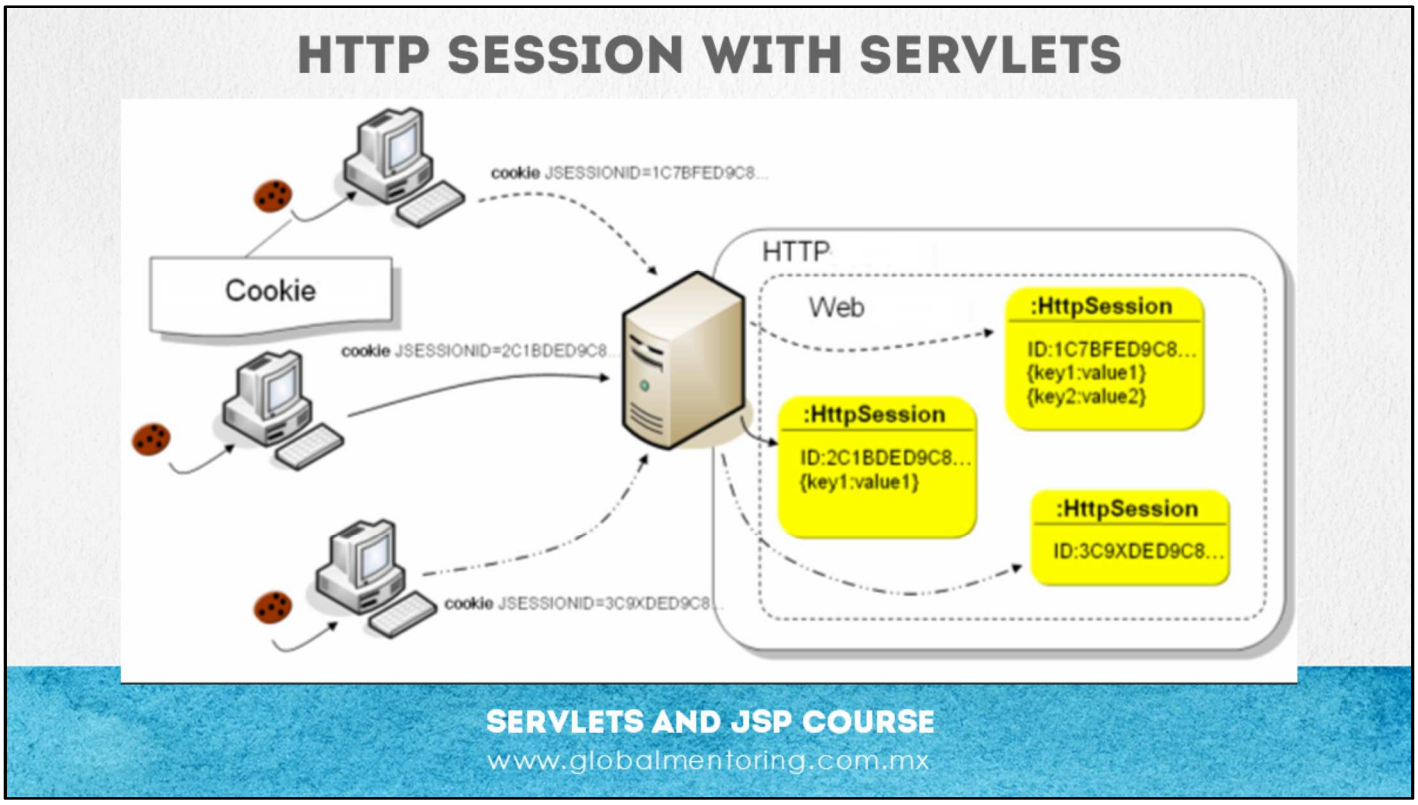
Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson ..

We will study the topic of Session Management with the object HttpSession.

Are you ready? Come on!



Let's now review the concept of HttpSession.

The Servlets API allows us to manage customer sessions through an object called HttpSession and it should be noted that this happens automatically, every request that a client makes to our web resource automatically creates a new session for example if we have two different browsers on the same PC, one using Internet Explorer and another using Firefox are considered different clients because it is different web browser and each client creates a new session on the web server, but if we have a new browser window web for example if we open a new Firefox window this is not considered a new client and therefore will not create a new session on the server side.

Now a session will allow us to manage the requests made by a user, we must remember that the HTTP protocol is considered to be a protocol that has no status, so it will NOT remember information that the user has previously sent.

The HttpSession object is obtained from the HttpServletRequest (request) object. So a session is used to manage the different requests (request) of the user.

A session then has a duration of life longer than a request or what is the same thing the request object. Therefore a session is destroyed until the time that we assign to the inactivity session elapses or we can also destroy it manually by means of the invalidate method. More ahead we will see an example of this subject.

We are now going to study the Servlets API to handle the concept of sessions.

