## SERVLET

- Servlets are discrete java classes which reside into the Container of a Web Server.
- Where Server follows REQUEST-RESPONSE model, where server takes the request, process the data and respond back to the client.

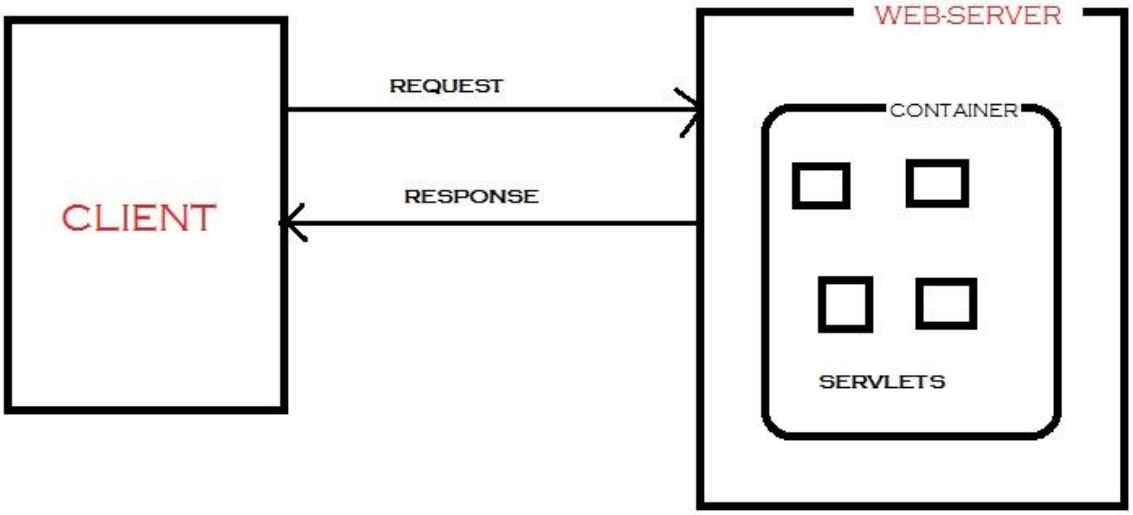



FIG: SERVLET DESCRIPTION WITH REQUEST-RESPONSE MODEL

- Client make request using HTTP and server respond back using HTML.
- Hence both the client and server should know HTTP-HTML
- The whole scenario can be overviewed as

  - Client makes an HTTP request for a servlet file
  - Web Server accepts the request, and invokes the specified servlet.
  - If servlet is still unloaded JVM loads it.
  - Servlet process the data and generates a HTML response.
  - Servlet passes the response to Web-Server and hence it passes it to Client1

## ADVANTAGES OF SERVLET OVER CGI

- **PERFORMANCE**
  - Performance of Servlet is significantly better.
  - Servlets execute within the address space of a Web server.
  - It is not necessary to create a separate process to handle each client request

- **PLATFORM-INDEPENDENT**
  - Servlets are platform-independent because they are written in Java.
  - A number of Web servers from different vendors offer the Servlet API

- **SECURITY**
  - Java Security Manager on server enforces restriction on resources, hence providing misuse.

- **INTER-COMMUNICATION**
  - As Servlets are written in Java, all the Java class libraries are available to a servlet.
  - Hence making it to communicate with various kind of software using SOCKET, RMI etc.

## LIFE CYCLE OF SERVLET

- Three methods are central to the life cycle of a servlet.
- These are **init( )**, **service( )**,and **destroy( )**.
- They are implemented by every servlet and are invoked at specific times by the server.
- Assume that a user enters a Uniform Resource Locator (URL) to a Web browser.
- The browser then generates an HTTP request for this URL.
- This request is then sent to the appropriate server.
- This HTTP request is received by the Web server.
- The server maps this request to a particular servlet.
- The servlet is dynamically retrieved and loaded into the address space of the server.
- Now the Server invokes the init()

- **init()**

- init() is invoked when servlet is loaded for the very first time.
- init() method is invoked by the web server when it is loaded into the memory.
- init() is called once in the life time of a servlet.
- Signature of init() is

  **public void init(ServletConfig config) throws ServletException**

- It is possible to pass initialization parameter to servlet so that it can configure itself.
- The parameter is passed using ServletConfig object.

- **service()**

- All the requests to a servlets are handled by service().

- service() cannot be invoked before invoking init().
- Signature of service()

  **public void service(ServletRequest request, ServletResponse response) throws ServletException,IOException.**

- It is very clear from the signature that service() takes the request, process the data and respond back to the client.
- service() accepts the HTTP request, and generate the HTTP response which is later converted to HTML format on browser.
- service() is invoked every time when a request for a servlet arrive.

- **destroy()**

- Server may decide to unload the servlet the memory.
- It unloads it using destroy().
- Signature of destroy()

  **public void destroy()**

## SERVLET-API

- Servlet API consists of two packages
  - javax.servlet
  - javax.servlet.http

- **THE javax.servlet package**

  - The **javax.servlet** package contains a number of interfaces and classes that establish the framework in which servlets operate.
  - We will go through some descriptions of the interfaces of javax.servlet package
  - The most significant of these is **Servlet**.
  - All servlets must implement this interface or extend a class that implements the interface.
  - The **ServletRequest** and **ServletResponse** interfaces are also very important.

