

Servlets

Onkar Deshpande



Onkar Deshpande





Strategic Training Expert

Table of Contents

Module	Topic
Module 1	Getting Started
Module 2	The Servlet's Life Cycle
Module 3	Form Data ,Context and Configuration Parameters
Module 4	Analyzing the Request Headers
Module 5	Setting the Response Header
Module 6	Redirection in Servlets
Module 7	Request Dispatching
Module 8	The Cookies
Module 9	Session Tracking
Module 10	Filters & Listeners

Module 1 : Getting Started

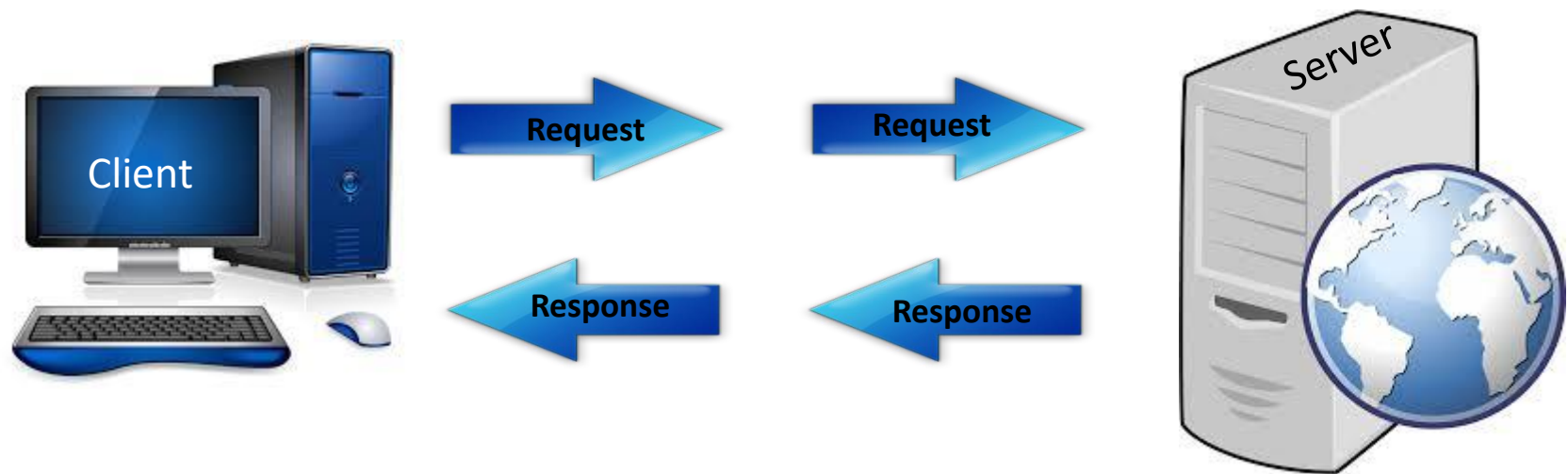
Overview :

-  Introduction to Servlet Technology
-  Setting up Servlet programming environment
-  Writing and deploying first Servlet
-  Role of web server

Flavours of Java

- J2SE(Standard Edition)
- J2EE(Enterprise Edition)
- J2ME(Micro Edition)

Client Server Technology



Need of Dynamic Web Page Building

- Data on web page depends on the client request
 - Ex : Search engines
- Data on web pages changes frequently
 - Ex : Weather reports
- Data on web pages uses data from corporate databases
 - Ex : Stock indexes

A Static HTML Page

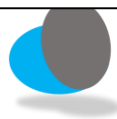
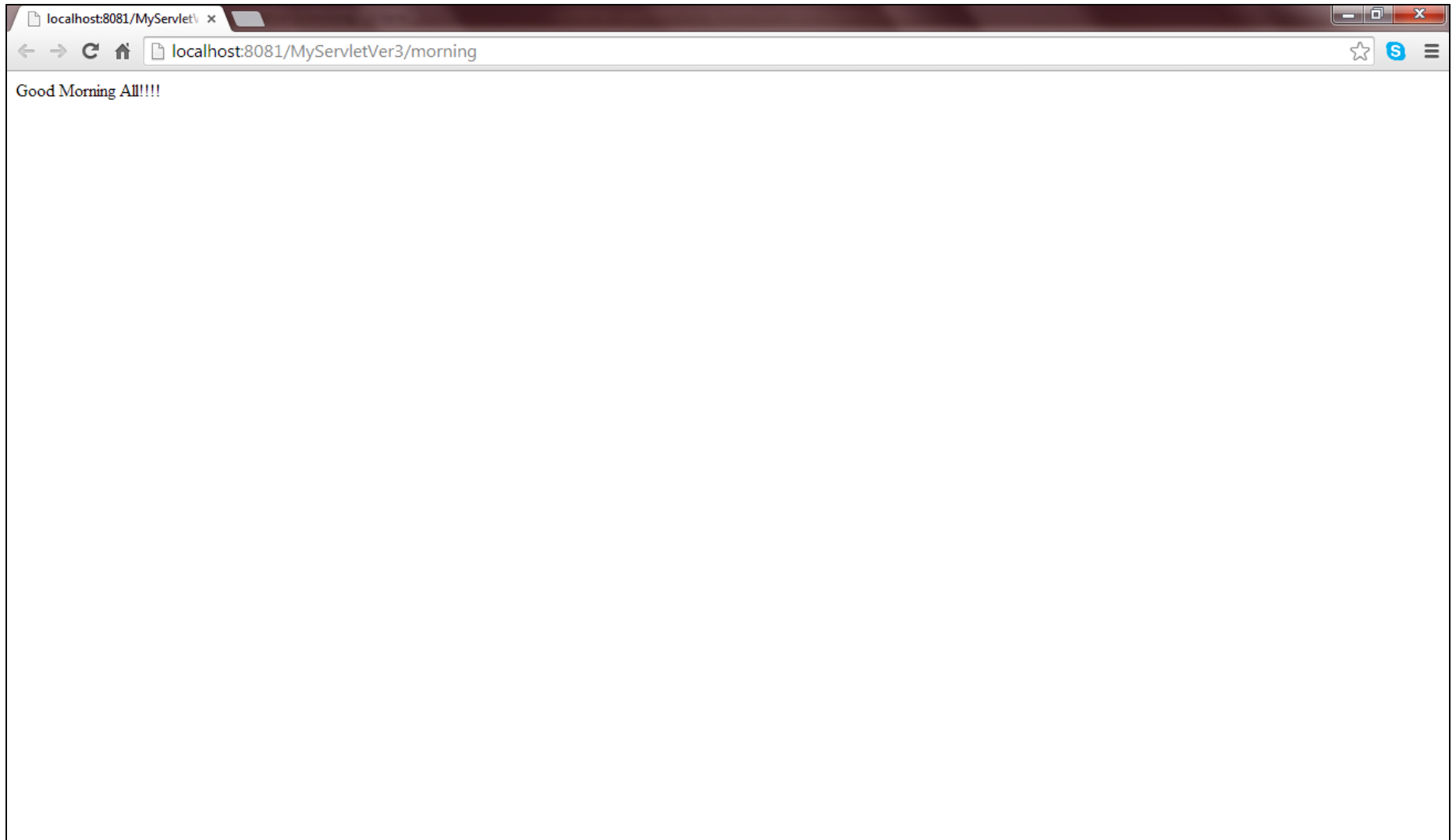
```
<html>
  <head>
    <title>
      Displaying Date
    </title>
  </head>
  <body>
    <script language="java script">
      <!-- document.write(Date( )); -->
    </script>
  </body>
</html>
```

Writing the First Servlet “GoodMorning.java”

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

public class GoodMorning extends HttpServlet {
    protected void doGet(HttpServletRequest request , HttpServletResponse response) throws
    IOException {
        response.setContentType( "text/html" );
        PrintWriter out = response.getWriter();
        out.println( "GOOD MORNING EVERYBODY !!!! " );
    }
}
```


Accessing Servlet



Servlet Showing Dynamic Nature

