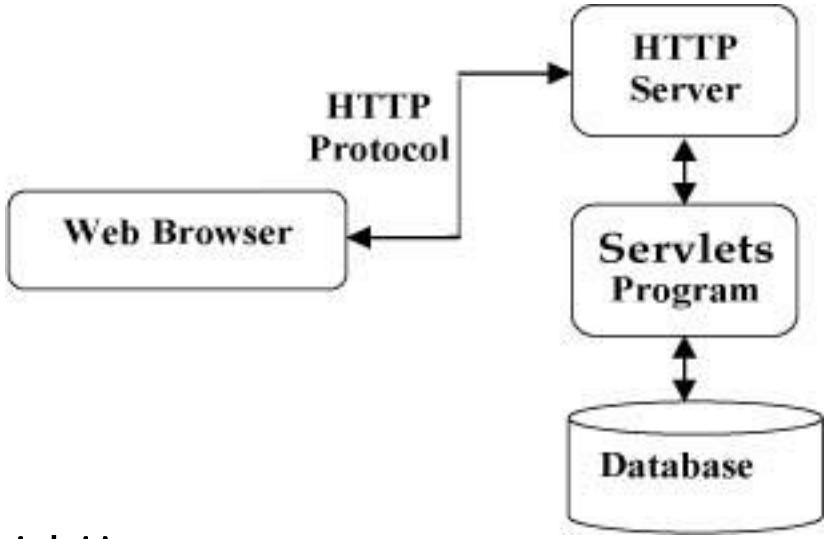
DEZVOLTAREA APLICATIILOR WEB LAB 2-3

Lect. Univ. Dr. Mihai Stancu

DEPARTAMENTUL DE INFORMATICĂ

Servlets Overview

- Servlets Architecture



- Servlets Packages
  - javax.servlet and javax.servlet.http
  - implement the Java Servlet and JSP specifications (Java Servlet 2.5, JSP 2.1)

Departamentul de Informatică

Servlets - Environment Setup

- Setting up Java Development Kit
  - JAVA_HOME
- Setting up Web Server: Tomcat
  - server.xml
  - startup.bat / shutdown.bat
- Setting up CLASSPATH
  - catalina.bat

Departamentul de Informatică

Servlets - Life Cycle

```
public void init() throws ServletException {
  // Initialization code...
}
```

```
public void service(ServletRequest request, ServletResponse response)
throws ServletException, IOException {
}
```

```
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // Servlet code
}
```

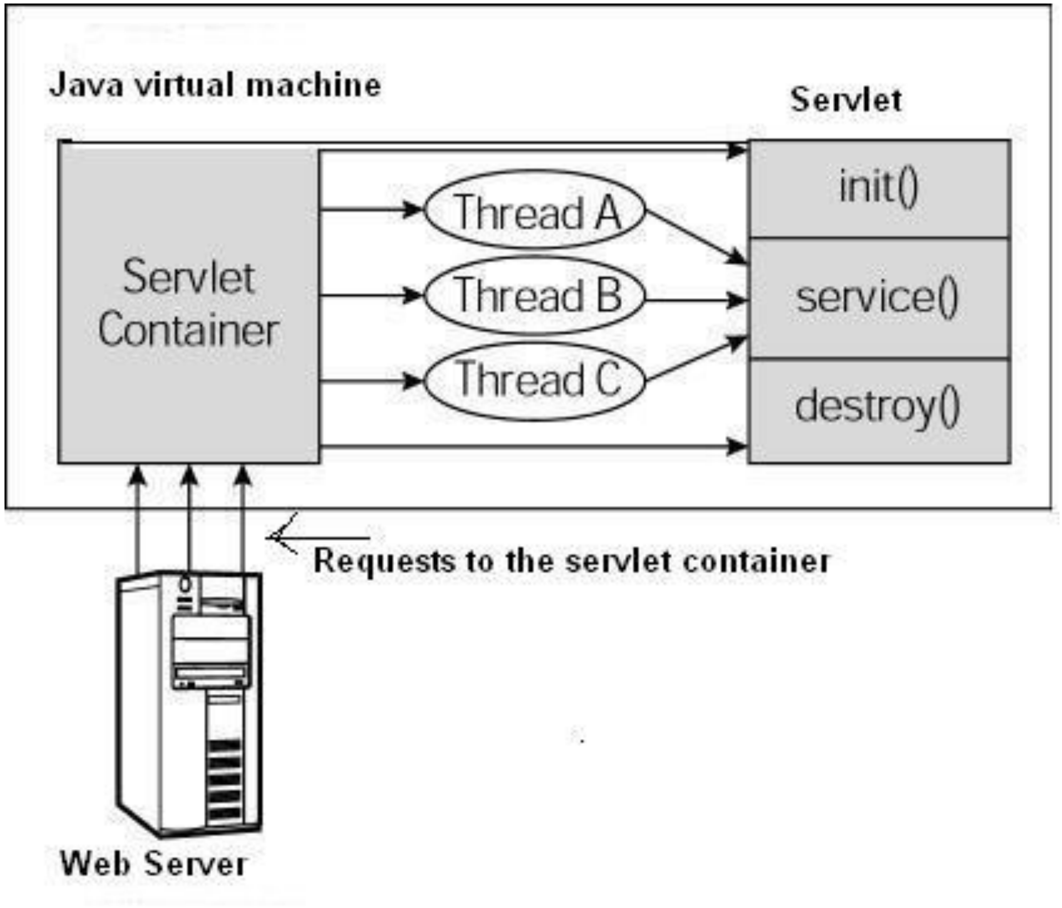
```
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // Servlet code
}
```

```
public void destroy () throws ServletException {
  // Finalization code...
}
```