Interfaces of javax.servlet package

**javax.servlet.Servlet**
- Declares life cycle methods for a servlet.

**javax.servlet.ServletConfig**
- Allows servlets to get initialization parameters.

**javax.servlet.ServletContext**
- Enables servlets to log events and access information about their environment.

**javax.servlet.ServletRequest**
- Used to read data from a client request.

**javax.servlet.ServletResponse**
- Used to write data to a client response.

**javax.servlet.SingleThreadModel**

- Indicates that the servlet is thread safe.

## Servlet Interface

- Servlet is the most important interface in the Servlet-API.
- Servlet interface resides in javax.servlet package.
- For being a servlet it is required that the class should implement Servlet interface or extend a class that implements Servlet interface.

- Servlet interface defines all the life-cycle methods of Servlet.
- These life-cycle methods are used to initialize a servlet so that they can accept the request, process the data and responds back.
- It also contain method to unload the servlet.

**METHODS**

- public void **init**(ServletConfig config)throws ServletException
  - init() is invoked by the servlet container to indicate to a servlet that the servlet is being placed into service.
  - The servlet container calls the init() exactly once after instantiating the servlet.
  - The init() must complete successfully before the servlet can receive any requests.

- public void **service**(ServletRequest request,ServletResponse response) throws ServletException,IOException
  - Called by the servlet container to allow the servlet to respond to a request.
  - This method is only called after the servlet's init() method has completed successfully.

- public void **destroy**()
  - Called by the servlet container to indicate to a servlet that the servlet is being taken out of service

- public String **getServletInfo**()
  - Returns information about the servlet, such as author, version, and copyright.

- public ServletConfig **getServletConfig**()
  - Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.
  - The ServletConfig object returned is the one passed to the init method.

## ServletConfig Interface

- ServletConfig is an important interface in javax.servlet package.
- ServletConfig encapsulates the servlet configuration.
- Hence it is used to pass to the server while initialization of Servlet.

**METHODS**

- public ServletContext **getServletContext**()
  - Returns a reference to the ServletContext in which the caller is executing.

- public String **getInitParameter**(String name)
  - Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

- public Enumeration **getInitParameterNames**()
  - Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters

- public String **getServletName**()
  - Returns the name of this servlet instance.

## ServletContext

- ServletContext is an interface which defines set of methods that servlet requires to communicate with server container.
- It enables servlet to get information about its environment.

**METHODS**

- public Object **getAttribute**(String name)
  - Returns the servlet container attribute with the given name, or null if there is no attribute by that name.

- public void **setAttribute**(String name,Object object)
  - Binds an object to a given attribute name in this servlet context.

- public ServletContext **getContext**(String uripath)
  - Returns a ServletContext object that corresponds to a specified URL on the server.

- public String **getMimeType**(String file)
  - Returns the MIME type of the specified file, or null if the MIME type is not known.

## ServletRequest

- ServletRequest object encapsulates the client request information to the servlet.
- ServletRequest object is passed as request parameter to the Servlets service().

**METHODS**

- public Object **getAttribute**(String name)
  - Returns the value named as attribute as Object or null if there is no attribute by that name.

- public void **setAttribute**(String name,Object object)
  - Binds an object to a given attribute name in the request.

- public String **getParameter**(String name)
  - Returns the value of a request parameter as a String, or null if the parameter does not exist.
  - Request parameters are extra information sent with the request.

- public Enumeration **getParameterNames**()

- Returns an Enumeration of String objects containing the names of the parameters contained in this request.
- If the request has no parameters, the method returns an empty Enumeration.

➢ public String **getContentType**()

- Returns the MIME type of the body of the request, or null if the type is not known.

## ServletResponse

- ServletResponse object assists the servlet to send response to the client.
- ServletResponse object is passed as an argument to the Servlets service().

### METHODS

➢ public PrintWriter **getWriter**() throws IOException
- Returns a PrintWriter object that can send character text to the client

➢ public ServletOutputStream **getOutputStream**()throws IOException
- Returns a ServletOutputStream suitable for writing binary data in the response.

➢ public void **setContentType**(String type)
- Sets the content type of the response being sent to the client, if the response has not been committed yet.

➢ public void **setBufferSize**(int size)
- Sets the preferred buffer size for the body of the response

➢ public int **getBufferSize**()
- Returns the actual buffer size used for the response. If no buffering is used, this method returns 0.

## Classes of javax.servlet package

**javax.servlet.GenericServlet**
- Implements the **Servlet** and **ServletConfig** interfaces.

**javax.servlet.ServletInputStream**
- Provides an input stream for reading requests from a client.

**javax.servlet.ServletOutputStream**
- Provides an output stream for writing responses to a client.

**javax.servlet.ServletException**

- Indicates a servlet error occurred.

**javax.servlet.UnavailableException**

- Indicates a servlet is unavailable

## GenericServlet

- GenericServlet is a generic, protocol independent servlet.
- GenericServlet implements Servlet, ServletConfig and Serializable interface.
- GenericServlet provides all the life cycle methods, only service() should be overrided to write a simple servlet.

**CONSTRUCTOR**

- public GenericServlet()

- The constructor usually does nothing.
- All the initialization work is done by init().

