Managing Session in Servlets

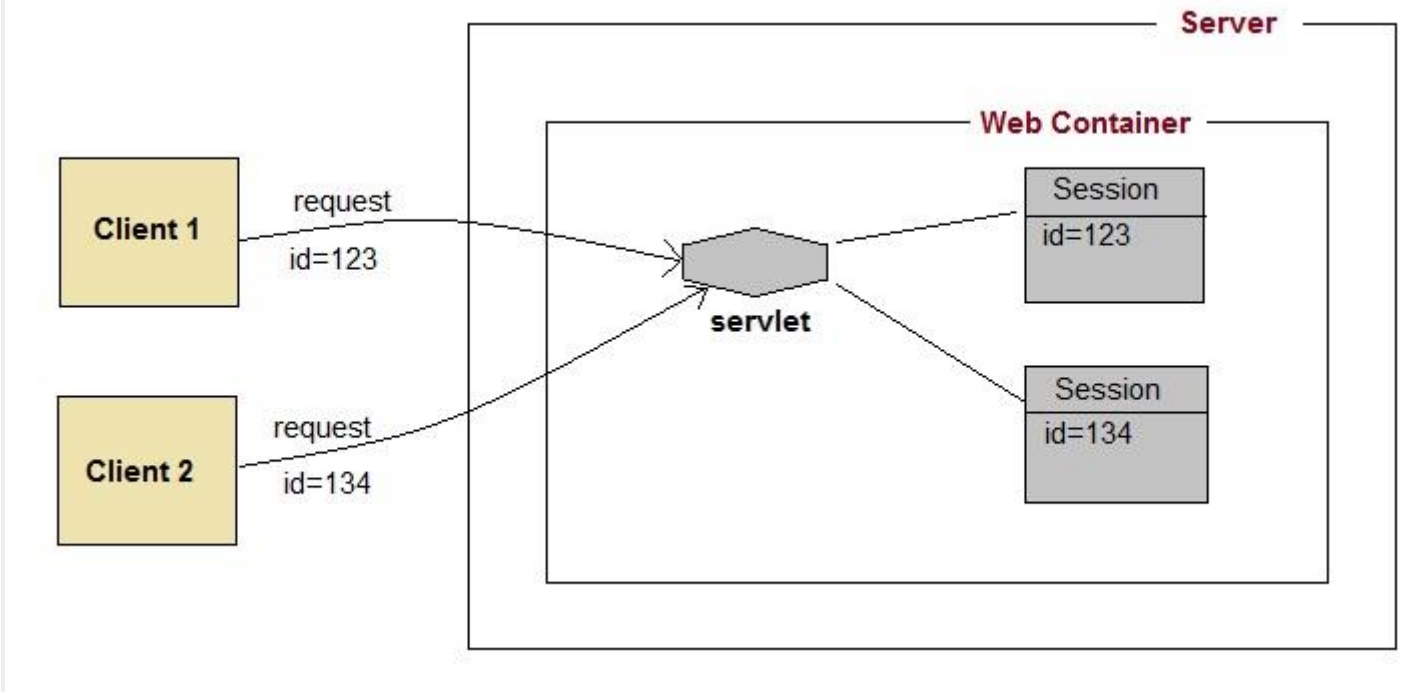
We all know that HTTP is a stateless protocol. All requests and responses are independent. But sometimes you need to keep track of client's activity across multiple requests. For eg. When a User logs into your website, not matter on which web page he visits after logging in, his credentials will be with the server, until he logs out. So this is managed by creating a session.

Session Management is a mechanism used by the Web container to store session information for a particular user. There are four different techniques used by Servlet application for session management. They are as follows:

1. **Cookies**
2. **Hidden form field**
3. **URL Rewriting**
4. **HttpSession**

Session is used to store everything that we can get from the client from all the requests the client makes.

