Understanding Java Servlet Session

- Servlet Session Tracking
- Cookie API
- Session API

What is a session?

A session is a collection of HTTP requests, over a period of time, between a client and a Web server.
When a session is created, the requirement is to set its lifetime too.
Upon expiration, the session is destroyed and its resources returned back to the Servlet engine.
However, if the developer wants, the session is prematurely destroyed by him.

Session Tracking

- Session tracking is a mechanism that servlets use to maintain state about a series of requests from the same user (that is, requests originating from the same browser) across some period of time.
- Sessions are shared among the servlets accessed by a client.
- This is convenient for applications made up of multiple servlets.
- For example, a **Bookstore** system uses session tracking to keep track of the books being ordered by a user.

Why do we need session tracking?

- HTTP is a **stateless protocol**:
  - it doesn't provides a way for a server to recognize which client is using what part of the application.
- Great number of web-oriented application requires that application has to keep track of the clients performing actions, and thus can't be supported without an additional API.
- To support the software that needs keep track of the state, **Java Servlet technology provides an API for managing sessions and allows several mechanisms for implementing sessions**.

Three ways for session tracking

- **Hidden Form Fields**
  - fields are added to an HTML form which are not displayed in the client's request.
    e.g., <input type ="hidden" name = "name" value="***">
- **Cookies**
  - By default the server creates a cookie and the cookie get stored on the client machine.
  - The cookie is sent back to the server when the user sends a new request. By this cookie, the server is able to identify the user. In this way the session is maintained. Cookie is nothing but a name- value pair, which is stored on the client machine.
- **URL Rewriting**
  - A session object is created when the first request goes to the server.
  - Then server creates a token which will be used to maintain the session.
  - The token is transmitted to the client by the response object and gets stored on the client machine.

What are cookies?

- A cookie is a *message* given to a *Web browser* by a *Web server*.
- The browser stores the message in a text file. The message is then sent back to the server each time the browser requests a page from the server.
- **Purpose**
  - to identify users and possibly prepare customized Web pages for them.
  - Example
  - When you enter a Web site using cookies, you may be asked to fill out a form providing such information as your name and interests.
  - This information is packaged into a cookie and sent to your Web browser which stores it for later use.
  - The next time you go to the same Web site, your browser will send the cookie to the Web server.
  - The server can use this information to present you with custom Web pages.
  - So, for example, instead of seeing just a generic welcome page you might see a welcome page with your name on it.

Cookie API

Cookie

- Creating Cookies
- Cookie Attributes
- Reading Cookies
  - Session & Persistent Cookies
- Examples
  - Repeat Visitor
  - Tracking User Access Counts
  - Remembering User Preferences

Creating Cookies

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

1. Cookie Constructor

- You create a new cookie by calling the Cookie constructor and specifying:
  - Name
  - Value
  - Example:
  - Cookie cookie = new Cookie ("school", "NYU");
  - Neither the name nor the value should contain *whitespace* or any of the following characters:
  - [ ] ( ) = , " / ? @ ;

2.Set Cookie Attributes

- Before adding your cookie to the response object, you can set any of its attributes.
- Attributes include:
  - Name/Value
  - Domain
  - Maximum Age
  - Path
  - Version

Cookie Name

public String getName();
public void setName (String name);

- You rarely call setName() directly, as you specify the name in the cookie constructor.
- getName() is useful for reading in cookies.

Cookie Value

public String getValue();
public void setValue (String value);

- You rarely call setValue() directly, as you specify the name in the cookie constructor.
- getValue() is useful for reading in cookies.

Domain Attributes

public String getDomain ();
public void setDomain(String domain);

- Normally, the browser only returns cookies to the exact same host that sent them.
- You can use setDomain() to instruct the browser to send cookies to other hosts within the same domain.
- Example: Cookies sent from a servlet at bali.vacations.com would not be forwarded to mexico.vacations.com.
- If you do want to the cookie to be accessible to both hosts, set the domain to the highest level:
  - cookie.setDomain (".vacations.com");
  - Note that you are always required to include at least two dots. Hence, you must specify .vacations.com, not just vacations.com

Cookie Age

public int getMaxAge ();
public void setMaxAge (int lifetime);

- In general there are two types of cookies:
  - Session Cookies: Temporary cookies that expire when the user exits the browser.
  - Persistent Cookies: Cookies that do not expire when the user exits the browser. These cookies stay around until their expiration date, or the user explicitly deletes them.

Cookie Expiration

- The **setMaxAge()** method tells the browser how long (in seconds) until the cookie expires.
- Possible values:
  - **negative value** (default): creates a session cookie that is deleted when the user exits the browser.
  - **0**: instructs the browser to delete the cookie.
  - **positive value**: any number of seconds.
    e.g., to create a cookie that lasts for one hour, setMaxAge (3600);

Path

public String getPath();
public void setPath (String path);

- By default, the browser will only return a cookie to URLs in or below the directory that created the cookie.
- Example: If you create a cookie at
  http://ecommerce.site.com/toys.html then:
  - The browser will send the cookie back to
    http://ecommerce.site.com/toys.html
  - The browser will not send the cookie back to
    http://ecommerce.site.com/cds
  - If you want the cookie to be sent to all pages, set the path to /
  - Cookie.setPath ("/");
  - Very common, widely used practice.