```
public class DateFormat extends HttpServlet {  
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException{  
        PrintWriter out = response.getWriter( );  
        out.println("<B>STATIC DATA</B><BR>");  
        out.println("GOOD MORNING EVERYBODY!!!!!!");  
  
        String [ ] monthNames = {"Jan", "Feb", "Mar", "Apr", "May",  
        "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };  
        Calendar timeNow = Calendar.getInstance( );  
        int date = timeNow.get(Calendar.DAY_OF_MONTH);  
        int month = timeNow.get(Calendar.MONTH);  
  
        out.println("<BR><BR><B>DYNAMICALLY GENERATED DATA</B><BR>");  
        out.println(" Today is : "+date+" th"+monthNames[month]+"  
        "+year+"<BR>");  
    }  
}
```

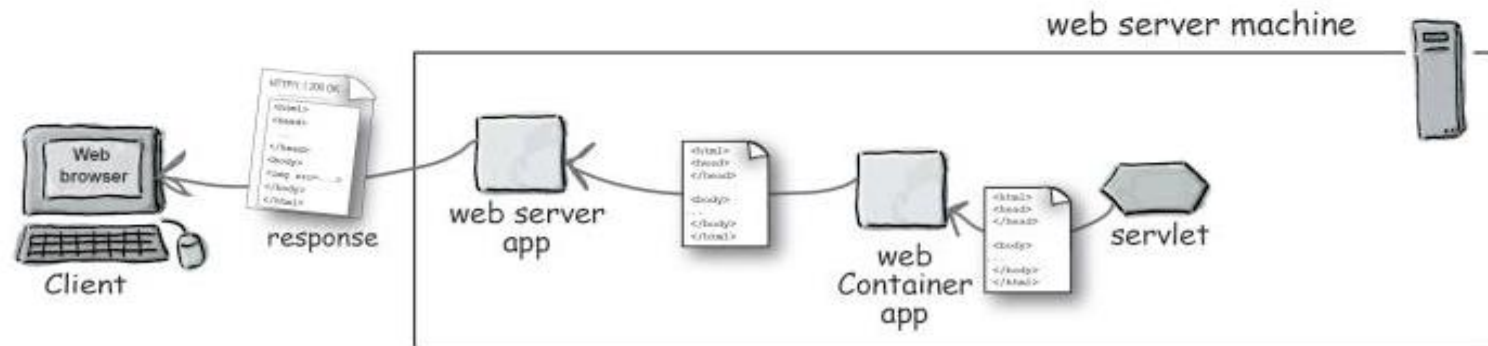
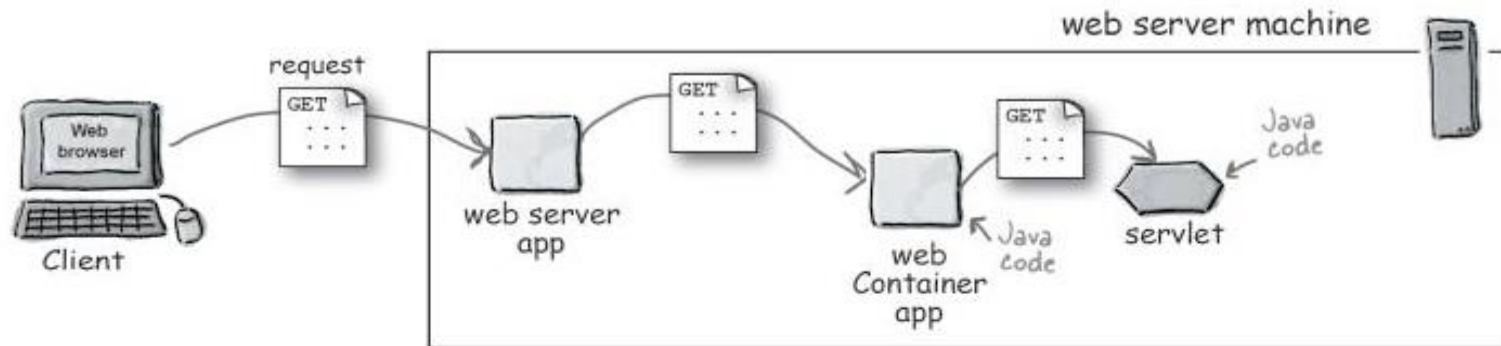
Other Competing Technologies

- HTML (HyperText Markup Language)
- CGI (Common Gateway Interface)
- ASP (Active Server Pages)
- JavaScript
- PHP (HyperText Pre-Processor)
- ColdFusion

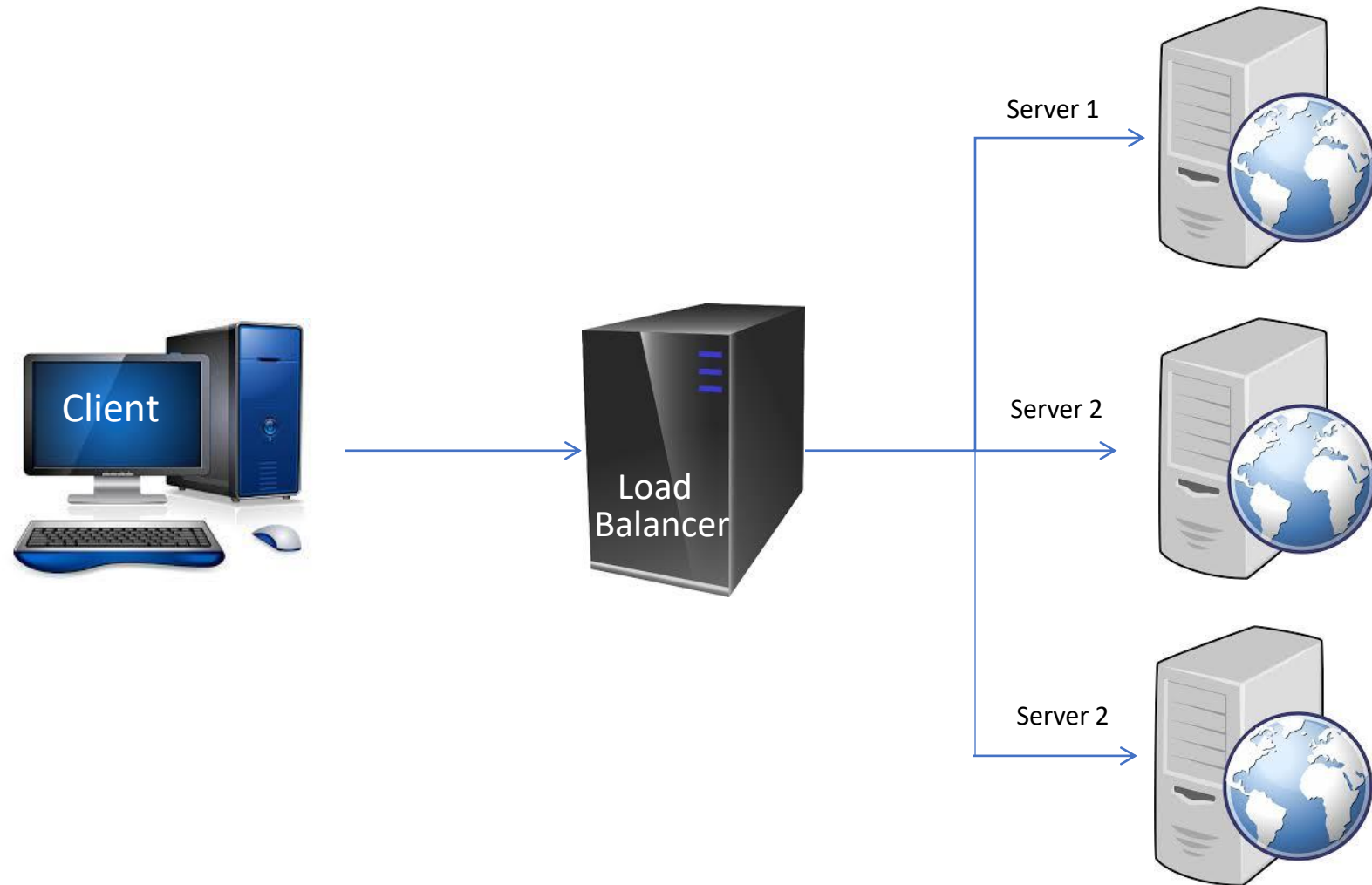
Servlets Features

- Portability
- Powerful
- Efficiency
- Safety