How Session Works

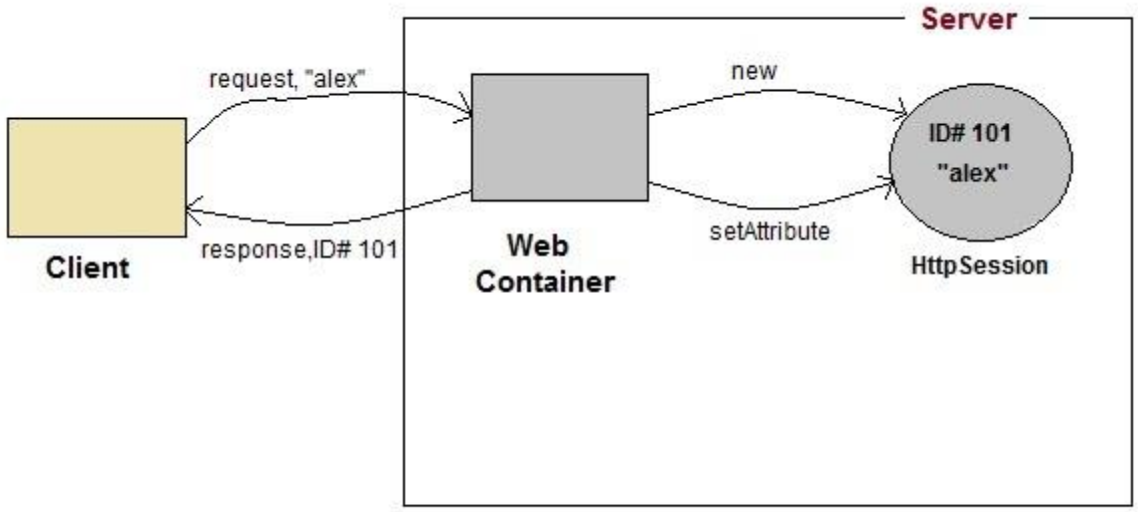


The basic concept behind session is, whenever a user starts using our application, we can save a unique identification information about him, in an object which is available throughout the application, until its destroyed. So wherever the user goes, we will always have his information and we can always manage which user is doing what. Whenever a user wants to exit from your application, destroy the object with his information.

What is HttpSession?

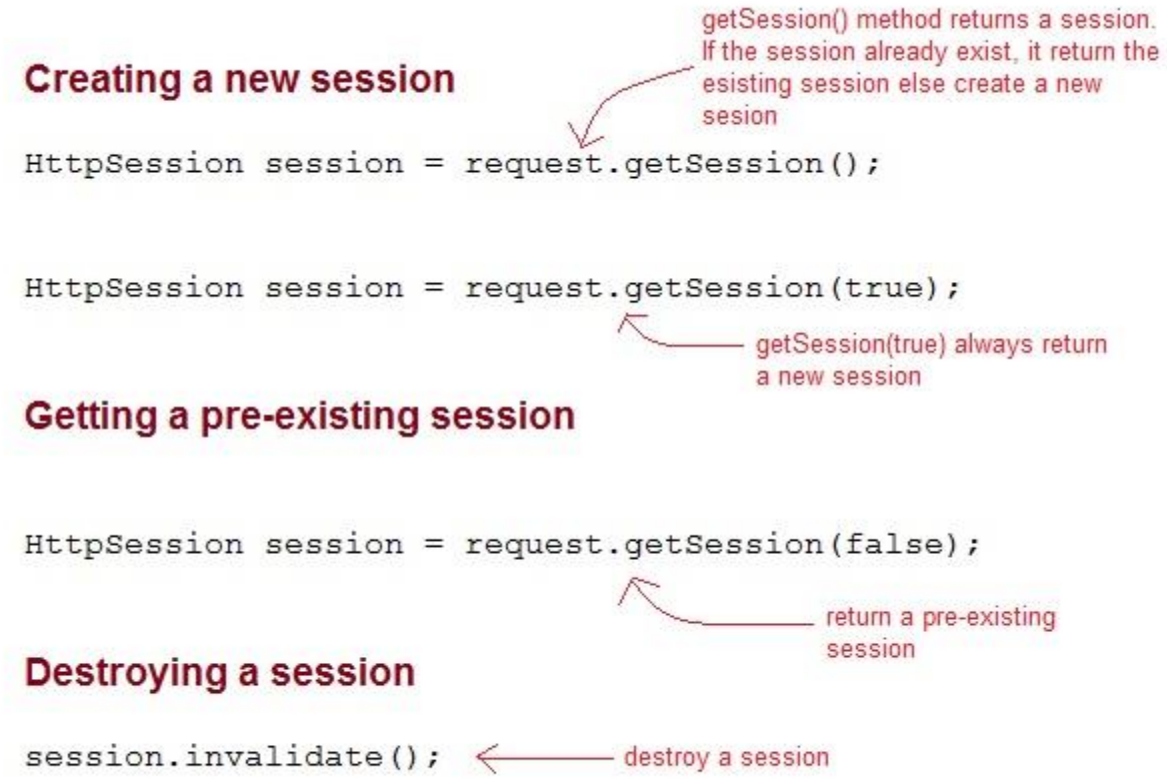
HttpSession object is used to store entire session with a specific client. We can store, retrieve and remove attribute from **HttpSession** object. Any servlet can have access to **HttpSession** object throughout the `getSession()` method of the **HttpServletRequest** object.

