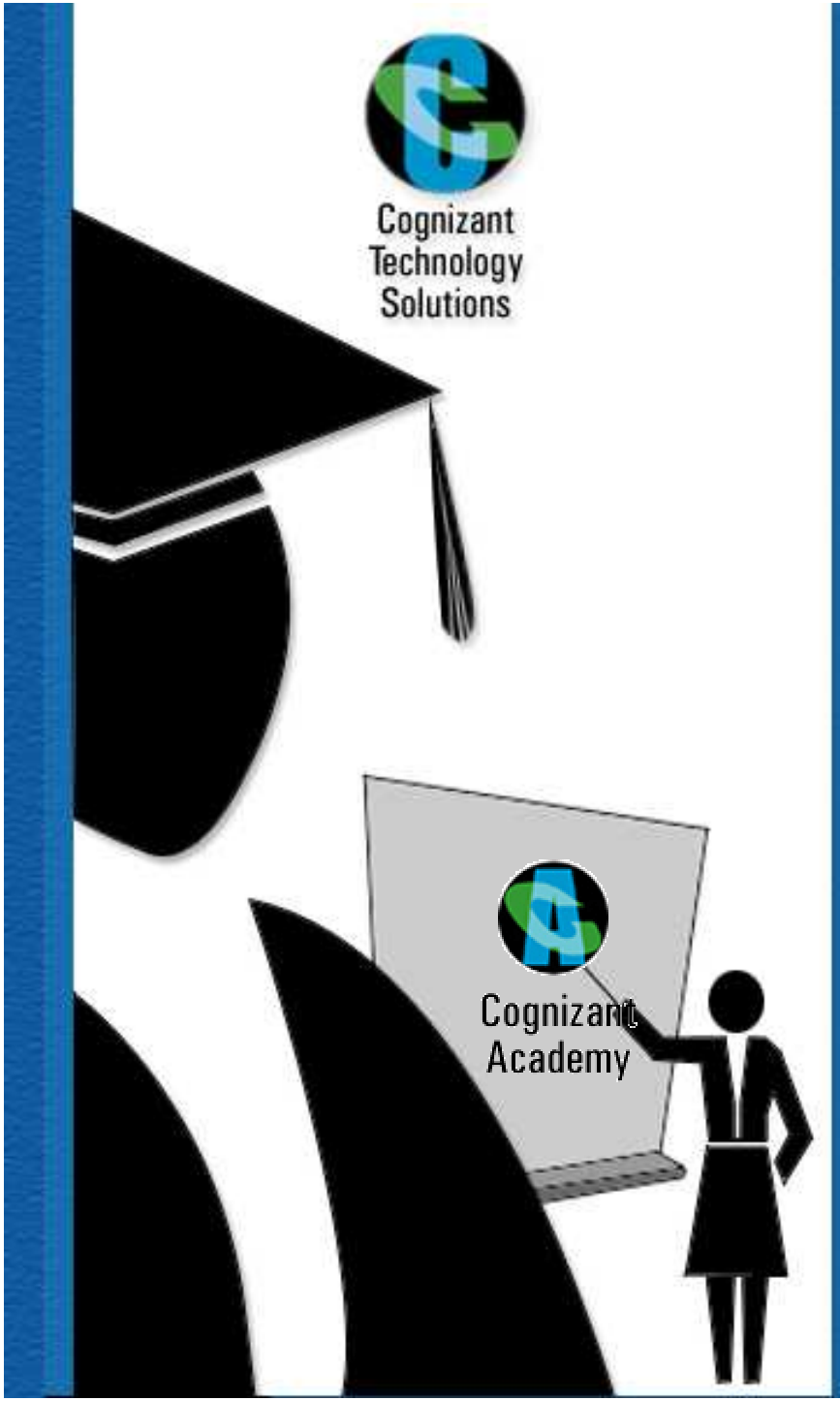


Servlets - Session Management

Basic

About the Author

Created By:	**Mohamed, Hajura (Cognizant), 117960**
Credential Information:	**Executive – Cognizant Academy**
Version and Date:	**J2EE/PPT/0306/1.0**

Cognizant Certified Official Curriculum

Icons Used


Questions


Hands-on Exercise


A Welcome Break

Test Your Understanding


Coding Standards


Reference


Demo


Key Contacts

Servlets - Session Management: Overview

- **Introduction:**

  HTTP is a stateless protocol. So, Web-based applications are responsible for maintaining state, which is called a session. To support applications that need to maintain state, the Java servlet technology provides an API for managing sessions and allows several mechanisms for implementing sessions.