Cookie Version

public int getVersion ();
public void setVersion (int version);

- By default, the Servlet API will create Version 0 cookies.
- Via the **setVersion()** method you can specify version 1. But, since this is not widely implemented, stick with the default.

Security

public int getSecure ();
public void setSecure (boolean);

- If you set Secure to true, the browser will only return the cookie when connecting over an encrypted connection.
- By default, cookies are set to non-secure.

Comments

public int getComment ();
public void Comment (String)

- Comments: you can specify a cookie comment via the setComment() method. But, comments are only supported in Version 1 cookies.
- Hence, no one really uses these methods.

3. Add Cookies to Response
   - Once you have created your cookie, and set any attributes, you add it to the response object.
   - By adding it to the response object, your cookie is transmitted back to the browser.
   - Example:
     Cookie school = new Cookie ("school", "NYU");
     school.setMaxAge (3600);
     response.addCookie (school);

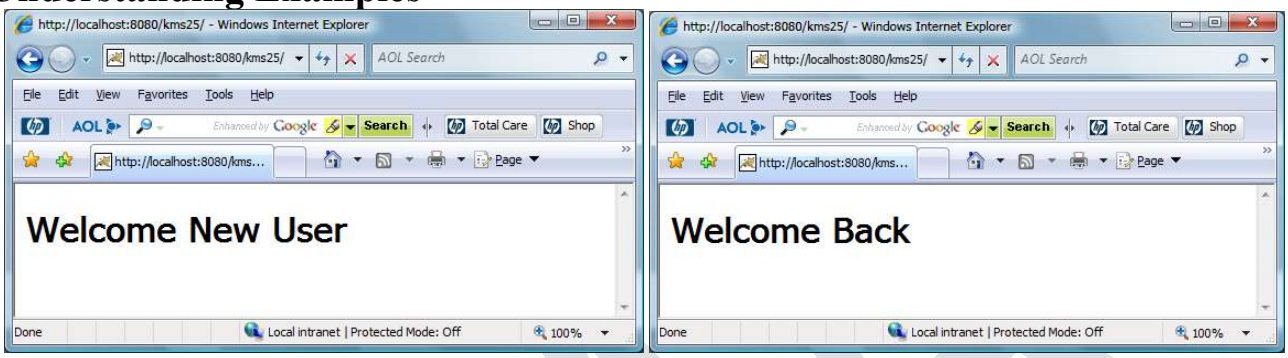
Reading Cookies

- To create cookies, **add** them ***to*** the **response object**.
- To read incoming cookies, **get** them ***from*** the **request object**.
- HttpServletRequest has a **getCookies()** method.
  - Returns an array of cookie objects. This includes all cookies sent by the browser.
  - Returns a zero-length array if there are no cookies.
- Once you have an array of cookies, you can iterate through the array and extract the one(s) you want.
- Our next few examples illustrate how this is done.

Session & Persistent Cookies

- **Session Cookies**
  - these are temporary and are erased when you close your browser at the end of your surfing session.
  - The next time you visit that particular site it will not recognise you and will treat you as a completely new visitor as there is nothing in your browser to let the site know that you have visited before.
- **Persistent Cookies**
  - these remain on your hard drive until you erase them or they expire.
  - How long a cookie remains on your browser depends on how long the visited website has programmed the cookie to last .

Understanding Examples

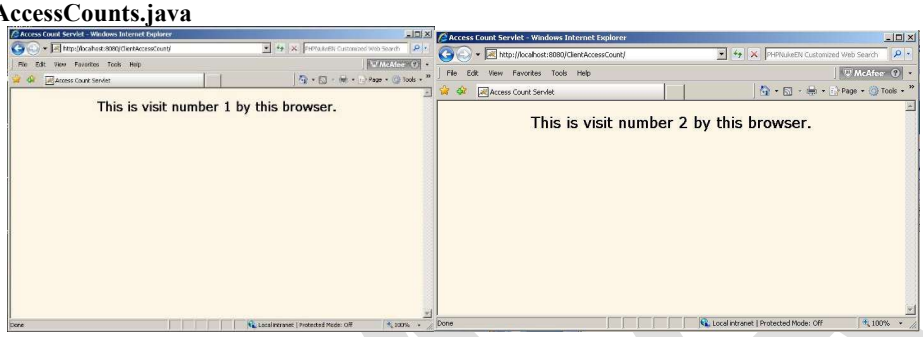


Example 1 RepeatVisitor.java

```
package myeg;

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

public class RepeatVisitors extends HttpServlet
{ public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
        { boolean newUser = true;
                Cookie[ ] cookies = request.getCookies();
                if (cookies != null)
                {
                        for(Cookie c: cookies)
                        { if ((c.getName().equals("repeatVisitor")) &&(c.getValue().equals("yes")))
                                { newUser = false; break;
                                }
                        }
                }

        String title=null;
        if (newUser)
        { Cookie returnVisitorCookie =new Cookie("repeatVisitor", "yes");
          returnVisitorCookie.setMaxAge(60*60*24*365);
          response.addCookie(returnVisitorCookie);
          title="Welcome New User";
        }
        else
        {
                title = "Welcome Back";
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<H1>" + title + "<H1>");
        }
        public void doPost(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException
        {
                doGet(request,response);
        }
}
```

```
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
      metadata-complete="true">
        <servlet>
                <servlet-name>RepeatVisitors</servlet-name>
                <servlet-class>myeg.RepeatVisitors</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>RepeatVisitors</servlet-name>
                <url-pattern>/Visitors</url-pattern>
        </servlet-mapping>
</web-app>
```

