# Servlets - seguimiento de la sesión

HTTP es un protocolo "sin estado", lo que significa que cada vez que un cliente recupera una página Web, el cliente abre una conexión independiente para el servidor Web y el servidor de forma automática no mantiene ningún registro de la petición del cliente anterior.

Aún existen las siguientes tres maneras de mantener sesiones entre el cliente web y el servidor web:

## Cookies:

Un servidor web puede asignar un ID de sesión único como una galleta a cada cliente web y las solicitudes posteriores del cliente pueden ser reconocidos mediante la cookie de recibido.

Esto no puede ser una manera eficaz porque muchos navegador momento no admite una cookie, por lo que no recomendaría utilizar este procedimiento para mantener las sesiones.

## Campos ocultos de formulario:

Un servidor web puede enviar a un campo de formulario HTML oculta junto con un ID de sesión único de la siguiente manera:

<input type="hidden" name="sessionid" value="12345">

Esta entrada significa que, cuando se envía el formulario, el nombre y el valor especificado se incluyen automáticamente en los datos GET o POST. Cada vez cuando navegador de web envía la solicitud de nuevo, entonces el valor session\_id puede ser usado para mantener la pista de diferentes navegadores web.

Esto podría ser una manera eficaz de hacer el seguimiento de la sesión, pero al hacer clic en un enlace regular (HREF...> <A) hipertexto no da lugar a un envío de formulario, los campos de formulario ocultos así tampoco puede respaldar seguimiento de la sesión general.

## Reescritura de URL:

Usted puede añadir algunos datos adicionales sobre el final de cada URL que identifica la sesión, y el servidor puede asociar ese identificador de sesión con los datos que ha almacenado sobre ese período de sesiones.

Por ejemplo, con http://tutorialspoint.com/file.htm; sessionid = 12345, el identificador de sesión se adjunta como sessionid = 12345 que se puede acceder en el servidor web para identificar al cliente.

Reescritura de URL es una mejor manera de mantener las sesiones y trabajos para los navegadores cuando no admiten cookies, pero aquí inconveniente es que habría generar cada URL dinámicamente para asignar un identificador de sesión si la página es simple página HTML estática.

## El objeto HttpSession:

Aparte de los mencionados tres maneras, servlet proporciona Interface HttpSession que proporciona una manera de identificar a un usuario a través de más de una solicitud de página o visita a un sitio Web y para almacenar información sobre ese usuario.

El contenedor de servlets utiliza esta interfaz para crear una sesión entre un cliente HTTP y un servidor HTTP. La sesión continúa por un período de tiempo especificado, a través de más de una conexión o de solicitud de la página del usuario.

Se podría obtener el objeto HttpSession llamando al método **getSession** público **()** de HttpServletRequest, de la siguiente manera:

HttpSession session = request.getSession ();

Tienes que llamar *request.getSession ()* antes de enviar cualquier contenido del documento al cliente. He aquí un resumen de los métodos más importantes disponibles a través del objeto HttpSession:

|  |  |
| --- | --- |
| **SN** | **Método y Descripción** |
| 1 | **getAttribute public Object (String nombre)**  Este método devuelve el objeto de destino con el nombre especificado en esta sesión, o null si no hay ningún objeto está vinculado con el nombre. |
| 2 | **getAttributeNames Enumeración pública ()**  Este método devuelve una enumeración de objetos String que contiene los nombres de todos los objetos vinculados a esta sesión. |
| 3 | **GetCreationTime mucho público ()**  Este método devuelve la hora en que se creó esta sesión, medido en milisegundos desde la medianoche del 1 de enero 1970 GMT. |
| 4 | **getId public String ()**  Este método devuelve una cadena que contiene el identificador único asignado a esta sesión. |
| 5 | **getLastAccessedTime mucho público ()**  Este método devuelve la última vez que el cliente envía una petición asociada a esta sesión, como el número de milisegundos desde la medianoche del 1 de enero 1970 GMT. |
| 6 | **getMaxInactiveInterval public int ()**  Este método devuelve el intervalo de tiempo máximo, en segundos, que el contenedor de servlets mantendrá esta sesión abierta entre los accesos de clientes. |
| 7 | **public void invalidate ()**  Este método invalida esta sesión y desenlaza todos los objetos vinculados a la misma. |
| 8 | **public boolean isNew (**  Este método devuelve true si el cliente no sabe todavía acerca de la sesión o si el cliente decide no unirse a la sesión. |
| 9 | **removeAttribute public void (String nombre)**  Este método elimina el objeto vinculado con el nombre especificado de esta sesión. |
| 10 | **setAttribute public void (String name, Object value)**  Este método se une un objeto de la presente sesión, utilizando el nombre especificado. |
| 11 | **void setMaxInactiveInterval público (int interval)**  Este método especifica el tiempo, en segundos, entre las peticiones de los clientes antes de que el contenedor de servlets se anula esta sesión. |

En este ejemplo se describe cómo utilizar el objeto HttpSession para saber la hora de creación y la última vez que se ha accedido durante una sesión. Queremos asociar una nueva sesión con la petición si no existe ya.