- Integration
- Extensibility
- Inexpensive

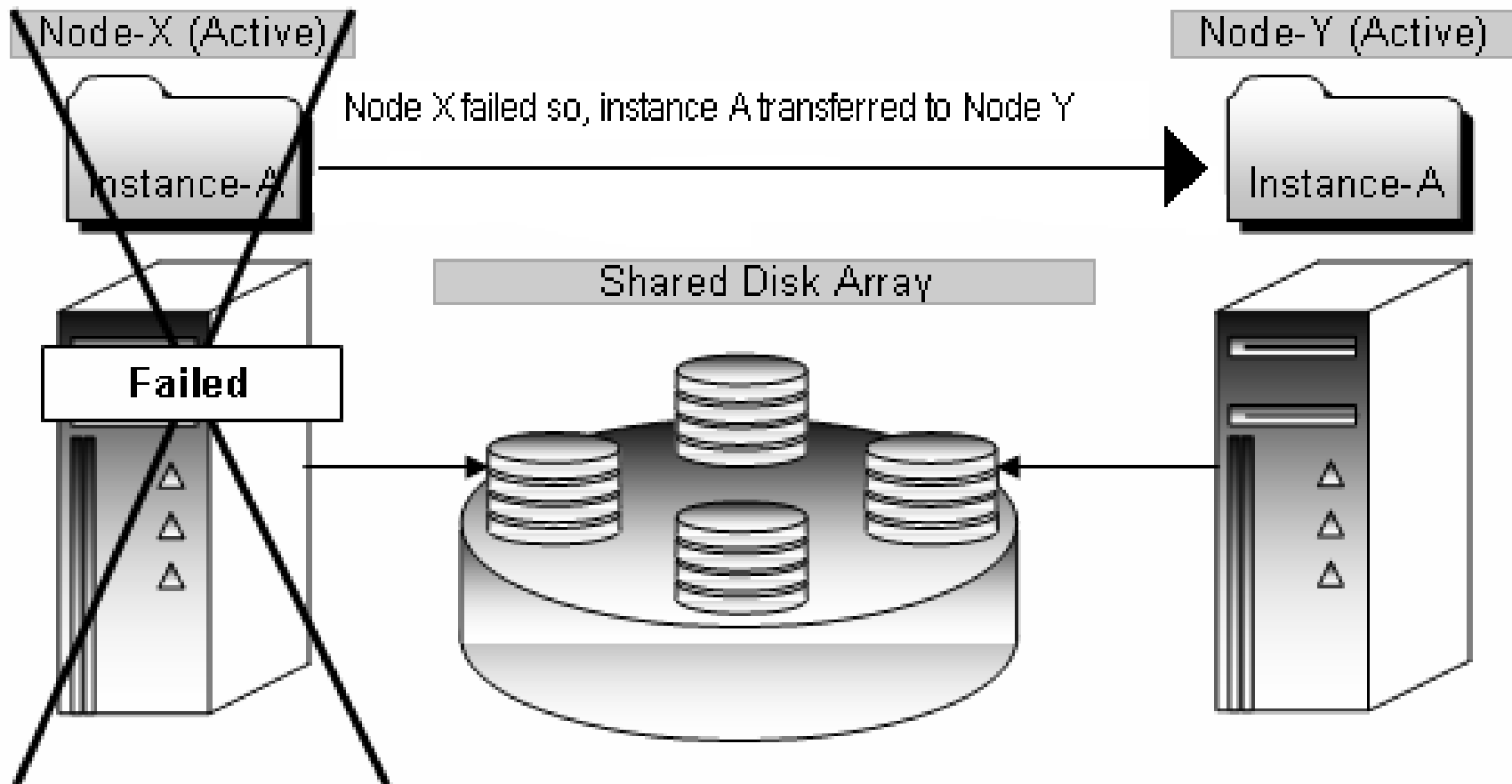
Role of a Server



Features of Server



Features of Server (contd...)

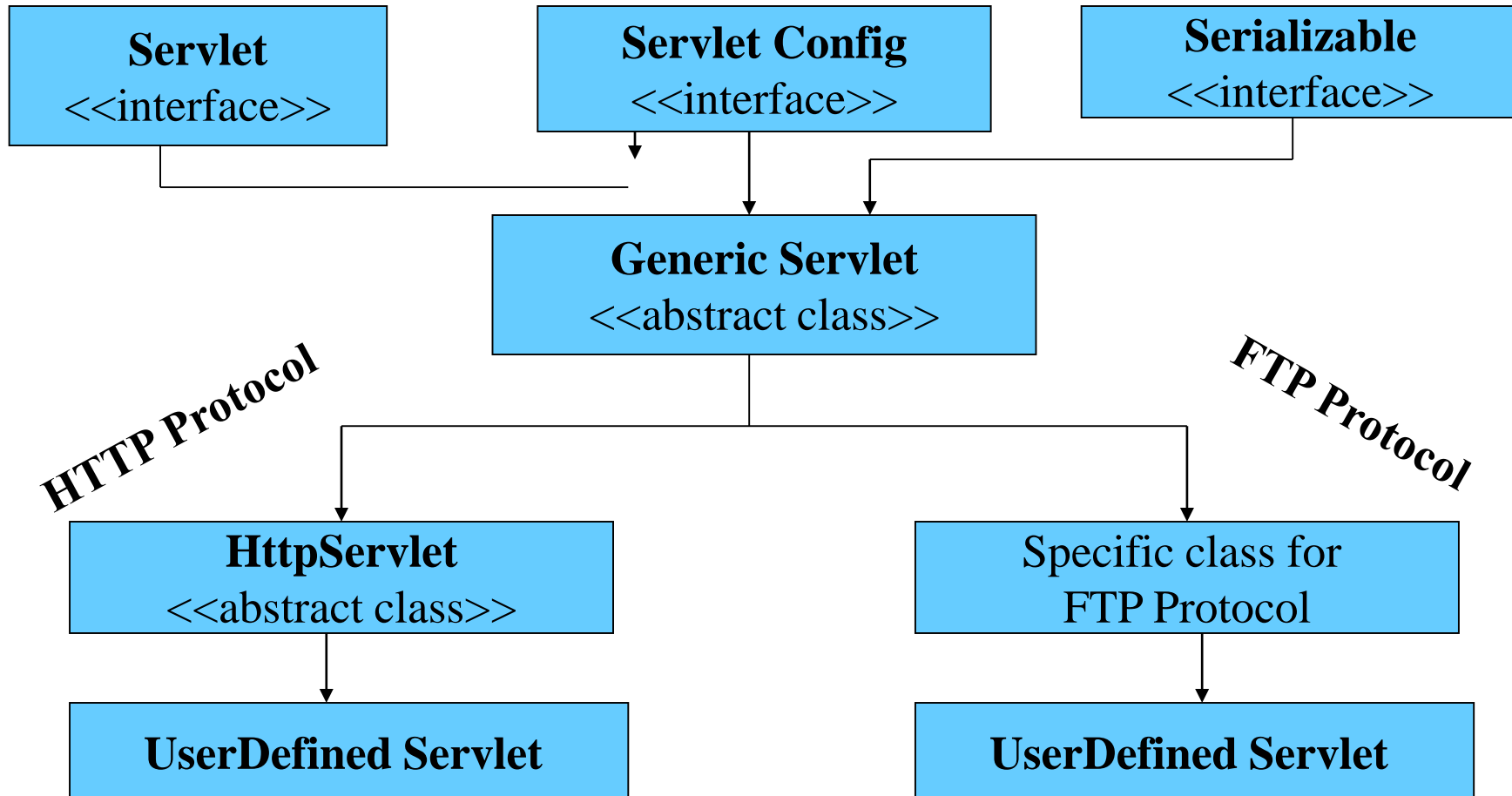


Web Servers

- Tomcat (Apache)
- Weblogic (Bea)
- Microsoft Internet Information Server (IIS)
- Websphere (IBM)
- WebStar (StarNine)
- Java Web Server (Sun's)

Servlet Architecture







Servlet Class Hierarchy



Servlet Architecture contd...

Directory Structure :

Working Directory

-  html
-  jsp
-  WEB-INF
 -  classes
 -  lib
 -  src

Servlet Architecture contd...

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

public class GoodMorning extends HttpServlet {
    protected void doGet(HttpServletRequest request , HttpServletResponse response)
    throws IOException {
        response.setContentType( "text/html" );
        PrintWriter out = response.getWriter();
        out.println( "GOOD MORNING EVERYBODY !!!!!" );
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException {
        doGet( request, response);
    }
}
```

Deployment Descriptor and Deployment





- Write and add web.xml in WEB-INF.