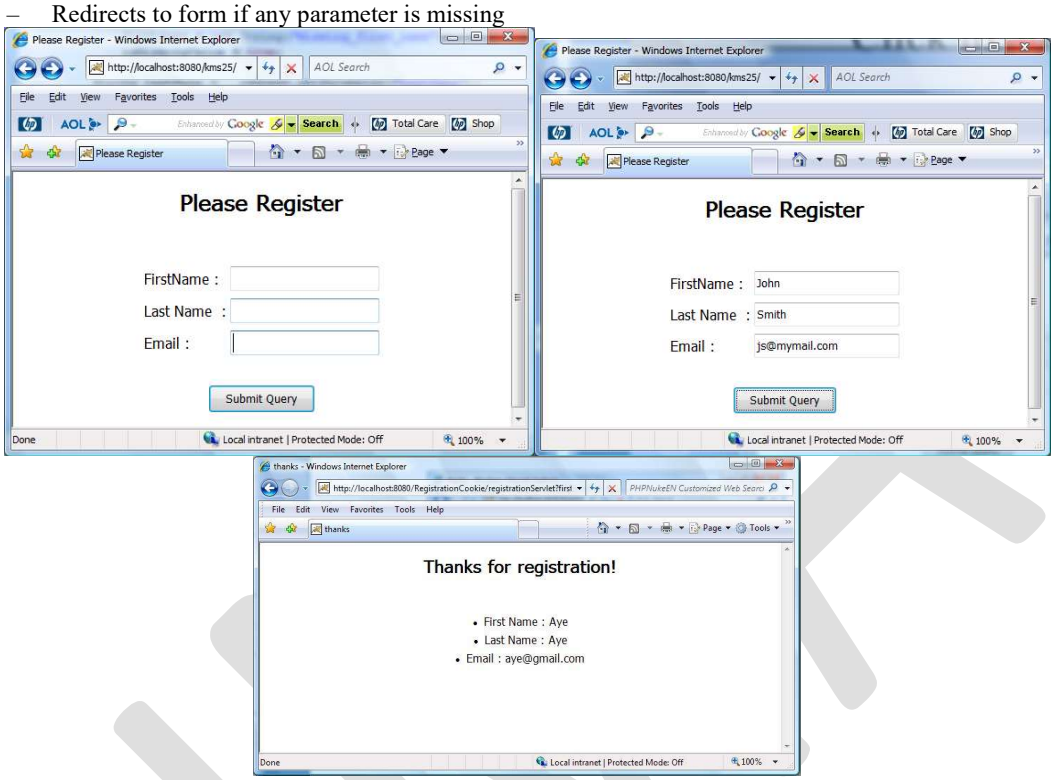
Example 2 ClientAccessCounts.java




```
package myeg;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ClientAccessCounts extends HttpServlet
{ public void doGet(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException
        { String countString="1";
          Cookie[ ] cookies = request.getCookies();
          if (cookies != null)
          { for(Cookie cookie: cookies)
            { if (cookie.getName().equals("accessCount"))
                countString=cookie.getValue();
            }
          }
          int count = 1;
          try
          { count = Integer.parseInt(countString); }
          catch(NumberFormatException nfe) { }
          Cookie c = new Cookie("accessCount",String.valueOf(count+1));
          response.addCookie(c);
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();
          out.println("<HTML>\n");
          out.println("<HEAD><TITLE> <H1>Access Count Servlet </H1> </TITLE>");
          out.println("<BODY BGCOLOR=\"#FDF5E6\">\n<CENTER>\n");
          out.println("<H2>This is visit number " + count + " by this browser.</H2>\n");
          out.println("</BODY>\n");
          out.println("</HTML>\n");
        }
}
<web-app>
   <servlet>
                <servlet-name>ClientAccessCounts</servlet-name>
                <servlet-class>myeg.ClientAccessCounts</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>ClientAccessCounts</servlet-name>
                <url-pattern>/ClientCounts</url-pattern>
        </servlet-mapping>
</web-app>
```

