CSC584 Enterprise Programming

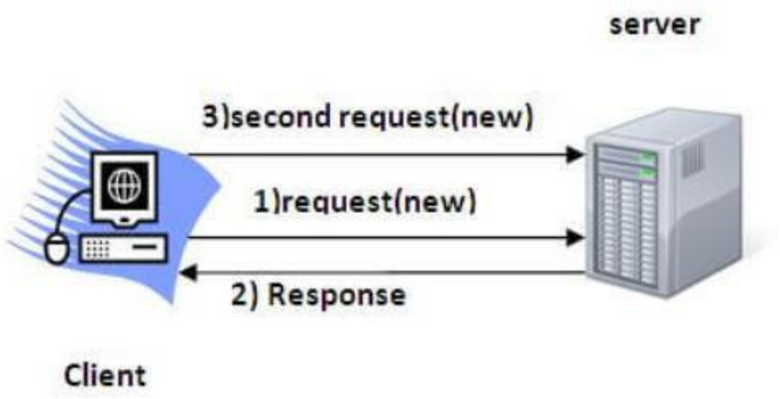
TOPIC 3 – SERVLET

Topic 3 Outline

- Creating & Running Servlets
- The Servlet API
- HTML forms
- Session tracking
- Database programming in servlets

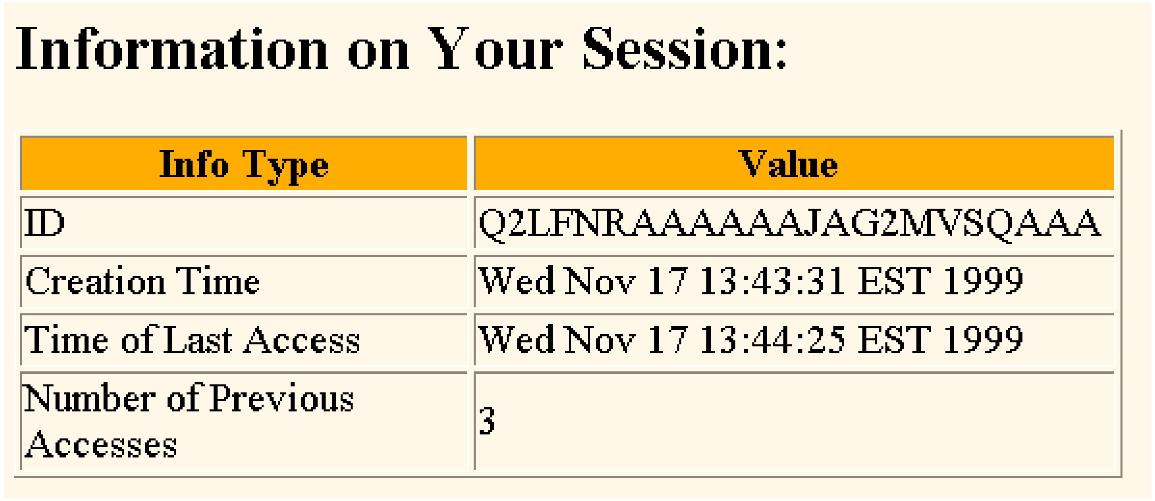
Session Tracking

- Web servers use Hyper-Text Transport Protocol (HTTP). HTTP is a stateless protocol. The HTTP Web server cannot associate requests from a client together.
- Each request is treated independently by the Web server. This protocol works fine for simple Web browsing, where each request typically results in an HTML file or a text file being sent back to the client.
- Such simple requests are isolated. However, the requests in interactive Web applications are often related.



Session Tracking

- A session can be defined as a series of related interactions between a single client and the Web server over a period of time.
- To track data among requests in a session is known as session tracking.
- Why?
  - The web server and the server automatically does not keep any record of previous client request each time a client retrieves a Web page



Information on Your Session:

Info Type	Value
ID	Q2LFNRAAAAAAJAG2MVSQAAA
Creation Time	Wed Nov 17 13:43:31 EST 1999
Time of Last Access	Wed Nov 17 13:44:25 EST 1999
Number of Previous Accesses	3

Session Tracking Techniques

1. Hidden input type
2. URL rewriting
3. Cookies
4. Session tracking with Servlet API (HttpSession object)

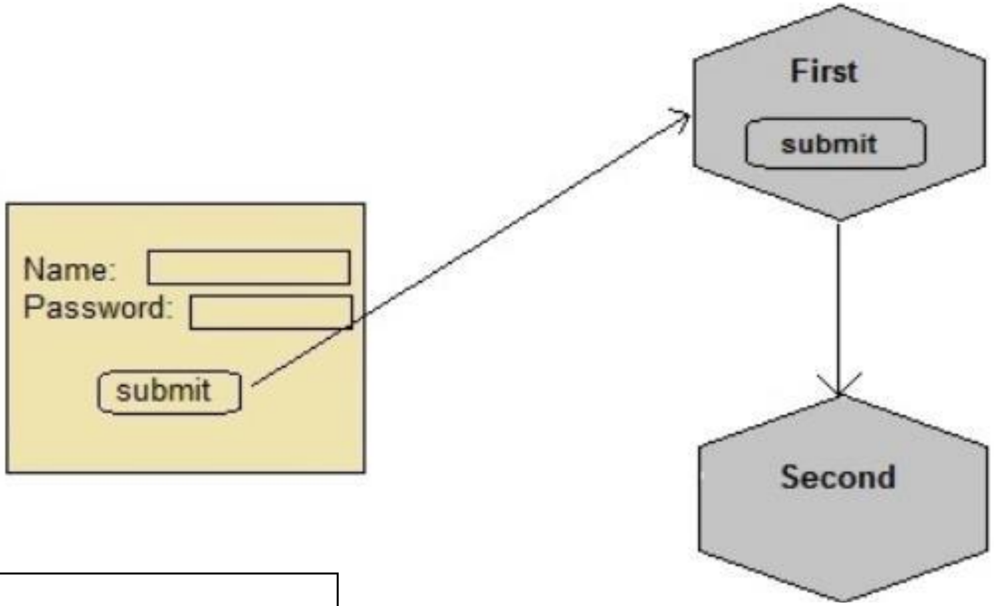
1. Session Tracking Using Hidden Values

1. You can track session by passing data from the servlet to the client as hidden value in a dynamically generated HTML form by including a field like this:

   ```
   <input type="hidden" name="lastName" value="Smith">
   ```

2. So the next request will submit the data back to the servlet.
3. The servlet retrieves this hidden value just like any other parameter value using the **getParameter()** method.