Departamentul de Informatică

SERVLETS - LIFE CYCLE

- Architecture Digram



Servlets - Examples

➢ Hello World Servlet

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

  private String message;

  public void init() throws ServletException {
      // Do required initialization
      message = "Hello World";
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      // Set response content type
      response.setContentType("text/html");

      // Actual logic goes here.
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
  }

  public void destroy() {
      // do nothing.
  }
}
```

Servlets - Examples

- Servlet Deployment
  - <Tomcat-installation-directory>/webapps/myapp/WEB-INF/classes
  - <Tomcat-installation-directory>/webapps/myapp/WEB-INF/web.xml

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

<servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

Servlets Form Data

- GET method
  - http://www.test.com/hello?key1=value1&key2=value2
  - **doGet()** method
  - QUERY_STRING header
- POST method
  - Reading Form Data using Servlet
    - getParameter(): get the value of a form parameter.
    - getParameterValues(): returns multiple values, for example checkbox.
    - getParameterNames(): a complete list of all parameters in the current request.

Servlets Form Data

- GET Method Example Using URL
  - http://localhost:8080/HelloForm?first_name=ZARA&last_name=ALI

```
public class HelloForm extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
      // Set response content type
      response.setContentType("text/html");

      PrintWriter out = response.getWriter();
      String title = "Using GET Method to Read Form Data";
      String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
      out.println(docType + "<html>\n<head><title>" + title
         + "</title></head>\n"
         + "<body bgcolor=\"#f0f0f0\">\n"
         + "<h1 align=\"center\">" + title + "</h1>\n"
         + "<ul>\n" + "   <li><b>First Name</b>: "
         + request.getParameter("first_name") + "\n"
         + "  <li><b>Last Name</b>: "
         + request.getParameter("last_name") + "\n" + "</ul>\n"
         + "</body></html>");
  }
}
```

Servlets Form Data

- GET Method Example Using FORM

```
<html>
<body>
     <form action="HelloForm" method="GET">
          First Name: <input type="text" name="first_name">
          <br />
          Last Name: <input type="text" name="last_name" />
          <input type="submit" value="Submit" />
     </form>
</body>
</html>
```

Servlets Form Data

- POST Method Example Using FORM

```
public class HelloForm extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws ServletException, IOException {
      // Set response content type
      response.setContentType("text/html");

      PrintWriter out = response.getWriter();
      String title = "Using GET Method to Read Form Data";
      String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " + "transitional//en\">\n";
      out.println(docType +
                "<html>\n" + "<head><title>" + title + "</title></head>\n" +
                "<body bgcolor=\"#f0f0f0\">\n" +
                "<h1 align=\"center\">" + title + "</h1>\n<ul>\n" +
                "  <li><b>First Name</b>: "
                + request.getParameter("first_name") + "\n" +
                "  <li><b>Last Name</b>: "
                + request.getParameter("last_name") + "\n" +
                "</ul>\n" + "</body></html>");
  }
  // Method to handle POST method request.
  public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
      throws ServletException, IOException {
     doGet(request, response);
  }
}
```

Servlets Form Data

- Passing Checkbox Data to Servlet Program

```
<html>
<body>
     <form action="CheckBox" method="POST" target="_blank">
     <input type="checkbox" name="maths" checked="checked" /> Maths
     <input type="checkbox" name="physics"  /> Physics
     <input type="checkbox" name="chemistry" checked="checked" />
          Chemistry
     <input type="submit" value="Select Subject" />
     </form>
</body>
</html>
```

Servlets Form Data

➢ Passing Checkbox Data to Servlet Program

```
public class CheckBox extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                   throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String title = "Reading Checkbox Data";
      String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
      out.println(docType + "<html>\n" + "<head><title>" + title
         + "</title></head>\n<body bgcolor=\"#f0f0f0\">\n<h1 align=\"center\">"
         + title + "</h1>\n<ul>"
         + "\n  <li><b>Maths Flag : </b>: " + request.getParameter("maths")
         + "\n <li><b>Physics Flag: </b>: " + request.getParameter("physics")
         + "\n <li><b>Chemistry Flag: </b>: " + request.getParameter("chemistry")
         + "\n</ul>\n</body></html>");
  }
  // Method to handle POST method request.
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
     doGet(request, response);
  }
}
```

Servlets Form Data

- Reading All Form Parameters

```
public class ReadParams extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String title = "Reading All Form Parameters";
      String docType = "<!doctype html public \"-//w3c//dtd html 4.0 "
          + "transitional//en\">\n";
      out.println(docType + "<html>\n<head><title>" + title + "</title></head>\n"
          + "<body bgcolor=\"#f0f0f0\">\n" + "<h1 align=\"center\">" + title
          + "</h1>\n"
          + "<table width=\"100%\" border=\"1\" align=\"center\">\n"
          + "<tr bgcolor=\"#949494\">\n<th>Param Name</th><th>Param Value(s) </th>\n"
          + "</tr>\n");