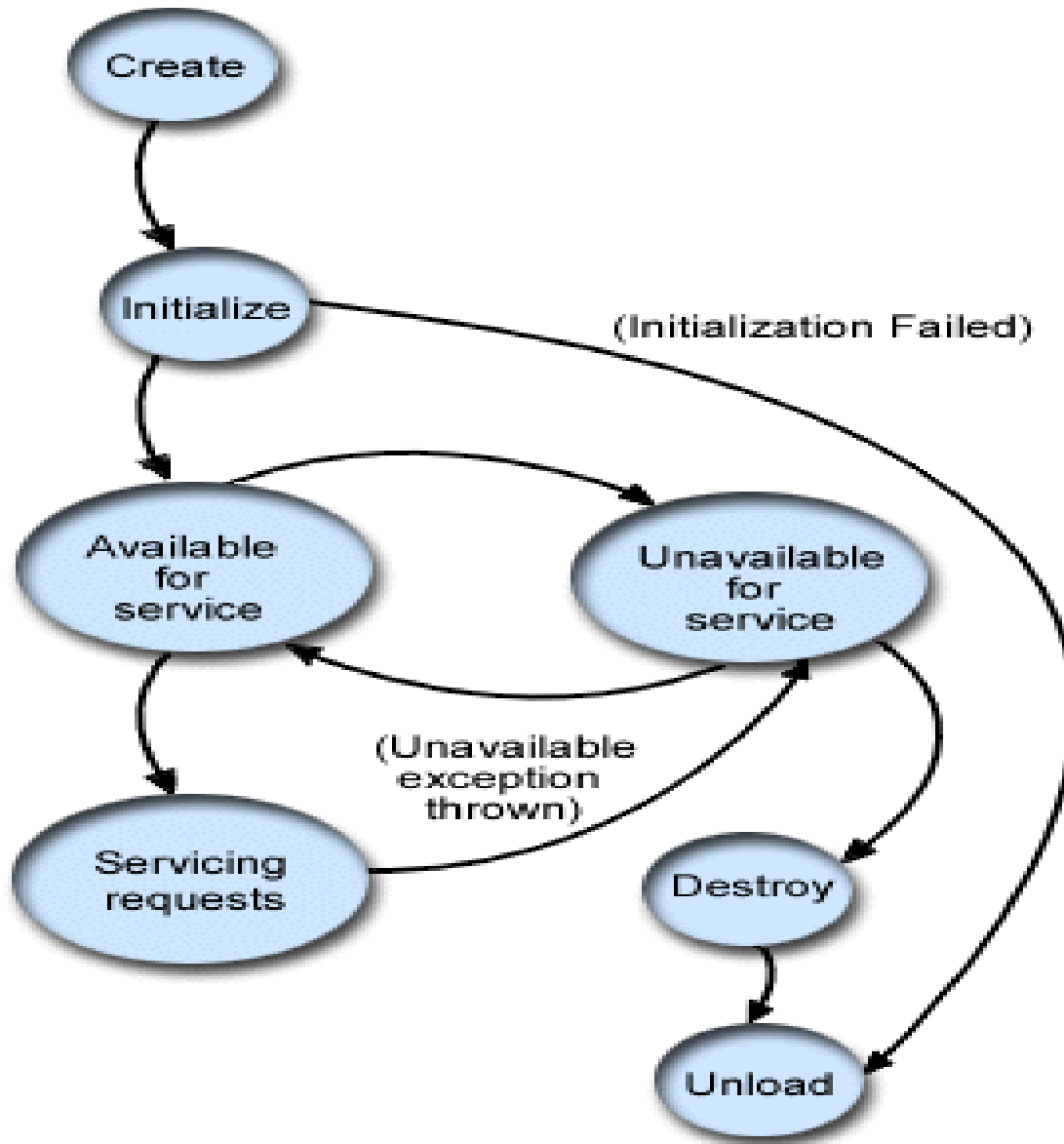
```
<web-app>
    <servlet>
        <servlet-name>MORNING</servlet-name>
        <servlet-class>pack010welcome.GoodMorning</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>MORNING</servlet-name>
        <url-pattern>/morning</url-pattern>
    </servlet-mapping>
</web-app>
```

- Optional : Convert an application into a JAR file.
- Mandatory : Deploy application on the server.

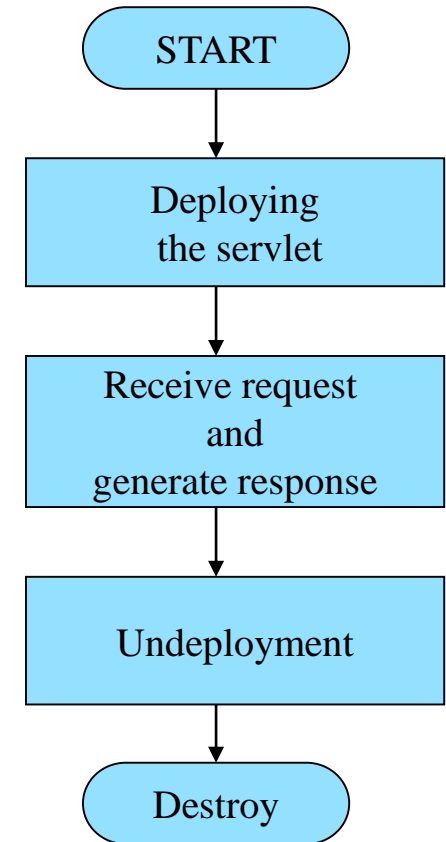
Module 2 : The Servlet Life Cycle

Overview :

-  Servlet Lifecycle
-  init(), service() and destroy() methods
-  The doGet() and doPost() methods
-  The servlet packaging



Servlet Life Cycle



The init() method

```
public void init( ) throws ServletException{
    f= new File("C:/Count.txt");
    FileInputStream fi=null;
    try {
        if (f.exists( )){
            fi = new FileInputStream(f);
            hitCount = fi.read( );
            fi.close( );
        }
        else
            hitCount = 0;
    }
    catch(IOException ie){
        throw new ServletException("Input file corrupted.", ie);
    }
}
```

The service() method

```
protected void doGet ( HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {
```

```
    System.out.println("Servlet receiving subsequent requests");
```

```
    PrintWriter out = response.getWriter();
```

```
    out.println("Pragati Software Private Limited");
```

```
    out.println("This website is hit "+(hitCount++)+" no. of times." );
```

```
}
```

```
protected void doPost (HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException{
```

```
    doGet(request, response);
```

```
}
```


The destroy() method






```
public void destroy(){
    System.out.println( " Object being undeployed from server.." );
    try{
        FileOutputStream fo = new FileOutputStream(f);
        fo.write(hitCount);
        fo.close();
    }
    catch(IOException io){
        io.printStackTrace();
    }
}
```

Servlet Packaging

- Various ways in which Packaging can be done.....
 - .jar file : Java Archive
 - .war file : Web Archive
 - .ear file : Enterprise Archive

Module 3: FormData, Config and Context Parameters

Overview :

-  The Form Data
-  Collecting form data for different HTML components
-  Roll of XML descriptor
-  The 'context' and 'config' parameters
-  The Context workspace

The Form Data

- The HTML elements
 - The FORM ACTION tag
 - The INPUT like component tags
 - The SUBMIT button tag
- The method types
 - The GET type of method
 - `http://localhost:7001/CollectParameters/collect?fname=abc&sname=xyz`
 - The POST type of method
 - The form data is sent on the separate line

Different HTML Components

▶ **Creating Form**

- ▶ `<FORM ACTION = "/CollectParameters22/param" METHOD = "GET">`

▶ **Creating Text Box**

- ▶ First Name : ` <INPUT TYPE="TEXT" NAME = "fname" ">`

▶ **Creating Password Box**

- ▶ PassWord : `<INPUT TYPE="PASSWORD" NAME = "pass">`

▶ **Creating Radio Buttons**

- ▶ Married `<INPUT TYPE="RADIO" NAME="marital" VALUE="Married">`

- ▶ Unmarried `<INPUT TYPE="RADIO" NAME="marital" VALUE="Unmarried">`

▶ **Creating Text Area**

- ▶ `<TEXTAREA name='address' COLS="50" ROWS="4"></TEXTAREA>`

Different HTML Components (Contd...)