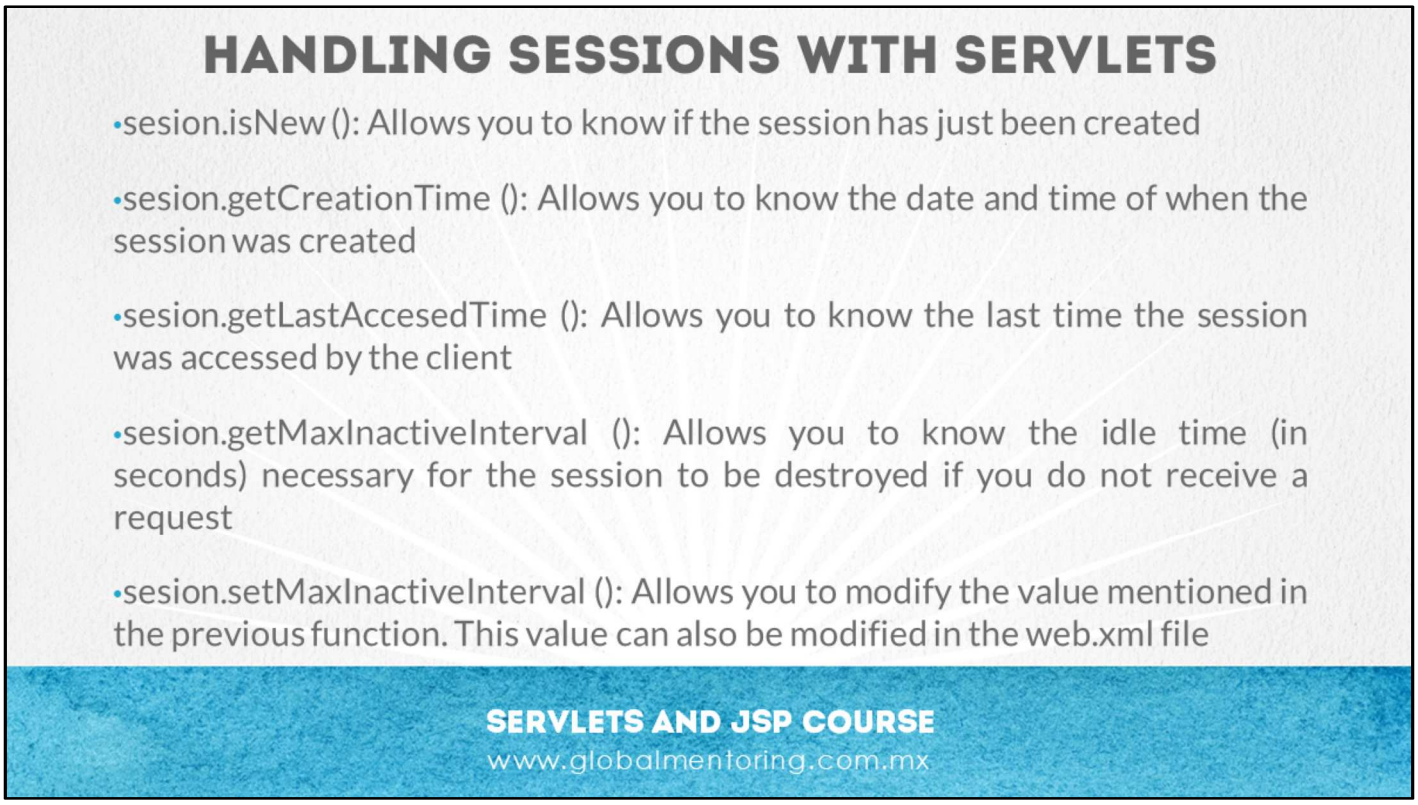
In the first instance, in order to obtain a session, we will use the object HTTPServletRequest and through this object we will create a session with the following method: reques.getSession ().

The created session is associated with the client that generated the request. We are going to review several more methods. Once we have the session object, the methods that we will most use are:

getAttribute (): This method allows us to obtain an attribute previously added to the client's session.

setAttribute (): This method is also housed within the HttSession object. This method setAttribute () allows to add new information of the client in the web session and as well as we can recover and add new attributes also the API of the Servlets allows us to remove the attributes that we are no longer using in an HTTP session.

sesion.invalidate (): This method allows us to delete the client's current session and will also delete any attribute that we have previously added through the serAttribute () method.



Other important methods for example the method sesion.isNew (): this method allows us to know if the session was recently created this is in the first request of the client if the client makes a second request to the web resource this session has already been created previously and therefore this method isNew () would return a false, however if it is the first request this method returns true.

We also have the sesion.get.CreationAccesedTime () method and this method allows us to know what time this HTTP session was created

We also have the sesion.get.LastAccesedTime () method, this method allows us to know once when this session was accessed by the client.

Later we have the method sesion.getMaxInactiveInterval () this method allows us to know the idle time with which this session has been defined and this is measured in seconds, so, if this method returns for example a value of 3600 seconds, it means that this session will wait an hour until it is destroyed if it does not receive any request from the client, then if an hour of inactivity passes by the user then the current session will be deleted and the user will lose all the information that was stored in this HTTP session.

It should be noted that this value can be modified using the setMaxInactiveInterval () method and it will allow us to specify the inactivity time that we will allow to pass if the user does not make an HTTP request. Now this inactivity value can also be configured from the web.xml file that we will see later also if we configure it within the web.xml file it is said to be a declarative configuration because we are not adding Java code to modify that value but we are simply declaring the value in the web.xml file. On the other hand, if we do it using the setMaxInactiveInterval () method, we say that we are modifying the value programmatically because we are using Java code to modify the value of that attribute, we will review some exercises below to review the API of Servlet sessions.