- ◆ Calls destroy method only once before removing the servlet
 - Syntax: `public void destroy()`

Servlet LifeCycle

- ◆ Methods doGet() and doPost() in HttpServlet class receive appropriate client request, and formats a response using
 - HttpServletRequest object
 - HttpServletResponse object
- ◆ Both these objects are created by the servlet container

Syntax :

```
public void doPost(HttpServletRequest req, HttpServletResponse  
res)
```

throws IOException, ServletException

```
public void doGet(HttpServletRequest req, HttpServletResponse res)  
throws IOException, ServletException
```


HttpServletRequest interface

- ◆ The HttpServletRequest object incorporates communication from client to servlet
- ◆ Provides methods that allow to retrieve incoming information
 - For example: HTTP request headers, form data or a client's hostname
- ◆ Methods to read parameters from a form
 - `getParameter(String pname)`
 - `getParameterNames()`
 - `getParameterValues(String pname)`

HttpServletResponse Interface

- ◆ The HttpServletResponse object incorporates communication from servlet to client
- ◆ Allows to specify outgoing information
 - For example: response headers and HTTP status codes
- ◆ Also enables to obtain a PrintWriter object for writing output back to the client
- ◆ Methods:
 - `getWriter`
 - `setContentType`
 - `sendRedirect`

1. What is a Web Server ?
 - A. A computer having server software installed within it which serves up web pages
2. Name few Web Servers
 - A. Apache HTTP Server (Apache)
Microsoft Internet Information Server (IIS)
Sun Java System Web Server
3. A servlet's life cycle methods are -----
 - A. `init()`
`service()`
`destroy()`