▶ **Creating Combo Box**

```
<SELECT NAME="city" SIZE="1">  
  <OPTION VALUE="Mumbai">Mumbai</ OPTION >  
  < OPTION VALUE ="Pune">Pune</ OPTION >  
  < OPTION VALUE ="Nasik">Nasik</ OPTION >  
</Select>
```

▶ **Creating Check Box**

- ▶ Times <INPUT TYPE="CHECKBOX" NAME="news" VALUE="Times ">
- ▶ DNA <INPUT TYPE="CHECKBOX" NAME="news" VALUE="DNA">

▶ **Creating Submit Button**

- ▶ <INPUT TYPE="RESET" VALUE = "RESET">
- ▶ <INPUT TYPE="SUBMIT" NAME="choice" VALUE = "Tie">
- ▶ <INPUT TYPE="SUBMIT" NAME="choice" VALUE = "Necklace">

The HttpServletRequest

- ▶ The ServletRequest methods :
 - ▶ String Request.getParameter(String)
 - ▶ String [] request.getParameterValues(String)
 - ▶ Enumeration request.getParameterNames()
 - ▶ BufferedReader request.getReader()
 - ▶ ServletInputStream request.getInputStream()

- ▶ The HttpServletRequest methods :
 - ▶ String getHeader (String)
 - ▶ Enumeration getHeaderNames()
 - ▶ String getMethod()



The HttpServletResponse

- The Servlet Response methods :
 - PrintWriter getWriter()
 - void setContentType(String type)
 - void setBufferSize(int size)
 - ServletOutputStream getOutputStream()
- The HttpServletResponse methods :
 - void addHeader(String name, String value)
 - void setHeader(String name, String value)
 - void setStatus(int code)

Collecting Form Data

```
private void doPost(HttpServletRequest arg0, HttpServletResponse arg1) {  
    String[] firstName = arg0.getParameter("fname");           // Text box  
    String surName = arg0.getParameter("sname");               // Text box  
    String passWord = arg0.getParameter("pass");               // Password box  
    String maritalStatus = arg0.getParameter("marital");        // Radio Buttons  
    String yourAddress = arg0.getParameter("address");         // Text Area  
    String yourCity = arg0.getParameter("city");               // Combo box  
    String yourChoice = arg0.getParameter("choice");           // Submit Button  
  
    String [ ]news = arg0.getParameterValues("news");          // Check boxes  
  
    if (news != null){  
        out.println("<TR><TD>News Paper/s");  
        out.println("<TD>" + news[0]);  
        for(int i=1; i<news.length; i++){  
            out.println(" & " + news[i]);  
        }  
    }  
}
```

Collecting Form data contd...

```
Enumeration e = arg0.getParameterNames( );  
    while(e.hasMoreElements( )){  
        String field = (String)e.nextElement( );  
        String value = arg0.getParameter(field);  
        out.println(field+" : "+value+"<BR>");  
    }
```

The ServletContext

- One per application
- Facilitates communication between server and the servlet
- Provides access to resources and facilities common to all servlets and JSPs in the application
- `getServletContext()` returns a reference to the ServletContext

The ServletConfig

- One per Servlet
- Used to store information specific to a particular servlet
- Carries servlet specific data, which is not accessible to any other servlet
- `getServletConfig ()` returns the reference to the ServletConfig

Initializing Parameters

`<context-param>`

`<param-name>drv</param-name>`

`<param-value>oracle.jdbc.driver.OracleDriver</param-value>`

`</context-param>`

`<context-param>`

`<param-name>url</param-name>`

`<param-value>jdbc:oracle:thin:@localhost:1521:xe</param-value>`

`</context-param>`



Initializing Parameters

`<servlet>`

`<servlet-name>USERENTRY</servlet-name>`

`<servlet-class>pack040user.UserEntry</servlet-class>`

`<init-param>`

`<param-name>user</param-name>`

`<param-value>scott</param-value>`

`</init-param>`

`<init-param>`

`<param-name>pass</param-name>`

`<param-value>tiger</param-value>`

`</init-param>`

`</servlet>`



ServletContext methods

- ▶ Accessing initial parameters and other server side information.
 - ▶ `getInitParameter()`
 - ▶ `getInitParameterNames()`
 - ▶ `getMajorVersion()`
 - ▶ `getMinorVersion()`
 - ▶ `getServerInfo()`

- ▶ To access server side file resources.
 - ▶ `getMimeType()`
 - ▶ `getResourceAsStream()`
 - ▶ `getRequestDispatcher(String path)`

- ▶ Handling server side log
 - ▶ `log()`

ServletContext methods contd...

- Accessing context workspace
- `setAttribute()`
- `getAttribute()`
- `getAttributeNames()`
- `removeAttribute()`



ServletConfig methods

Obtaining Config Parameters

-  `getInitParameter()`
-  `getInitParameterNames()`

Obtaining Context reference

-  `getServletContext()`

Obtaining Servlet Name

-  `getServletName()`

ServletConfig vs ServletContext

- ▶ Accessibility :
 - ▶ ServletContext : One per Application , Context parameters are available across servlets under same application.
 - ▶ ServletConfig : One per servlet. The config parameters are private to the servlet and cannot be accessed by any other servlet.
 - ▶ Getting the parameter values :
 - ▶ `ServletContext sct = this.getServletContext();`
 - ▶ `String driverName = sct.getInitParameter("drv");`
 - ▶ `ServletConfig sc = this.getServletConfig();`
 - ▶ `String passwd = sc.getInitParameter("pass");`
 - ▶ Setting attributes :
 - ▶ ServletConfig has only Parameters.
 - ▶ Cannot set the config parameters via methods hence, only getter methods are available.
 - ▶ ServletContext has both Parameters and Attributes.
 - ▶ Context Parameters can be set via the setter methods provided
-

Module 4 : Analysing Request Header

- Overview :
 - The GET and POST requests
 - HTTP Request Parameter
 - The Request Headers
 - Analyzing request header

Typical Http Request

- ▶ For GET method :

GET /requestheadersdemo?name=abc&surname=xyz http/1.1

Host : <http://localhost:8081/Myervlets/collectparameters>

User-Agent: Mozilla/4.0

Accept : */*

Accept Encoding : gzip,deflate

- ▶ For POST method :

POST /requestheadersdemo http/1.1

Host : <http://localhost:8081/Myervlets/collectparameters>

User-Agent: Mozilla/4.0

Accept : */*

Accept Encoding : gzip,deflate

name=abc&surname=xyz

Reading Request Header

▶ **General :**

- ▶ `getHeader("HeaderName")`
- ▶ `getHeaderNames()`

▶ **Specialized :**

- ▶ `getCookies()`
- ▶ `getRemoteUser()`
- ▶ `getContentTypeLength()`
- ▶ `getDateHeader()`
- ▶ `getIntHeader()`

▶ **Related Info :**

- ▶ `getMethod()`
- ▶ `getRequestURI()`
- ▶ `getQueryString()`
- ▶ `getProtocol()`







Analysing Request Header

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException {
    PrintWriter out = response.getWriter( );
    out.println("<B>Request Method : </B>" + request.getMethod( ) + "<BR>");
    out.println("<B>Request Protocol : </B>" + request.getProtocol( ) + "<BR>");
    out.println("<B>Request URI : </B>" + request.getRequestURI( ) + "<BR>");

