

# Java Servlets

CSE 305 – Principles of Database Systems

Paul Fodor

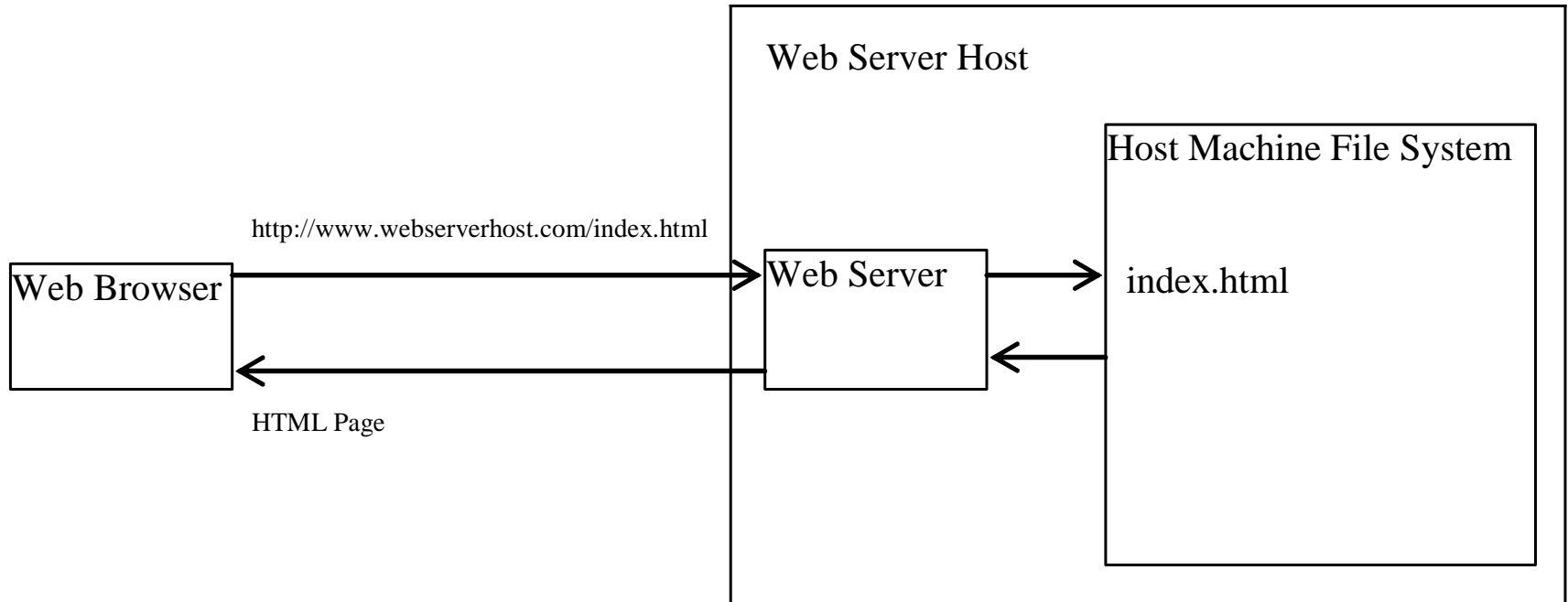
Stony Brook University

<http://www.cs.stonybrook.edu/~cse305>

# Servlets

- Servlet technology is primarily designed for use with the HTTP protocol of the Web.
- Servlets are Java programs that run on a Web server.
- Java servlets can be used to process client requests or produce dynamic Web pages.

