Introduction to Session Tracking

When you visit a website and interact with it (for example, by logging in, adding items to a shopping cart, or browsing pages), the website needs to remember your actions to give you a personalized experience. This process of keeping track of what you do is called **session tracking**.

Why Do We Need Session Tracking?

The web is **stateless**, which means that every time you load a new page or make a request (like clicking a button), the server treats it like a new interaction. It doesn't automatically remember anything you did before. Without session tracking, a website wouldn't know who you are or what actions you took previously.

Session tracking solves this problem by helping the server remember information about you (like your login status or shopping cart) as you move from one page to another.

Example:

- Imagine you log in to an online store. The server needs to "remember" who you are while you shop, so it can show your name and track the items in your cart. Without session tracking, you'd have to log in every time you click on a new page.

Methods of Session Tracking

There are a few common ways websites track sessions

1. **Cookies**
    - These are small pieces of data stored on your computer by the website. When you make another request, the browser automatically sends the cookie back to the server, so the server knows it's you.
    - **Example**: When you log in to a site, the website might store your session ID in a cookie, so it knows you're still logged in the next time you visit.
2. **URL Rewriting**
    - The session information is added to the URL. This way, every time you click a link, the session data (like your session ID) is passed along with the URL.
    - **Example**: Instead of a simple link like www.example.com/home, you might see something like www.example.com/home?sessionId=12345.
3. **Hidden Form Fields**
    - Websites can store session data in hidden form fields. When you submit a form (like a login form), the session data is sent along with the form, even though you don't see it.
    - **Example**: A login form might have hidden fields that pass along your session ID when you submit it.

4. **HTTP Sessions**
    - This is managed by the server itself. The server keeps track of session data, and the browser only needs to send a session ID, which the server uses to retrieve the stored information.
    - **Example**: When you log in to a website, the server creates a session for you. Each time you make a request, the server checks your session ID to remember your login details, items in your cart, and so on.

Focus for This Week

In this week's material, we'll mainly look at two methods:

1. **Cookies**: How websites use small files in your browser to keep track of your session.
2. **HTTP Sessions**: How servers store your session data to remember your actions during your visit.

These are the most used session-tracking methods for modern web applications.

Session Tracking with Cookies

Cookies are small text files that a website stores on your computer through your web browser. They work like a memory for the website, storing small pieces of information (like your preferences or session ID) in the form of **key-value pairs** (a pair like "username=JohnDoe"). Cookies are used to track user data across different interactions or page visits, allowing the server to recognize the user each time they return to the site.

Steps for Using Cookies in Servlets

Here's how cookies are used in **Servlets** (the programs that run on a web server to handle user requests):

1. **Create a Cookie**
    - A website (through a servlet) can create a cookie to store some information about the user. For example, the server might create a cookie with your username after you log in.
    - The cookie is created with two parts: a **name** (like "username") and a **value** (like "JohnDoe").
    - The server then **sends this cookie to the client** (your web browser) using an HTTP response. The browser stores the cookie.
2. **Retrieve a Cookie**
    - When the user interacts with the website again (for example, by clicking on a new page), the browser **automatically sends the cookie back** to the server.
    - The server can now access the cookie's **value** (e.g., "JohnDoe") and remember who the user is without needing them to log in again.

Example:

- **Step 1 (Create a Cookie)**: After logging into an online store, the server might create a cookie like username=JohnDoe and send it to your browser.
- **Step 2 (Retrieve a Cookie)**: When you visit another page on the same site, your browser sends the username=JohnDoe cookie back to the server, and the server can greet you with, "Hello, JohnDoe!"

In summary, cookies make it possible for websites to remember important details about users (like login status or shopping cart items) between different page visits, creating a smoother and more personalized experience.

Implementing Session Tracking Using Cookies in NetBeans

Step-by-Step Example

1. **Create a New Servlet Project**:
    - Open **NetBeans**.
    - Go to File > New Project, select Java Web > Web Application, and click Next.
    - Name the project SessionTrackingExample and select **Apache Tomcat** as the server.
2. **Create the First Servlet**: This servlet will set a cookie for the user.
    - Right-click on the Source Packages and create a new servlet named SetCookieServlet.

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/setcookie")
public class SetCookieServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Create a new cookie
        Cookie userCookie = new Cookie("username", "JohnDoe");

        // Set the cookie's max age to 1 day (86400 seconds)
        userCookie.setMaxAge(86400);

        // Add the cookie to the response
        response.addCookie(userCookie);

        // Send a response message
        response.setContentType("text/html");
        response.getWriter().println("Cookie for username set.");
    }
}
```

```java
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/setcookie")
public class SetCookieServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Create a new cookie
        Cookie userCookie = new Cookie("username", "JohnDoe");

        // Set the cookie's max age to 1 day (86400 seconds)
        userCookie.setMaxAge(86400);

        // Add the cookie to the response
        response.addCookie(userCookie);