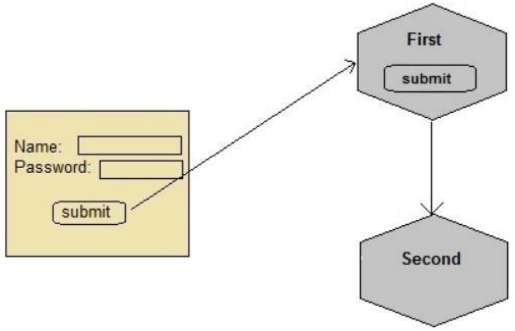
Example demonstrating usage of Hidden Form Field for Session



index.html

```
<h2>Creating Hidden Form</h2>
<form method="post" action="First">
Name:<input type="text" name="user" /><br/>
Password: <input type="text" name="pass" ><br/>
<input type="submit" value="submit">
```

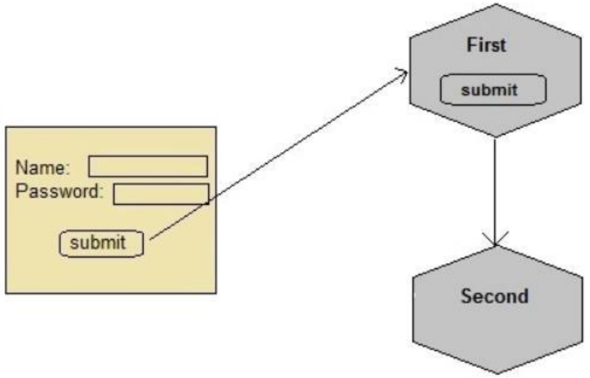
Example demonstrating usage of Hidden Form Field for Session



First.java

```
public class First extends HttpServlet {
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        //getting value submitted in form from HTML file
        String user = request.getParameter("user");

        out.println("<form action='Second'>");
        out.println("<input type='hidden' name='user' value='"+user+"'>");
        out.println("<input type='submit' value='submit' >");
        out.println("</form>");
    }
}
```

Example demonstrating usage of Hidden Form Field for Session

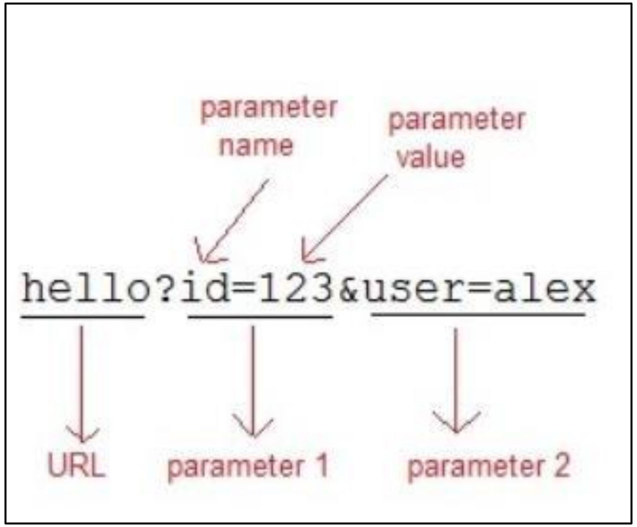


Second.java

```java
public class Second extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        //getting parameter from hidden field
        String user = request.getParameter("user");
        out.println("Welcome "+user);
    }
}
```

2. Session Tracking Using URL Rewriting

- A token(parameter) is added at the end of the URL.
- The token consist of name/value pair separated by an equal (=) sign
- When the User clicks on the URL having parameters, the request goes to the Web Container with extra bit of information at the end of URL.
- The Web Container will fetch the extra part of the requested URL and use it for session management.
- The **getParameter()** method is used to get the parameter value at the server side.



Example demonstrating usage of URL rewriting

index.html

```html
<h2>Creating Hidden Form</h2>
<form method="post" action="validate">
Name:<input type="text" name="user" /><br/>
Password: <input type="text" name="pass" ><br/>
<input type="submit" value="submit">
```

validate.java

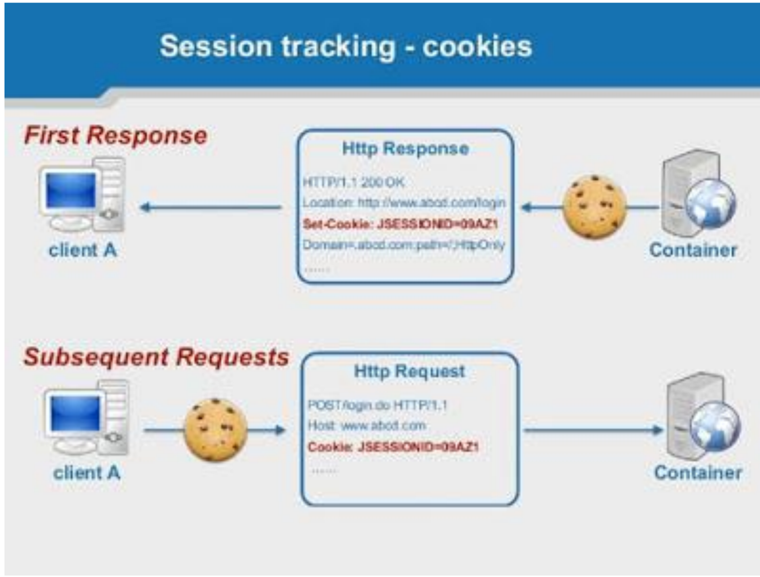
```java
public class validate extends HttpServlet {
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        //getting value submitted in form from HTML file
        String uname = request.getParameter("user");
        String pass = request.getParameter("pass");

        if (pass.equals("1234")){
            response.sendRedirect("Second?user="+uname+"");
        }
    }
}
```

Second.java

```java
public class Second extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        //getting parameter from hidden field
        String user = request.getParameter("user");
        out.println("Welcome "+user);
    }
}
```

3. Session Tracking Using Cookies

1. You can track sessions using cookies. Cookies are small text files that store sets of name=value pairs on the disk in the client's computer.
2. Cookies are sent from the server through the instructions in the header of the HTTP response.
3. The instructions tell the browser to create a cookie with a given name and its associated value. If the browser already has the cookie with the key name, the value will be updated.
4. The browser will then send the cookie with any request submitted to the same server. Cookies can have expiration dates set, after which the cookies will not be sent to the server.