      Enumeration paramNames = request.getParameterNames();
  ...
```

Servlets Form Data

➢ Reading All Form Parameters

```
...
      Enumeration paramNames = request.getParameterNames();
      while(paramNames.hasMoreElements()) {
         String paramName = (String)paramNames.nextElement();
         out.print("<tr><td>" + paramName + "</td>\n<td>");
         String[] paramValues = request.getParameterValues(paramName);
         // Read single valued data
         if (paramValues.length == 1) {
           String paramValue = paramValues[0];
           if (paramValue.length() == 0)
             out.println("<i>No Value</i>");
           else
             out.println(paramValue);
         } else {
             // Read multiple valued data
             out.println("<ul>");
             for(int i=0; i < paramValues.length; i++) {
                out.println("<li>" + paramValues[i]);
             }
             out.println("</ul>");
         }
      }
      out.println("</tr>\n</table>\n</body></html>");
  }
  // Method to handle POST method request.
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
     doGet(request, response);
  }
}
```

Servlets Form Data

- Reading All Form Parameters

```
<html>
<body>
     <form action="ReadParams" method="POST" target="_blank">
          <input type="checkbox" name="maths" checked="checked" /> Maths
          <input type="checkbox" name="physics"  /> Physics
          <input type="checkbox" name="chemistry" checked="checked" /> Chem
          <input type="submit" value="Select Subject" />
     </form>
</body>
</html>
```

Servlets - Client HTTP Request

Header	Description
Accept	This header specifies the MIME types that the browser or other clients can handle. Values of image/png or image/jpeg are the two most common possibilities.
Accept-Charset	This header specifies the character sets the browser can use to display the information. For example ISO-8859-1.
Accept-Encoding	This header specifies the types of encodings that the browser knows how to handle. Values of gzip or compress are the two most common possibilities.
Accept-Language	This header specifies the client's preferred languages in case the servlet can produce results in more than one language. For example en, en-us, ru, etc.
Authorization	This header is used by clients to identify themselves when accessing password-protected Web pages.
Connection	This header indicates whether the client can handle persistent HTTP connections. Persistent connections permit the client or other browser to retrieve multiple files with a single request. A value of Keep-Alive means that persistent connections should be used
Content-Length	This header is applicable only to POST requests and gives the size of the POST data in bytes.
Cookie	This header returns cookies to servers that previously sent them to the browser.
Host	This header specifies the host and port as given in the original URL.
If-Modified-Since	This header indicates that the client wants the page only if it has been changed after the specified date. The server sends a code, 304 which means Not Modified header if no newer result is available.
If-Unmodified-Since	This header is the reverse of If-Modified-Since; it specifies that the operation should succeed only if the document is older than the specified date.
Referer	This header indicates the URL of the referring Web page. For example, if you are at Web page 1 and click on a link to Web page 2, the URL of Web page 1 is included in the Referer header when the browser requests Web page 2.
User-Agent	This header identifies the browser or other client making the request and can be used to return different content to different types of browsers.

Servlets - Client HTTP Request

Methods to read HTTP Header	
Cookie[] getCookies()	String getMethod()
Enumeration getAttributeNames()	String getParameter(String name)
Enumeration getHeaderNames()	String getPathInfo()
Enumeration getParameterNames()	String getProtocol()
HttpSession getSession()	String getQueryString()
HttpSession getSession(boolean create)	String getRemoteAddr()
Locale getLocale()	String getRemoteHost()
Object getAttribute(String name)	String getRemoteUser()
ServletInputStream getInputStream()	String getRequestURI()
Enumeration getAttributeNames()	String getRequestedSessionId()
String getAuthType()	String getServletPath()
String getCharacterEncoding()	String[] getParameterValues(String name)
String getContentType()	boolean isSecure()
String getContextPath()	int getContentLength()
String getHeader(String name)	int getIntHeader(String name)

Servlets - Client HTTP Request

```
public class DisplayHeader extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
                              throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String title = "HTTP Header Request Example";
      String docType = "<!doctype html public \"-//w3c//dtd html 4.0 "
          + "transitional//en\">\n";
      out.println(docType + "<html>\n" + "<head><title>" + title + "</title></head>\n"+
        "<body bgcolor=\"#f0f0f0\">\n" + "<h1 align=\"center\">" + title + "</h1>\n" +
        "<table width=\"100%\" border=\"1\" align=\"center\">\n" +
        "<tr bgcolor=\"#949494\">\n<th>Header Name</th><th>Header Value(s)</th>\n"+
        "</tr>\n");

      Enumeration headerNames = request.getHeaderNames();

      while(headerNames.hasMoreElements()) {
         String paramName = (String)headerNames.nextElement();
         out.print("<tr><td>" + paramName + "</td>\n");
         String paramValue = request.getHeader(paramName);
         out.println("<td> " + paramValue + "</td></tr>\n");
      }
      out.println("</table>\n</body></html>");
  }
  // Method to handle POST method request.
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
     doGet(request, response);
  }
}
```

Servlets - Server HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/html
Header2: ...
...
HeaderN: ...
  (Blank Line)
<!doctype ...>
<html>
<head>...</head>
<body>
...
</body>
</html>
```