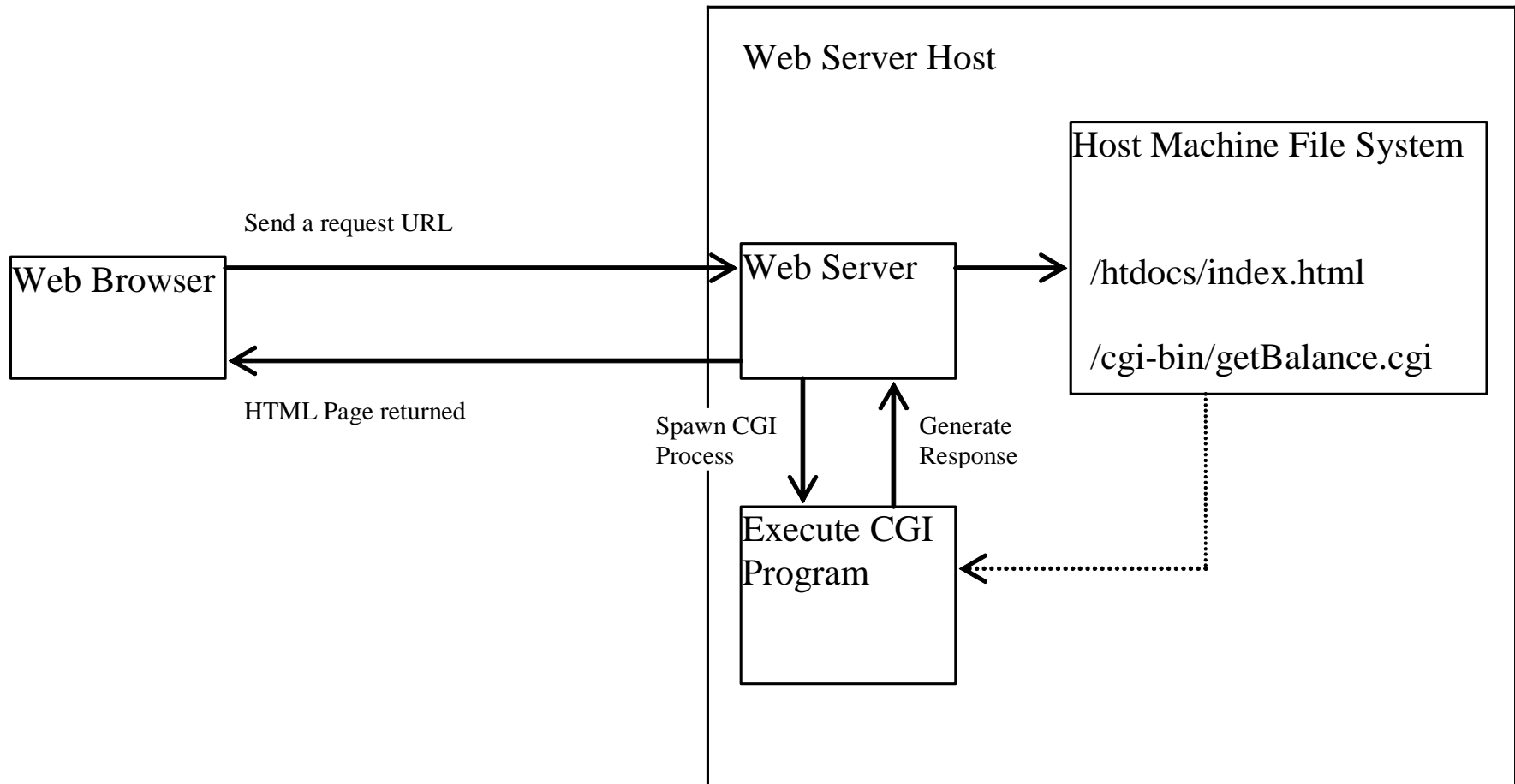
# HTTP and HTML



# CGI

- The Common Gateway Interface, or CGI, was proposed to generate dynamic Web contents.
- The interface provides a standard framework for Web servers to interact with external programs, known as the CGI programs.

# How Does CGI Work?



# The GET and POST Methods

- The two most common HTTP requests, also known as methods, are GET and POST.
  - GET - Requests data from a specified resource
    - **the query string (name/value pairs) is sent in the URL of a GET request**  
`Servlet?name1=value1&name2=value2`
    - GET requests can be cached
    - GET requests remain in the browser history
  - POST - Submits data to be processed to a specified resource
    - **the query string (name/value pairs) is sent in the HTTP message body of a POST request** (never cached)
- The Web browser issues a request using a URL or an HTML form to trigger the Web server to execute a CGI program.
- When issuing a CGI request directly from a URL, the GET method is used.

# GET vs. POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL  Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

# Other HTTP Request Methods

Method	Description
HEAD	Same as GET but returns only HTTP headers and no document body
PUT	Uploads a representation of the specified URI
DELETE	Deletes the specified resource
OPTIONS	Returns the HTTP methods that the server supports
CONNECT	Converts the request connection to a transparent TCP/IP tunnel



# Query String

- The URL query string consists of the location of the CGI program, parameters and their values.

`http://www.webserverhost.com/cgi-bin/  
getBalance.cgi?accountId=john+smith&password=tiger`

- The ? symbol separates the program from the parameters.
- The parameter name and value are associated using the = symbol.
- The parameter pairs are separated using the & symbol.
- The + symbol denotes a space character.

# HTML Forms

HTML forms enable you to submit data to the Web server in a convenient form.

The form can contain text fields, text area, check boxes, combo boxes, lists, radio buttons, and buttons.

Student Registration Form

Last Name  First Name  MI

Gender:  Male  Female

Major  Minor

Hobby:  Tennis  Golf  Ping Pong

Remarks:

# From CGI to Java Servlets

- Java servlets are Java programs.
- They function like CGI programs.
- They are executed upon the request from Web browser.
- All the servlets run inside a servlet container.
- A servlet container is also referred to as a servlet server, or a servlet engine.
- A servlet container is a single process that runs a JVM.
- The JVM creates a thread to handle each servlet.
- Java threads have much less overhead than full-brown processes.
- All the threads share the same memory allocated to the JVM.

# Creating and Running Servlets

- To run Java servlets, you need a servlet container.
- Many servlet containers are available.
- Tomcat, developed by Apache ([www.apache.org](http://www.apache.org)), is a standard reference implementation for Java Servlet and Java Server Pages.