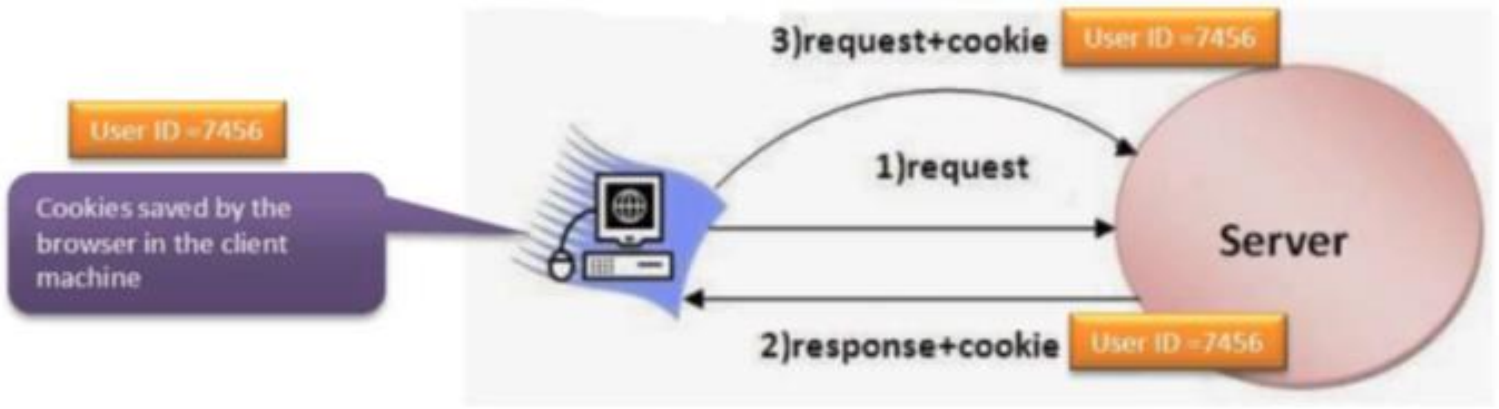
```
HTTP/1.1 200 OK
Location: http://www.abcd.com/login
Set-Cookie: JSESSIONID=09AZ1
Domain=.abcd.com;path=/;HttpOnly
......
```

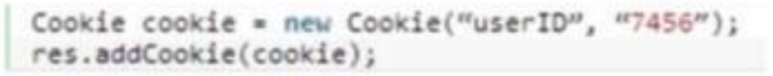
```
POST/login.do HTTP/1.1
Host: www.abcd.com
Cookie: JSESSIONID=09AZ1
......
```

Example: online shopping

Cookies in Servlet

- Cookies are the mostly used technology for session tracking. Cookie is a key value pair of information, sent by the server to the browser. This should be saved by the browser in its space in the client computer. Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie.
- Session tracking is easy to implement and maintain using the cookies. Disadvantage is that, the users can opt to disable cookies using their browser preferences. In such case, the browser will not save the cookie at client computer and session tracking fails.
- Cookies are small files which are stored on user's computer. They are designed to hold a modest amount of data specific to a particular client and website.
- Cookies are small files that websites put on your computer hard disk drive when you first visit. Think of a cookie as an identification card that's uniquely yours. Its job is to notify the site when you've returned.

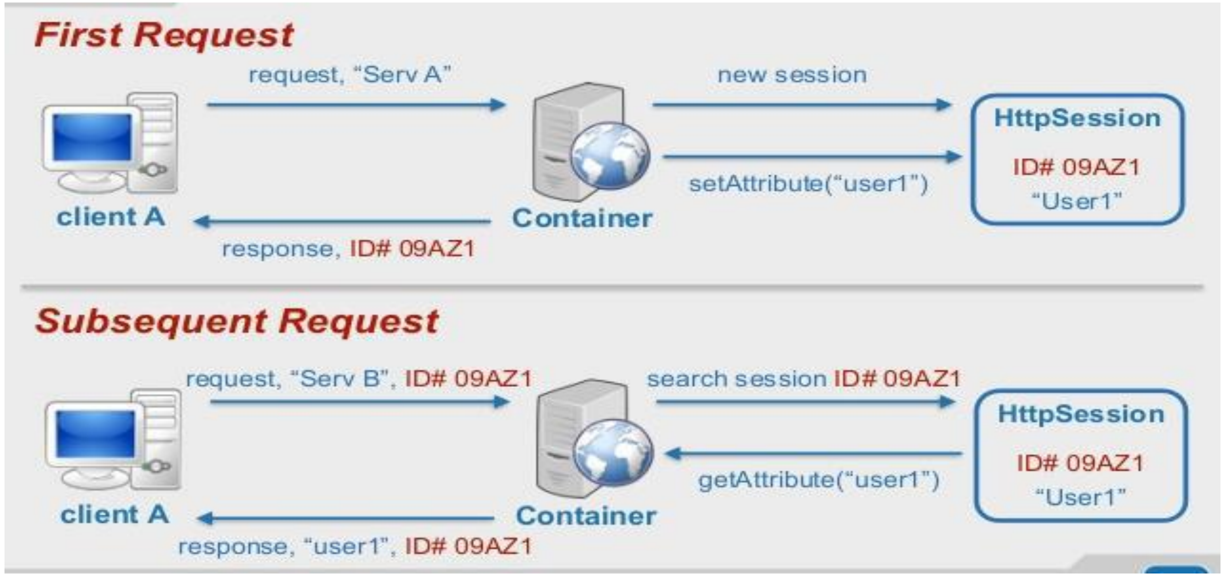




```
Cookie cookie = new Cookie("userID", "7456");
res.addCookie(cookie);
```

4. Session Tracking Using the Servlet API

1. The problems of session tracking with hidden data and cookies are that data are not secured and difficult to deal with large set of data.
2. Java servlet API provides a session tracking tool, which enables tracking of a large set of data.
3. Data can be stored as objects. Data are kept on the server side so they are secure.



The HttpSession Class

1. To use the Java servlet API for session tracking, first create a session object using the getSession method in the HttpServletRequest interface like this:

   ```
   HttpSession session = request.getSession(true);
   ```

2. This obtains the session or creates a new session if the client does not have a session on the server.

3. The HttpSession class provides the methods for reading and storing data to the session, and for manipulating the session.