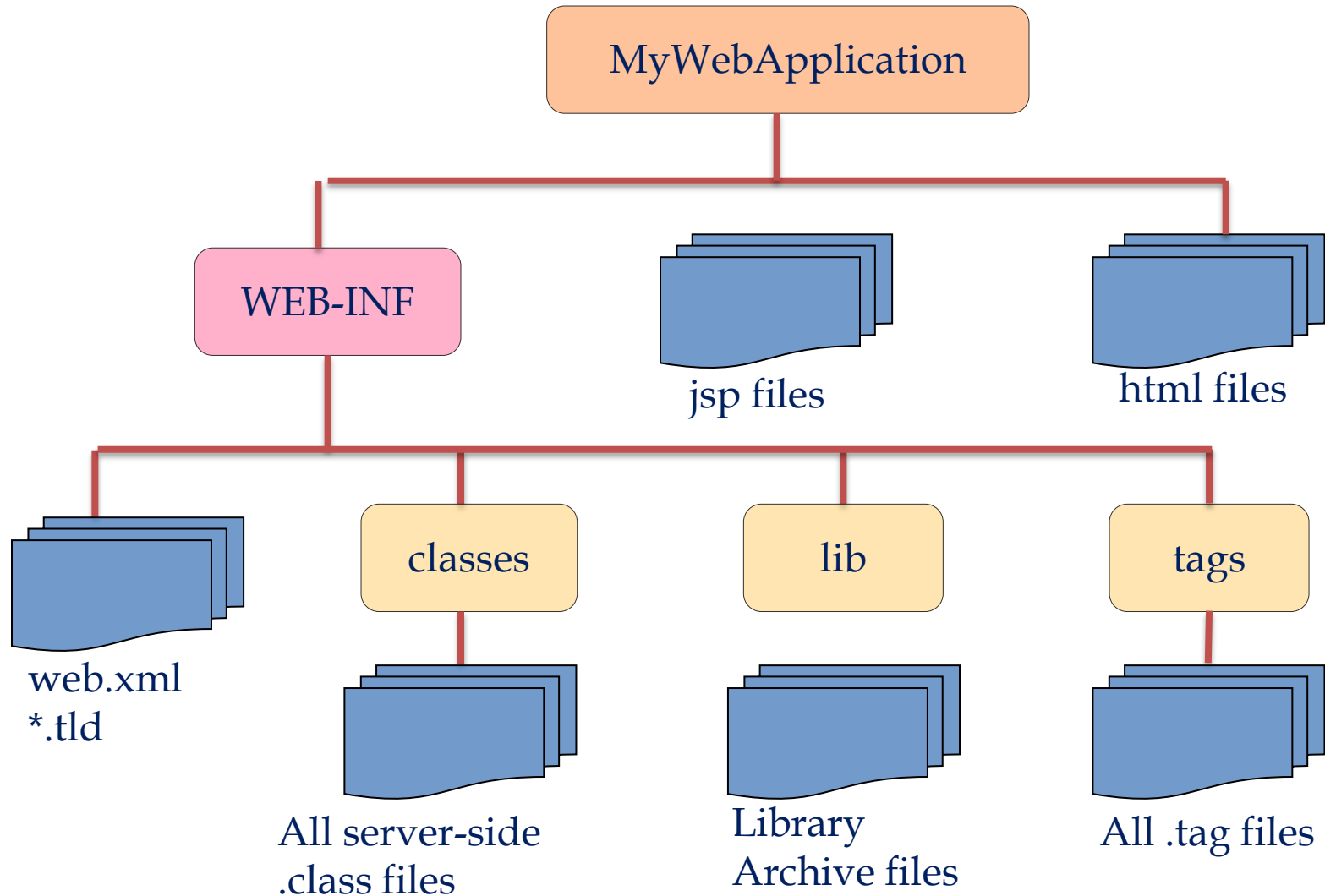
Deploying a Simple Servlet

- ◆ Set environment variables: JAVA_HOME, CATALINA_HOME, CLASSPATH, and PATH
- ◆ Create a directory, say “MyWebApplication” in C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps
- ◆ Compile your servlet say “WelcomeServlet.java” in C:\Program Files\Apache Software Foundation\Tomcat5.0\webapps\MyWebApp\WEB-INF\classes directory

Deploying a Simple Servlet

- ◆ Create web.xml file and place it in C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\MyWebApplication\WEB-INF folder
- ◆ Start Tomcat server by double-clicking C:\Program Files\Apache Software Foundation\Tomcat 5.0\bin\startup.batch file OR by selecting Start -> Programs -> Apache Tomcat 5.0 -> Monitor Tomcat
- ◆ Open browser with URL: <http://localhost:8090/> to check Tomcat is successfully installed
- ◆ Give URL as <http://localhost:8090/MyWebApplication/WelcomeServlet> to test your servlet