Example 3 RegistrationForm.java

- RegistrationForm servlet
  - Uses cookie values to prepopulate form field values
  - Uses default values if no cookies are found
  - Will also be done in JSP
- ServletRegistration servlet
  - Creates cookies based on request parameters received
  - Displays values if all parameters are present
  - Redirects to form if any parameter is missing




```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class RegistrationForm extends HttpServlet {
            public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
      {
           String firstName = "";
           String lastName = "";
           String emailAddress = "";
           Cookie[] cookies = request.getCookies();



            if (cookies != null)
            {
            for(Cookie cookie: cookies)
            {
              if (cookie.getName().equals("firstName"))
                 firstName=cookie.getValue();
              else if (cookie.getName().equals("lastName"))
                 lastName=cookie.getValue();
              else if (cookie.getName().equals("emailAddress"))
                 emailAddress=cookie.getValue();
             }
            }
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<HTML>");
           out.println("<HEAD> <CENTER> <TITLE> Please Register </TITLE> </CENTER></HEAD>");
            out.println("<BODY><CENTER>");
```

```
            out.println("<H2> Please Register </H2><BR/>");
            out.println("<FORM method='GET' ACTION=\"registrationServlet\">")
            out.println("<TABLE>" +
                    "<TR>" +
                    "<TD>FirstName : </TD> " +
                     "<TD><INPUT Type=\"text\" name=\"firstName\" value=" + firstName + "> </TD>" +
                    "</TR>" +
                     "<TR>" +
                       "<TD>Last Name  : </TD>" +
                       "<TD><INPUT Type=\"text\" name=\"lastName\" value=" + lastName + "> </TD>" +
                     "</TR>" +
                      "<TR>" +
                      "<TD>Email : </TD>" +
                      "<TD> <INPUT Type=\"text\" name=\"emailAddress\" value=" + emailAddress + "> </TD>" +
                      "</TR>" +
                      "</TABLE>" +
                       "<BR/><INPUT Type=\"submit\" name=\"submit\"> ");
                     out.println("</FORM>");
                      out.println("</CENTER></BODY>");
                     out.println("</HTML>");
                }
}
```

RegistrationServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class RegistrationServlet extends HttpServlet {
      public void doGet(HttpServletRequest request,HttpServletResponse response)throws
                                        ServletException, IOException
      {
       response.setContentType("text/html");
       boolean isMissingValue = false;
       String firstName =request.getParameter("firstName");
       if (firstName.equals("") || firstName.equals("Missing_first_name"))
         {
          firstName = new String("Missing_first_name");
          isMissingValue = true;
         }
       String lastName =request.getParameter("lastName");
       if (lastName.equals("")|| lastName.equals("Missing_last_name"))
                                                    {
                                                            lastName = new String("Missing_last_name");
                                                            isMissingValue = true;
                                                    }
          String email = request.getParameter("emailAddress");
          if (email.equals("")|| email.equals("Missing_email")) {
                                                    email = new String("Missing_email");
                                                    isMissingValue = true;
           }
          Cookie c1 =new Cookie("firstName",firstName);
          c1.setMaxAge(60);
          response.addCookie(c1);
          Cookie c2 = new Cookie("lastName", lastName);
          response.addCookie(c2);
          c2.setMaxAge(60);
          Cookie c3 = new Cookie("emailAddress",email);
          response.addCookie(c3);
          c3.setMaxAge(60);
          String formAddress ="./register";
          if (isMissingValue) {
                response.sendRedirect(formAddress);
           }
          else
           {
            response.setContentType("text/html");
```