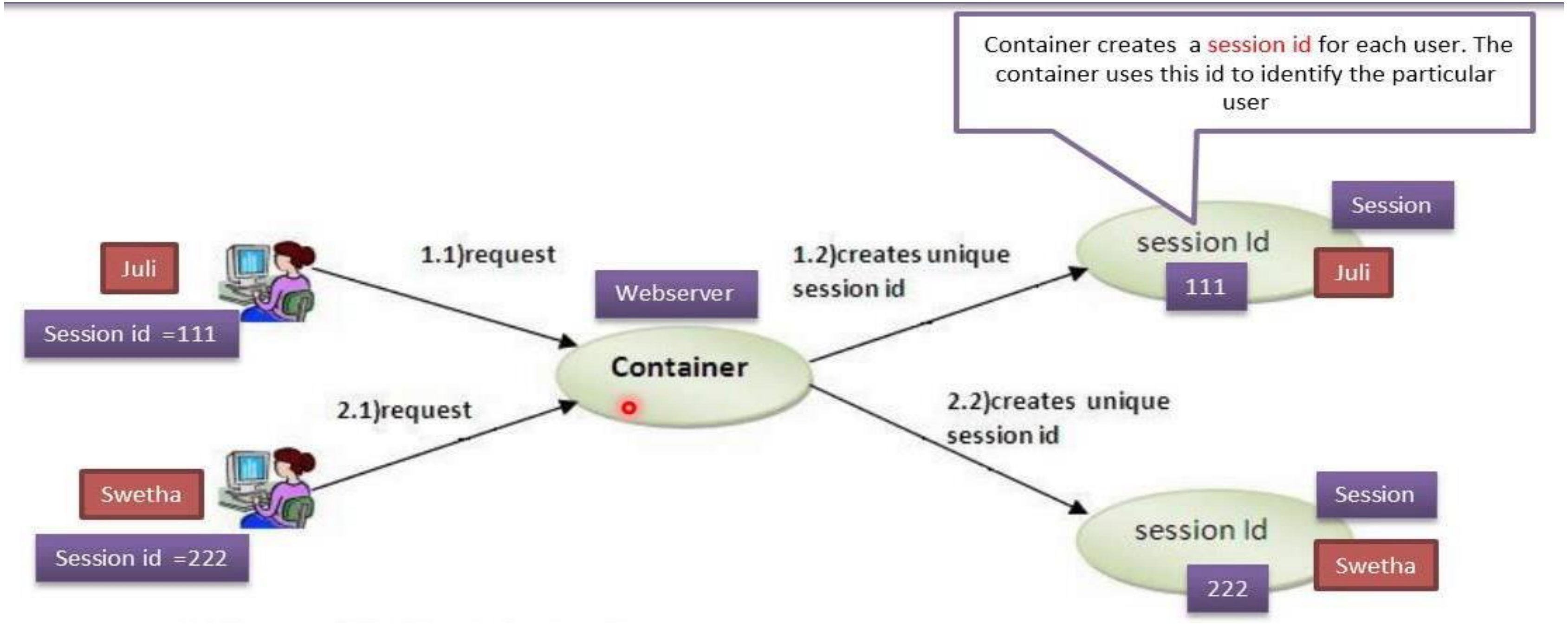
Session tracking with HttpSession object

HttpSession interface

The servlet container uses this HttpSession interface to create a session between an HTTP client and an HTTP server.

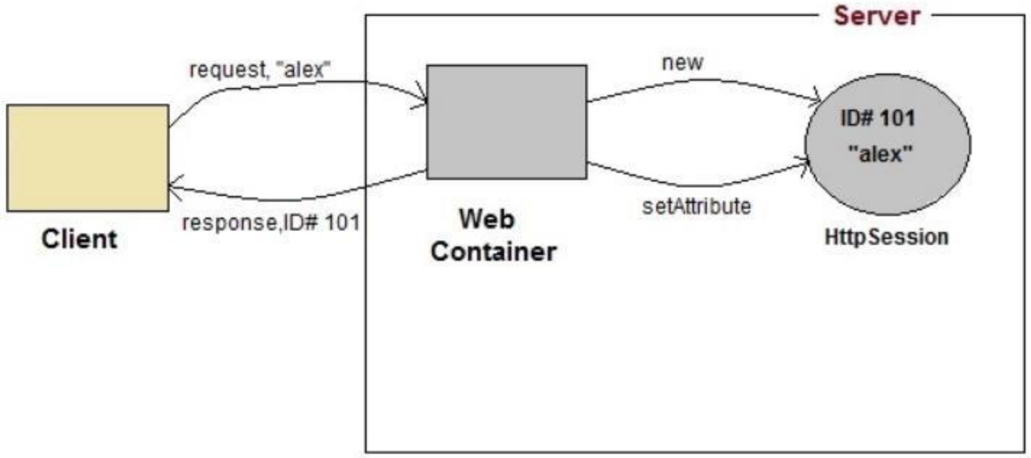
The session persists for a specified time period, across more than one connection or page request from the user. A session usually corresponds to one user, who may visit a site many times.

An object of HttpSession can be used to perform two tasks:
1. Bind objects to sessions, allowing user Information to persist across multiple user connections
2. View and manipulate information about a session, such as the session identifier, creation time, and last accessed time.