# The Servlet Interface

```
/**Invoked for every servlet constructed*/  
public void init(ServletConfig p0) throws ServletException;  
  
/**Invoked to respond to incoming requests*/  
public void service(ServletRequest p0, ServletResponse p1)  
    throws ServletException, IOException;  
  
/**Invoked to release resource by the servlet*/  
public void destroy();  
  
/**Return information about the servlet*/  
public String getServletInfo();  
  
/**Return configuration objects of the servlet*/  
public ServletConfig getServletConfig();
```

# Servlet Life-Cycle

1. The `init` method is called when the servlet is first created, and is not called again as long as the servlet is not destroyed.
2. The `service` method is invoked each time the server receives a request for the servlet. The server spawns a new thread and invokes `service`.
3. The `destroy` method is invoked once all threads within the servlet's `service` method have exited or after a timeout period has passed. This method releases resources for the servlet.

# The HttpServlet Class

- The HttpServlet class defines a servlet for the HTTP protocol.
  - It extends GenericServlet and implements the service method.
  - The service method is implemented as a dispatcher of HTTP requests.
  - The HTTP requests are processed in the following methods: **doGet**, **doPost**, **doDelete**, **doPut**, **doOptions**, and **doTrace**. All these methods have the same signature as follows:

```
protected void doXxx (HttpServletRequest req,  
HttpServletRequest resp) throws ServletException,  
java.io.IOException
```



# The HttpServletRequest Interface

- Every doXxx method in the HttpServletRequest class has an argument of the HttpServletRequest type, which is an object that contains HTTP request information including parameter name and values, attributes, and an input stream.
- HttpServletRequest is a subinterface of ServletRequest.
- ServletRequest defines a more general interface to provide information for all kinds of clients.

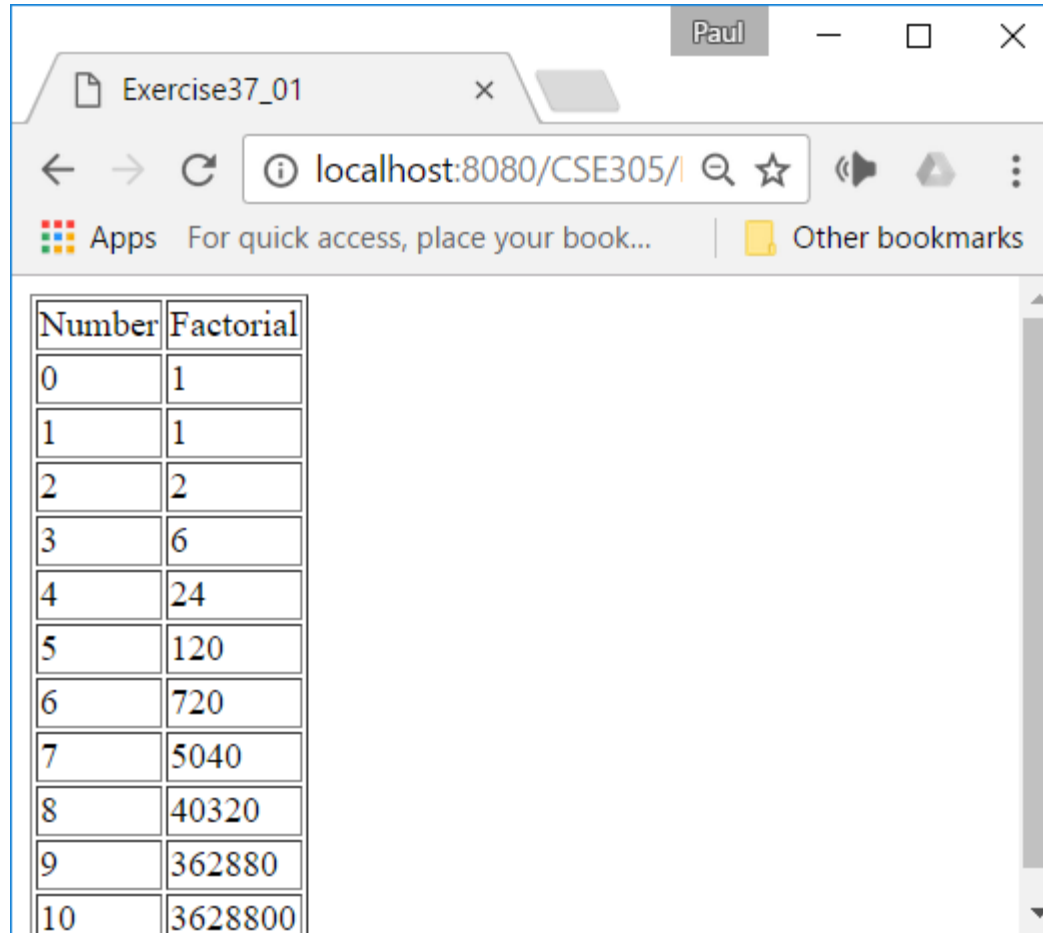
# The HttpServletResponse Interface

- Every doXxx method in the HttpServletResponse class has an argument of the HttpServletResponse type, which is an object that assists a servlet in sending a response to the client.
- HttpServletResponse is a subinterface of ServletResponse.
- ServletRequest defines a more general interface for sending output to the client.

