State Management

Sandeep, SunBeam

PowerPoint

Introduction

- HTTP protocol and Web Servers are stateless, what it means is that, for web server every request is a new request to process and they can’t identify if it’s coming from client that has been sending request previously.

- State management techniques.

    1. Client side state management technique

    2. Server side state management technique

Client side state management

- State of the client is stored on client machine.
- Less memory is needed at server side.
- Since state is maintained at client side, it is less secure.
- Following are the client side state management techniques:
    1. Cookie
    2. Query string
    3. Hidden form field

Server side state management

- State of the client is stored on server machine.
- more memory is needed at server side.
- Since state is maintained at server side, it is more secure.
- Following are the server side state management techniques:
    - 1. session
    - 2. application

Cookies

- A cookie is a small amount of information sent by a Server (Servlet/JSP) to a Web browser, saved by the browser, and later sent back to the server.
- Cookies were originally developed by Netscape Communications.
- Cookies are currently standardized by the Internet Engineering Task Force(IETF)
- Cookie represents data shared across multiple dynamic pages from the same web application.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

Cookie API

- Cookie is a class which is declared in **javax.servlet.http** package.
- Methods declared in Cookie class:
  1. public String **getValue**()
  2. public void **setValue**(String newValue)
  3. public int **getMaxAge**()
  4. public void **setMaxAge**(int expiry)

How to create and add cookie in response header?

- **Cookie Constructor**
    - public Cookie(String name, String value)
- **Example:**

```
Cookie cookie = new Cookie("UserName", "sandeep");

response.addCookie(cookie);
```

How to get the cookies from request header?

```
String userName = "";
Cookie[] cookies = request.getCookies();
if( cookies != null ){
        for (Cookie cookie : cookies){
                if( cookie.getName().equals("UserName")){
                        userName = cookie.getValue();
                break;
                }
        }
}
```

Cookie Types

- Depending on the cookie age there are two type of cookie:
  1. Persistent cookie:
     - It is stored on your computer hard disk. It stays on hard disk and can be accessed by web servers until they are deleted or have expired.
     - use "setMaxAge( )" method with positive value to create persistent cookie.
  2. Non-persistent cookie:
     - It is stored in web browser's memory. They can be used by a web server only until we close the browser.
     - Use "setMaxAge( )" method with negative value to create persistent cookie.

Cookie Deletion

- If we want to delete cookie then we should set age of the cookie to 0.

```
cookie.setMaxAge( 0 );
response.addCookie(cookie);
```

- Browser may get cookies with same name but their path attribute can be different.

- The browser is expected to support 20 cookies for each Web server, 300 cookies total, and may limit cookie size to 4 KB each.

Disadvantages of cookie

1. Cookies can handle only text data. We can not store state of java instance or binary data in cookie.

2. If number of cookies are increased then it may increase network traffic.

3. Entire state of the client is saved at client side. If the client browser rejects the cookies then state will be lost.

Query string

- It is client side state management technique.
- It passes request parameters / state of the client with URL.
- **Syntax :** URL?key1=value1&key2=value2
- **Advantages:**
    1. It always work whether cookie is enabled or disabled.
    2. Form submission is not required.
- **Disadvantages:**
    1. It works with only link.
    2. It can send only textual information.
    3. Some browsers put restrictions on length of the URL hence we can not send large data with URL.

Hidden form fields

- It is hidden/invisible text field that is used to maintain state of the client.
- **Example :** *<input type="hidden" name="UserName" value="Sandeep"/>*
- **Advantage:**
    1. It always work whether cookie is enabled or disabled.
- **Disadvantages:**
    1. It is used to store only textual information.
    2. Extra form submission is required.
    3. It makes client thick.

Session

- Session is a conversational state between client and server and it can consists of multiple request and response between client and server.
- If we want to save state of the client at server side then we should HTTPSession object.
- **Methods of HTTPSession interface:**

  1. Object **getAttribute**(String name)
  2. void **setAttribute**(String name,Object value)
  3. void **removeAttribute**(String name)
  4. boolean **isNew**()
  5. void **invalidate**()
  6. String **getId**()

Session

- This interface allows servlets to:
    - View and manipulate information about a session, such as the session identifier, creation time, and last accessed time.
    - Bind objects to sessions, allowing user information to persist across multiple user connections.
- To get the reference of HttpSession instance we should use methods declared in HttpServletRequest interface.
    1. HttpSession getSession()
        - Returns the current session associated with this request, or if the request does not have a session, creates one.
    2. HttpSession getSession(boolean create)
        - Returns the current HttpSession associated with this request.
        - If there is no current session and create is true, returns a new session.
        - If there is no current session and create is false and the request has no valid HttpSession, this method returns null.

Working With Session

- **How to bind object to session?**

```
HttpSession session = request.getSession();
session.setAttribute("joinDate", new Date(107, 0, 1));
```

- **How to get the object from session?**

```
Date date = (Date) session.getAttribute("joinDate");
```

- **How to remove the object from session?**

```
session.removeAttribute("joinDate");
```

Session Time-out

- All HttpSession objects have a finite life.
- A session either time-outs when it has been inactive for a certain period of time, or a session can also be invalidated by the application.
- An application can be told to invalidate a session at any time by invoking the HttpSession invalidate() method.
- If a session is not explicitly invalidated, it will automatically be invalidated after a given period of time. The period of time a session can be idle before it is timed-out can be set via web.xml.

```
    <session-config>
            <session-timeout>15</session-timeout>
    </session-config>
```

- If an integer value is specified in the body of the session-timeout element, it will be treated as the number of minutes a Web Application should save session information about a client.
- A negative number or 0 value number means indefinitely.

Session Tracking

- When clients at an online store add an item to their shopping carts, how does the server know what's already in the carts? Similarly, when clients decide to proceed to checkout, how can the server determine which previously created shopping carts are theirs?

- There are three typical solutions to this problem:

  1. Cookies (Part of Servlet Spec.)
  2. URL rewriting and (Part of Servlet Spec.)
  3. Hidden form fields. (Not a Part of Servlet Spec.)

Session tracking using cookie

- When we use “**HttpServletRequest getSession()**” method and it creates a new request, it creates the new HttpSession object and also add a Cookie to the response object with name JSESSIONID and value as session id.

- For example, on the initial request a servlet could do something like the following:

```
String sessionID = makeUniqueString();
HashMap sessionInfo = new HashMap();
HashMap globalTable = findTableStoringSessions();
globalTable.put(sessionID, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sessionID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

- This cookie is used to identify the HttpSession object in further requests from client.

Session tracking using URL Rewriting

- We can manage a session with HttpSession but if we disable the cookies in browser then it won't work because server will not receive the JSESSIONID cookie from client.
- URL rewriting is a moderately good solution for session tracking and even has the advantage that it works when browsers don't support cookies or when the user has disabled them.
- With this approach, browser appends some JSESSIONID at the end of URL.
- The HttpServletResponse object provides the following methods for rewriting a URL.

  1. **String encodeURL(String url)**
  2. **String encodeRedirectURL(String url)**

Application

- It is an instance of ServletContext()
- If we want to add anything global then we should use application object.
- **How to get the reference of application object?**

```
ServletContext application = this.getServletContext();
```

- **How to bind object to application?**

```
application.setAttribute("joinDate", new Date(107, 0, 1));
```

- **How to get the object from application?**

```
Date date = (Date) application.getAttribute("joinDate");
```

Thank you