**METHOD**

- public void **init**(ServletConfig config)throws ServletException
  - init() is invoked by the servlet container to indicate to a servlet that the servlet is being placed into service.

- public abstract void **service**(ServletRequest request,ServletResponse response) throws ServletException,IOException
  - Called by the servlet container to allow the servlet to respond to a request.

- public void **destroy**()
  - Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.

- public void **log**(String msg)
  - Writes the specified message to a servlet log file, prepended by the servlet's name

- public ServletConfig **getServletConfig**()
  - Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.

- public ServletContext **getServletContext**()
  - Returns a ServletContext object,in which servlet is currently running.

## ServletInputStream

- ServletInputStream is an abstract class.
- ServletInputStream is direct sub class of java.io.InputStream.
- ServletInputStream provides an input stream for reading binary data from a client request.

**CONSTRUCTOR**

- protected **ServletInputStream**()
  - Does nothing, because this is an abstract class.

**METHOD**

- public int **readLine**(byte[] b, int off, int len)
  throws IOException
  - Reads the input stream, one line at a time

## ServletOutputStream

- ServletOutputStream is an abstract class.
- ServletOutputStream is direct sub class of java.io.OutputStream.
- ServletOutputStream provides an output stream for sending binary data to a client.

**CONSTRUCTOR**

- protected **ServletOutputStream**()
  - Does nothing, because this is an abstract class.

**METHODS**

- public void **print**(String s)throws IOException
  - Writes a String to the client, with no carriage return-line feed (CRLF) at the end.

- public void **println**(String s)throws IOException
  - Writes a String to the client, with carriage return-line feed (CRLF) at the end.

## • THE javax.servlet.http package

- The **javax.servlet.http** package contains a number of interfaces and classes that are commonly used by servlet developers.
- It is commonly used to build servlet which is used for HTTP request and response.

**INTERFACES OF javax.servlet.http package**

**HttpServletRequest**
- Enables servlets to read data from an HTTP request.

**HttpServletResponse**
- Enables servlets to write data to an HTTP response.

**HttpSession**
- Allows session data to be read and written.

**HttpSessionBindingListener**
- Informs an object that it is bound to or unbound from a session.

## HttpServletRequest

- HttpServletRequest is an interface used to provide request information for the HTTP servlet.
- HttpServletRequest is sub-interface of **javax.servlet.ServletRequest**
- The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).

**METHODS**

➢ public String **getRequestedSessionId**()
- Returns the session ID specified by the client.

➢ public Cookie[] **getCookies**()
- Returns an array containing all of the Cookie objects the client sent with this request.
- This method returns null if no cookies were sent.

➢ public String **getHeader**(String name)
- Returns the value of the specified request header as a String.

➢ public HttpSession **getSession**()
- Returns the current session associated with this request, or if the request does not have a session, creates one.

➢ public HttpSession **getSession**(boolean create)
- Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

## HttpServletResponse

- HttpServletResponse is an interface used to provide HTTP specific response to the client.
- HttpServletResponse is sub-interface of **javax.servlet.ServletResponse**
- The servlet container creates an HttpServletResponse object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).

**METHODS**

- public void **addCookie**(Cookie cookie)
- Adds the specified cookie to the response.
- This method can be called multiple times to set more than one cookie

- public void **setHeader**(String name,String value)
- Sets a response header with the given name and value.

## HttpSession

- HttpSession provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
- The servlet container uses this interface to create a session between an HTTP client and an HTTP server.
- The session persists for a specified time period, across more than one connection or page request from the user.
- A session usually corresponds to one user, who may visit a site many times.
- The server can maintain a session in many ways such as using cookies or rewriting URLs.

### METHODS

- public long **getCreationTime**()
- Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT

- public long **getLastAccessedTime**()
- Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT

- public ServletContext **getServletContext**()
- Returns the ServletContext to which this session belongs.

- public Object **getAttribute**(String name)
- Returns the object bound with the specified name in this session, or null if no object is bound under the name.

- public void **setAttribute**(String name,Object value)
- Binds an object to this session, using the name specified.
- If an object of the same name is already bound to the session, the object is replaced.

- public void **removeAttribute**(String name)
- Removes the object bound with the specified name from this session.
- If the session does not have an object bound with the specified name, this method does nothing.

- public void **invalidate**()
- Invalidates this session then unbinds any objects bound to it.

## Classes in javax.servlet.http package

**Cookie**
- Allows state information to be stored on a client machine.

**HttpServlet**
- Provides methods to handle HTTP requests and responses.

**HttpSessionEvent**
- Encapsulates a session-changed event.

**HttpSessionBindingEvent**
- Indicates when a listener is bound to or unbound from a session value, or that a session attribute changed.

## Cookie

- The **Cookie** class encapsulates a cookie.
- A cookie is stored on a client and contains state information.
- Cookies are valuable for tracking user activities. For example, assume that a user visits an online store.
- A cookie can save the user's name, address, and other information. The user does not need to enter this data each time he or she visits the store.