Servlets - Server HTTP Response

Header	Description
Allow	This header specifies the request methods (GET, POST, etc.) that the server supports.
Cache-Control	This header specifies the circumstances in which the response document can safely be cached. It can have values public, private or no-cache etc. Public means document is cacheable, Private means document is for a single user and can only be stored in private (nonshared) caches and no-cache means document should never be cached.
Connection	This header instructs the browser whether to use persistent in HTTP connections or not. A value of close instructs the browser not to use persistent HTTP connections and keep-alive means using persistent connections.
Content-Disposition	Lets you request that the browser ask the user to save the response to disk in a file of the given name.
Content-Encoding	This header specifies the way in which the page was encoded during transmission.
Content-Language	This header signifies the language in which the document is written. For example en, en-us, ru, etc.
Content-Length	This header indicates the number of bytes in the response. This information is needed only if the browser is using a persistent (keep-alive) HTTP connection.
Content-Type	This header gives the MIME (Multipurpose Internet Mail Extension) type of the response document.
Expires	This header specifies the time at which the content should be considered out-of-date and thus no longer be cached.
Last-Modified	This header indicates when the document was last changed. The client can then cache the document and supply a date by an If-Modified-Since request header in later requests.
Location	This header should be included with all responses that have a status code in the 300s. This notifies the browser of the document address. The browser automatically reconnects to this location and retrieves the new document.
Refresh	This header specifies how soon the browser should ask for an updated page. You can specify time in number of seconds after which a page would be refreshed.
Retry-After	This header can be used in conjunction with a 503 (Service Unavailable) response to tell the client how soon it can repeat its request.

Servlets - Server HTTP Response

Methods to read HTTP Header	
String encodeRedirectURL(String url)	void sendError(int sc)
String encodeURL(String url)	void sendError(int sc, String msg)
boolean containsHeader(String name)	void sendRedirect(String location)
Enumeration getParameterNames()	void setBufferSize(int size)
boolean isCommitted()	void setCharacterEncoding(String charset)
void addCookie(Cookie cookie)	void setContentLength(int len)
void addDateHeader(String name, long date)	void setContentType(String type)
void addHeader(String name, String value)	void setDateHeader(String name, long date)
void addIntHeader(String name, int value)	void setHeader(String name, String value)
void flushBuffer()	void setIntHeader(String name, int value)
void reset()	void setLocale(Locale loc)
void resetBuffer()	void setStatus(int sc)

Servlets - Server HTTP Response

➢ HTTP Header Response Example

```
public class Refresh extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                          throws ServletException, IOException {
      // Set refresh, autoload time as 5 seconds
      response.setIntHeader("Refresh", 5);

      response.setContentType("text/html");

      // Get current time
      Calendar calendar = new GregorianCalendar();
      String am_pm;
      int hour = calendar.get(Calendar.HOUR);
      int minute = calendar.get(Calendar.MINUTE);
      int second = calendar.get(Calendar.SECOND);
      if(calendar.get(Calendar.AM_PM) == 0)
        am_pm = "AM";
      else
        am_pm = "PM";

      String CT = hour+":"+ minute +":"+ second +" "+ am_pm;

      PrintWriter out = response.getWriter();
      String title = "Auto Refresh Header Setting";
      String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
                                                    "transitional//en\">\n";
      out.println(docType + "<html>\n<head><title>" + title + "</title></head>\n"+
        "<body bgcolor=\"#f0f0f0\">\n" + "<h1 align=\"center\">" + title + "</h1>\n" +
        "<p>Current Time is: " + CT + "</p>\n");
  }
  // Method to handle POST method request...
}
```

Servlets - HTTP Status Codes

Code:	Message:	Description:
100	Continue	Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request
101	Switching Protocols	The server switches protocol.
200	OK	The request is OK
201	Created	The request is complete, and a new resource is created
202	Accepted	The request is accepted for processing, but the processing is not complete.
203	Non-authoritative Information	
204	No Content	
205	Reset Content	
206	Partial Content	
300	Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses
301	Moved Permanently	The requested page has moved to a new url
302	Found	The requested page has moved temporarily to a new url
303	See Other	The requested page can be found under a different url
304	Not Modified	
305	Use Proxy	
306	Unused	This code was used in a previous version. It is no longer used, but the code is reserved.
307	Temporary Redirect	The requested page has moved temporarily to a new url.

