Session variable State variable and Cookies

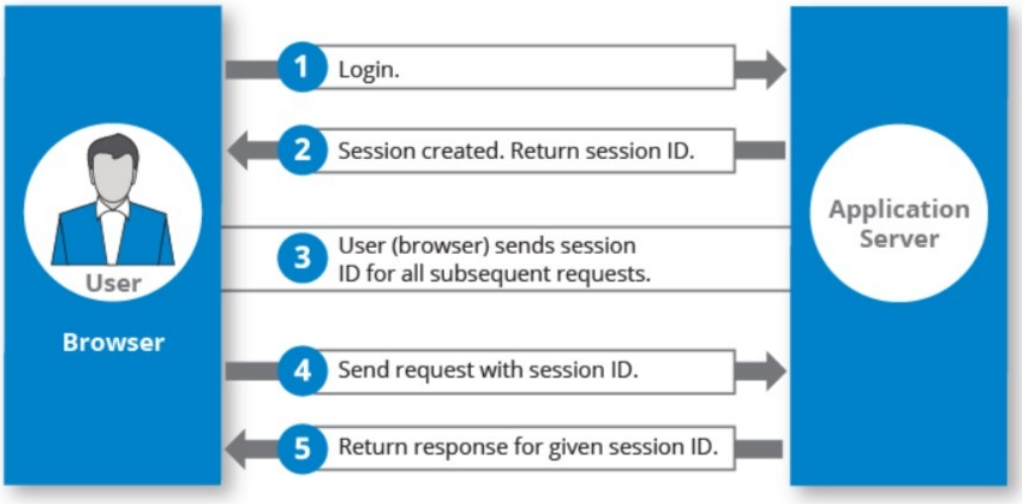
Session:

Once we login in a system we will stay on the page or system until we logout or we shout down the browser or we manually clear the HTTP authentication.

There is option “remember me” using this option we can login in a site only in one click. Together Session and cookies store user data persistently. Which means they ensure that a user connected with the same server again and again whenever they request within a boundary of single session.

Session id are create and destroy continuously because of security reason.each user gets a unique session ID which send to the server for validation after validation the user can enter the web application.
HTTP is a stateless protocol which means each request is independent.the server don’t know anything about the previous made request.

A web session is a series of contiguous action perform by a visitor on a website during a time frame.this could be search, enter and retrieve data from the web site, scrolling the pages. Clicking on the links etc. Any interaction perform on the web site is recorded as a session to that website property.



To track session. A web session Id is stored in the user web browser. And that session Id is sent with every HTTP request. Session ID is created on server.

You can store sensitive information on a session because it is kept on the server, but be aware that the session ID can still be stolen if the user, let's say, logged in over an insecure WiFi. (An attacker can sniff the cookies, and set it as its own, he won't see the variables themselves, but the server will identify the attacker as the user).

How Session Works

Imagine a website containing two pages: **main** and **water**. Suppose a visitor sees **main** first and then moves on to **water**. HTTP is a stateless protocol. So, if a typical web server is managing this site, any knowledge gathered at **main** is lost when the visitor browses over to **water**. In other words, **water** cannot take advantage of any information that the visitor might have provided at **main**.

To get around this limitation, application servers detect when a visitor first enters a website. At that point, the application server starts a **session** for this visitor. In the preceding example, when the visitor requests the **main** page, the application server starts a session. The website designer can use **main** to gather information about the visitor and store that information in **session variables**. The information in session variables is available to all subsequent pages. So, for example, if Bob provides his age to **main**, and **main**'s designer wrote the age to a session variable, then **water** could easily access Bob's age.

Session ID:

Session ID is generate by the server. Client machine use cookies to store a session ID. And session ID is used by the server to uniquely identify a request from a client. Using session ID server distinguish a user.session ID is randomly generated by the ASP.NET and store in cookies of the browser.

Session Variable

Session variable contain values available for the duration of the session. When the session ends, the application server destroys the session variables associated with that session. Each session variable consumes memory on the application server, so creating unnecessary session variables can hurt performance.

Session Variables

Session Variable	What it Holds
`SessionVariables.currentUser`	The `id` of the visitor logged in.
`SessionVariables.currentAcl`	The comma-separated list of all ACLs to which this visitor belongs. If the visitor has not explicitly logged in, the default ACL is `Browser`.
`SessionVariables.username`	The username under which this visitor is logged in. If the visitor has not explicitly logged in, the default username is `DefaultReader`.
`SessionVariables.iniFile`	The name of the file containing WebCenter Sites properties.

Logging In and Logging Out

When a visitor first hits the site, WebCenter Sites creates a session and implicitly logs in the visitor as **DefaultReader**. During the session, if the visitor explicitly logs in, WebCenter Sites automatically updates the values of **SessionVariables.currentUser, SessionVariables.currentAcl**, and **SessionVariables.username**. Logging in does not affect the values of any other session variables. In other words, if your pages create session variables prior to a login, then those values are still valid after the login. When a visitor explicitly logs out, the WebCenter Sites-generated session variables automatically revert to the values they held prior to login. For example, consider the following sequence:

> A visitor first hits a page, so the value of **SessionVariables.username** is **DefaultReader**.
>
> The visitor logs in as marilyn, so the value of **SessionVariables.username** is **marilyn**.
>
> If marilyn logs out, the value of **SessionVariables.username** reverts to **DefaultReader**.

To trigger a logout, you call the **<CATALOGMANAGER>** tag with the **ftcmd=logout** modifier. When issuing this tag, you can optionally supply the **killsession** modifier, which destroys the current session. You can then create a new session by invoking the **<CATALOGMANAGER>** tag with the **ftcmd=login** modifier.

Cookies:

Cookies are used to store session ID and store in client side of the browser.it is store in small text file format. page requests that follow return the cookie name and value. A cookie can only be read from the domain that it has been issued from. For example, a cookie set using the domain www.guru99.com cannot be read from the domain

A **cookie** is a string that your application writes to the visitor's browser. A cookie stores information about visitors that lasts between sessions. The visitor's browser writes this string to a special cookie file on the visitor's disk. When that visitor returns to your website, the visitor's browser sends a copy of the cookie back to the web server that set it. Once a cookie has been created, it is available as a variable to elements on a page.

For example, your application might store the visitor's favorite sports team in a cookie. Then, when the visitor returns, your application could retrieve the cookie and use its information to display the team logo in a banner.

When cookies are no longer needed, you can delete them

Cookie Tags

Tag	Use
`satellite.cookie`	Sets a cookie on the client's browser.
`REMOVECOOKIE`	Deletes a cookie from the client's browser.

Cookie Attributes

Attribute	Value
`name`	Name of the cookie. This also serves as the name of the incoming variable containing the value of the cookie. **Important:** Cookies in the WebCenter Sites page context are treated as variables. Therefore, when a cookie and an asset attribute share the same name, they are treated as the same variable.
`expiration`	Time in seconds after which the cookie no longer is sent to the web server.
`security`	Optionally set security on the cookie.
`URL`	Restrict that the cookie only be sent on this URL
`Domain`	Restrict that the cookie only be sent to URLs in the specified domain.