- A servlet can write a cookie to a user's machine via the **addCookie( )** method of the **HttpServletResponse** interface.
- The data for that cookie is then included in the header of the HTTP response that is sent to the browser.
- The names and values of cookies are stored on the user's machine.
- Some of the information that is saved for each cookie includes the following:

- **The name of the cookie**
- **The value of the cookie**
- **The expiration date of the cookie**
- **The domain and path of the cookie**

- The expiration date determines when this cookie is deleted from the user's machine.
- If an expiration date is not explicitly assigned to a cookie, it is deleted when the current browser session ends.
- Otherwise, the cookie is saved in a file on the user's machine.
- The domain and path of the cookie determine when it is included in the header of an HTTP request.
- If the user enters a URL whose domain and path match these values the cookie is then supplied to the Web server.

**CONSTRUCTOR**

- public **Cookie**(String name,String value)
- Constructs a cookie with a specified name and value.

**METHODS**

- public String **getName**()
- Returns the name of the cookie. The name cannot be changed after creation.

- public void **setMaxAge**(int expiry)
- Sets the maximum age of the cookie in seconds.

- public int **getMaxAge**()
- Returns the maximum age of the cookie, specified in seconds.
- By default, -1 indicating the cookie will persist until browser shutdown.

- public String **getValue**()
- Returns the value of the cookie.

- public void **setValue**(String newValue)
- Assigns a new value to a cookie after the cookie is created.

## HttpServlet

- HttpServlet is an abstract class in javax.servlet.http package.
- Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site.
- A subclass of HttpServlet must override at least one method, usually one of these:

- doGet, if the servlet supports HTTP GET requests
- doPost, for HTTP POST requests
- doPut, for HTTP PUT requests
- doDelete, for HTTP DELETE requests
- init and destroy, to manage resources that are held for the life of the servlet
- getServletInfo, which the servlet uses to provide information about itself

### CONSTRUCTOR

- public **HttpServlet**()
  Does nothing, because this is an abstract class.

### METHODS

- public void **doGet**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
- Called by the server (via the service method) to allow a servlet to handle a GET request.

- public void **doPost**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
- Called by the server (via the service method) to allow a servlet to handle a POST request.

- public void **doDelete**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
  - Called by the server (via the service method) to allow a servlet to handle a DELETE request.

- public void **doPut**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
  - Called by the server (via the service method) to allow a servlet to handle a PUT request.

- public void **doOptions**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
  - Called by the server (via the service method) to allow a servlet to handle an OPTIONS request.

- public void **doHead**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
  - Called by the server (via the service method) to allow a servlet to handle a HEAD request.

- public void **doTrace**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
  - Called by the server (via the service method) to allow a servlet to handle a TRACE request.

- public void **service**(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException
  - Receive standard HTTP request sent by public service method and then dispatch to any doXXX() defined in the class.

## Handling HTTP Requests and Responses

- The **HttpServlet** class provides specialized methods that handle the various types of HTTP requests.
- A servlet developer typically overrides one of these methods.
- These methods are
  - **doDelete( )**
  - **doGet( )**
  - **doHead( )**
  - **doOptions( )**
  - **doPost( )**
  - **doPut( )**
  - **doTrace( )**

### Handling HTTP GET Requests

- Here we will develop a servlet that handles an HTTP GET request.
- The servlet is invoked when a form on a Web page is submitted.
- The example contains two files. A Web page is defined in **ColorGet.html** and a servlet is defined in **ColorGetServlet.java**.
- The HTML source code for **ColorGet.htm** is shown in the following listing.
- It defines a form that contains a select element and a submit button.
- Notice that the action parameter of the form tag specifies a URL.

- The URL identifies a servlet to process the HTTP GET request.

**ColorGet.html**

```
<html>
<body>
<center>
<form
action="http://localhost:8080/examples/servlet/ColorGetServlet">
<B>Color:</B>
<select name="color" size="1">
<option value="Red">Red</option>
<option value="Green">Green</option>
<option value="Blue">Blue</option>
</select>
<br><br>
<input type=submit value="Submit">
</form>
</body>
</html>
```

- The source code for **ColorGetServlet.java** is shown in the following listing.
- The **doGet( )** method is overridden to process any HTTP GET requests that are sent to this servlet.
- It uses the **getParameter( )** method of **HttpServletRequest** to obtain the selection that was made by the user.
- A response is then formulated.

**ColorGetServlet.java**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ColorGetServlet extends HttpServlet
        {
                public void init(ServletConfig config) throws ServletException
                {
                        super.init(config);
                }
      public void doGet(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException
          {
```

```
                String color = request.getParameter("color");
                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<html>");
                pw.println("<head>");
                pw.println("<body>");
                pw.println("<B>The selected color is: "+color);
                pw.println("</body>");
                pw.println("</head>");
                pw.println("</html>");
                pw.close();
    }
}
```

**Handling HTTP POST Requests**

- Here we will develop a servlet that handles an HTTP POST request.
- The servlet is invoked when a form on a Web page is submitted.
- The example contains two files.
- A Web page is defined in **ColorPost.html** and a servlet is defined in **ColorPostServlet.java**.
- The HTML source code for **ColorPost.html** is shown in the following listing.
- It is identical to **ColorGet.html** except that the method parameter for the form tag explicitly specifies that the POST method should be used, and the action parameter for the form tag specifies a different servlet.

**ColorPost.html**

```
<html>
<body>
<center>
<form name=
method="post"
action="http://localhost:8080/examples/servlet/ColorPostServlet">
<B>Color:</B>
<select name="color" size="1">
<option value="Red">Red</option>
<option value="Green">Green</option>
<option value="Blue">Blue</option>
</select>
<br><br>
<input type=submit value="Submit">
```