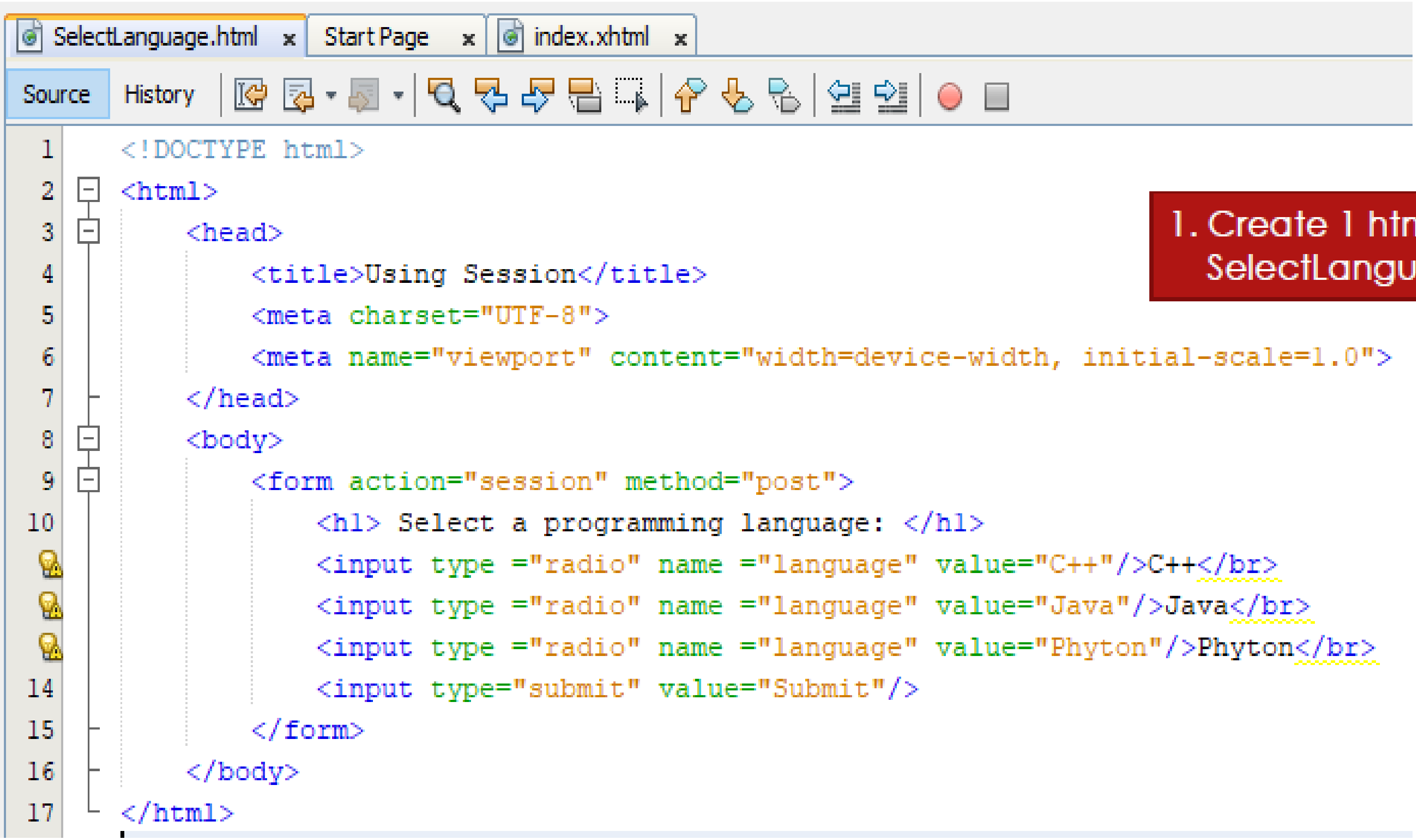

Container creates a session id for each user. The container uses this id to identify the particular user
Juli
Session id =111
1.1)request
Webserver
1.2)creates unique session id
Session
session Id
111
Juli
Container
2.1)request
2.2)creates unique session id
Swetha
Session id =222
Session
session Id
222
Swetha

How HttpSession works?

1. On client's first request, the **Web Container** generates a unique session ID and gives it back to the client with response. This is a temporary session created by web container.
2. The client sends back the session ID with each request. Making it easier for the web container to identify where the request is coming from.
3. The **Web Container** uses this ID, finds the matching session with the ID and associates the session with the request.



Example of session tracking



```
<!DOCTYPE html>
<html>
    <head>
        <title>Using Session</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="session" method="post">
            <h1> Select a programming language: </h1>
            <input type ="radio" name ="language" value="C++"/>C++</br>
            <input type ="radio" name ="language" value="Java"/>Java</br>
            <input type ="radio" name ="language" value="Phyton"/>Phyton</br>
            <input type="submit" value="Submit"/>
        </form>
    </body>
</html>
```

1. Create 1 html file name SelectLanguage.html

Example of session tracking

Code
1 of 4

2. Create a servlet file that contains the following codes:

```
import java.io.IOException;
import java.util.Map;
import java.util.*;
import java.io.*;
import javax.servlet.http.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/**
* Servlet implementation class session
*/
@WebServlet("/session")
public class session extends HttpServlet {
      private final Map books = new HashMap();
      public void init() {
        books.put("C++","001");
        books.put("Java","002");
        books.put("Phyton","003");
      }
```

Example of session tracking

Code 2 of 4

```
/**
* @see HttpServlet#HttpServlet()  */
public session() {
 super();
}
/**
* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
*/
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    String language = request.getParameter("language");
    HttpSession session = request.getSession(true);
    session.setAttribute(language, books.get(language));
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html xmlns= \"http://www.w3.org/1999/xhtml\">");
    out.println("<head>");
    out.println("<title>A simple Session Example 2</title>");
    out.println("<head>");
    out.println("<body>");
    out.println("<h2> Welcome to session !You selected "+ language+ "</h2>");
    out.println("<h2> Your unique session ID is: " + session.getId() + "<br/>");
    out.println("This "+(session.isNew()? " is " : " is not ") + " a new session <br/>");
    out.println("The session was created at: " + new Date(session.getCreationTime()) + "<br/>");
    out.println("You last accessed the session at : " + new Date(session.getLastAccessedTime()) + "<br/></h2>");
    out.println("<a href= \"session\">" + "Click here to get book recommendations</a>");
    out.println("<body>");
    out.println("</html>");
    out.close();
}
```

Example of session tracking

Code 3 of 4

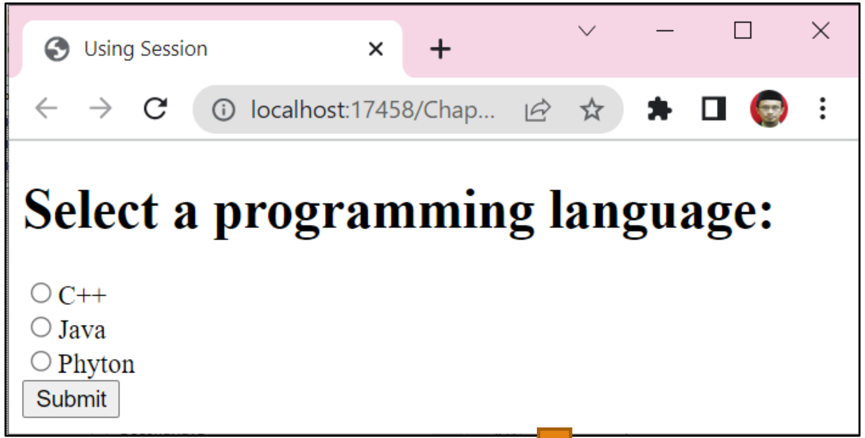
```
/**
* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
*/
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    HttpSession session = request.getSession(false);
    Enumeration valueNames;
    if (session != null)
      valueNames= session.getAttributeNames();
    else
     valueNames= null;

    PrintWriter out = response.getWriter();
    response.setContentType("text/html");
    out.println("<html xmlns= \"http://www.w3.org/1999/xhtml\">");
    out.println("<head>");
    out.println("<title>Recommendation</title>");
    out.println("<head>");
    out.println("<body>");
    if (valueNames!= null && valueNames.hasMoreElements()) {
      out.println("<h1>Recommendations</h1>");
      String name,value;
```