# Creating Servlets

- To write a Java servlet, you define a class that extends the `HttpServlet` class.
- The servlet engine controls the servlets using the `init`, `doGet`, `doPost`, `destroy`, and other methods.
- By default, the `doGet` and `doPost` methods do nothing.
- To handle the GET request, you need to override the `doGet` method; to handle the POST request, you need to override the `doPost` method.

# Creating Servlets: FactorialServlet



A screenshot of a web browser window. The browser has a single tab titled 'Exercise37\_01'. The address bar shows 'localhost:8080/CSE305/'. Below the address bar, there are sections for 'Apps' and 'Other bookmarks'. The main content area of the browser displays a table with two columns: 'Number' and 'Factorial'. The table contains data for numbers from 0 to 10, with their corresponding factorial values.

Number	Factorial
0	1
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class FactorialServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html";

    /** Process the HTTP Get request */
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Exercise37_01</title></head>");
        out.println("<body>");
        out.println("<table border=\"1\">");
        out.println("<tr>");
        out.println("<td>Number</td>");
        out.println("<td>Factorial</td>");
        out.println("</tr>");
    }
}
```

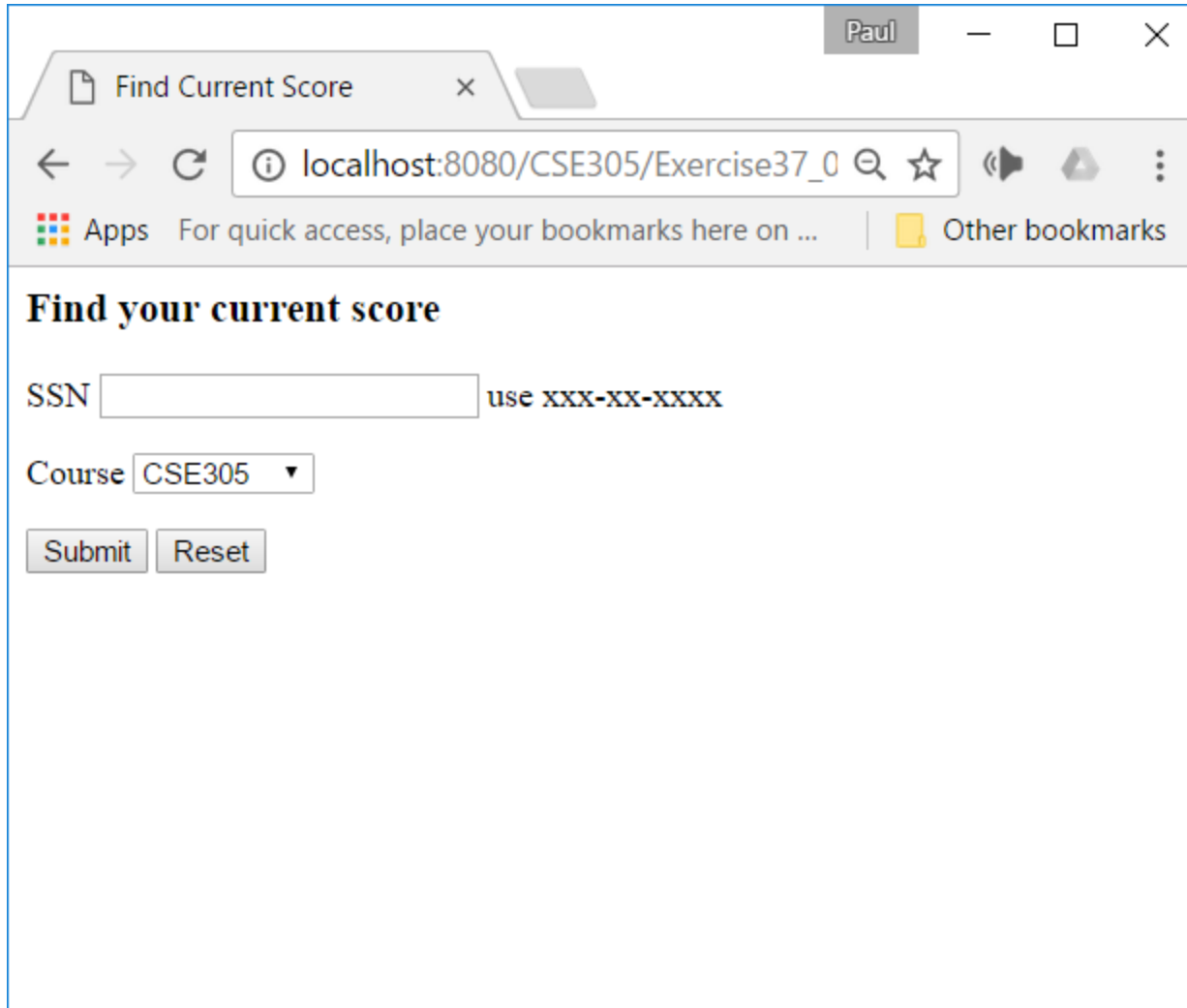
```
for (int i = 0; i <= 10; i++) {
    out.println("<tr>");
    out.println("<td>" + i + "</td>");
    out.println("<td>" + computeFactorial(i) + "</td>");
    out.println("</tr>");
}
out.println("</body></html>");
```