```
</form>
</body>
</html>
```

- The source code for **ColorPostServlet.java** is shown in the following listing.
- The **doPost( )** method is overridden to process any HTTP POST requests that are sent to this servlet.
- It uses the **getParameter( )** method of **HttpServletRequest** to obtain the selection that was made by the user.
- A response is then formulated.

**ColorPostServlet.java**

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

public class ColorPostServlet extends HttpServlet
        {
                public void init(ServletConfig config) throws ServletException
                {
                        super.init(config);
                }
      public void doPost(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException
          {
                String color = request.getParameter("color");
                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<html>");
                pw.println("<head>");
                pw.println("<body>");
                pw.println("<B>The selected color is: "+color);
                pw.println("</body>");
                pw.println("</head>");
                pw.println("</html>");

                pw.close();
        }
}
```

## Using Cookies

- Now, let's develop a servlet that illustrates how to use cookies.
- The servlet is invoked when a form on a Web page is submitted.
- The example contains three files as summarized here:

**File Description**

AddCookie.html
- Allows a user to specify a value for the cookie named **MyCookie**.

AddCookieServlet.java
- Processes the submission of **AddCookie.html**.

GetCookiesServlet.java
- Displays cookie values.

- The HTML source code for **AddCookie.html** is shown in the following listing.
- This page contains a text field in which a value can be entered.
- There is also a submit button on the page.
- When this button is pressed, the value in the text field is sent to **AddCookieServlet** via an HTTP POST request.

**AddCookie.html**

```
<html>
<body>
<center>
<form
method="post"
action="http://localhost:8080/examples/servlet/AddCookieServlet">
<B>Enter a value for MyCookie:</B>
<input type=textbox name="data" size=25 value="">
<input type=submit value="Submit">
</form>
</body>
</html>
```

- The source code for **AddCookieServlet.java** is shown in the following listing.
- It gets the value of the parameter named "data".
- It then creates a **Cookie** object that has the name "MyCookie" and contains the value of the "data" parameter.
- The cookie is then added to the header of the HTTP response via the **addCookie( )** method.
- A feedback message is then written to the browser.

**AddCookieServlet.java**

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

public class AddCookieServlet extends HttpServlet
```

```
        {
                        public void init(ServletConfig config) throws ServletException
                {
                        super.init(config);
                }
public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
        {
// Get parameter from HTTP request.
String data = request.getParameter("data");
// Create cookie.
Cookie cookie = new Cookie("MyCookie", data);
// Add cookie to HTTP response.
response.addCookie(cookie);
// Write output to browser.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<html>");
pw.println("<head>");
pw.println("<body>");
pw.println("<B>MyCookie has been set to");
pw.println(data);
pw.println("</body>");
pw.println("</head>");
pw.println("</html>");
pw.close();
  }
}
```

- The source code for **GetCookiesServlet.java** is shown in the following listing.
- It invokes the **getCookies( )** method to read any cookies that are included in the HTTP GET request.
- The names and values of these cookies are then written to the HTTP response.
- Observe that the **getName( )** and **getValue( )** methods are called to obtain this information.

**GetCookiesServlet.java**

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

public class GetCookiesServlet extends HttpServlet
        {
```

```
                public void init(ServletConfig config) throws ServletException
        {
                super.init(config);
        }

public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
        {
                // Get cookies from header of HTTP request.
                 Cookie[] cookies = request.getCookies();
                // Display these cookies.
                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<html>");
                pw.println("<head>");
                pw.println("<body>");
                pw.println("<B>");
                for(int i = 0; i < cookies.length; i++)
                {
                String name = cookies[i].getName();
                String value = cookies[i].getValue();
                pw.println("name = " + name +"; value = " + value);
                pw.println("</body>");
                pw.println("</head>");
                pw.println("</html>");
                pw.close();
  }
}
```

## Session Tracking

- HTTP is a stateless protocol.
- Each request is independent of the previous one.
- However, in some applications, it is necessary to save state information so that information can be collected from several interactions between a browser and a server.
- Sessions provide such a mechanism.NG JAVA
- A session can be created via the **getSession( )** method of **HttpServletRequest**.
- An **HttpSession** object is returned.
- This object can store a set of bindings that associate names with objects.
- The **setAttribute( )**, **getAttribute( )**, **getAttributeNames( )**, and **removeAttribute( )** methods of **HttpSession** manage these bindings.
- It is important to note that session state is shared among all the servlets that are associated with a particular client.

- The following servlet illustrates how to use session state.

- The **getSession( )** method gets the current session.
- A new session is created if one does not already exist.
- The **getAttribute( )** method is called to obtain the object that is bound to the name “date”.
- That object is a **Date** object that encapsulates the date and time when this page was last accessed. (Of course, there is no such binding when the page is first accessed.)
- A **Date** object encapsulating the current date and time is then created.
- The **setAttribute( )** method is called to bind the name “date” to this object.