Servlets - Http Status Codes

Code:	Message:	Description:
400	Bad Request	The server did not understand the request
401	Unauthorized	The requested page needs a username and a password
402	Payment Required	You can not use this code yet
403	Forbidden	Access is forbidden to the requested page
404	Not Found	The server can not find the requested page.
405	Method Not Allowed	The method specified in the request is not allowed.
406	Not Acceptable	The server can only generate a response that is not accepted by the client.
407	Proxy Authentication Required	You must authenticate with a proxy server before this request can be served.
408	Request Timeout	The request took longer than the server was prepared to wait.
409	Conflict	The request could not be completed because of a conflict.
410	Gone	The requested page is no longer available.
411	Length Required	The "Content-Length" is not defined. The server will not accept the request without it.
412	Precondition Failed	The precondition given in the request evaluated to false by the server.
413	Request Entity Too Large	The server will not accept the request, because the request entity is too large.
414	Request-url Too Long	The server will not accept the request, because the url is too long. Occurs when you convert a "post" request to a "get" request with a long query information.
415	Unsupported Media Type	The server will not accept the request, because the media type is not supported.
417	Expectation Failed	
500	Internal Server Error	The request was not completed. The server met an unexpected condition
501	Not Implemented	The request was not completed. The server did not support the functionality required.
502	Bad Gateway	The request was not completed. The server received an invalid response from the upstream server
503	Service Unavailable	The request was not completed. The server is temporarily overloading or down.
504	Gateway Timeout	The gateway has timed out.
505	HTTP Version Not Supported	The server does not support the "http protocol" version.

Servlets - Http Status Codes

Method	Description
public void setStatus (int statusCode)	This method sets an arbitrary status code. The setStatus method takes an int (the status code) as an argument. If your response includes a special status code and a document, be sure to call setStatus before actually returning any of the content with the PrintWriter.
public void sendRedirect(String url)	This method generates a 302 response along with a Location header giving the URL of the new document.
public void sendError(int code, String message)	This method sends a status code (usually 404) along with a short message that is automatically formatted inside an HTML document and sent to the client.

Servlets - Writing Filters

Method	Description
public void doFilter (ServletRequest, ServletResponse, FilterChain)	This method is called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
public void init(FilterConfig filterConfig)	This method is called by the web container to indicate to a filter that it is being placed into service.
public void destroy()	This method is called by the web container to indicate to a filter that it is being taken out of service.

```
<filter>
   <filter-name>LogFilter</filter-name>
   <filter-class>LogFilter</filter-class>
   <init-param>
            <param-name>test-param</param-name>
            <param-value>Initialization Paramter</param-value>
   </init-param>
</filter>
<filter-mapping>
   <filter-name>LogFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

Servlets - Writing Filters

```
// Implements Filter class
public class LogFilter implements Filter  {
   public void  init(FilterConfig config) 
                         throws ServletException{
      // Get init parameter 
      String testParam = config.getInitParameter("test-param"); 

      //Print the init parameter 
      System.out.println("Test Param: " + testParam); 
   }
   public void  doFilter(ServletRequest request, 
                 ServletResponse response,
                 FilterChain chain) 
                 throws java.io.IOException, ServletException {

      // Get the IP address of client machine.   
      String ipAddress = request.getRemoteAddr();

      // Log the IP address and current timestamp.
      System.out.println("IP "+ ipAddress + ", Time "
                                      + new Date().toString());

      // Pass request back down the filter chain
      chain.doFilter(request,response);
   }
   public void destroy( ){
      /* Called before the Filter instance is removed 
      from service by the web container*/
   }
}
```

Servlets - Exception Handling

- web.xml Configuration

```
<!-- servlet definition -->
<servlet>
        <servlet-name>ErrorHandler</servlet-name>
        <servlet-class>ErrorHandler</servlet-class>
</servlet>
<!-- servlet mappings -->
<servlet-mapping>
        <servlet-name>ErrorHandler</servlet-name>
        <url-pattern>/ErrorHandler</url-pattern>
</servlet-mapping>

<!-- error-code related error pages -->
<error-page>
    <error-code>404</error-code>
    <location>/ErrorHandler</location>
</error-page>
<error-page>
    <error-code>403</error-code>
    <location>/ErrorHandler</location>
</error-page>

<!-- exception-type related error pages -->
<error-page>
    <exception-type>javax.servlet.ServletException</exception-type>
    <location>/ErrorHandler</location>
</error-page>

<error-page>
    <exception-type>java.io.IOException</exception-type >
    <location>/ErrorHandler</location>
</error-page>
```

Servlets - Exception Handling

Attribute	Description
javax.servlet.error.status_code	This attribute give status code which can be stored and analysed after storing in a java.lang.Integer data type.
javax.servlet.error.exception_type	This attribute gives information about exception type which can be stored and analysed after storing in a java.lang.Class data type.
javax.servlet.error.message	This attribute gives information exact error message which can be stored and analysed after storing in a java.lang.String data type.
javax.servlet.error.request_uri	This attribute gives information about URL calling the servlet and it can be stored and analysed after storing in a java.lang.String data type.
javax.servlet.error.exception	This attribute gives information the exception raised which can be stored and analysed after storing in a java.lang.Throwable data type.
javax.servlet.error.servlet_name	This attribute gives servlet name which can be stored and analysed after storing in a java.lang.String data type.

Servlets - Exception Handling