```
}
```

```
private long computeFactorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * computeFactorial(n - 1);
}
```

```
}
```

# Submitting forms



The screenshot shows a web browser window with a single tab titled "Find Current Score". The address bar displays "localhost:8080/CSE305/Exercise37\_0". The page content includes a heading "Find your current score", an SSN input field with a placeholder "use xxx-xx-xxxx", a "Course" dropdown menu set to "CSE305", and two buttons labeled "Submit" and "Reset".

**Find your current score**

SSN  use xxx-xx-xxxx

Course

## find\_score.html

```
<html>
<head>
<title>Find Current Score</title>
</head>
<body>
<h3>Find your current score</h3>

<form method="get" action="FindScoreServlet">
<p>SSN <input type="text" name="ssn"/> use xxx-xx-xxxx
  <p>Course <select size="1" name="course">
    <option value="CSE305">CSE305</option>
    <option value="CSE307">CSE305</option>
    <option value="CSE114">CSE114</option>
  </select> 
<p><input type="submit" name="Submit" value="Submit" />
  <input type="reset" value="Reset" />
</form>
</body>
</html>
```



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

public class FindScore extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html";

    /**Process the HTTP Get request*/
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>Find score</title></head>");
        out.println("<body>");

        String course = request.getParameter("course");
        String ssn = request.getParameter("ssn");

        BufferedReader in = null;
        boolean found = false;
        try {
            in = new BufferedReader(new FileReader(
                "c:\\\" + course + ".txt"));
```

```

String line;
while ( (line = in.readLine()) != null) {
    StringTokenizer st = new StringTokenizer(line, "#\n\r\t");
    String name = st.nextToken();
    String ssnFromFile = st.nextToken();
    String score = st.nextToken();
    System.out.println(name + " " + ssnFromFile + " " + score);
    if (ssn.equals(ssnFromFile)) {
        out.print(name + " " + score);
        found = true;
        break;
    }
}
if (!found) {
    out.print(ssn + " not found in " + course);
}
out.println("</body></html>");
}
catch (Exception ex) {
    out.println(ex.toString());
    ex.printStackTrace();
}

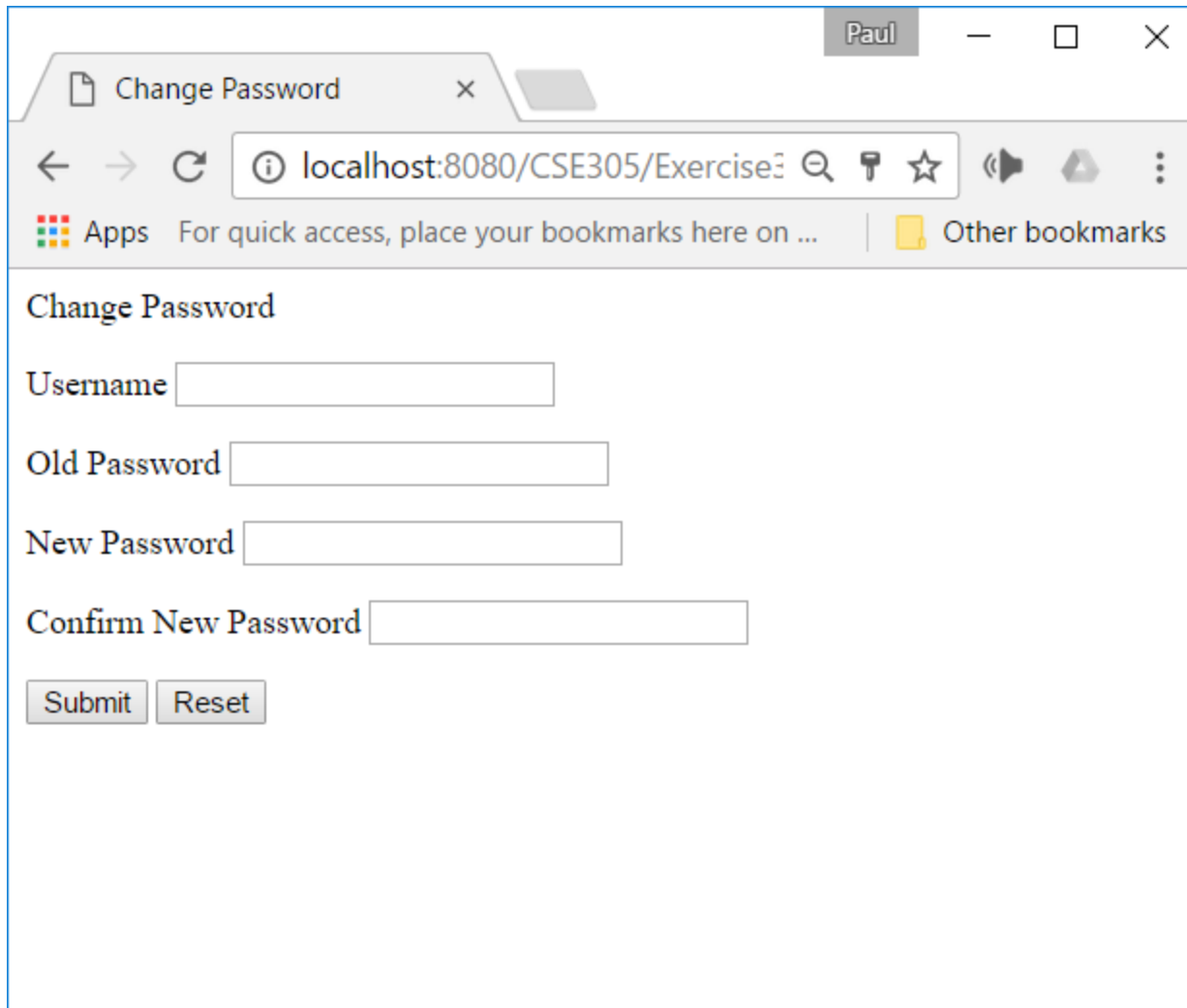
```

```
finally {  
    try {  
        if (in != null) in.close();  
        if (out != null) out.close();  
    }  
    catch (Exception ex) { }  
}  
  
/**Clean up resources*/  
public void destroy()  
{  
}  
}
```