**DateServlet.java**

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

public class DateServlet extends HttpServlet
	{
			public void init(ServletConfig config) throws ServletException
		{
			super.init(config);
		}

public void doGet(HttpServletRequest request,
HttpServletResponse response)throws ServletException, IOException
	{
		// Get the HttpSession object.
		HttpSession hs = request.getSession(true);
		// Get writer.
		response.setContentType("text/html");
		PrintWriter pw = response.getWriter();
		pw.println("<html>");
		pw.println("<head>");
		pw.println("<body>");
		pw.print("<B>");
		// Display date/time of last access.
		Date dt = (Date)hs.getAttribute("date");
		if(dt== null)
			{
		  pw.print("Hello User !! <BR> THIS IS YOUR FIRST VISIT");
			}
		else
		  {
			pw.println("Last Accessed Date :"+dt)
```

```
            }

            // Display current date/time.
            dt = new Date();
            hs.setAttribute("date", date);

            pw.println("Current date: " + dt);
            pw.println("</body>");
            pw.println("</head>");
            pw.println("</html>");
            pw.close();
  }
}
```

- When you first request this servlet, the browser displays one line with the current date and time information.
- On subsequent invocations, two lines are displayed.
- The first line shows the date and time when the servlet was last accessed.
- The second line shows the current date and time.

## EXAMPLES OF GENERIC SERVLET

**Q. A servlet to display Hello World when invoked.**

**MyFirstGenericServlet.java**

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

public class MyFirstGenericServlet extends GenericServlet
{
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
 }
```

```
 public void service(ServletRequest request, ServletResponse response) throws
ServletException,IOException
 {
  response.setContentType("text/html");
  PrintWriter pw=response.getWriter();
  pw.println("<HTML>");
  pw.println("<HEAD>");
  pw.println("<BODY>");
  pw.println("Welcome to the world of Servlet");
  pw.println("</BODY>");
  pw.println("</HEAD>");
  pw.println("</HTML>");
  pw.close();
  }
}
```

**Q. A Servlet where user name is passed from url as request parameter, and output is given as Hello User Name!**

**NameGenericGetServlet.java**

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

public class NameGenericGetServlet extends GenericServlet
{
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
 }

 public void service(ServletRequest request, ServletResponse response) throws
ServletException,IOException
 {
  response.setContentType("text/html");
  PrintWriter pw=response.getWriter();
   String nam=request.getParameter("name");
  pw.println("<HTML>");
```

```
  pw.println("<HEAD>");
  pw.println("<BODY>");
  pw.println("Hello "+nam);
  pw.println("</BODY>");
  pw.println("</HEAD>");
  pw.println("</HTML>");
  pw.close();
  }
}
```

**NOTE**
way to fire the url when using Get
http://localhost:8080/examples/servlet/NameGenericGetServlet?name=Tom

**Q. A Servlet where user name is passed from an HTML form as request parameter, and output is given as Hello User Name!!**

**Name.html**

```
<html>
<head>
<body>
<form method=post action="http://localhost:8080/examples/servlet/NameGenericServlet">
ENTER YOUR NAME : <input type=text name="name">
<br>
<br>
<input type=submit value="Submit">
</form>
</body>
</head>
</html>
```

**NameGenericServlet.java**

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

public class NameGenericServlet extends GenericServlet
{
 public void init(ServletConfig config) throws ServletException
 {
```

```
 super.init(config);
 }

 public void service(ServletRequest request, ServletResponse response) throws
ServletException,IOException
 {
  response.setContentType("text/html");
  PrintWriter pw=response.getWriter();
  String nam=request.getParameter("name");
  pw.println("<HTML>");
  pw.println("<HEAD>");
  pw.println("<BODY>");
  pw.println("Hello "+nam);
  pw.println("</BODY>");
  pw.println("</HEAD>");
  pw.println("</HTML>");
  pw.close();
  }
}
```

**Q. A Servlet to where a user passes a number and servlet respond back whether number is even or odd.**

**EvenOdd.html**

```
<html>
<head>
<body>
<form method=post action="http://localhost:8080/examples/servlet/EvenOddGenericServlet">
ENTER A NUMBER : <input type=number name="number">
<br>
<br>
<input type=submit value="Find Even/Odd">
</form>
</body>
</head>
</html>
```

**EvenOddGenericServlet.java**

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

public class EvenOddGenericServlet extends GenericServlet
{
 public void init(ServletConfig config) throws ServletException
 {
```

```
 super.init(config);
 }

 public void service(ServletRequest request, ServletResponse response) throws ServletException,IOException
 {
  response.setContentType("text/html");
  PrintWriter pw=response.getWriter();
  String num=request.getParameter("number");
   int number=Integer.parseInt(num);
  pw.println("<HTML>");
  pw.println("<HEAD>");
  pw.println("<BODY>");
  if(number%2==0)
{

  pw.println("Given number is an EVEN number");
}
else
{
pw.println("Given number is an ODD number");
}
  pw.println("</BODY>");
  pw.println("</HEAD>");
  pw.println("</HTML>");
  pw.close();
  }
}
```

**Q. A Servlet to where a user passes a number and servlet respond back whether number is prime or not.**

**Prime.html**

```
<html>
 <head>
 <title> Prime Number </title>
 </head>
 <body>

<form method = "GET"
 action = "http://localhost:8080/examples/servlet/PrimeGenericServlet">