Servlets - Session Management: Objectives

- **Objective:**

  After completing this chapter, you will be able to:

  - Maintain client state
  - Describe cookies
  - Track a session

Maintaining Client State

Session management is a mechanism to maintain client state across a series of requests from a same user (or originating from the same browser) over a period of time because HTTP is a stateless protocol.

Example:

An online shopping cart uses:

- `HttpSession` to maintain a client state: Used by servlets to set and get the values of session scope attributes
- `getSession()` method of a `Request` object (`HttpRequest`)

Maintaining Client State - Example

Example

HttpSession

```
public class CashierServlet extends HttpServlet
{
   public void doGet (HttpServletRequest request,
   HttpServletResponse response)throws ServletException,
   IOException
   {
    // Get the user's session and shopping cart
   HttpSession session = request.getSession(); // Accessing a
   Session
   ShoppingCart cart =
   (ShoppingCart)session.getAttribute("cart");//Associating
   objects with session
       ...
   // Determine the total price of the user's books
   double total = cart.getTotal();
   }
}
```

Session Tracking

HTTP is stateless. When it gets a page request, it cannot associate or reuse the processing of any previous requests from the same client. This makes it difficult to hold a "*conversation*".

Example:

In case of an online shopping cart, putting things one at a time into the shopping cart and then checking out each page request should be associated with previous requests.

Ideally:

- The server must be able to keep track of multiple conversations with multiple users.
- Session tracking is keeping a track of what has gone before in a particular conversation. HTTP being stateless cannot perform this function for you. You have to do it yourself, by using servlets.

Session Tracking (Contd.)

Session tracking solutions:

- Cookies: Are small files that the servlet can store on the client computer, and retrieve later.
- URL rewriting: You can append a unique ID after the URL to identify the user
- Hidden <form> fields: Can be used to store a unique ID
- Java's Session Tracking API can be used to do most of the work for you

Hidden <form> Fields

Syntax:

```
<input type="hidden" name="S1" value="somevalue">
```

- **Advantages:**
  - Requires the least knowledge of programming language. You just need to know how to read and write parameters
  - Can use `request.getParameter("S1");`
- **Disadvantages:**
  - Not maintained across sessions, so are useless for maintaining persistent information about a user
  - Because the session ID must be incorporated into every HTML page, every HTML page must be dynamically generated

Cookies

- A cookie is a small bit of text sent to the client that can be read again later.
- Limitations (for the protection of the client):
  - Not more than 4KB per cookie (more than enough in general)
  - Not more than 20 cookies per site
  - Not more than 300 cookies total
- Cookies are not a security threat
- Cookies can be a privacy threat:
  - Cookies can be used to customize advertisements
  - Outlook Express allows cookies to be embedded in email
  - A servlet can read any and all of your cookies:
    - Unethical organizations might keep your credit card info in a cookie
- Netscape lets you refuse cookies to sites other than that to which you connected

Cookies (Contd.)

To use a cookie (Assuming request is an `HttpServletRequest` and response is an `HttpServletResponse`):

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

Cookie(String name, String value)//Constructor
```

To add a cookie:

```
response.addCookie(cookie);
```

To retrieve a cookies:

```
Cookie[ ] cookies = request.getCookies();

String name = cookies[i].getName();

String value = cookies[i].getValue();
```

Demo1 - Session Tracking


```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;
import java.util.*;
/*Simple example of session tracking. See the shopping
* cart example for a more detailed one.
*/
public class ShowSession extends HttpServlet {
   public void doGet(HttpServletRequest request,
   HttpServletResponse response)
   throws ServletException, IOException {
       response.setContentType("text/html");
       PrintWriter out = response.getWriter();
       String title = "Session Tracking Example";
       HttpSession session = request.getSession(true);
       String heading;
```

Demo1- Session Tracking (Contd.)


```
    // Use getAttribute instead of getValue in version 2.2.
    Integer accessCount =
(Integer)session.getValue("accessCount");
    if (accessCount == null) {
    accessCount = new Integer(0);
    heading = "Welcome, Newcomer";
    } else {
    heading = "Welcome Back";
    accessCount = new Integer(accessCount.intValue() + 1);
    }
    // Use setAttribute instead of putValue in version 2.2.
    session.putValue("accessCount", accessCount);
```

Demo1- Session Tracking (Contd.)