    Enumeration headerNames = request.getHeaderNames( );
    while(headerNames.hasMoreElements( )){
        String headerName =(String)headerNames.nextElement( );
        out.println("<TR><TD>" + headerName);
        out.println("    <TD>" + request.getHeader(headerName));
    }
}
```

Module 5 : Setting Response Header

Overview :

-  HTTP Response Header
-  Setting MIME types
-  Status Code, Refreshing, Sending Error Page, Logging in
-  Setting Encoding Type



Typical HttpResponse

HTTP Response Message

status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

HTTP/1.1 200 OK

Connection: close

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998 ...

Content-Length: 6821

Content-Type: text/html

data data data data data ...



Setting Response Header

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

public class ResponseHeaders extends HttpServlet{

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

        res.setContentType("text/html");
        PrintWriter out = res.getWriter( );
        res.addHeader("MyHeader" , "This is My custom header");
        out.println("<html>");
        out.println("<body>");
        out.println("My Custom Header Set");
        out.println("</body>");
        out.println("<html>");

    }
}
```

Various Response Headers

- Allow
- Cache-Control
- Connection
- Content-Encoding
- Content-Language
- Content-Length
- Content-Type
- Refresh

Setting MIME type

- ▶ Method to set MIME type :
 - ▶ `setContentType(Mime type)`
- ▶ List of all mime types :
 - ▶ `text/plain` : Plain text
 - ▶ `text/html` : HTML document
 - ▶ `text/xml` : XML document
 - ▶ `audio/basic` : Sound file in .snd format
 - ▶ `image/gif` : GIF image
 - ▶ `image/jpeg` : JPEG image
 - ▶ `application/msword` : Microsoft Word Document
 - ▶ `application/pdf` : Acrobat (.pdf) file .
 - ▶ `application/x-java-archive` : JAR file
 - ▶ `video/mpeg` : MPEG video clip

Setting Encoding Type

```
public class Gzipping extends HttpServlet {  
    protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException, IOException {  
        String encodings = arg0.getHeader("Accept-Encoding");  
        PrintWriter out; String title;  
        arg1.setContentType("text/plain");  
        if ((encodings != null) && (encodings.indexOf("gzip") != -1)) {  
            title = "Page Encoded with GZip";  
            OutputStream out1 = arg1.getOutputStream();  
            out = new PrintWriter(new GZIPOutputStream(out1), false);  
            arg1.setHeader("Content-Encoding", "gzip");  
        } else {  
            title = "Un-encoded Page.";  
            out = arg1.getWriter();  
        }  
        for (int i = 0; i < 10000; i++)  
            out.println("-foo-faa");  
        out.close();  
    }  
}
```

Module 6 : The Redirection

- Overview :
 - Redirection
 - Redirection by setting response header
 - Linking to another websites

Redirection in HTML

```
protected void doGet(HttpServletRequest request,
HttpServletRequest response) throws ServletException, IOException {
    String url1 = "http://localhost:8081/MyServlet/readword";
    String url2 = "http://www.Xoriant.com";
    String url3 = "/MyServlet/goodmorning";

    // HTML Hyper Link Way of Redirection
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html><head><title>301 Moved Permanently</title></head>");
    out.println("<body>");
    out.println("Website moved to <a href=\"\" + url1 + "\">here.</a>");
    out.println("</body></html>");
    out.close();
}
```

Redirection using Servlet API

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
    PrintWriter out = response.getWriter();
    out.println("<B><CENTER> *****WELCOME*****");

    // Redirection by specifying the absolute path of the site to whom the request is
    directed
    response.sendRedirect(url2);

    // Redirection by specifying the relative path of the site to whom the request is
    directed
    response.sendRedirect(url3);





    // Redirection by setting response header
    response.setStatus(HttpServletResponse.SC_MOVED_TEMPORARILY);
    response.setHeader("Location", url1);
}
```

Setting the Refresh Header

```
public class RefreshHeader extends HttpServlet {  
    String url = "www.Xoriant.com";  
  
    protected void doGet(HttpServletRequest request,  
        HttpServletResponse response) throws ServletException, IOException {  
  
        PrintWriter out = response.getWriter();  
        out.println("<H1>HELLO !!!!!<H1>");  
  
        // The Browser asks for an updated page after 30 seconds  
        response.setIntHeader("Refresh", 30);  
  