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.

HttpSession Interface



Some Important Methods of HttpSession

Methods	Description
long `getCreationTime()`	returns the time when the session was created, measured in milliseconds since midnight January 1, 1970 GMT.
String `getId()`	returns a string containing the unique identifier assigned to the session.
long `getLastAccessedTime()`	returns the last time the client sent a request associated with the session

int `getMaxInactiveInterval()`	returns the maximum time interval, in seconds.
void `invalidate()`	destroy the session
boolean `isNew()`	returns true if the session is new else false
void `setMaxInactiveInterval(int interval)`	Specifies the time, in seconds,after servlet container will invalidate the session.

Using Cookies for Session Management

Cookies are small pieces of information that are sent in response from the web server to the client. **Cookies**are the simplest technique used for storing client state.

Cookies are stored on client's computer. They have a lifespan and are destroyed by the client browser at the end of that lifespan.

Using Cookies for storing client state has one shortcoming though, if the client has turned of COokie saving settings in his browser then, client state can never be saved because the browser will not allow the application to store cookies.

Cookies API

Cookies are created using **Cookie** class present in Servlet API. Cookies are added to **response** object using the `addCookie()` method. This method sends cookie information over the HTTP response stream. `getCookies()` method is used to access the cookies that are added to response object.

Creating a new Cookie

```
Cookie ck = new Cookie("username",name);
```

creating a new cookie object

Setting up lifespan for a cookie

```
ck.setMaxAge(30*60);
```

setting maximum age of cookie

Sending the cookie to the client

```
response.addCookie(ck);
```

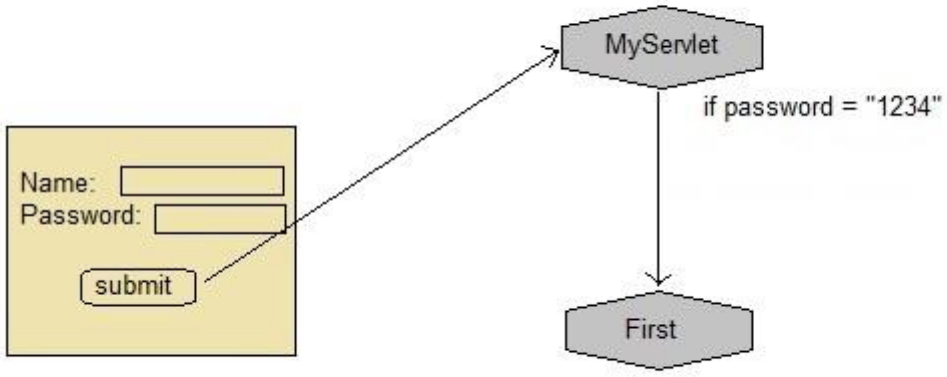
adding cookie to **response** object

Getting cookies from client request

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

getting the cookie for **request** object

Example demonstrating usage of Cookies

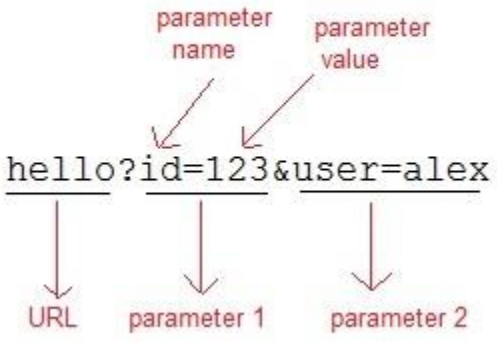

MyServlet
if password = "1234"
Name:
Password:
submit
First

Using URL Rewriting for Session Management

If the client has disabled cookies in the browser then session management using cookie wont work. In that case **URL Rewriting** can be used as a backup. **URL rewriting** will always work.

In URL rewriting, a token(parameter) is added at the end of the URL. The token consist of name/value pair seperated by an `equal(=)` sign.

For Example:



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.

Using Hidden Form Field for Session Management

Hidden form field can also be used to store session information for a particular client. In case of hidden form field a hidden field is used to store client state. In this case user information is stored in hidden field value and retrieved from another servlet.

Advantages :

- Does not have to depend on browser whether the cookie is disabled or not.
- Inserting a simple HTML Input field of type hidden is required. Hence, its easier to implement.

Disadvantage :

- Extra form submission is required on every page. This is a big overhead.

Example demonstrating usage of Hidden Form Field for Session