Enter Number : <BR><input type = "number" name = "num">
```

```
<input type = "Submit" Value = "Check Prime">
</form>
 </body>
</html>
```

**PrimeGenericServlet.java**

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

public class PrimeGenericServlet extends GenericServlet
{
	public void init(ServletConfig config) throws ServletException
	{
		super.init(config);
	}

	public void service(ServletRequest request, ServletResponse response) throws
ServletException,IOException
	{
	int num = Integer.parseInt(request.getParameter("num"));		PrintWriter pw =
response.getWriter();		response.setContentType("text/html");
	pw.println("<HTML><HEAD><TITLE>Prime Number</TITLE></HEAD>");

	for (int i = 2 ; i <= num ; ++i)
```

```
        {
                if (num%i == 0)
                {
                        if (num == i)
                        {

                        break;
                        }
                        else
                        {
                        pw.println("Given Number is not a prime number");
                        break;
                        }
                }
        }
}
```

- **EXAMPLES OF HTTP SERVLET**

**Q. A Servlet where user passes a number from a url and servlet respond back with its multiplication table.**

**MultiplicationGetHttpServlet.java**

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

public class MultiplicationGetHttpServlet extends HttpServlet
{
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
 }

 public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
 {
  response.setContentType("text/html");
  PrintWriter pw=response.getWriter();
  String num=request.getParameter("number");
  int number=Integer.parseInt(num);
  pw.println("<HTML>");
```

```
  pw.println("<HEAD>");
  pw.println("<BODY>");
  for(int i=1;i<=10;i++)
  {
  pw.println(""+number+" * "+i+" = "+(number*i));
  }
  pw.println("</BODY>");
  pw.println("</HEAD>");
  pw.println("</HTML>");
  pw.close();
  }
}
```

**Q. A servlet where a user sends a string literal from a html form and servlet responds whether the string is palindrome or not.**

**Palindrome.html**

```
<html>
<head>
<body>
<form method=post action="http://localhost:8080/examples/servlet/PalindromeHttpServlet">
ENTER A String : <input type=name name="string">
<br>
<br>
<input type=submit value="Find Palindrome">
</form>
</body>
</head>
</html>
```

**PalindromeHttpServlet.java**

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

public class PalindromeHttpServlet extends HttpServlet
{
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
```

```
 }
 public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
 {
  response.setContentType("text/html");
  PrintWriter pw=response.getWriter();
  String orirginal=request.getParameter("string");
  pw.println("<HTML>");
  pw.println("<HEAD>");
  pw.println("<BODY>");
  StringBuffer sb=new StringBuffer(original);
  sb=sb.reverse();
  String reverse=sb.toString();
  if(original.equalsIgnoreCase(reverse))
  {
  pw.println("Given String is a Palindrome");
  }
  else
  {
  pw.println("Given String is not a Palindrome");
  }
  pw.println("</BODY>");
  pw.println("</HEAD>");
  pw.println("</HTML>");
  pw.close();
  }
}
```

- **SERVLET WITH JDBC**

**Q. A Servlet where user inserts the username and password and the values are stored in a sql table Login_Details.**

**InsertData.html**

```
<html>
<head>
<body>
<form method=post action="http://localhost:8080/examples/servlet/InsertDataHttpServlet">
UserName: <input type=name name="UN">
<br>
Password: <input type=password name="PASSWD">
<br>
<input type=submit value="Submit Values">
```

```
</form>
</body>
</head>
</html>
```

**InsertDataHttpServlet.java**

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

public class InsertDataHttpServlet extends HttpServlet
{
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
 }

 public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
 {
  response.setContentType("text/html");
  PrintWriter pw=response.getWriter();
  String uN=request.getParameter("UN");
  String passwd=request.getParameter("PASSWD");

   pw.println("<HTML>");
  pw.println("<HEAD>");
  pw.println("<BODY>");

  try
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:LoginDSN");
String query="Insert into LoginDetail values(?,?)";
PreparedStatement pstmt=con.prepareStatement(query);
pstmt.setString(1,uN);
pstmt.setString(2,passwd);
pstmt.executeUpdate();
```

```
pw.println("Value inserted Successfully");
pstmt.close();
con.close();
}
catch(Exception e)
{
System.out.println("Exception Occured "+(e.getMessage()));
}


  pw.println("</BODY>");
  pw.println("</HEAD>");
  pw.println("</HTML>");
  pw.close();
  }
}
```

## SESSION TRACKING EXAMPLE

**Q. A Servlet that displays the number of time the user has visited the page.**

**CounterServlet.java**

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

public class CounterServlet extends HttpServlet
{
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
 }

 public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
 {
  HttpSession hs=request.getSession();
  response.setContentType("text/html");
```

```
 PrintWriter pw=response.getWriter();
 int newCount=0;
 pw.println("<HTML>");
 pw.println("<HEAD>");
 pw.println("<BODY>");

 Integer count=(Integer)hs.getAttribute("counter");

 if(count==null)
 {
 pw.println("Hello User!! This is your First Visit");
 hs.setAttribute("counter",new Integer(1));
 }
 else
 {
 newCount=count.intValue()+1;
 pw.println("Visit No : "+newCount);
 hs.setAttribute("counter",new Integer(newCount));
 }

 pw.println("</BODY>");
 pw.println("</HEAD>");
 pw.println("</HTML>");
 pw.close();
 }
}
```

**Inter Servlet Communication**

There are various instances where Servlet within the same Container or different container require communicating with each other.