        // The browser goes to the specified page after specified time delay  
        response.setHeader("refresh", "5; URL=http://www.Xoriant.com");  
    }  
}
```

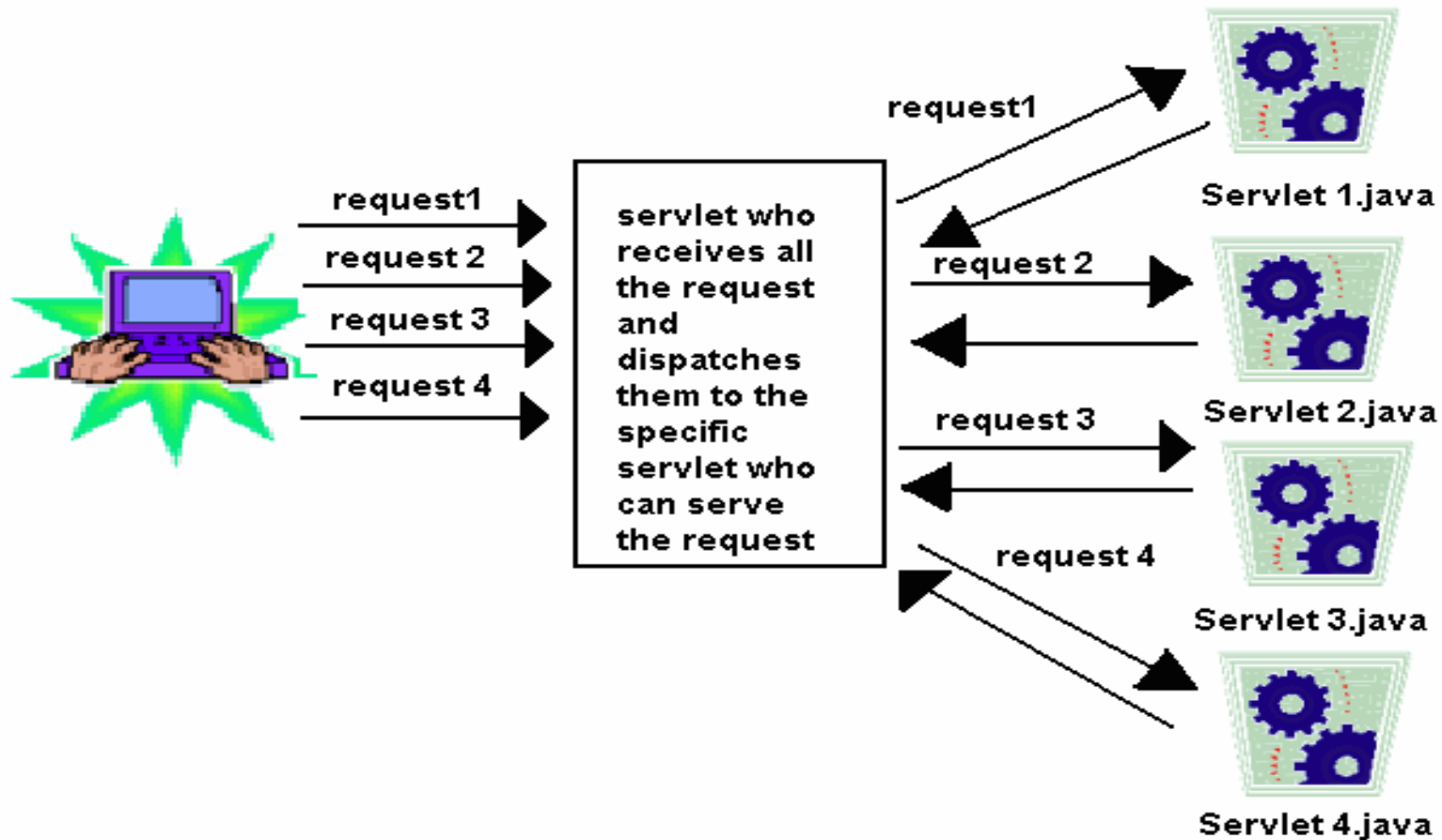

Module 7 : The Request Dispatching

Overview

-  The Request Dispatcher
-  The forward and include requests
-  Sending request between servlets
-  The request workspace

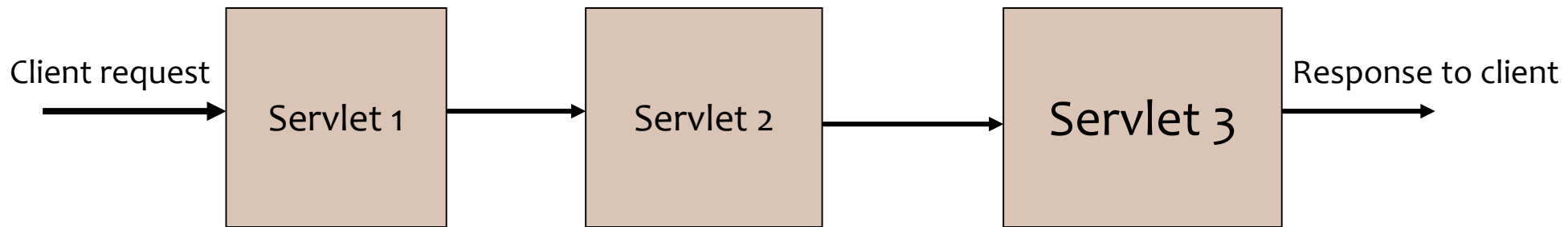


Request Dispatcher



Dispatching using forward() method

The request is received by one servlet and response is generated by another.



Receives the request and dispatches it to other servlet

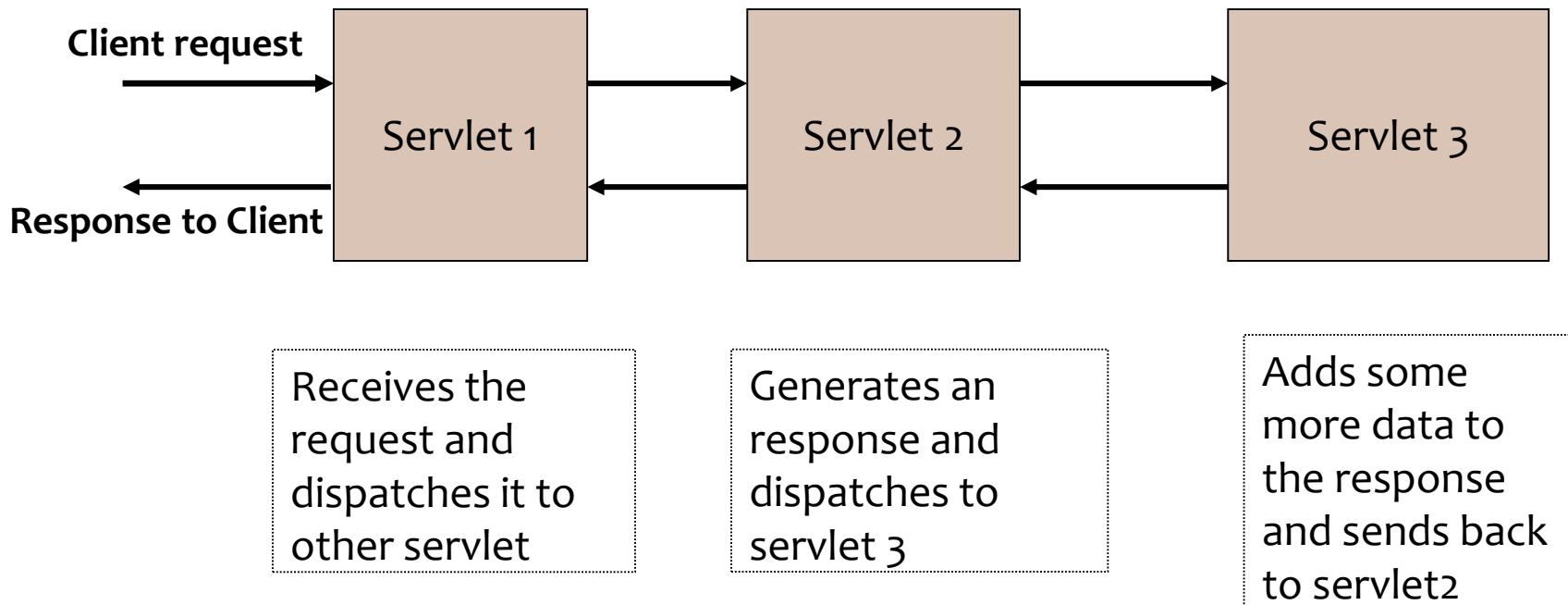
It may manipulate the request coming from servlet1 and propagate it to next servlet. It will overwrite the response generated by servlet1

Generates a response for the client. It will overwrite the response generated by servlet2



Dispatching using include() method

The request is received by one servlet and response is generated by same after including responses from other servlets .



MainPage.java

```
protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1)
throws ServletException, IOException {
    PrintWriter out = arg1.getWriter();
    out.println("Xoriant Solution Pvt Ltd.");
    String accounttype = arg0.getParameter("accounttype");
    RequestDispatcher rd;
    if (accounttype.equalsIgnoreCase("Current")) {
        rd = arg0.getRequestDispatcher("/current");
        rd.forward(arg0, arg1);
    } else {
        rd = arg0.getRequestDispatcher("/saving");
        rd.include(arg0, arg1);
    }
    out.println("Winchester Buliding ,Powai, Andheri(E) .");
}
```

Module 8 : The Cookies





- Overview :
 - Pros and Cons of using Cookies
 - Session and Persistent Cookies
 - Sending and Receiving Cookies
 - Application of Cookies

Preserving client data




- At the server side in database
- At the server side in any other storage
 - Text files
 - Serialized form
- At the client side in the form of cookies

More about Cookies

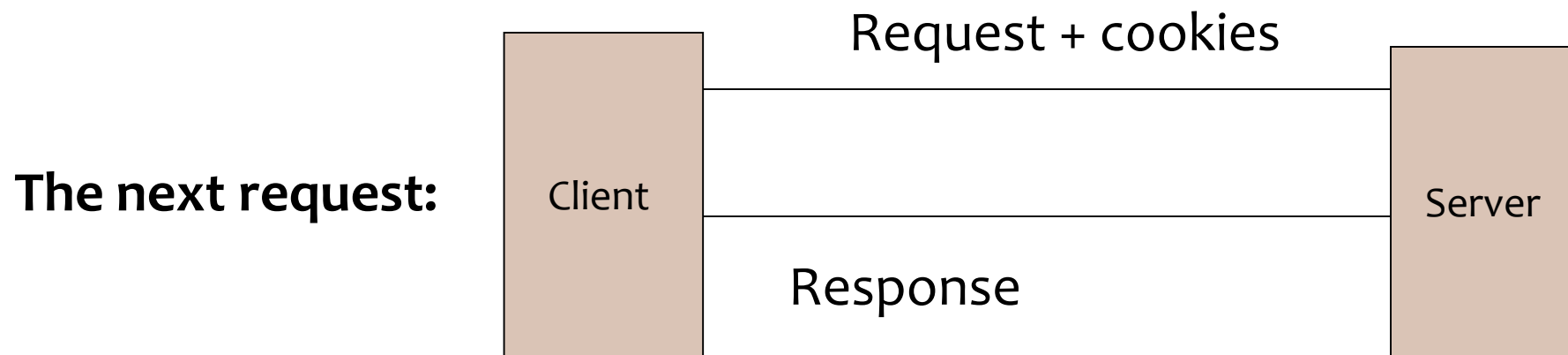
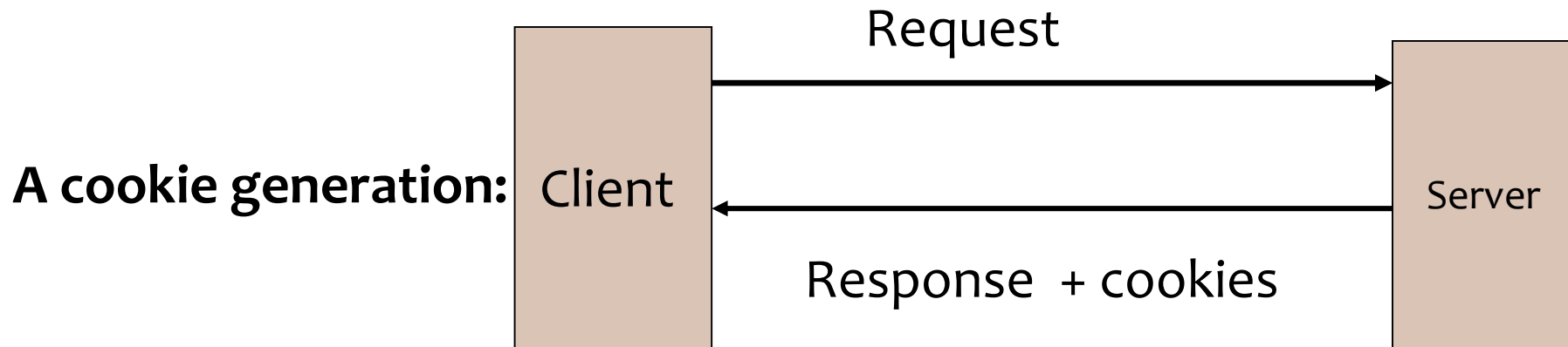
Cookies is:

-  a textual information stored at client's machine by the server.
-  neither a virus nor carrier of a virus.
-  cannot contain Executable code.
-  carries less confidential client specific information.

Browser restricts:

-  20 cookies for a website.
-  300 cookies for a browser instance.
-  4kb size for a single cookie.

Working of Cookies :



Creating Cookies

- Three steps to create a new cookie:
 - Create a new Cookie Object
 - `Cookie cookie = new Cookie (name, value);`
 - Set cookie attributes
 - `cookie.setMaxAge (60);`
 - Add your cookie to the response object:
 - `Response.addCookie (cookie)`

Creating Cookies (contd...)