Directory Structure



Demo for a Simple Servlet

- ◆ A simple HTTP Servlet that displays a Welcome message on the web page



- ◆ Files required:
 - WelcomeServlet.java
 - web.xml

Demo for a Simple Servlet

- ◆ An HTTP Servlet that displays a Welcome message

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

public class WelcomeServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html"); // set header field first
        PrintWriter pw = res.getWriter(); // then get writer & write response data
        pw.println("<HTML>");
        pw.println("<HEAD><TITLE>Welcome</TITLE></HEAD>");
        pw.println("<BODY>");
        pw.println("<H3>" + "Welcome to Java Servlet Technology!!" + "</H3>");
        pw.println("</BODY>");
        pw.println("</HTML>");
        pw.close(); // closes the writer
    }
}
```

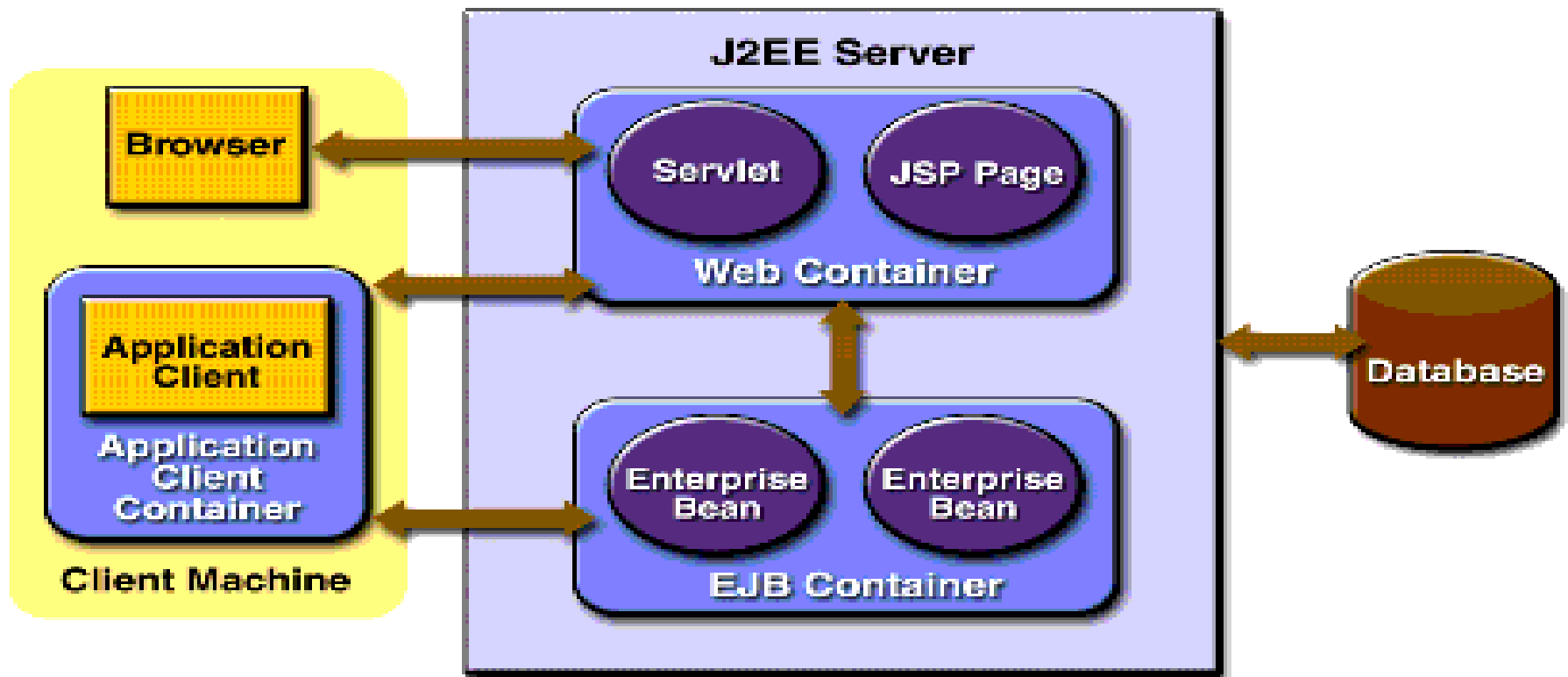

Web Deployment Descriptor

- ◆ Web.xml is a deployment descriptor file
- ◆ Example: web.xml

```
<web-app>
  <display-name>A Small Web Application</display-name>
  <servlet>
    <servlet-name>MyFirstServlet</servlet-name>
    <servlet-class>WelcomeServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyFirstServlet</servlet-name>
    <url-pattern>/welcome.msg</url-pattern>
  </servlet-mapping>
</web-app>
```

Web Container

- ◆ Web components and their container run on J2EE server



Role of a Web Container

- ◆ Communication Support
- ◆ Lifecycle Management
- ◆ Multithreading Support
- ◆ Declarative Security
- ◆ JSP Support

Servlet Requests

- ◆ Container creates 2 objects on receiving a request for a servlet:
 - `HttpServletRequest` and `HttpServletResponse`
- ◆ Finds right servlet based on URL request
- ◆ Creates a thread for that request
- ◆ Passes request and response objects to the servlet thread
- ◆ Calls servlet's `service()` method

Check Your Understanding

◆ Suppose you are a web developer working for an Online Movie Service. You want to use a servlet called `MovieServlet` so that clients can access the latest film shows for the day in a particular city from your movie database. Determine the correct sequence of following steps carried out by `MovieServlet` when processing a request from a client

SNo	Description
1	Check information included in the Http request
2	Access any necessary business components or data storage
3	Set the appropriate Http response parameters
4	Read data submitted by the client
5	Send the response to the client
6	Format the results in a response

1. ----- URL is used to test servlet
 - A. <http://localhost:8090/MyWebApplication/WelcomeServlet>
2. Container creates ----- and ----- objects on receiving a request for a servlet:
 - A. HttpServletRequest and HttpServletResponse

Handling Form Data

Simpleform.html