Example of session tracking

```
    while (valueNames.hasMoreElements()) {
       name = valueNames.nextElement().toString();
       value = session.getAttribute(name).toString();
       out.println(name + " How to Program. " + " ISBN# : " + value
       +"</br>");
    }
    out.println("<br>");
    }
    else {
      out.println("<h1>No recommendations<h1>");
      out.println("You did not select a language.");
    }
    out.println("</body>");
    out.println("</html>");
    out.close();
    } //end doGet()
} //end class session
```

Code
4 of 4

Sample Output

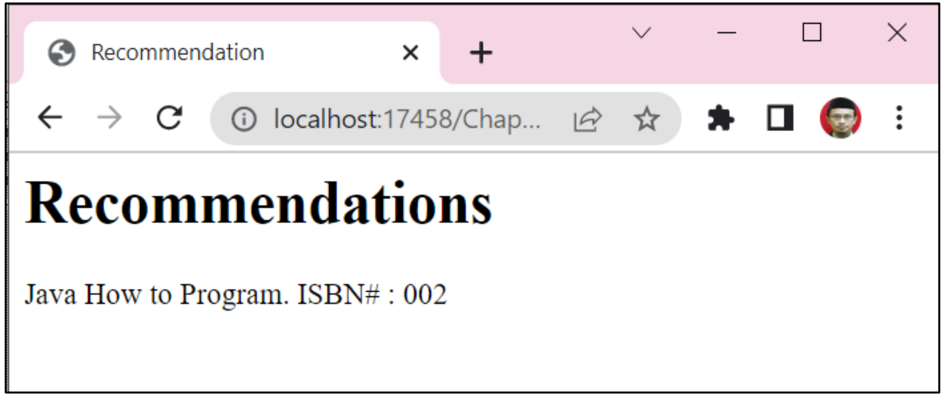

Using Session

localhost:17458/Chap...

Select a programming language:

- C++
- Java
- Phyton

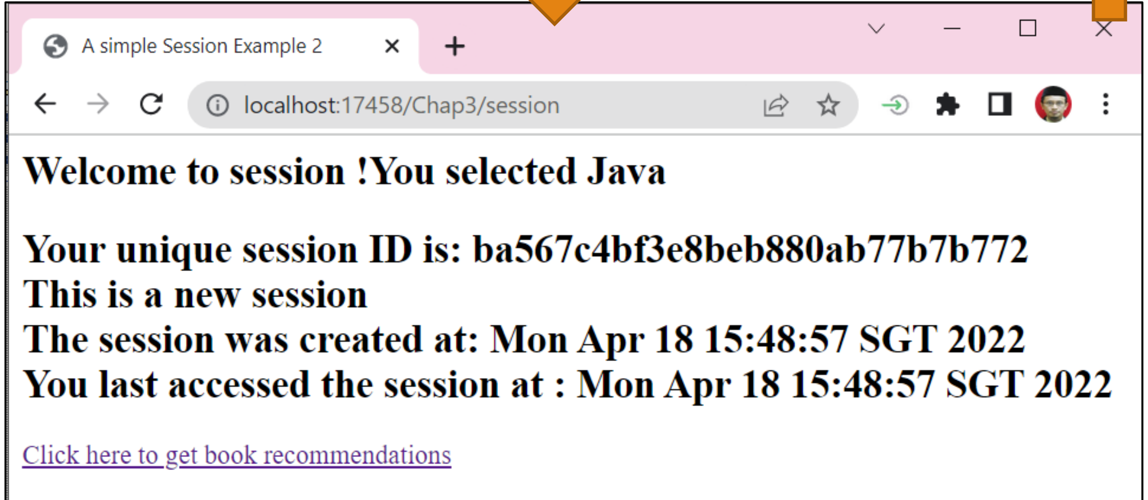
Submit


Recommendation

localhost:17458/Chap...

Recommendations

Java How to Program. ISBN# : 002


A simple Session Example 2

localhost:17458/Chap3/session

Welcome to session !You selected Java

Your unique session ID is: ba567c4bf3e8beb880ab77b7b772
This is a new session
The session was created at: Mon Apr 18 15:48:57 SGT 2022
You last accessed the session at : Mon Apr 18 15:48:57 SGT 2022

Click here to get book recommendations

Database programming in servlets

- Server can communicate with databases via JDBC (Java Database Connectivity)
- Developers do not need to be familiar with the specifics of each database system.
- Developers use SQL-based queries and the JDBC driver handles the specifics of interacting with each database system.
- You have to download JDBC library to driving the connection with the Database.
- For examples, JDBC libraries for **Oracle, MySQL, SQL Server**

Multitier Applications: Using JDBC from a Servlet

Servlets and databases

- Communicate via JDBC
  - Connect to databases in general manner
  - Use SQL-based queries

Three tier distributed applications

- User interface
  - Often in HTML, sometimes applets
  - HTML preferred, more portable
- Business logic (middle tier)
  - Accesses database
- Database access
- Three tiers may be on separate computers
  - Web servers for middle tier

Multitier Applications: Using JDBC from a Servlet

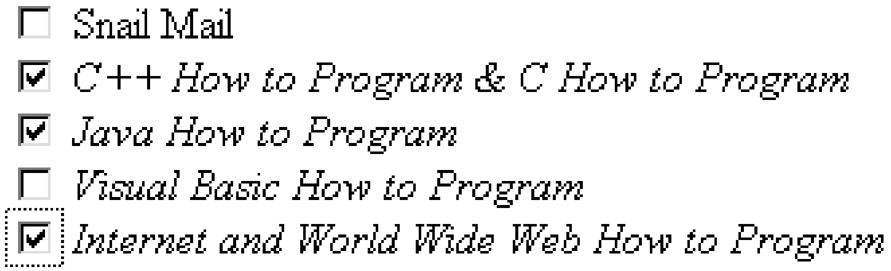
Servlets

- Method **init**
  - Called exactly once, before client requests
  - Initialization parameters
- Method **destroy**
  - Called automatically, cleanup method
  - Close files, connections to databases, etc.