```
public class ErrorHandler extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException {
      // Analyze the servlet exception
     Throwable throwable = (Throwable) request.getAttribute("javax.servlet.error.exception");
     Integer statusCode = (Integer) request.getAttribute("javax.servlet.error.status_code");
     String servletName = (String) request.getAttribute("javax.servlet.error.servlet_name");
     if (servletName == null){
        servletName = "Unknown";
     }
     String requestUri = (String) request.getAttribute("javax.servlet.error.request_uri");
     if (requestUri == null){
        requestUri = "Unknown";
     }
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     String title = "Error/Exception Information";
     String docType = "<!doctype html public \"-//w3c//dtd html 4.0 transitional//en\">\n";
     out.println(docType + "<html>\n<head><title>" + title + "</title></head>\n" +
           "<body bgcolor=\"#f0f0f0\">\n");
     if (throwable == null && statusCode == null) {
        out.println("<h2>Error information is missing</h2>");
        out.println("Please return to the <a href=\"" +
           response.encodeURL("http://localhost:8080/") + "\">Home Page</a>.");
     } else if (statusCode != null){
        out.println("The status code : " + statusCode);
     } else {
        out.println("<h2>Error information</h2>");
        out.println("Servlet Name : " + servletName + "</br></br>");
        out.println("Exception Type : " + throwable.getClass().getName() + "</br></br>");
        out.println("The request URI: " + requestUri + "<br><br>");
        out.println("The exception message: " + throwable.getMessage( ));
     }
     out.println("</body></html>");
  }
}
```

Servlets - Cookies Handling

```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;
                 path=/; domain=mysite.com
Connection: close
Content-Type: text/html




GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)
Host: zink.demon.co.uk:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz
```

- Servlet Cookies Methods
  - void *setDomain*(String pattern), String *getDomain*(), void *setMaxAge*(int expiry), int *getMaxAge*(), String *getName*(), void *setValue*(String newValue), String *getValue*(), void *setPath*(String uri), String *getPath*(), void *setSecure*(boolean flag), void *setComment*(String purpose), String *getComment*()

Servlets - Cookies Handling

- Setting Cookies with Servlet

```
 public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
      // Create cookies for first and last names.
    Cookie firstName = new Cookie("first_name", request.getParameter("first_name"));
    Cookie lastName = new Cookie("last_name", request.getParameter("last_name"));

    // Set expiry date after 24 Hrs for both the cookies.
    firstName.setMaxAge(60*60*24);
    lastName.setMaxAge(60*60*24);

    // Add both the cookies in the response header.
    response.addCookie( firstName );
    response.addCookie( lastName );

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Setting Cookies Example";
    String docType = "<!doctype html public \"-//w3c//dtd html 4.0
transitional//en\">\n";
    out.println(docType + "<html>\n" +
                "<head><title>" + title + "</title></head>\n" +
                "<body bgcolor=\"#f0f0f0\">\n" +
                "<h1 align=\"center\">" + title + "</h1>\n" +
                "<ul>\n" +
                "  <li><b>First Name</b>: "
                + request.getParameter("first_name") + "\n" +
                "  <li><b>Last Name</b>: "
                + request.getParameter("last_name") + "\n" +
                "</ul>\n" +
                "</body></html>");
  }
```

Servlets - Cookies Handling

- Reading Cookies with Servlet

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                throws ServletException, IOException {
      Cookie cookie = null;
      Cookie[] cookies = null;
      // Get an array of Cookies associated with this domain
      cookies = request.getCookies();

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String title = "Reading Cookies Example";
      String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";
      out.println(docType +
                "<html>\n" +
                "<head><title>" + title + "</title></head>\n" +
                "<body bgcolor=\"#f0f0f0\">\n" );
      if( cookies != null ){
         out.println("<h2> Found Cookies Name and Value</h2>");
         for (int i = 0; i < cookies.length; i++){
            cookie = cookies[i];
            out.print("Name : " + cookie.getName( ) + ",  ");
            out.print("Value: " + cookie.getValue( )+" <br/>");
         }
      }else{
          out.println(
            "<h2>No cookies founds</h2>");
      }
      out.println("</body>");
      out.println("</html>");
   }
```

Servlets - Cookies Handling

- Delete Cookies with Servlet

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                              throws ServletException, IOException {
     Cookie cookie = null;
     Cookie[] cookies = null;
     // Get an array of Cookies associated with this domain
     cookies = request.getCookies();

     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     String title = "Delete Cookies Example";
     out.println("<html>\n<head><title>" + title + "</title></head>\n" +
               "<body bgcolor=\"#f0f0f0\">\n" );
      if( cookies != null ){
        out.println("<h2> Cookies Name and Value</h2>");
        for (int i = 0; i < cookies.length; i++){
           cookie = cookies[i];
           if((cookie.getName()).compareTo("first_name") == 0 ){
                cookie.setMaxAge(0);
                response.addCookie(cookie);
                out.print("Deleted cookie : " + cookie.getName( ) + "<br/>");
           }
           out.print("Name : " + cookie.getName() + ",  ");
           out.print("Value: " + cookie.getValue()+" <br/>");
        }
     } else {
         out.println("<h2>No cookies founds</h2>");
     }
     out.println("</body>");
     out.println("</html>");
  }
```

SERVLETS - SESSION TRACKING

- Servlet Cookies Methods
  - HttpSession session = request.getSession();
  - Servlet Cookies Methods
    - Object getAttribute(String name), Enumeration getAttributeNames(), long getCreationTime(), String getId(), long getLastAccessedTime(), int getMaxInactiveInterval(), void invalidate(), boolean isNew(), void removeAttribute(String name), void setAttribute(String name, Object value), void setMaxInactiveInterval(int interval)
  - web.xml