        // Send a response message
        response.setContentType("text/html");
        response.getWriter().println("Cookie for username set.");
    }
}
```

3. **Create the Second Servlet**: This servlet will retrieve and read the cookie.
    - Right-click and create another servlet named GetCookieServlet.

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/getcookie")
public class GetCookieServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      // Get all cookies from the request
      Cookie[] cookies = request.getCookies();
      String username = null;

      // Check if cookies are not null
      if (cookies != null) {
         // Loop through each cookie to find the one with the name 'username'
         for (Cookie cookie : cookies) {
            if (cookie.getName().equals("username")) {
```

```
                username = cookie.getValue();
                break;
            }
        }
    }

    // Set the response content type and display the username
    response.setContentType("text/html");
    if (username != null) {
        response.getWriter().println("Hello, " + username + "!");
    } else {
        response.getWriter().println("No user found.");
    }
  }
}
```

```java
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/getcookie")
public class GetCookieServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Get all cookies from the request
        Cookie[] cookies = request.getCookies();
        String username = null;

        // Check if cookies are not null
        if (cookies != null) {
            // Loop through each cookie to find the one with the name 'username'
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("username")) {
                    username = cookie.getValue();
                    break;
                }
            }
        }

        // Set the response content type and display the username
        response.setContentType("text/html");
        if (username != null) {
            response.getWriter().println("Hello, " + username + "!");
        } else {
            response.getWriter().println("No user found.");
        }
    }
}
```

4. **Run the Project**:
    - Access the first servlet via http://localhost:8080/SessionTrackingExample/setcookie to set the cookie.
    - Then, go to http://localhost:8080/SessionTrackingExample/getcookie to retrieve the cookie and see the stored value ("Hello, JohnDoe!").

Using HTTP Sessions

An **HTTP session** is a server-side mechanism to track user interactions with a web application. Unlike cookies, session information is stored on the server, and only the session ID is passed between the client and the server.

Steps for Using HTTP Sessions in Servlets

1. **Create or Retrieve a Session**: When the user interacts with the servlet, you can create a session or retrieve the existing session.
2. **Store Data in the Session**: You can store any user-specific data, like user preferences or login information.
3. **Retrieve Data from the Session**: On subsequent requests, you can retrieve this session data.

Implementing Session Tracking Using HTTP Sessions in NetBeans

Step-by-Step Example

1. **Create a Servlet to Set Session Attributes**:
    - Right-click on the Source Packages and create a new servlet named SetSessionServlet.

```
import java.io.IOException;
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;

@WebServlet("/setsession")
public class SetSessionServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      // Get or create a session
      HttpSession session = request.getSession();

      // Store a username in the session
      session.setAttribute("username", "JaneDoe");

      // Send a response message
      response.setContentType("text/html");
      response.getWriter().println("Session for username set.");
   }
```

}

```java
import java.io.IOException;
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;

@WebServlet("/setsession")
public class SetSessionServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Get or create a session
        HttpSession session = request.getSession();

        // Store a username in the session
        session.setAttribute("username", "JaneDoe");

        // Send a response message
        response.setContentType("text/html");
        response.getWriter().println("Session for username set.");
    }
}
```

2. **Create a Servlet to Get Session Attributes**:
    - Right-click and create another servlet named GetSessionServlet.

```
import java.io.IOException;
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;

@WebServlet("/getsession")
public class GetSessionServlet extends HttpServlet {

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    // Retrieve the session, if it exists
    HttpSession session = request.getSession(false);

    String username = null;

    // Check if session exists and retrieve the username attribute
    if (session != null) {
      username = (String) session.getAttribute("username");
    }

    // Set the response content type and display the username
    response.setContentType("text/html");
```

```
        if (username != null) {
            response.getWriter().println("Hello, " + username + "!");
        } else {
            response.getWriter().println("No active session.");
        }
    }
}
```

```java
import java.io.IOException;
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;

@WebServlet("/getsession")
public class GetSessionServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Retrieve the session, if it exists
        HttpSession session = request.getSession(false);

        String username = null;

        // Check if session exists and retrieve the username attribute
        if (session != null) {
            username = (String) session.getAttribute("username");
        }

        // Set the response content type and display the username
        response.setContentType("text/html");
        if (username != null) {
            response.getWriter().println("Hello, " + username + "!");
        } else {
            response.getWriter().println("No active session.");
        }
    }
}
```

3. **Run the Project**:
    - Access the first servlet via http://localhost:8080/SessionTrackingExample/setsession to set the session.
    - Then, go to http://localhost:8080/SessionTrackingExample/getsession to retrieve the session data and see the stored value ("Hello, JaneDoe!").

Inter-Servlet Communication

Inter-Servlet communication allows servlets to share information and communicate with each other. This is essential when multiple servlets work together to handle complex requests in a web application.

Methods of Inter-Servlet Communication:

1. **Forwarding Requests**: One servlet can forward a request to another servlet using the RequestDispatcher.
2. **Sharing Data through Session**: Multiple servlets can access session attributes to share data.
3. **Redirecting Requests**: One servlet can send a client-side redirect to another servlet using response.sendRedirect().

Exercise: Build a Multi-Servlet Application

Objective: Build a simple multi-servlet application that tracks a user's name and age across multiple requests and pages.

Steps:

1. Create two servlets:
    - One to collect the user's name (NameServlet).
    - One to collect the user's age (AgeServlet).
2. Store both the name and age in session attributes.
3. Create a final servlet (SummaryServlet) that retrieves both the name and age and displays them to the user.

Expected Outcome: A multi-page form where the user enters their name on one page, their age on another, and then sees a summary of their data on the final page.

Additional Resources

- **Java EE Servlet Documentation**: Java Servlet API Documentation
- **Session Management in Servlets**: Session Handling in Java

Recommended Books

- Marty Hall and Larry Brown, *Core Servlets and JavaServer Pages (JSP)*, Sun Microsystems Press.
- Jason Hunter, *Java Servlet Programming*, O'Reilly Media.