/ / Importar requiere java bibliotecas import java.io. \*; javax.servlet import \*,. ​​Importaciones javax.servlet.http \*;. Import java.util \*,. / / Extender clase HttpServlet SessionTrack clase pública se extiende HttpServlet {public void doGet (HttpServletRequest request, response HttpServletResponse) throws ServletException, IOException {/ / Crear un objeto de sesión si ya no se crea. HttpSession session = request.getSession (true); / / Obtener el tiempo de creación de sesiones. Fecha createTime = new Date (session.getCreationTime ()) / / Obtener última vez el acceso de esta página web. Fecha LastAccessTime = new Date (session.getLastAccessedTime ()); título String = "Bienvenido de nuevo a mi sitio web"; Integer visitCount = new Integer (0) Cadena visitCountKey = new String ("visitCount"); userIdKey cadena = new String ( "ID de usuario"), ID de usuario String = new String ("ABCD") / / Comprobar si esto es recién llegado en su página web. if (session.isNew ()) {title = "Bienvenido a mi sitio web"; session.setAttribute (userIdKey, ID de usuario);} else {visitCount = (Integer) session.getAttribute (visitCountKey); visitCount = visitCount + 1, ID de usuario = (String) session.getAttribute (userIdKey);} session.setAttribute (visitCountKey, visitCount) / / Set respuesta response.setContentType tipo de contenido ("text / html"); PrintWriter cabo response.getWriter = (); doctype Cadena = " \ n - <DOCTYPE HTML public \ "/ / W3C / / DTD HTML 4.0" + "de transición / / es \"> "; out.println (docType +" \ n <head> <html> "+" <título > "+ titulo +" </ title> </ head> \ n "+" <body bgcolor=\"#f0f0f0\"> \ n "+" <h1 align=\"center\"> "+ titulo +" </ h1> \ n "+" <h2 align=\"center\"> Sesión Infomation </ h2> \ n "+" <table border=\"1\" align=\"center\"> \ n " + "<tr bgcolor=\"#949494\"> \ n" + "<th> Session info </ th> <th> valor </ th> </ tr> \ n" + "<tr> \ n" + "<td> id </ td> \ n" + "<td>" session.getId + () + "</ td> </ tr> \ n" + "<tr> \ n" + "<td > Hora de creación </ td> \ n "+" <td> "createTime + +" </ td> </ tr> \ n "+" <tr> \ n "+" El tiempo de <td> Ultimo acceso </ td> \ n "+" <td> "LastAccessTime + +" </ td> </ tr> \ n "+" <tr> \ n "+" <td> ID de usuario </ td> \ n "+" <td> "+ ID de usuario +" </ td> </ tr> \ n "+" <tr> \ n "+" <td> número de visitas </ td> \ n "+" <td> "+ visitCount + "</ td> </ tr> \ n" + "</ table> \ n" + "</ body> </ html>");}}

Compilar anterior **SessionTrack** servlet y crear la entrada correspondiente en el archivo web.xml. Ahora ejecuta *http://localhost:8080/SessionTrack* le mostrará el siguiente resultado cuando se debe ejecutar por primera vez:

**Bienvenidos a mi sitio web**

**Infomation Sesión**

|  |  |
| --- | --- |
| **Session info** | **valor** |
| identificación | 0AE3EC93FF44E3C525B4351B77ABB2D5 |
| Hora de creación | Mar Jun 08 17:26:40 GMT +04:00 2010 |
| Hora del último acceso | Mar Jun 08 17:26:40 GMT +04:00 2010 |
| ID de usuario | ABCD |
| Número de visitas | 0 |

Ahora intenta ejecutar el mismo servlet por la segunda vez, mostraría siguiente resultado.

**Bienvenidos de nuevo a mi sitio web**

**Infomation Sesión**

|  |  |
| --- | --- |
| **Tipo info** | **valor** |
| identificación | 0AE3EC93FF44E3C525B4351B77ABB2D5 |
| Hora de creación | Mar Jun 08 17:26:40 GMT +04:00 2010 |
| Hora del último acceso | Mar Jun 08 17:26:40 GMT +04:00 2010 |
| ID de usuario | ABCD |
| Número de visitas | 1 |

**Eliminación de datos de sesión:**

Cuando haya terminado con los datos de sesión de un usuario, usted tiene varias opciones:

* **Eliminar un atributo particular:** Puede llamar al método *removeAttribute public void (String nombre)* para eliminar el valor asociado a una clave particular.
* **Eliminar toda la sesión:** Se puede llamar *public void invalida ()* para descartar una sesión entera.
* **Configuración de tiempo de espera de la sesión:** Se puede llamar al método *setMaxInactiveInterval public void (int intervalo)* para establecer el tiempo de espera para una sesión individual.
* **Entrar al usuario out:** Los servidores que soporte servlets 2.4, puede llamar al **cierre de sesión** para iniciar el cliente de un servidor Web e invalidar todas las sesiones que pertenecen a todos los usuarios.
* **configuración web.xml:** Si está utilizando Tomcat, además de los métodos mencionados anteriormente, se puede configurar el tiempo de sesión en el archivo web.xml de la siguiente manera.

<session-config>

<session-timeout> 15 </ session-timeout>

</ Session-config>

El tiempo de espera se expresa en minutos, y anula el tiempo de espera predeterminado, que está a 30 minutos en Tomcat.

El método getMaxInactiveInterval () en un servlet devuelve el tiempo de espera para la sesión en cuestión de segundos. Así que si la sesión se configura en web.xml durante 15 minutos, getMaxInactiveInterval () devuelve 900.