Multitier Applications: Using JDBC from a Servlet

HTML files

- **<INPUT TYPE=CHECKBOX NAME=name VALUE=value>**
    - Creates checkbox, any number can be selected



☐ Snail Mail
☑ *C++ How to Program & C How to Program*
☑ *Java How to Program*
☐ *Visual Basic How to Program*
☑ *Internet and World Wide Web How to Program*

- **<INPUT TYPE=TEXT NAME=name>**
    - Creates text field, user can input data

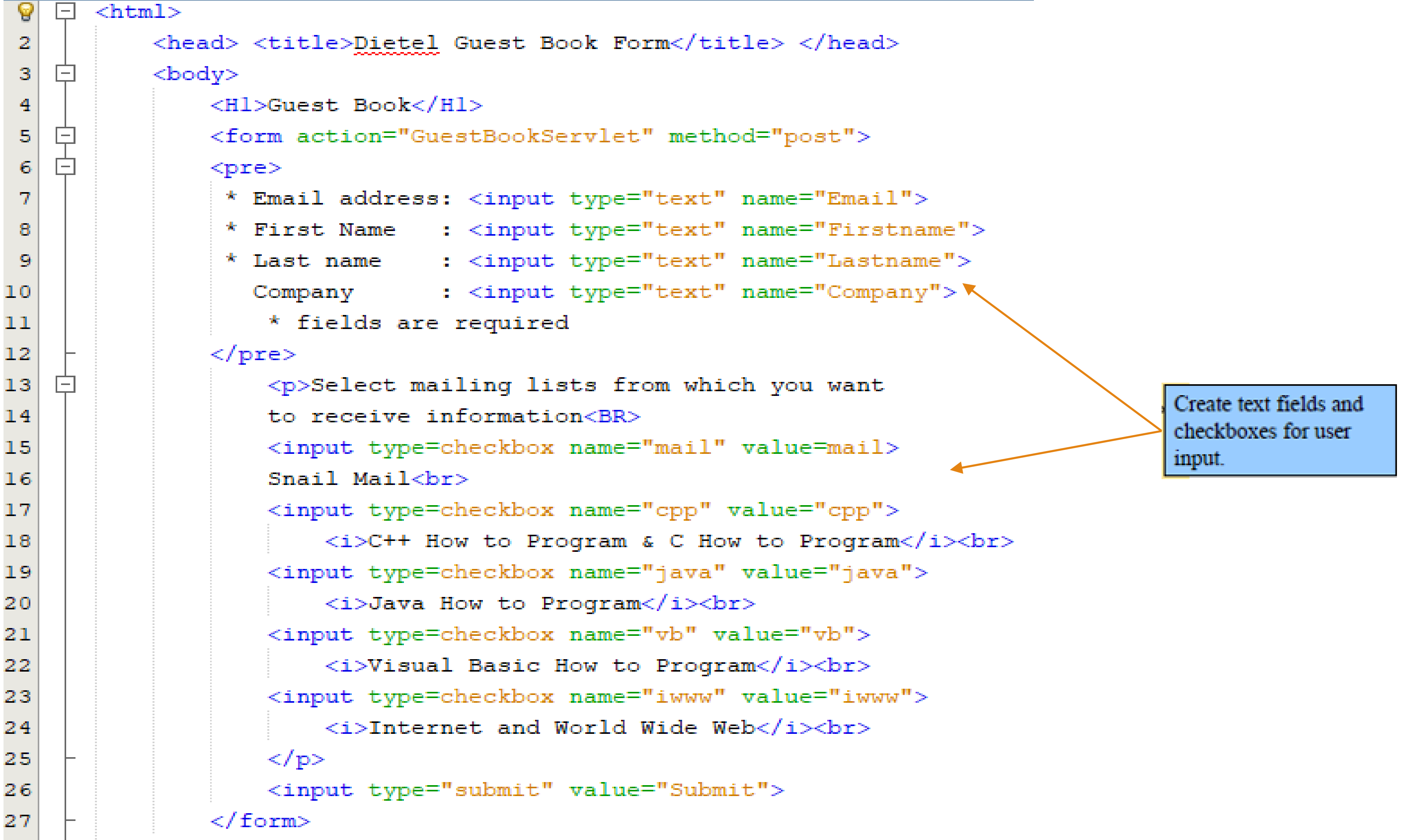
Multitier Applications: Using JDBC from a Servlet

Example servlet

- Guest book to register for mailing lists
- HTML document first tier
  - Get data from user

- Use servlet as middle tier
  - Provides access to database
  - Set up connection in **init**

- Microsoft Access database (third tier)



```html
<html>
    <head> <title>Dietel Guest Book Form</title> </head>
    <body>
        <H1>Guest Book</H1>
        <form action="GuestBookServlet" method="post">
        <pre>
         * Email address: <input type="text" name="Email">
         * First Name    : <input type="text" name="Firstname">
         * Last name     : <input type="text" name="Lastname">
           Company       : <input type="text" name="Company">
            * fields are required
        </pre>
            <p>Select mailing lists from which you want
            to receive information<BR>
            <input type=checkbox name="mail" value=mail>
            Snail Mail<br>
            <input type=checkbox name="cpp" value="cpp">
                <i>C++ How to Program & C How to Program</i><br>
            <input type=checkbox name="java" value="java">
                <i>Java How to Program</i><br>
            <input type=checkbox name="vb" value="vb">
                <i>Visual Basic How to Program</i><br>
            <input type=checkbox name="iwww" value="iwww">
                <i>Internet and World Wide Web</i><br>
            </p>
            <input type="submit" value="Submit">
        </form>
```

Create text fields and checkboxes for user input.

```java
// Fig. 19.16: GuestBookServlet.java
// Three-Tier Example
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.sql.*;

public class GuestBookServlet extends HttpServlet {
   private Statement statement = null;
   private Connection connection = null;
   private String URL = "jdbc:odbc:GuestBook";

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

      try {
         Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
         connection =
            DriverManager.getConnection( URL, "", "" );
      }
      catch ( Exception e ) {
         e.printStackTrace();
         connection = null;
      }
   }

```

`init` called exactly once, before client requests are processed. Note the first line format.