```
        out.println(ServletUtilities.headWithTitle(title) +"<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" + "</TABLE>\n" +"</BODY></HTML>");
    }
    /* Handle GET and POST requests identically. */
    public void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        doGet(request, response);
    }
}
```

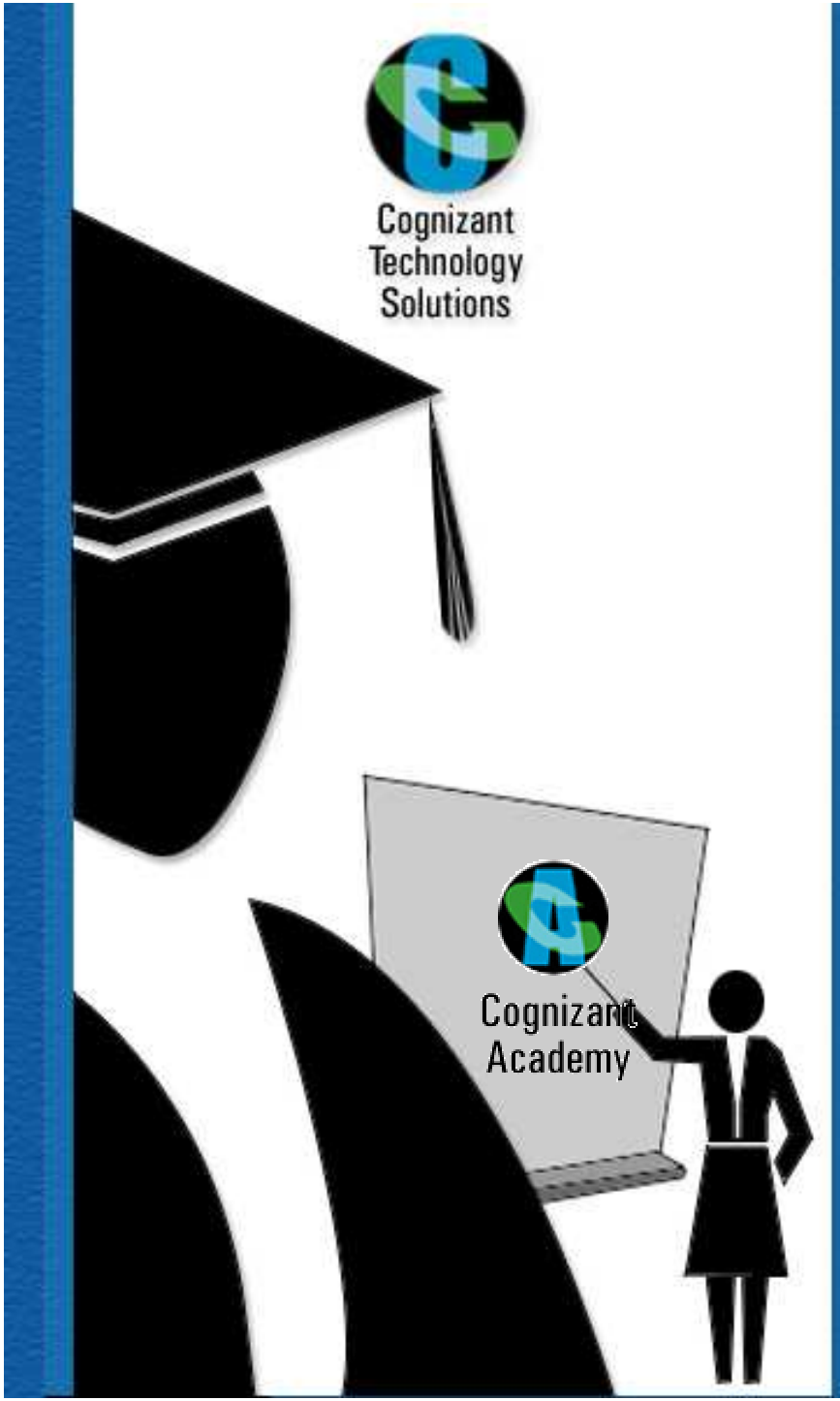
- **Allow time for questions from participants**

Servlets - Session Management: Summary

- Session management is a mechanism to maintain client state across a series of requests from a same user (or originating from the same browser) over a period of time because http is a stateless protocol.
- Session tracking can be done through cookies, URL writing, Hidden <form> fields, and Java session tracking APIs.
- A cookie is a small bit of text sent to the client that can be read again later.
- Cookies are not a security threat rather it can be a privacy threat.

Servlets - Session Management: Source

- http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html
- Professional Java Server Programming J2EE Edition By Wrox Author Team



You have successfully completed Servlets - Session Management.