**Session Management in Java**

This article is aimed to explain about session management in servlets using different techniques and with example programs.

1. [What is a Session?](https://www.journaldev.com/1907/java-session-management-servlet-httpsession-url-rewriting#servlet-session)
2. [Session Management in Java – Cookies](https://www.journaldev.com/1907/java-session-management-servlet-httpsession-url-rewriting#servlet-cookies)
3. [Session in Java Servlet – HttpSession](https://www.journaldev.com/1907/java-session-management-servlet-httpsession-url-rewriting#servlet-httpsession)
4. [Session Management in Java Servlet – URL Rewriting](https://www.journaldev.com/1907/java-session-management-servlet-httpsession-url-rewriting#servlet-url-rewriting)
5. **What is a Session?**

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.

But sometimes in web applications, we should know who the client is and process the request accordingly. For example, a shopping cart application should know who is sending the request to add an item and in which cart the item has to be added or who is sending checkout request so that it can charge the amount to correct client.

**Session** is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

There are several ways through which we can provide unique identifier in request and response.

* 1. **User Authentication** – This is the very common way where we user can provide authentication credentials from the login page and then we can pass the authentication information between server and client to maintain the session. This is not very effective method because it wont work if the same user is logged in from different browsers.
  2. **HTML Hidden Field** – We can create a unique hidden field in the HTML and when user starts navigating, we can set its value unique to the user and keep track of the session. This method can’t be used with links because it needs the form to be submitted every time request is made from client to server with the hidden field. Also it’s not secure because we can get the hidden field value from the HTML source and use it to hack the session.
  3. **URL Rewriting** – We can append a session identifier parameter with every request and response to keep track of the session. This is very tedious because we need to keep track of this parameter in every response and make sure it’s not clashing with other parameters.
  4. **Cookies** – Cookies are small piece of information that is sent by web server in response header and gets stored in the browser cookies. When client make further request, it adds the cookie to the request header and we can utilize it to keep track of the session. We can maintain a session with cookies but if the client disables the cookies, then it won’t work.
  5. **Session Management API** – Session Management API is built on top of above methods for session tracking. Some of the major disadvantages of all the above methods are:
     + Most of the time we don’t want to only track the session, we have to store some data into the session that we can use in future requests. This will require a lot of effort if we try to implement this.
     + All the above methods are not complete in themselves, all of them won’t work in a particular scenario. So we need a solution that can utilize these methods of session tracking to provide session management in all cases.

That’s why we need **Session Management API** and J2EE Servlet technology comes with session management API that we can use.