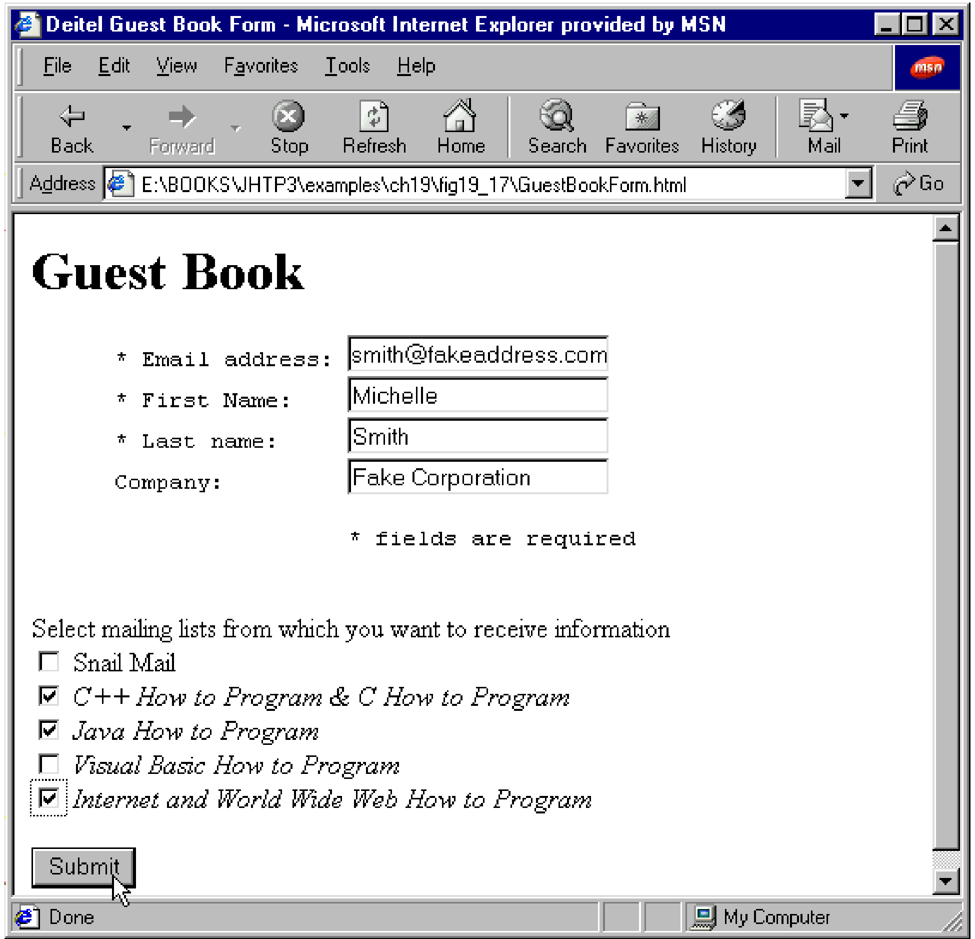
Get connection to database (no name/password).

```java
   public void doPost( HttpServletRequest req,
                       HttpServletResponse res )
      throws ServletException, IOException
   {
      String email, firstName, lastName, company,
             snailmailList, cppList, javaList, vbList,
             iwwwList;

      email = req.getParameter( "Email" );
      firstName = req.getParameter( "FirstName" );
      lastName = req.getParameter( "LastName" );
      company = req.getParameter( "Company" );
      snailmailList = req.getParameter( "mail" );
      cppList = req.getParameter( "c_cpp" );
      javaList = req.getParameter( "java" );
      vbList = req.getParameter( "vb" );
      iwwwList = req.getParameter( "iwww" );

      PrintWriter output = res.getWriter();
      res.setContentType( "text/html" );

      if ( email.equals( "" ) ||
           firstName.equals( "" ) ||
           lastName.equals( "" ) ) {
         output.println( "<H3> Please click the back " +
                         "button and fill in all " +
                         "fields.</H3>" );
         output.close();
         return;
      }

      /* Note: The GuestBook database actually contains fields
       * Address1, Address2, City, State and Zip that are not
       * used in this example. However, the insert into the
       * database must still account for these fields. */
      boolean success = insertIntoDB(
         "'" + email + "','" + firstName + "','" + lastName +
         "','" + company + "',' ',' ',' ',' ',' ','" +
         ( snailmailList != null ? "yes" : "no" ) + "','" +
         ( cppList != null ? "yes" : "no"  ) + "','" +
         ( javaList != null ? "yes" : "no"  ) + "','" +
         ( vbList != null ? "yes" : "no"  ) + "','" +
         ( iwwwList != null ? "yes" : "no"  ) + "'" );

      if ( success )
         output.print( "<H2>Thank you " + firstName +
                       " for registering.</H2>" );
      else
         output.print( "<H2>An error occurred. " +
                       "Please try again later.</H2>" );

      output.close();
   }

   private boolean insertIntoDB( String stringtoinsert )
   {
      try {
         statement = connection.createStatement();
         statement.execute(
            "INSERT INTO GuestBook values (" +
            stringtoinsert + ");" );
         statement.close();
      }
      catch ( Exception e ) {
         System.err.println(
            "ERROR: Problems with adding new entry" );
         e.printStackTrace();
         return false;
      }

      return true;
   }

   public void destroy()
   {
      try {
         connection.close();
      }
      catch( Exception e ) {
         System.err.println( "Problem closing the database" );
      }
   }
}
```

Insert data into database.

`destroy` called automatically, closes connection to database.

PROGRAM OUTPUT


Deitel Guest Book Form - Microsoft Internet Explorer provided by MSN
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print
Address E:\BOOKS\JHTP3\examples\ch19\fig19_17\GuestBookForm.html
Go
Done
My Computer

Guest Book

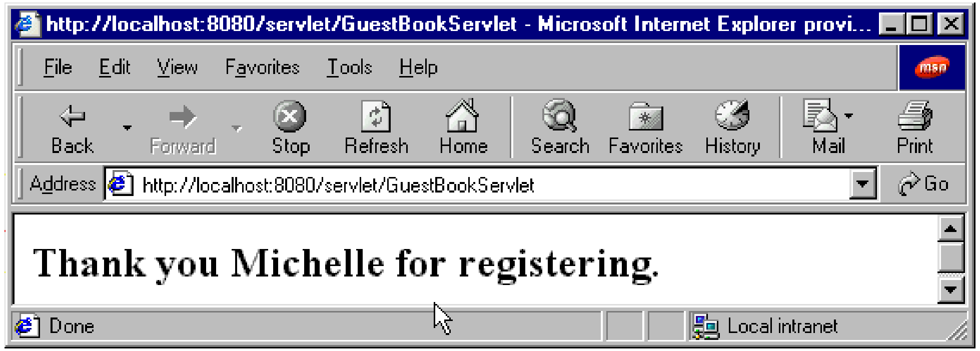
```
* Email address:   smith@fakeaddress.com
* First Name:      Michelle
* Last name:       Smith
Company:           Fake Corporation

                   * fields are required
```

Select mailing lists from which you want to receive information

- [ ] Snail Mail
- [✓] *C++ How to Program & C How to Program*
- [✓] *Java How to Program*
- [ ] *Visual Basic How to Program*
- [✓] *Internet and World Wide Web How to Program*

Submit

PROGRAM OUTPUT


http://localhost:8080/servlet/GuestBookServlet - Microsoft Internet Explorer provi...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print
Address http://localhost:8080/servlet/GuestBookServlet
Go
Done
Local intranet

```
Thank you Michelle for registering.
```