```
<html>
<head><title>Simple Form</title></head>
<body>
  <form method="post" action="SimpleFormServlet">
    <h3>Enter user details</h3>
    <br>Name: <input type="text" name="userName" />
    <br>Address: <input type="text" name="userAddress" />
    <input type="submit" value="Submit" />
  </form>
</body>
```

Handling Form Data

```
public class SimpleFormServlet extends HttpServlet {  
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws IOException, ServletException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        String name = request.getParameter("userName");  
        String address = request.getParameter("userAddress");  
        out.println("<html>");  
        out.println("<h1>" + "User Details" + "</h1>");  
        out.println("<p>The Name you entered was: " + name + "</p>");  
        out.println("<p> The Address you entered was: " + address + "</p>");  
    }  
}
```


Handling Simple Form Data

Address  http://localhost:10000/ServletsModularization/Simpleform.html

Enter user details

Name:

Address:

Address  http://localhost:10000/ServletsModularization/SimpleFormServlet

User Details

The Name you entered was: Anny

The Address you entered was: 24th Main Bangalore

Reason it out

- ◆ What difference does it make if SimpleFormServlet used doGet method and accordingly method="GET" in the html form?



Using HTML Components

- ◆ A Job seeker Company needs basic information about the user's name, address, state, highest qualification and skills

comp.html

```
<html>
<head><title>Information Details</title></head>
<body>
  <form action="differentCompServlet" method=POST>
    <BR>Name: <input type=text name="name" />
    <BR>Address: <textarea name="address" rows=5 cols=20></textarea>
    <BR>State: <select name="state">
      <option value="Andhra Pradesh"> Andhra Pradesh </option>
      <option value="Karnataka"> Karnataka </option>
      <option value="Uttar Pradesh"> Uttar Pradesh</option>
    </select>
```

Using HTML Components


comp.html