# Database Programming Using Servlets

- Many dynamic Web applications use databases to store and manage data.
- Servlets can connect to any relational database via JDBC.
- Connecting a servlet to a database is no different from connecting a Java application or applet to a database.
- If you know Java servlets and JDBC, you can combine them together to develop interesting and practical Web based interactive projects immediately.
- Example: Registering Student into a Database

# Example: change a password



A screenshot of a web browser window. The window title is "Paul". The browser tab is titled "Change Password". The address bar shows "localhost:8080/CSE305/Exercise3". The page content includes a form titled "Change Password" with the following fields and buttons:

Change Password

Username

Old Password

New Password

Confirm New Password

## change\_password.html

```
<html>
<head>
<title>Change Password</title>
</head>
<body>Change Password
<p></font>
<form method="post" action="ChangePassword">
<p> Username
  <input type="text" name="username">
<p> Old Password
  <input type="password" name="oldPassword">
<p> New Password
  <input type="password" name="newPassword">
<p> Confirm New Password
  <input type="password" name="confirmNewPassword">
<p>
  <input type="submit" name="Submit" value="Submit">
  <input type="reset" value="Reset">
</p>
</form>
</body>
</html>
```

```

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

public class ChangePassword extends HttpServlet {

    private static final String CONTENT_TYPE = "text/html";
    PreparedStatement pstmtCheckPassword;
    PreparedStatement pstmtUpdate;
    PreparedStatement pstmtGetName;

    public void init() throws ServletException {
        try {
            initializeJdbc();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /**Initialize database connection*/
    private void initializeJdbc() {
        try {
            // Explicitly load a MySQL driver
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("Driver loaded");
        }
    }
}

```

```

// Establish a connection
Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost/test", "root", "tiger");

// Create prepared statements
pstmtCheckPassword = conn.prepareStatement(
    "SELECT COUNT(*) FROM account WHERE username = ? "
    + "AND password = ?");
pstmtUpdate = conn.prepareStatement(
    "UPDATE account SET password = ? WHERE username = ?");
pstmtGetName = conn.prepareStatement(
    "SELECT name FROM Account WHERE username = ?");
} catch (Exception ex) {
    ex.printStackTrace();
}
}

/**Process the HTTP Post request*/
public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
// Write HTML back to a browser
response.setContentType(CONTENT_TYPE);

```



```

// Obtain username and password
String username = request.getParameter("username").trim();
String oldPassword = request.getParameter("oldPassword").trim();
String newPassword = request.getParameter("newPassword").trim();
String confirmNewPassword = request.getParameter("confirmNewPassword").trim();

// Check password
try {
    pstmtCheckPassword.setString(1, username.trim());
    pstmtCheckPassword.setString(2, oldPassword.trim());
    ResultSet resultSet = pstmtCheckPassword.executeQuery();
    resultSet.next();
    if (resultSet.getInt(1) < 1) {
        out.println("Your old password is incorrect. Please try again<p>");
        out.println("<form method=\"post\" action=ChangePassword>");
        out.println("<p>Username <input type=\"text\" name=\"username\">"
            + "<p>Old Password <input type=\"password\" name=\"oldPassword\">"
            + "<p>New Password <input type=\"password\" name=\"newPassword\">"
            + "<p>Confirm New Password <input type=\"password\" "
            + "name=\"confirmNewPassword\">"
            + "<p><input type=\"submit\" name=\"Submit\" value=\"Submit\">"
            + " <input type=\"reset\" value=\"Reset\"></p>"
            + "</form>");
        return;
    }
}