```
            PrintWriter out = response.getWriter();
            out.println("<HTML>");
            out.println("<HEAD> <CENTER> <TITLE> thanks </TITLE> </CENTER></HEAD>");
            out.println("<BODY><CENTER>");
            out.println("<H2> Thanks for registration! </H2><BR/>");
            out.println("<LI> First Name : " + firstName);
            out.println("<LI> Last Name : " + lastName);
            out.println("<LI> Email : " + email);
            out.println("</CENTER> </BODY>");
             out.println("</HTML>");
                                                    }
                                                  }
}
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
      metadata-complete="true">
  <servlet>
                  <servlet-name>RegistrationForm</servlet-name>
                  <servlet-class>RegistrationForm</servlet-class>
        </servlet>
        <servlet-mapping>
                  <servlet-name>RegistrationForm</servlet-name>
                  <url-pattern>/register</url-pattern>
        </servlet-mapping>
        <servlet>
                  <servlet-name>RegistrationServlet</servlet-name>
                  <servlet-class>RegistrationServlet</servlet-class>
        </servlet>
        <servlet-mapping>
                  <servlet-name>RegistrationServlet</servlet-name>
                  <url-pattern>/registrationServlet</url-pattern>
        </servlet-mapping>
</web-app>
```

Session API

Session

- Using the Java Session API
  - Overview of what the Session API provides
  - Extracting Data from the Session
  - Extracting Session Information
  - Adding Data to the Session
- Example:
  - Repeat Visitor with accessCount
  - Shopping Cart System

Overview of Session API Functionality

Overview of Session API

- Servlets include a built-in Session API.
- Enables you to very easily create applications that depend on individual user data.
- For example:
  - Shopping Carts
  - Personalization Services
  - Maintaining state about the user's preferences.

Using the Session API

- Steps to using the Java Session API
  1) Get the Session object from the HttpRequest object.
  2) Extract Data from the user's Session Object
  3) Extract information about the session object, e.g. when was the session created?
  4) Add data to the user's Session Object.

Getting the Session Object

- To get the user's session object, call the getSession() method of the HttpServletRequest class.
- Example:
  HttpSession session = request.getSession();
- If user already has a session, the existing session is returned. If no session exists, a new one is created and returned.
- If you want to know if this is a new session, call the Session isNew() method.
- If you want to disable creation of new sessions, pass false to the getSession() method.
- For example:
  HttpSession session = request.getSession(false);
- If no current session exists, you will now get back a null object.

Behind the scenes

- When you call getSession() there is a lot going on behind the scenes.
  - Each user is automatically assigned a unique session ID.
  - How does this sessionID get to the user?
  - Option 1: If the browser supports cookies, the servlet will automatically create a session cookie, and store the session ID within the cookie. (In Tomcat, the cookie is called: JSESSIONID)
  - Option 2: If the browser does not support cookies, the servlet will try to extract the session ID from the URL.

Extracting Data from the Session

- The Session object works like a Hash Map that enables you to store any type of Java object.
- You can therefore store any number of keys and their associated values.
- To extract an existing object, use the **getAttribute()** method.
- The getAttribute () method will return an Object type, so you will need to perform a type cast.
- Example:

  Integer accessCount = (Integer)session.getAttribute("accessCount");

- Tip:
  - If you want to get a list of all "keys" associated with a Session, use the **getAttributeNames()** method.
  - This method returns an Enumeration of all Attribute names.

Additional Session Info.

- The Session API includes methods for determining Session specific information.
- **public String getId();**
  - returns the unique session ID associated with this user, e.g. *gj9xswvw9p*
  - **public boolean isNew();**
  - indicates if the session was just created.
  - **public long getCreationTime();**
  - indicates when the session was first created.
  - **public long getLastAccessedTime();**
  - indicates when the session was last sent from the client.
- **public int getMaxInactiveInterval()**
  - Determine the length of time (in seconds) that a session should go without access before being automatically invalidated.
- **public void setMaxInactiveInterval (int seconds)**
  - Sets the length of time (in seconds) that a session should go without access before being automatically invalidated.
  - A negative value specifies that the session should never time out.

Adding Data to the Session

- To add data to a session, use the setAttribute() or putValue() methods, and specify the key name and value.
- e.g.,
  - session.setAttribute("accessCount", accessCount);
  - To remove a value, you can use the removeAttribute (String name) method.

Terminating Sessions

- public void invalidate()
  - If the user does not return to a servlet for XX minutes*, the session is automatically invalidated and deleted.
  - If you want to manually invalidate the session, you can call invalidate().

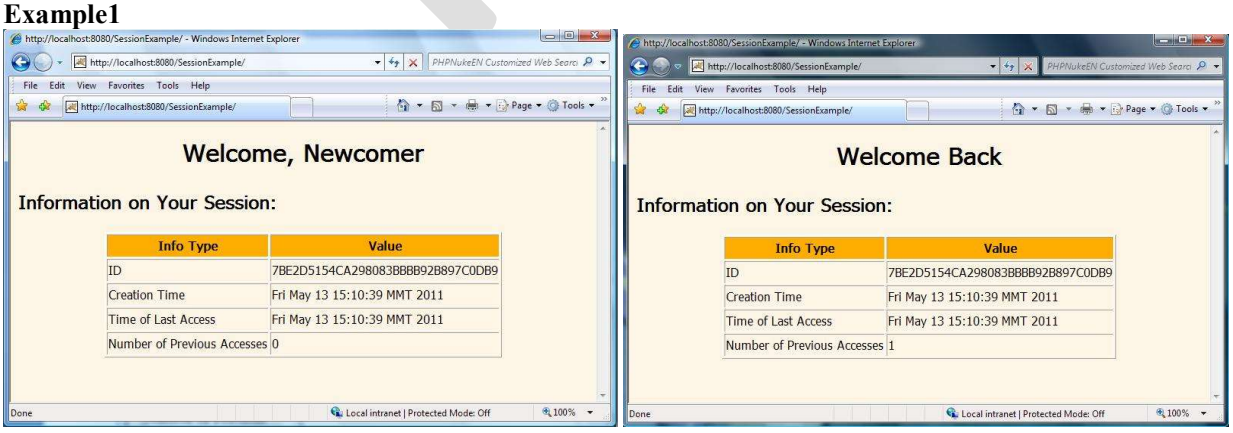
* For the exact number of minutes before automatic expiration, check the getMaxInactiveInterval() method.

Encoding URLs

- If a browser does not support cookies, you need some other way to maintain the user's session ID.
- The Servlet API takes care of this for you by automatically appending the session ID to URLs if the browser does not support cookies.
- To automatically append the session ID, use the encodeURL() method.
- Example:
  - String url = response.encodeURL (originalURL);
  - Remember that if you do this, every single URL must include the sessionID.
- Since this is hard to ensure, lots of sites (e.g. Yahoo require cookies.)

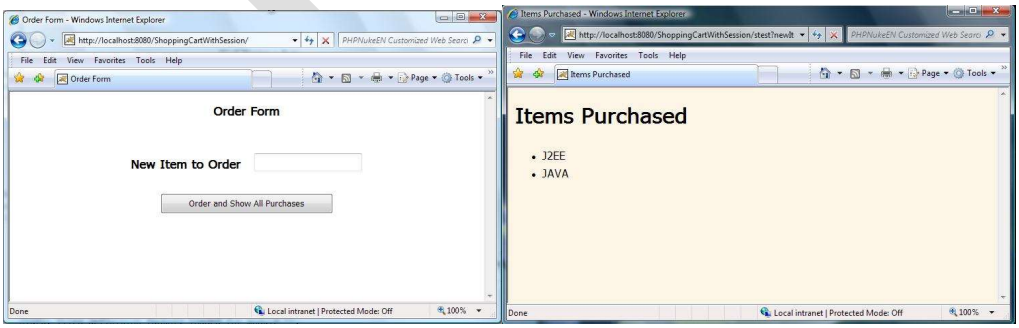
Understanding Examples

Example1