```
protected void doGet(HttpServletRequest arg0, HttpServletResponse arg1) throws ServletException,
IOException{
    for(int i=0; i<3; i++){
        Cookie cook = new Cookie("Session"+i, "Value "+i);
        arg1.addCookie(cook);// This is a cookie with age for a session only.
        Cookie cook1 = new Cookie("Persistent"+i, "Value "+i);
        if (i==0)
            cook1.setMaxAge(60);
        if (i==1)
            cook1.setMaxAge(120);// Age set for two minutes.
        if (i==2)
            cook1.setMaxAge(60*60);
            arg1.addCookie(cook1);// This is a cookie persistent for an hour.
    }
    PrintWriter out = arg1.getWriter();
    out.println("To see the cookies, visit the");
    out.println("<AHREF = \"/MyServlets/showcookies\">SHOWCOOKIES</A>");
}
```

Reading Cookies

```
protected void doGet ( HttpServletRequest arg0, HttpServletResponse arg1)
throws ServletException, IOException {
    PrintWriter out = arg1.getWriter( );
    Cookie [ ] cookies = arg0.getCookies( );
    for(int i= 0 ; i < cookies.length; i++){
        out.println( "<TR>"+" <TD>" + cookies[i].getName()+
            " <TD>" + cookies[i].getValue() ) ;
    }
}
```

Applications of cookies





- Customizing the sites
- Focused advertising
- Storing information about the client

Disadvantages of cookies

- Browsers block cookies
- Threat to privacy

Module 9 : The Session Tracking

Overview :

-  Session Object
-  Setting and getting attributes
-  Session Tracking API
-  Browser and Server sessions



Need of Session Tracking

- To identify a user if he re-visits your site.
- To maintain the state , when there are series of requests from the same user.

Session Handling

- ▶ Accessing session object associated with current request :
 - ▶ `request.getSession();`
- ▶ Looking up information associated with a session :
 - ▶ `session.getAttribute();`
- ▶ Storing information in a session :
 - ▶ `session.setAttribute();`
- ▶ Discarding Session :
 - ▶ `session.invalidate();`

Various Ways of Session Tracking

- Cookies
- URL Rewriting
- Hidden Form Fields

The Hidden Form Fields

- ▶ The HTML :
 - ▶ `<html> <body>`
 - ▶ `<form action = "/MyServlet/hiddenfield" Method = POST align =Center>`
 - ▶ `USERNAME : <input type = text name = "user" align = center>`
 - ▶ `<input type = submit value = "Login" align = center>`
 - ▶ `</form> </body> </html>`

- ▶ `public class HiddenField extends HttpServlet {`
- ▶ `protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {`

- ▶ `String username = request.getParameter("user");`
- ▶ `PrintWriter out = response.getWriter();`
- ▶ `out.println("Hello !! click Submit to proceed further");`
- ▶ `out.println("<form action = \"/MyServlet/secondservlet\">");`
- ▶ `out.println("<input type = \"hidden\" name = \"user\" value = \"+username+\">");`
- ▶ `out.println("SURNAME : <input type = \"text\" name = \"surname\");`
- ▶ `out.println("<input type = \"Submit\" value = \"Submit\"</form>");`
- ▶ `}`

The Hidden Form Fields (contd...)

```
▶ public class SecondServlet extends HttpServlet {  
▶  
▶     protected void doGet(HttpServletRequest request, HttpServletResponse  
response) throws ServletException, IOException {  
▶         doPost(request,response);  
▶     }  
▶  
▶     protected void doPost(HttpServletRequest request, HttpServletResponse  
response) throws ServletException, IOException {  
▶         String uname = request.getParameter("user");  
▶         String surname = request.getParameter("surname");  
▶         PrintWriter out = response.getWriter();  
▶         out.println("Hello !!! "+uname);  
▶     }  
▶ }
```

The URL Rewriting

```
▶ public class URLRewriting extends HttpServlet{
▶
▶     public void doGet(HttpServletRequest request, HttpServletResponse response) throws
        IOException, ServletException{
▶
▶         response.setContentType("text/html");
▶         PrintWriter out = response.getWriter();
▶         String str = "/MyServlet/SessionTracking";
▶         HttpSession ses = request.getSession();
▶         String encodedURL = response.encodeURL(str);
▶         out.println("test session");
▶         out.println("<html><body>");
▶         out.println("<form action =" + encodedURL + ">");
▶         out.println("<input type = \"Submit\" value = \"Submit\"</form>");
▶         out.println("</body></html>");
▶
▶     }
▶ }
```

The Cookies

- Can pass information:
 - For both GET and POST type of requests.
 - Even if application has static web pages.
- Cannot pass information:
 - If Browser blocks cookies.
 - If web page is bookmarked

The Session Tracking API

- ▶ Provides a simple interface to manage the session automatically.
 - ▶ The API manages session through:
 - ▶ Cookies
 - ▶ URL rewriting
- ▶ The API can do:
 - ▶ Provides implicit session object
 - ▶ Creates unique session ID
 - ▶ Tracks session using session ID
 - ▶ Provides listeners for implicit session objects
- ▶ API Hierarchy :
 - ▶ public interface HttpSession

Methods of HttpSession

- isNew()
- invalidate()
- getAttribute(String name)
- setAttribute(String name)
- getMaxInactiveInterval()
- setMaxInactiveInterval()

The Session creation

```
class SessionTracking extends HttpServlet {  
    protected void doGet(HttpServletRequest request, HttpServletResponse  
        response) throws ServletException, IOException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        out.println("Test session");  
        HttpSession ses = request.getSession();  
        if (ses.isNew()) {  
            out.println("This is a new Sesion");  
        } else {  
            out.println("Welcome back");  
        }  
    }  
  
    protected void doPost(HttpServletRequest request, HttpServletResponse  
        response) throws ServletException, IOException {  
        doGet(request, response);  
    }  
}
```