```

```

if (newPassword.equals(confirmNewPassword)) {
    pstmtUpdate.setString(1, newPassword);
    pstmtUpdate.setString(2, username);
    pstmtUpdate.executeUpdate();

    pstmtGetName.setString(1, username);
    resultSet = pstmtGetName.executeQuery();

    resultSet.next();
    String name = resultSet.getString(1);

    out.println("Hello, " + name + ", your password has been updated!");
} else {
    out.println("Your new password is not confirmed correctly. "
        + "Please try again<p>");
    out.println("<form method=\"post\" action=Exercise37_8>");
    out.println("<p>Username <input type=\"text\" name=\"username\">"
        + "<p>Old Password <input type=\"password\" name=\"oldPassword\">"
        + "<p>New Password <input type=\"password\" name=\"newPassword\">"
        + "<p>Confirm New Password <input type=\"password\" "
        + "name=\"confirmNewPassword\">"
        + "<p><input type=\"submit\" name=\"Submit\" value=\"Submit\">"
        + "    <input type=\"reset\" value=\"Reset\"></p>"
        + "</form>");
    return;
}
} catch (Exception ex) {
    ex.printStackTrace();
    return;
}
}
}

```

# 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.

# What is a Session ?

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.

## Session Tracking Techniques

Using hidden values, using cookies, and using the session tracking tools from servlet API.

# Session Tracking Using Hidden Values

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">
```

So the next request will submit the data back to the servlet.

The servlet retrieves this hidden value just like any other parameter value using the `getParameter` method.

# For Example: Using Hidden Values in the Registration form

- This example creates a servlet that processes a registration form.
- The client first submits the form using the GET method.
- The server collects the data in the form, displays the data to the client, and asks the client for confirmation.
- The client confirms it by submitting the request with the hidden values using the POST method.
- Finally, the servlet writes the data to a database.

# Session Tracking Using Cookies

- 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.
- Cookies are sent from the server through the instructions in the header of the HTTP response.
- 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.
- 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.

```

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

public class RegistrationWithCookie extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html";
    // Use a prepared statement to store a student into the database
    private PreparedStatement pstmt;

    /** Initialize global variables */
    public void init() throws ServletException {
        initializeJdbc();
    }

    /** Process the HTTP Get request */
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Obtain data from the form
        String lastName = request.getParameter("lastName");
        String firstName = request.getParameter("firstName");
        String mi = request.getParameter("mi");
        String telephone = request.getParameter("telephone");
        String email = request.getParameter("email");
        String street = request.getParameter("street");
        String city = request.getParameter("city");
    }
}

```



```
String state = request.getParameter("state");
String zip = request.getParameter("zip");

// Create cookies and send cookies to browsers
Cookie cookieLastName = new Cookie("lastName", lastName);
// cookieLastName.setMaxAge(1000);
response.addCookie(cookieLastName);
Cookie cookieFirstName = new Cookie("firstName", firstName);
response.addCookie(cookieFirstName);
// cookieFirstName.setMaxAge(0);
Cookie cookieMi = new Cookie("mi", mi);
response.addCookie(cookieMi);
Cookie cookieTelephone = new Cookie("telephone", telephone);
response.addCookie(cookieTelephone);
Cookie cookieEmail = new Cookie("email", email);
response.addCookie(cookieEmail);
Cookie cookieStreet = new Cookie("street", street);
response.addCookie(cookieStreet);
Cookie cookieCity = new Cookie("city", city);
response.addCookie(cookieCity);
Cookie cookieState = new Cookie("state", state);
response.addCookie(cookieState);
Cookie cookieZip = new Cookie("zip", zip);
response.addCookie(cookieZip);

// Ask for confirmation
out.println("You entered the following data");
out.println("<p>Last name: " + lastName);
```

```

// Set the action for processing the answers
out.println("<p><form method=\"post\" action=" +
    "/RegistrationWithCookie>");
out.println("<p><input type=\"submit\" value=\"Confirm\" >");
out.println("</form>");
out.close(); // Close stream
}
/** Process the HTTP Post request */
public void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();

    String lastName = "";
    String firstName = "";
    String mi = "";
    String telephone = "";
    String email = "";
    String street = "";
    String city = "";
    String state = "";
    String zip = "";

    // Read the cookies
    Cookie[] cookies = request.getCookies();

    // Get cookie values
    for (int i = 0; i < cookies.length; i++) {
        if (cookies[i].getName().equals("lastName"))
            lastName = cookies[i].getValue();
    }
}

```