```
<BR><BR>Highest Qualification:<BR>
    Under Graduate<input type=radio name=qualification
value="UG">
    Post Graduate<input type=radio name=qualification
value="PG">
<BR><BR>Skills:<BR>
    Java<input type=checkbox name=skills value=Java>
    Servlets<input type=checkbox name=skills value=Servlets>
    JSPs<input type=checkbox name=skills value=JSPs>
    EJB 3.0<input type=checkbox name=skills value=EJB>
<BR><BR><input type=submit value=submit><input type=reset>
</form>
</body>
</html>
```

Using HTML Components

- ◆ `public Enumeration getParameterNames()`
- ◆ `public String[] getParameterValues(String name)`

Using HTML Components

Address  http://localhost:10000/ServletsModularization/comp.html

Name:

Address:

12th Cross
Sector 6
Bangalore

12th Cross
Sector 6
Bangalore

State: ▼

Highest Qualification:
Under Graduate ☐ Post Graduate ☒

Skills:
Java ☒ Servlets ☒ JSPs ☐ EJB 3.0 ☐

Address  http://localhost:10000/ServletsModularization/differentCompServlet

address : 12th Cross Sector 6 Bangalore

skills : Java Servlets

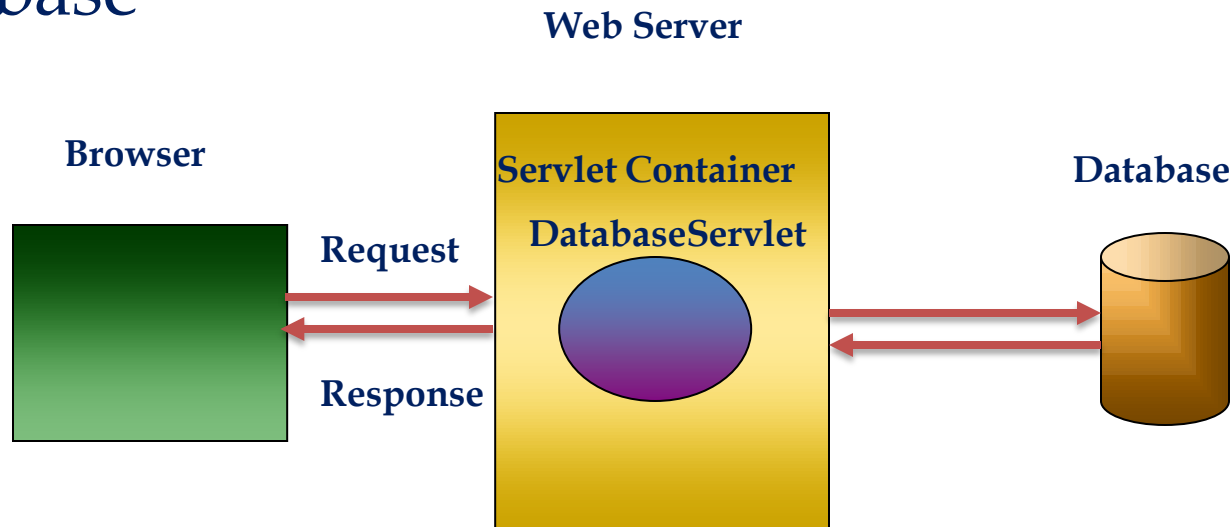
name : Harry

qualification : PG

state : Karnataka

Using JDBC in a Servlet

- ◆ A servlet can retrieve information from a database
- ◆ Can perform update/delete/insert queries to/from a database



Check Your Understanding

1. When you use a servlet to interact with database, why is it preferred to include database connection statements in the init method?
2. In the demo example for information retrieval, what would happen if you use doPost method instead of doGet method?
3. Let us say your servlet class is “DatabaseServlet.java”. How do you access the servlet through the URL:
<http://localhost:10000/ServletsModularization/books.show>

Summary

- ◆ In this module, you were able to:
 - Describe the role of HTTP Servlet in Web Programming
 - Describe and use the Servlet Life Cycle methods appropriately
 - Process parameters from HTML forms
 - Establish Database Connectivity through servlets

Quiz

1. The doGet() or doPost() method of a Servlet are invoked by -----

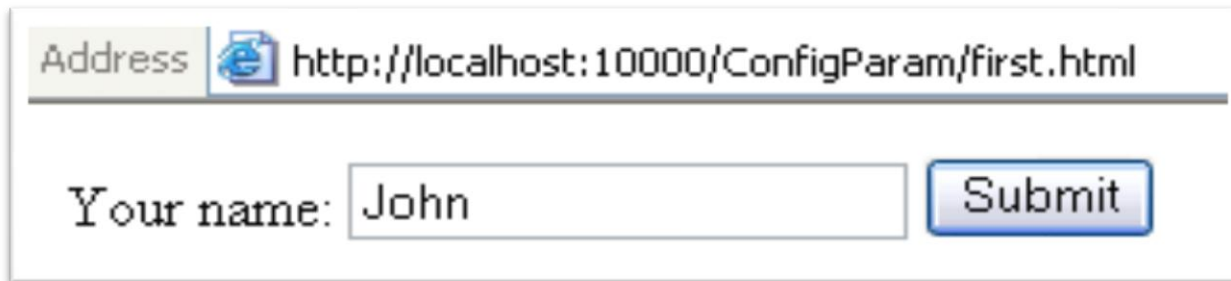
 - a. init() method
 - b. service() method
 - c. destroy() method
2. ----- is the deployment descriptor file for Servlets
 - a. servlet-config.xml
 - b. web.xml
 - c. struts-config.xml


Servlet Config

- ◆ Provided to a servlet after initialization by the web container
- ◆ Simple read only interface to get configuration details
 - `String getInitParameter(String name)`
 - `Enumeration getInitParameterNames()`
 - `String getServletName ()`
- ◆ Can also access `ServletContext`

Servlet Config

- ◆ Consider an html form “first.html” which accepts the user name
- ◆ A servlet “Second.java” takes in this parameter and displays it on the web page



Address  http://localhost:10000/ConfigParam/first.html

Your name:



Where did this value come from?

Servlet Config

web.xml