```
<session-config>
   <session-timeout>15</session-timeout>
 </session-config>
```

➢ Session Tracking Example

```
public class SessionTrack extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                throws ServletException, IOException {
      // Create a session object if it is already not created.
      HttpSession session = request.getSession(true);
      // Get session creation time.
      Date createTime = new Date(session.getCreationTime());
      // Get last access time of this web page.
      Date lastAccessTime = new Date(session.getLastAccessedTime());

      String title = "Welcome Back to my website";
      Integer visitCount = new Integer(0);
      String visitCountKey = new String("visitCount");
      String userIDKey = new String("userID");
      String userID = new String("ABCD");

      // Check if this is new comer on your web page.
      if (session.isNew()){
         title = "Welcome to my website";
         session.setAttribute(userIDKey, userID);
      } else {
         visitCount = (Integer)session.getAttribute(visitCountKey);
         visitCount = visitCount + 1;
         userID = (String)session.getAttribute(userIDKey);
      }
      session.setAttribute(visitCountKey,  visitCount);

      // Set response content type
      ...
```

Session Tracking Example

```
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html>\n<head><title>" + title + "</title></head>\n" +
          "<body bgcolor=\"#f0f0f0\">\n<h1 align=\"center\">" + title + "</h1>\n" +
          "<h2 align=\"center\">Session Infomation</h2>\n" +
          "<table border=\"1\" align=\"center\">\n" +
          "<tr bgcolor=\"#949494\">\n" +
          "  <th>Session info</th><th>value</th></tr>\n" +
          "<tr>\n" +
          "  <td>id</td>\n" +
          "  <td>" + session.getId() + "</td></tr>\n" +
          "<tr>\n" +
          "  <td>Creation Time</td>\n" +
          "  <td>" + createTime +
          "  </td></tr>\n" +
          "<tr>\n" +
          "  <td>Time of Last Access</td>\n" +
          "  <td>" + lastAccessTime +
          "  </td></tr>\n" +
          "<tr>\n" +
          "  <td>User ID</td>\n" +
          "  <td>" + userID +
          "  </td></tr>\n" +
          "<tr>\n" +
          "  <td>Number of visits</td>\n" +
          "  <td>" + visitCount + "</td></tr>\n" +
          "</table>\n" +
          "</body></html>");
   }
}
```

Servlets - File Uploading

- File Upload Form

```
<html>
<head>
<title>File Uploading Form</title>
</head>
<body>
          <h3>File Upload:</h3>
          Select a file to upload: <br />
          <form action="UploadServlet" method="post" enctype="multipart/form-data">
          <input type="file" name="file" size="50" /><br />
          <input type="submit" value="Upload File" />
          </form>
</body>
</html>
```

- Backend Servlet

```
<web-app>
....
<context-param>
    <description>Location to store uploaded file</description>
    <param-name>file-upload</param-name>
    <param-value>
         c:\apache-tomcat-5.5.29\webapps\data\
     </param-value>
</context-param>
....
</web-app>
```

Servlets - File Uploading

➢ Backend Servlet

```
public class UploadServlet extends HttpServlet {
   private boolean isMultipart;
   private String filePath;
   private int maxFileSize = 50 * 1024;
   private int maxMemSize = 4 * 1024;
   private File file ;

   public void init() {
      // Get the file location where it would be stored.
      filePath = getServletContext().getInitParameter("file-upload");
   }
   public void doPost(HttpServletRequest request, HttpServletResponse response)
             throws ServletException, java.io.IOException {
      // Check that we have a file upload request
      isMultipart = ServletFileUpload.isMultipartContent(request);
      response.setContentType("text/html");
      java.io.PrintWriter out = response.getWriter( );
      if( !isMultipart ){
         out.println("<html><head><title>Servlet upload</title></head>");
         out.println("<body><p>No file uploaded</p></body></html>");
         return;
      }
      DiskFileItemFactory factory = new DiskFileItemFactory();
      // maximum size that will be stored in memory
      factory.setSizeThreshold(maxMemSize);
      // Location to save data that is larger than maxMemSize.
      factory.setRepository(new File("c:\\temp"));
      // Create a new file upload handler
      ServletFileUpload upload = new ServletFileUpload(factory);
      upload.setSizeMax( maxFileSize ); // maximum file size to be uploaded.
```

SERVLETS - FILE UPLOADING

```
   try{
      List fileItems = upload.parseRequest(request); //Parse the req to get fileitems.
      Iterator i = fileItems.iterator(); // Process the uploaded file items
      out.println("<html><head><title>Servlet upload</title></head><body>");
      while (i.hasNext()) {
         FileItem fi = (FileItem)i.next();
         if (!fi.isFormField()) {
            // Get the uploaded file parameters
            String fieldName = fi.getFieldName();
            String fileName = fi.getName();
            String contentType = fi.getContentType();
            boolean isInMemory = fi.isInMemory();
            long sizeInBytes = fi.getSize();
            // Write the file
            if( fileName.lastIndexOf("\\") >= 0 ){
         file = new File(filePath + fileName.substring(fileName.lastIndexOf("\\")));
            } else {
         file = new File(filePath + fileName.substring(fileName.lastIndexOf("\\")+1));
            }
            fi.write( file ) ;
            out.println("Uploaded Filename: " + fileName + "<br>");
         }
      }
      out.println("</body</html>");
   } catch(Exception ex) {
       System.out.println(ex);
   }
  }
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                          throws ServletException, java.io.IOException {
        throw new ServletException("GET method used with " + getClass().getName()+":
POST method required.");
  }
}
```

Servlets - Page Redirection

- Example