```
import java.io.*;
import javax.servlet.*;
 import javax.servlet.http.*;
 import java.util.*;
public class ShowSession extends HttpServlet
{
        public void doGet(HttpServletRequest request,HttpServletResponse response) throws
                                                                        ServletException,IOException
        {
                String title = "Session Tracking Example";
                HttpSession session = request.getSession(true);
                String heading; Integer accessCount = (Integer)session.getAttribute("accessCount");
                 if (accessCount == null) {
                accessCount = new Integer(0);
                heading = "Welcome, Newcomer"; }
                else {
                 heading = "Welcome Back"; accessCount = new Integer(accessCount.intValue() + 1);
                 }
                session.setAttribute("accessCount", accessCount);


                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println("<HTML>");
                out.println("<HEAD> <CENTER> <TITLE>" + title + "</TITLE> </CENTER> </HTML>");
                out.println("<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=\"CENTER\">" + heading + "</H1>\n" +
                "<H2>Information on Your Session:</H2>\n" +
                "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" + " <TH>Info Type<TH>Value\n" + "<TR>\n" +
                " <TD>ID\n" + " <TD>" + session.getId() + "\n" + "<TR>\n" +
                " <TD>Creation Time\n" +
                " <TD>" + new Date(session.getCreationTime()) + "\n" +
                "<TR>\n" + " <TD>Time of Last Access\n" +
                " <TD>" + new Date(session.getLastAccessedTime()) + "\n" +
                "<TR>\n" + " <TD>Number of Previous Accesses\n" +
                " <TD>" + accessCount + "\n" +
                "</TR>"+
                "</TABLE>\n" +
                "</BODY></HTML>");
        }
}
```

Example2



OrderForm.html

```
<html>
<head>
	<center>
		<title> Order Form </title>
	</center>
</head>
<body>
	<center>
		 <h3> Order Form </h3><br>
		<form method="get" action="stest">
			New Item to Order  
			<input type="text" name="newItem"><br><br>
			<input type="submit" value="Order and Show All Purchases"><br>
		 </form>
	</center>
</body>
</html>
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class ShowItems extends HttpServlet
{
	public void doGet(HttpServletRequest request, HttpServletResponse response)throws ServletException,
	IOException
	{
		HttpSession session = request.getSession();
		ArrayList<String>previousItems=(ArrayList<String>) session.getAttribute("previousItems");
		if (previousItems == null)
		{	previousItems = new ArrayList<String>(); s
			ession.setAttribute("previousItems", previousItems);
		}
		String newItem = request.getParameter("newItem");
		if ((newItem != null) && !newItem.trim().equals(""))
		 previousItems.add(newItem);

		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		String title = "Items Purchased";
		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=\"#FDF5E6\">\n" + "<H1>" + title + "</H1>");
		if (previousItems.size() == 0)
		 out.println("<I>No items</I>");
		else {
			out.println("<UL>");
			for(String item : previousItems)
			{
				out.println(" <LI>" + item);
			}
			out.println("</UL>");
		}
		 out.println("</BODY></HTML>");
	}
}
```

Exercises

Ex1.

Ex1.

```
<web-app>
 <welcome-file-list>
  <welcome-file>Survey</welcome-file>
 </welcome-file-list>
 <servlet>
  <servlet-name>LikeDislikeServlet</servlet-name>
  <servlet-class>LikeDislikeServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>LikeDislikeServlet</servlet-name>
  <url-pattern>/LikeDislikeServlet</url-pattern>
 </servlet-mapping>
</web-app>
```

Survey.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Product Survey</title>
</head>
<body>
<form action="LikeDislikeServlet">
        <table>
        <caption>Survey Form</caption>
        <tr>
                <td>Name:</td>
                <td> <INPUT TYPE="TEXT" NAME="txtName"></td>
        </tr>
        <tr>
                <td>Email:</td>
                <td> <INPUT TYPE="TEXT" NAME="txtEmail"></td>
        </tr>
        <tr>
                <td>Like or Dislike:</td>
                <td>
```

```
                <table>
                        <tr>
                                <td><INPUT TYPE="radio" NAME="r1" value="like">Like</td>
                                <td><INPUT TYPE="radio" NAME="r1" value="dislike">Dislike</td>
                        </tr>
                </table>
                </td>
        </tr>

        <tr>
                <td>Reason:</td>
                <td>
                        <table>
                                <tr>
                                        <td><INPUT TYPE="checkbox" NAME="c1" value="quality">Quality</td>
                                </tr>
                                <tr>
                                        <td><INPUT TYPE="checkbox" NAME="c1" value="price">Price</td>
                                </tr>
                                <tr>
                                        <td><INPUT TYPE="checkbox" NAME="c1" value="service">Service</td>
                                </tr>
                                <tr>
                                        <td><INPUT TYPE="checkbox" NAME="c1" value="usefulness">Usefulness</td>
                                </tr>
                                <tr>
                                        <td><INPUT TYPE="checkbox" NAME="c1" value="other">Other</td>
                                </tr>
                        </table>
                </td>
        </tr>

        <tr align="center">
                <td>
                 <input type="submit" value="submit">
                </td>
                <td>
                 <input type="reset" value="clear">
                </td>
        </tr>
        </table>
        </form>
</body>
</html>
```