```
<web-app>
  <servlet>
    <servlet-name>Second</servlet-name>
    <servlet-class>Second</servlet-class>
    <init-param>
      <param-name>homeName</param-name>
      <param-value>Welcome to www.simple.com</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>Second</servlet-name>
    <url-pattern>/Second</url-pattern>
  </servlet-mapping>
</web-app>
```

The init parameter values are configured in the web.xml deployment descriptor file

Servlet Config

◆ Second.java servlet code

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Second extends HttpServlet {

    String homeName;
    ServletConfig config;

    public void init() { //get the initialization parameters
        //Returns this servlet's ServletConfig object
        config = getServletConfig();

        /*Returns a String containing the value of the named
        initialization parameter,
        or null if the parameter does not exist. */
        homeName = config.getInitParameter("homeName");
    }
}
```

Servlet Config

```
public void doGet(HttpServletRequest req, HttpServletResponse
    res)
    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    String urname = req.getParameter("name");

    out.println("<h2>" + homeName + "</h2>");
    out.println("<hr>");
    out.println("Hello! " + urname);
    }
}
```

Servlet Config

```
<web-app>
  <servlet>
    <servlet-name>DBConfigParamServlet</servlet-name>
    <servlet-class>DBConfigParamServlet</servlet-class>
    <init-param>
      <param-name>driverName</param-name>
      <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-
value>
    </init-param>
    <init-param>
      <param-name>urlName</param-name>
      <param-value>Jdbc:Odbc:vdsn2</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>DBConfigParamServlet</servlet-name>
    <url-pattern>/booksconfig.show</url-pattern>
  </servlet-mapping>
</web-app>
```


Discussion

- ◆ What is the advantage of setting up a database connection by reading init parameter values?



1. What information is contained in an HTTP request?

A. The HTTP request contains the request URL, HTTP method, and form parameter data (if any). The request URL is the resource that the client is trying to access

Servlet Context

- ◆ Allows a servlet to communicate with the servlet container
- ◆ Access container-managed resources, dispatch requests, write to logs
- ◆ Defines a set of methods that a servlet uses to communicate with its servlet container
 - For example, to get the MIME type of a file, dispatch requests, or write to a log file
- ◆ There is one context per "web application"
- ◆ ServletContext object is contained within ServletConfig object

Servlet Context

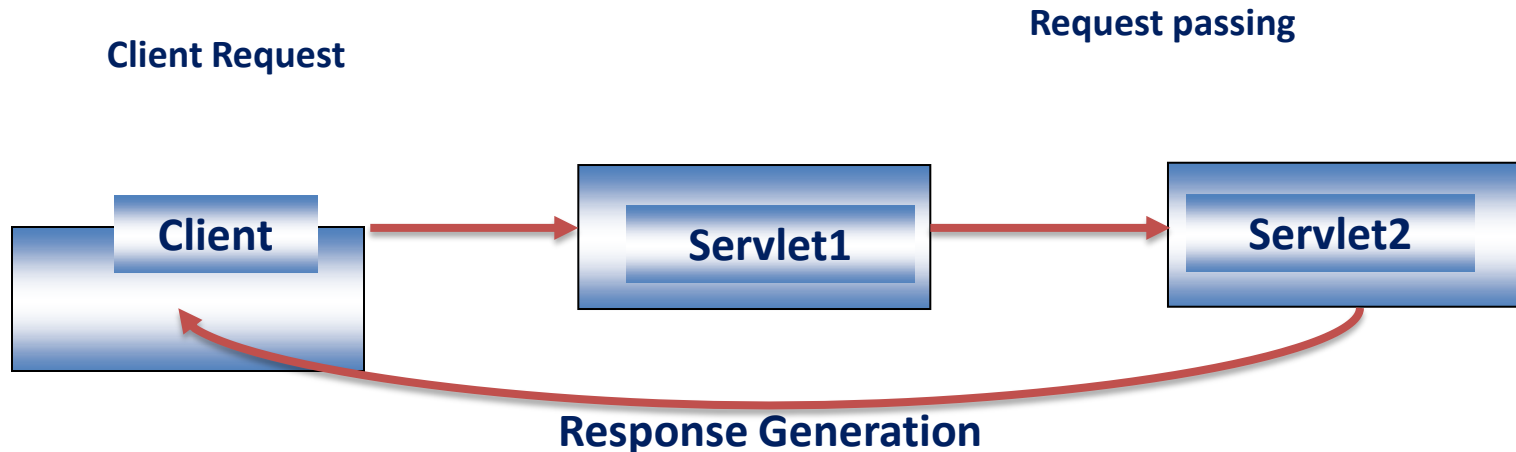
- ◆ Resources such as index.html can be accessed through web server or by servlet
 - Servlet uses `request.getContextPath()` to identify its context path, for example: `/app`
 - Servlet uses `getResource()` and `getResourceAsStream(request.getContextPath() + "/index.html")`
- ◆ To retrieve context-wide initialization parameters, servlet uses `getInitParameter()` and `getInitParameterNames()`
- ◆ Information shared with other servlets in the same servlet context can be accessed using `getAttribute()`, `setAttribute()`, `removeAttribute()`, `getAttributeNames()`