There are basically two ways of making Servlet inter communicate with each other.

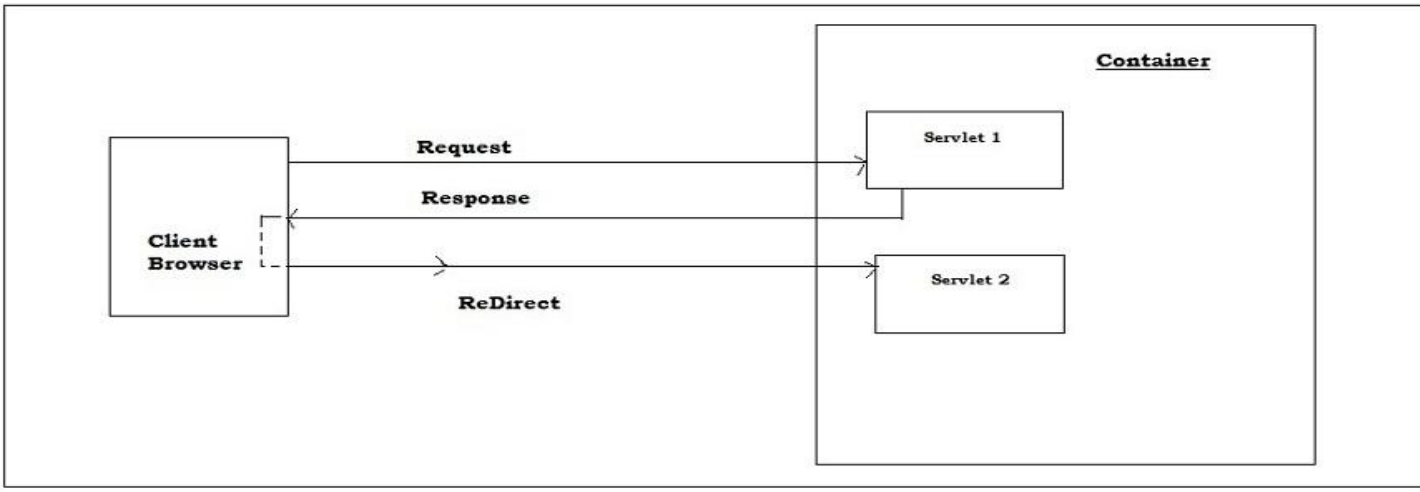
1. sendRedirect()
2. RequestDispatcher

- sendRedirect()

- The sendRedirect() method is a method of javax.servlet.http.HttpServletResponse.
- This method sends a temporary redirect response to the client using specified redirect location URL which is passed as an argument.

  response.sendRedirect("http://localhost:8080/TechWebApp/s5");
- This method also allows relative URL to be passed; later Container converts it into absolute URL.
- For eg.
  response.sendRedirect("/s5");

- Given that s5 and the servlet through which the call is made are in the same container.
- This is an old way of communicating and not so efficient way of working; because response is being sent toward Client and then redirected.
- The following diagram shows the overheads.




- Another disadvantage of this method is that all the parameter which comes with initial request has to be resend or else they will be lost.
- But this way of inter communication cannot be ruled out because it is majorly used in Payment System.

- RequestDispathcer
  - RequestDispatcher provides an elegant way of communicating amongst servlet with no overheads of client trips.
  - RequestDispatcher is an interface in javax.servlet package.

- As servlets within same container are allowed to be communicated here communication between them does not involve any client trips.
- With addition Request Parameters from initial request are simply propagated to target servlet.

  - **javax.servlet.RequestDispatcher**
- RequestDispather is an interface in above mentioned package.
- RequestDispatcher defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server.

**METHODS**

- **public void include(ServletRequest request,ServletResponse response)**
  - Includes the content of a resource (servlet, JSP page, HTML file) in the response.
  - Here the control is being passed to assigned resource till execution of the resource and then given back to calling servlet.
- **public void forward(ServletRequest request,ServletResponse response)**
  - Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
  - Once the control is being passed permanently to the given resource.

**CheckServlet.java**

```
import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

public class CheckServlet  extends HttpServlet

{

        public void init(ServletConfig config) throws ServletException

        {

                super.init(config);

        }
```

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


        PrintWriter pw;

        response.setContentType("text/html");

        pw=response.getWriter();

        String name,age;

        name=request.getParameter("name");

        age=request.getParameter("age");

        pw.println("<HTML>");

        pw.println("Hello TOm");

        RequestDispatcher rd;

rd=request.getRequestDispatcher("http://localhost:8080/TechWebApp/next");

        rd.forward(request,response);

        }

}
```

## NextServlet.java

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

public class NextServlet extends HttpServlet
{
    public void init(ServletConfig config) throws ServletException
    {
            super.init(config);
    }
```

```
    public void doGet(HttpServletRequest request,HttpServletResponse response) throws
ServletException,IOException
    {
    PrintWriter pw;
    response.setContentType("text/html");
    pw=response.getWriter();

    String name,age;
    name=request.getParameter("name");
    age=request.getParameter("age");

    pw.println("<HTML>");
    pw.println("--"+name);
    pw.println("<--"+age);

    }


}
```