LikeDislikeServlet.java

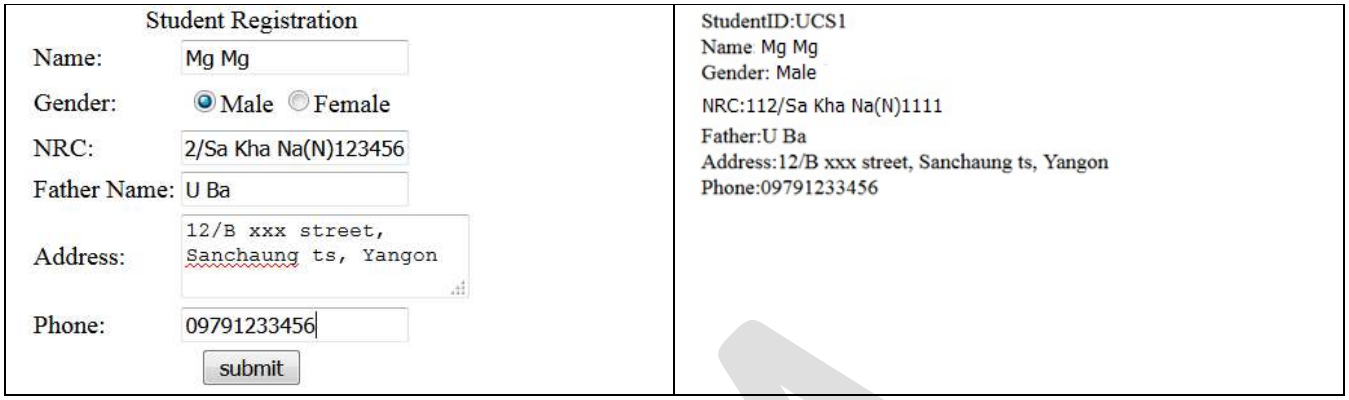
```
import java.io.*;
import java.util.ArrayList;

import javax.servlet.*;
import javax.servlet.http.*;

public class LikeDislikeServlet extends HttpServlet {

        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
                response.setContentType("text/html");
            response.setBufferSize(8192);

            HttpSession session = request.getSession(true);

            String name = request.getParameter("txtName");
            String email = request.getParameter("txtEmail");
            String like = request.getParameter("r1");
```

```
String reason[]=request.getParameterValues("c1");
        String caption[]={"Quality","Price","Service","Usefulness","Other"};

        ArrayList<Integer> count=(ArrayList<Integer>)session.getAttribute("count");
        if(count==null)
        { count=new ArrayList<Integer>();
                count.add(new Integer(0));
                count.add(new Integer(0));
        }
        ArrayList<Integer> resonsCount1=(ArrayList<Integer>)session.getAttribute("resonsCount1");
        if(resonsCount1==null)
        { resonsCount1=new ArrayList<Integer>();
         resonsCount1.add(new Integer(0));
         resonsCount1.add(new Integer(0));
         resonsCount1.add(new Integer(0));
         resonsCount1.add(new Integer(0));
         resonsCount1.add(new Integer(0));

        }
        ArrayList<Integer> resonsCount2=(ArrayList<Integer>)session.getAttribute("resonsCount2");
        if(resonsCount2==null)
        { resonsCount2=new ArrayList<Integer>();
         resonsCount2.add(new Integer(0));
         resonsCount2.add(new Integer(0));
         resonsCount2.add(new Integer(0));
         resonsCount2.add(new Integer(0));
         resonsCount2.add(new Integer(0));

        }

        switch(like)
        {case "like":count.set(0,(Integer)(count.get(0).intValue()+1));
                for(String s:reason)
                {
                 switch(s)
                 {case "quality":resonsCount1.set(0,(Integer)(resonsCount1.get(0).intValue()+1));break;
                  case "price":resonsCount1.set(1,(Integer)(resonsCount1.get(1).intValue()+1));break;
                  case "service":resonsCount1.set(2,(Integer)(resonsCount1.get(2).intValue()+1));break;
                 case "usefulness":resonsCount1.set(3,(Integer)(resonsCount1.get(3).intValue()+1));;break;
                  case "other":resonsCount1.set(4,(Integer)(resonsCount1.get(4).intValue()+1));break;
                 }
                }break;
         case "dislike":count.set(1,(Integer)(count.get(1).intValue()+1));
         for(String s:reason)
         {
                 switch(s)
                  {case "quality":resonsCount2.set(0,(Integer)(resonsCount2.get(0).intValue()+1));break;
                   case "price":resonsCount2.set(1,(Integer)(resonsCount2.get(1).intValue()+1));break;
                   case "service":resonsCount2.set(2,(Integer)(resonsCount2.get(2).intValue()+1));break;
                   case "usefulness":resonsCount2.set(3,(Integer)(resonsCount2.get(3).intValue()+1));;break;
                    case "other":resonsCount2.set(4,(Integer)(resonsCount2.get(4).intValue()+1));break;
                  }
         }
          break;
         }
        session.setAttribute("count",count);
        session.setAttribute("resonsCount1",resonsCount1);
        session.setAttribute("resonsCount2",resonsCount2);
PrintWriter out = response.getWriter();
out.println("<html>" + "<head><title>Student Registration</title></head>");

out.println("<body>");
```

```
                out.println("<h1>Servey Result</h1>");
                out.println("<h3>Like</h3>");
                out.println("<b>"+count.get(0).intValue()+" customers like the products.</b>");
                out.println("<table>");
                out.println("<tr>");
                out.println("<th>Resons</th>");
                out.println("<th>No. of liked customers</th>");
                out.println("</tr>");
                for(int i=0;i<5;i++)
                {
                        out.println("<tr>");
                          out.println("<td>"+caption[i]+"</td>");
                          out.println("<td>"+resonsCount1.get(i).intValue()+"</th>");
                          out.println("</tr>");
                }

                out.println("</table>");

                out.println("<h3>Dislike</h3>");
                out.println("<b>"+count.get(1).intValue()+" customers dislike the products.</b>");
                out.println("<table>");
                out.println("<tr>");
                out.println("<th>Resons</th>");
                out.println("<th>No. of liked customers</th>");
                out.println("</tr>");
                for(int i=0;i<5;i++)
                {
                        out.println("<tr>");
                          out.println("<td>"+caption[i]+"</td>");
                          out.println("<td>"+resonsCount2.get(i).intValue()+"</th>");
                          out.println("</tr>");
                }

                out.println("</table>");

                /*out.println("Name:"+name+"<br/>");
                out.println("Eamil:"+email+"<br/>");
                out.println("Like Or Dislike:"+like+"<br/>");
                out.println("Reason:");
                for(int i=0;i<reason.length;i++)
                                        out.println(reason[i]);*/
                out.println("</body>");
                out.close();

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

}
```