Servlet Chaining

- ◆ Used in order to FORWARD or INCLUDE a request from one servlet to another Servlet/JSP
- ◆ The RequestDispatcher interface provides two methods
 - RequestDispatcher.forward(request,response)
 - RequestDispatcher.include(request,response)
- ◆ Both these methods take ServletRequest and ServletResponse object as an argument

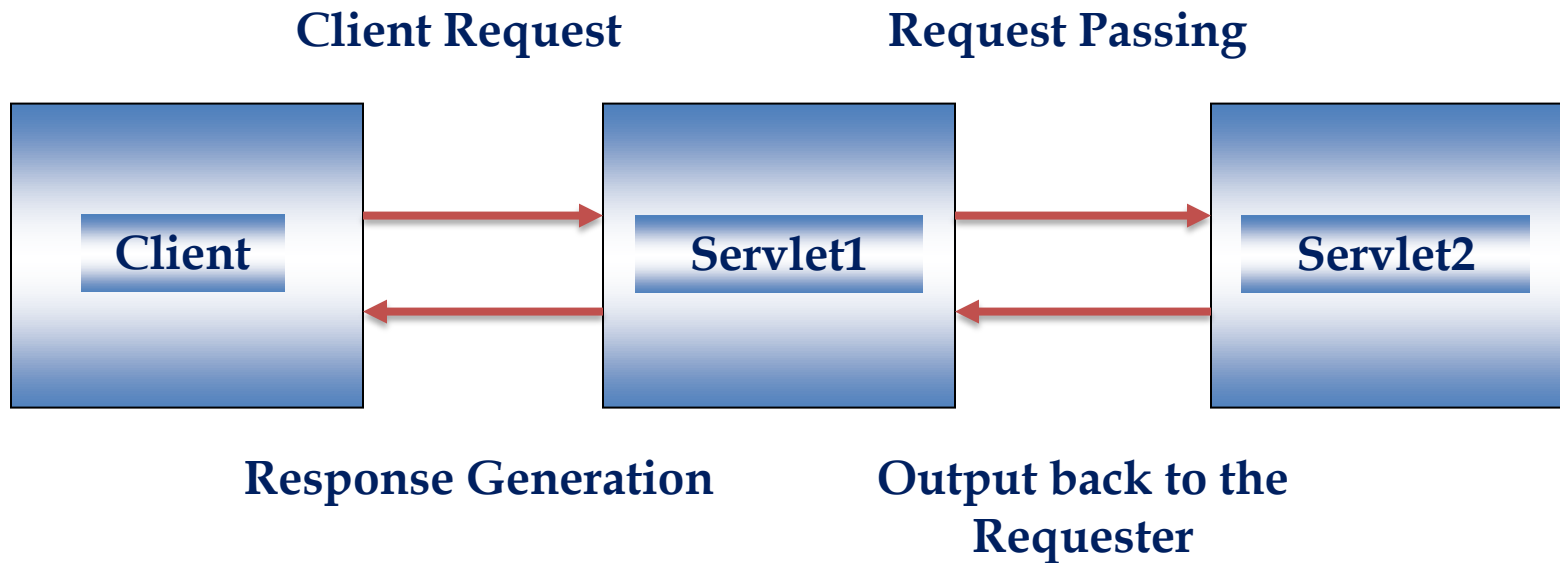
Servlet Chaining

```
Using forward(request,response);  
ServletContext ctx=getServletContext();  
RequestDispatcher  
dis=ctx.getRequestDispatcher("/servlet/ AnotherServlet");  
dis.forward(request,response);
```