```

else if (cookies[i].getName().equals("firstName"))
    firstName = cookies[i].getValue();
else if (cookies[i].getName().equals("mi"))
    mi = cookies[i].getValue();
else if (cookies[i].getName().equals("telephone"))
    telephone = cookies[i].getValue();
else if (cookies[i].getName().equals("email"))
    email = cookies[i].getValue();
else if (cookies[i].getName().equals("street"))
    street = cookies[i].getValue();
else if (cookies[i].getName().equals("city"))
    city = cookies[i].getValue();
else if (cookies[i].getName().equals("state"))
    state = cookies[i].getValue();
else if (cookies[i].getName().equals("zip"))
    zip = cookies[i].getValue();
}

try {
    storeStudent(lastName, firstName, mi, telephone, email, street,
        city, state, zip);

    out.println(firstName + " " + lastName +
        " is now registered in the database");

    out.close(); // Close stream
}
catch(Exception ex) {
    out.println("Error: " + ex.getMessage());
    return; // End the method
} }

```

# Session Tracking Using the Servlet API

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.

Java servlet API provides a session tracking tool, which enables tracking of a large set of data. Data can be stored as objects. Data are kept on the server side so they are secure.

# The HttpSession Class

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);
```

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

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

```

public class RegistrationWithHttpSession extends HttpServlet {
    // Use a prepared statement to store a student into the database
    private PreparedStatement pstmt;

    /** Initialize global variables */
    public void init() throws ServletException {
        initializeJdbc();
    }

    /** Process the HTTP Get request */
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        // Set response type and output stream to the browser
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Obtain data from the form
        String lastName = request.getParameter("lastName");
        String firstName = request.getParameter("firstName");
        String mi = request.getParameter("mi");
        String telephone = request.getParameter("telephone");
        String email = request.getParameter("email");
        String street = request.getParameter("street");
        String city = request.getParameter("city");
        String state = request.getParameter("state");
        String zip = request.getParameter("zip");
        if (lastName.length() == 0 || firstName.length() == 0) {
            out.println("Last Name and First Name are required");
            return; // End the method
        }
    }
}

```

```

// Create a Student object
Student student = new Student(lastName, firstName,
    mi, telephone, email, street, city, state, zip);

// Get an HttpSession or create one if it does not exist
HttpSession httpSession = request.getSession();

// Store student object to the session
httpSession.setAttribute("student", student);

// Ask for confirmation
out.println("You entered the following data");
out.println("<p>Last name: " + lastName);
out.println("<p>First name: " + firstName);
out.println("<p>MI: " + mi);
out.println("<p>Telephone: " + telephone);
out.println("<p>Email: " + email);
out.println("<p>Address: " + street);
out.println("<p>City: " + city);
out.println("<p>State: " + state);
out.println("<p>Zip: " + zip);

// Set the action for processing the answers
out.println("<p><form method=\"post\" action=" +
    "/RegistrationWithHttpSession>");
out.println("<p><input type=\"submit\" value=\"Confirm\" >");
out.println("</form>");
out.close(); // Close stream

```

```
/** Process the HTTP Post request */
public void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    // Set response type and output stream to the browser
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    // Obtain the HttpSession
    HttpSession httpSession = request.getSession();

    // Get the Student object in the HttpSession
    Student student = (Student)(httpSession.getAttribute("student"));

    try {
        storeStudent(student);

        out.println(student.firstName + " " + student.lastName +
            " is now registered in the database");
        out.close(); // Close stream
    }
    catch(Exception ex) {
        out.println("Error: " + ex.getMessage());
        return; // End the method
    }
}
```



```

/** Initialize database connection */
private void initializeJdbc() {
    try {
        // Declare driver and connection string
        String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
        String connectionString = "jdbc:odbc:exampleMDBCDataSource";

        // Load the Oracle JDBC Thin driver
        Class.forName(driver);
        System.out.println("Driver " + driver + " loaded");

        // Connect to the sample database
        Connection conn = DriverManager.getConnection
            (connectionString);
        System.out.println("Database " + connectionString +
            " connected");

        // Create a Statement
        pstmt = conn.prepareStatement("INSERT INTO address " +
            "(lastName, firstName, mi, telephone, email, street, city, " +
            + "state, zip) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");
    }
    catch (Exception ex) {
        System.out.println(ex);
    }
}

```

```

private void storeStudent(Student student) throws SQLException {
    pstmt.setString(1, student.getLastName());
    pstmt.setString(2, student.getFirstName());
    pstmt.setString(3, student.getMi());
    pstmt.setString(4, student.getTelephone());
    pstmt.setString(5, student.getEmail());
    pstmt.setString(6, student.getAddress());
    pstmt.setString(7, student.getCity());
    pstmt.setString(8, student.getState());
    pstmt.setString(9, student.getZip());
    pstmt.executeUpdate();
}

```

```

class Student {
    private String lastName = "";
    private String firstName = "";
    private String mi = "";
    private String telephone = "";
    private String email = "";
    private String street = "";
    private String city = "";
    private String state = "";
    private String zip = "";

```

```

Student(String lastName, String firstName,
String mi, String telephone, String email, String street,
String city, String state, String zip) {
    this.lastName = lastName;
    this.firstName = firstName;
    this.mi = mi; ...

```

# Sending Images From the Servlets

Java servlets are not limited to sending text to a browser. Java servlets can return images in GIF, JPEG, or PNG format. This section demonstrates returning images in GIF format.

To send contents as a GIF image, the content type must be set to image/gif like this:

```
response.setContentType("image/gif");
```

Images are binary data. You have to use a binary output stream like this:

```
OutputStream out = response.getOutputStream();
```