```
public class PageRedirect extends HttpServlet{

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
      // Set response content type
      response.setContentType("text/html");

      // New location to be redirected
      String site = new String("http://www.photofuntoos.com");

      response.setStatus(response.SC_MOVED_TEMPORARILY);
      response.setHeader("Location", site);
    }
}
```

Servlets - Sending Email

- Basic

```
public class SendEmail extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                      throws ServletException, IOException {
      String to = "abcd@gmail.com"; // Recipient's email ID needs to be mentioned.
      String from = "web@gmail.com"; // Sender's email ID needs to be mentioned
      String host = "localhost"; // Assuming you are sending email from localhost
      Properties properties = System.getProperties(); // Get system properties
      properties.setProperty("mail.smtp.host", host); // Setup mail server
      Session session = Session.getDefaultInstance(properties); // Get default Session obj

      response.setContentType("text/html"); // Set response content type
      PrintWriter out = response.getWriter();
      try {
         MimeMessage message = new MimeMessage(session); // Create a default MimeMessage obj
         message.setFrom(new InternetAddress(from)); // Set From: header field of the header
         message.addRecipient(Message.RecipientType.TO,
                              new InternetAddress(to)); // Set To: header field of the header
         message.setSubject("This is the Subject Line!"); // Set Subject: header field
         message.setText("This is actual message"); // Now set the actual message
         Transport.send(message); // Send message


         String title = "Send Email";
         String res = "Sent message successfully....";
         String docType = "<!doctype html public \"-//w3c//dtd html 4.0 " +
                                                     "transitional//en\">\n";
         out.println(docType + "<html>\n" + "<head><title>" + title + "</title></head>\n" +
         "<body bgcolor=\"#f0f0f0\">\n" + "<h1 align=\"center\">" + title + "</h1>\n" +
         "<p align=\"center\">" + res + "</p>\n" + "</body></html>");
      } catch (MessagingException mex) {
         mex.printStackTrace();
      }
   }
}
```

Servlets - Sending Email

- HTML Email

```
public class SendEmail extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                       throws ServletException, IOException {
      String to = "abcd@gmail.com"; // Recipient's email ID needs to be mentioned.
      String from = "web@gmail.com"; // Sender's email ID needs to be mentioned
      String host = "localhost"; // Assuming you are sending email from localhost
      Properties properties = System.getProperties(); // Get system properties
      properties.setProperty("mail.smtp.host", host); // Setup mail server
      Session session = Session.getDefaultInstance(properties); // Get default Session obj

      response.setContentType("text/html"); // Set response content type
      PrintWriter out = response.getWriter();
      try {
         MimeMessage message = new MimeMessage(session); // Create a default MimeMessage obj
         message.setFrom(new InternetAddress(from)); // Set From: header field of the header
         message.addRecipient(Message.RecipientType.TO,
                              new InternetAddress(to)); // Set To: header field of the header
         message.setSubject("This is the Subject Line!"); // Set Subject: header field

         // Send the actual HTML message, as big as you like
         message.setContent("<h1>This is actual message</h1>",
                               "text/html" );
         Transport.send(message); // Send message

         ...

      } catch (MessagingException mex) {
         mex.printStackTrace();
      }
   }
}
```

Servlets - Sending Email

- Email with attachment

```
public class SendEmail extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                       throws ServletException, IOException {
      ...
      try {
         MimeMessage message = new MimeMessage(session); // Create a default MimeMessage obj
         message.setFrom(new InternetAddress(from)); // Set From: header field of the header
         message.addRecipient(Message.RecipientType.TO,
                           new InternetAddress(to)); // Set To: header field of the header
         message.setSubject("This is the Subject Line!"); // Set Subject: header field

         BodyPart messageBodyPart = new MimeBodyPart(); // Create the message part
         messageBodyPart.setText("This is message body"); // Fill the message
         Multipart multipart = new MimeMultipart(); // Create a multipart message
         multipart.addBodyPart(messageBodyPart); // Set text message part
         messageBodyPart = new MimeBodyPart(); // Part two is attachment
         String filename = "file.txt";
         DataSource source = new FileDataSource(filename);
         messageBodyPart.setDataHandler(new DataHandler(source));
         messageBodyPart.setFileName(filename);
         multipart.addBodyPart(messageBodyPart);

         message.setContent(multipart ); // Part two is attachment
         Transport.send(message); // Send message

         ...

      } catch (MessagingException mex) {
         mex.printStackTrace();
      }
   }
}
```

Resurse utile

- Homework
  - Login servlet
  - JDBC connection
  - Login Filter