Servlet Chaining

```
ServletContext ctx=getServletContext();  
RequestDispatcher  
    dis=ctx.getRequestDispatcher("/servlet/AnotherServlet");  
dis.include(request,response);
```



Session Management

- ◆ Http is a stateless protocol
- ◆ Applications require a series of requests from the same client to be associated with one another
- ◆ Session management is a mechanism which maintains state across a series of requests over some period of time
 - Example: Online shopping cart
- ◆ Session tracking is keeping track of what has gone before in a particular conversation

Session Tracking

Three different session tracking mechanisms using
“session id”

- ◆ Cookies –Can use a single cookie containing a session id
- ◆ URL rewriting - Can append a unique ID after the URL to identify user
- ◆ Hidden <form> fields – Can use these to store a unique ID

Cookies

- ◆ A cookie is a small bit of textual information sent to the client by a web server
- ◆ Web server can later read it back from the browser
- ◆ The process of using cookies in servlets
 - Servlet sends a cookie with its response to the client
 - The client saves the cookie
 - The client returns a cookie back with subsequent requests

Cookies

◆ Uses of cookies

- Identifying a user during a session
- Avoiding username and password
- Customizing a website

◆ Limitations for using cookies for protection of the client

- A Cookie size is limited to 4KB
- Supports 20 cookies per website
- Supports 300 cookies in total

Session Tracking API

- ◆ Session tracking API are in `javax.servlet.http.HttpSession`
 - It is built on top of cookies and URL rewriting
- ◆ Servlet container creates `HttpSession` object
 - Attached to `HttpServletRequest` object in `doXXXX` methods

Session Tracking API

◆ Create a session

- `HttpSession session = request.getSession();`
 - ◆ Returns session associated with this request
 - ◆ If there was no associated session, new one is created
- `getSession()` method is overloaded

Session Tracking API

◆ We can also get an HttpSession object by using the following method :

- HttpSession session = request.getSession(true);
 - getSession(true) works exactly like getSession()
- HttpSession session = request.getSession(false);
 - This method returns session associated with this request
 - If there was no associated session, new one is NOT created

Http Session

- ◆ Store and retrieve user-defined data in the session
 - To store values:
 - `session.setAttribute(name, value);`
 - To retrieve values:
 - `Object obj = session.getAttribute(name);`
 - `boolean session.isNew()`
 - Determines if session is new to client (not page)
 - `public void invalidate()`
 - Expires the session and unbinds all objects with it

Quiz

1. Pick up the valid methods from the following list, using which we can get an HttpSession object.
 1. `request.getSession()`
 2. `request.getSession("true")`
 3. `request.getSession("false")`
 4. `request.getSession(true)`
 5. `request.getSession(false)`
 6. `request.getSession(1);`
2. The servlet sends cookies to the browser by using the method
A. `HttpServletResponse.addCookie(Object of type Cookie)`
3. The servlet retrieves cookies by using the method
A. `HttpServletRequest.getCookies()`

Check Your Understanding

1. What is the use of the Web container in the Web application?
 - A. The Web container provides communications support, lifecycle management, multithreading support, security, and JSP support to the Web application. This enables you to concentrate on the business logic of the Web application.

2. What are the different session tracking mechanisms ?
 - A. Cookies – You can use a single cookie containing a session id
 - URL rewriting - You can append a unique ID after the URL to identify user
 - Hidden <form> fields – You can use these to store a unique ID

Summary

- ◆ In this module, you were able to:
 - Use ServletConfig and ServletContext object in web applications
 - Create web applications that implement Servlet Chaining
 - Develop web applications that use Cookies
 - Implement Session tracking in web applications