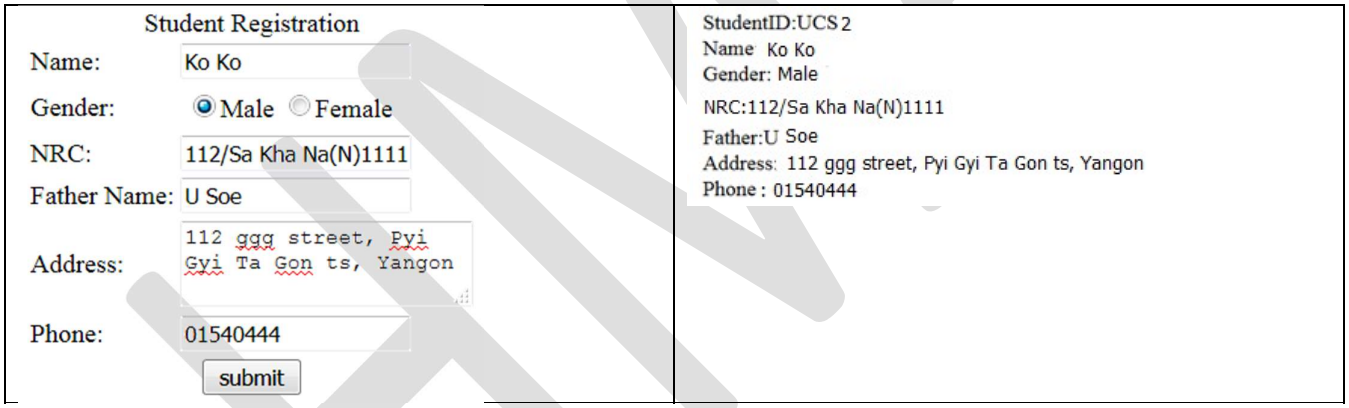
ASSIGNMENT

Ex1. Suppose you and your friends are intended to develop an online Student Registration System for your university. So, you start to gather students' details in electronic format. You can create a html page for data entry as follow:



When the first student data is submitted to the server, a servlet called StudentServlet, gets the student's details (name, gender, nrc, father's name, address, phone) and set student ID to 'UCS1'. The servlet then shows the student details as above figure.

When the second student data is submitted to the server, the servlet sets student ID to 'UCS2'. The servlet then shows the student details as follows.



Write the web component, *StudentServlet*, to keep track of student ID, to get student data and to present back the student data. You can use Cookie or Session for session tracking if you need.

Ex2.

Suppose you and your members have to create online voting system for MIDOL. There is a constraint one email can vote one candidate. After voting submitted to server, it will show the numbers of votes for each candidate.