Servlets y el seguimiento de sesiones

El seguimiento de sesión es un mecanismo que los servlets utilizan para mantener el estado sobre la serie de peticiones desde un mismo usuario (esto es, peticiones originadas desde el mismo navegador) durante un periodo de tiempo.

Las sesiones son compartidas por los servlets a los que accede el cliente. Esto es conveniente para aplicaciones compuestas por varios servlets. Para utilizar el seguimiento de sesión debemos:

- Obtener una sesión (un objeto **HttpSession**) para un usuario.
- Almacenar u obtener datos desde el objeto **HttpSession**.
- Invalidar la sesión (opcional).

Obtener una Sesión

El método **getSession** del objeto **HttpServletRequest** devuelve una sesión de usuario. Cuando llamamos al método con su argumento **create** como **true**, la implementación creará una sesión si es necesario.

Para mantener la sesión apropiadamente, debemos llamar a **getSession** antes de escribir cualquier respuesta. Si respondemos utilizando un **PrintWriter**, entonces debemos llamar a **getSession** antes de acceder al **PrintWriter**, no sólo antes de enviar cualquier respuesta.

```
public class CatalogoServlet extends HttpServlet {

      public void doGet (HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
      {
         // Obtiene la variable sesión para el usuario.
               HttpSession session = request.getSession(true);
         ...
               out = response.getWriter();
         ...
      }
   }
```

Almacenar y Obtener Datos desde la Sesión

El Interface **HttpSession** proporciona métodos que almacenan y recuperan:

- Propiedades de Sesión estándar, como un identificador de sesión.

- Datos de la aplicación, que son almacenados como parejas **nombre-valor**, donde el nombre es un **string** y los valores son **objetos** del lenguaje de programación Java. Como varios servlets pueden acceder a la sesión de usuario, deberemos adoptar una convención de nombrado para organizar los nombres con los datos de la aplicación. Esto evitará que los servlets sobrescriban accidentalmente otros valores de la sesión. Una de esas convenciones es ***servletname.name*** donde ***servletname*** es el nombre completo del servlet, incluyendo sus paquetes. Por ejemplo, **es.uah.comunica.estado** es una cookie con el *servletname* **es.uah.comunica** y el *name* **estado**.

```
public class CatalogServlet extends HttpServlet {

    public void doGet (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException  {
      // Obtiene la variable de sesión del usuario con un objeto de tipo Usuario
      //  el cual guarda los datos personales del usuario.
          HttpSession session = request.getSession(true);
          Usuario miuser =
        (Usuario)session.getValue(session.getId());

    // Si el usuario no creó una sesión hasta ahora se crea un objeto Usuario nuevo
    // con los datos recibidos (por ejemplo)
    if (miuser == null) {
       miuser = new Usuario(response.getParameter("nombre"),
                            response.getParameter("apellidos")
                            response.getParameter("edad"));
       session.putValue(session.getId(), miuser);
    }
       ...
    }
  }
```

Como un objeto puede ser asociado con una sesión, el ejemplo anterior autentica al usuario, para ello comprueba si ha iniciado sesión anteriormente viendo si el objeto asociado a la sesión es nulo o no. En caso de serlo creará la sesión con un nuevo objeto **Usuario** cuyos datos los recuperará con la instrucción getParameter del objeto HttpServletResonse.

Una sesión puede ser designada como *nueva*. Una sesión nueva hace que el método **isNew** de la clase **HttpSession** devuelva **true**, indicando que, por ejemplo, el cliente, todavía no sabe nada de la sesión. Una nueva sesión no tiene datos asociados.

Invalidar la Sesión

Una sesión de usuario puede ser invalidada manual o automáticamente, dependiendo de donde se esté ejecutando el servlet. (Por ejemplo, el Java Web Server, invalida una sesión cuando no hay peticiones de página por un periodo de tiempo, unos 30 minutos por defecto). Invalidar una sesión significa eliminar el objeto **HttpSession** y todos sus valores del sistema.

Para invalidar manualmente una sesión, se utiliza el método **invalidate** de la clase HttpSession. Algunas aplicaciones tienen un punto natural en el que invalidar la sesión.

```
public class servletSalidaUsuario extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {

        ...
        miUser = (Usuario)session.getValue(session.getId());
        ...
        // Borra todos los datos del usuario e invalida la sesión.
        session.invalidate();

        // Asigna el actual tipo de contexto en la cabecera y después crea el
PrintWirter
        response.setContentType("text/html");
        out = response.getWriter();
        ...
    }
}
```

Manejar todos los Navegadores

Por defecto, el seguimiento de sesión utiliza cookies para asociar un identificador de sesión con un usuario. Para soportar también a los usuarios que acceden al servlet con un navegador que no soporta cookies, o si este está programado para no aceptarlas, se debe utilizar **reescritura de URL** en su lugar.

Cuando se utiliza la reescritura de URL se llama a los métodos que, cuando es necesario, incluyen el ID de sesión en un enlace. Debemos llamar a esos métodos por cada enlace en la respuesta del servlet.

El método que asocia un ID de sesión con una URL es **HttpServletResponse.encodeUrl**. Si se redirecciona al usuario a otra página, el método para asociar el ID de sesión con la URL redireccionada se llama **HttpServletResponse.encodeRedirectUrl**.

Los métodos **encodeUrl** y **encodeRedirectUrl** deciden si las URL necesitan ser reescritas, y devolver la URL cambiada o sin cambiar. (Las reglas para las URLs y las URLs redireccionadas son diferentes, pero en general si el servidor detecta que el navegador soporta cookies, entonces la URL no se reescribirá).

Por ejemplo, el siguiente servlet devuelve un catalogo con dos enlaces para cada libro. Un enlace ofrece detalles sobre el libro y el otro ofrece al usuario añadir el libro a su hoja de pedidos. Ambas URLs son reescritas:

```
public class CatalogServlet extends HttpServlet {

    public void doGet (HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
      // Get the user's session and shopping cart, the Writer, etc.
      ...
      // then write the data of the response

      out.println("<html>" + ...);
      ...

      // Get the catalog and send it, nicely formatted

      BookDetails[] books = database.getBooksSortedByTitle();
      ...
      for(int i=0; i < numBooks; i++) {
        ...
        //Print out info on each book in its own two rows
        out.println("<tr>" + ...

              "<a href=\"" +
              response.encodeUrl("/servlet/bookdetails?bookId=" +
                      bookId) +
              "\"> <strong>" + books[i].getTitle() +
              "  </strong></a></td>" + ...

              "<a href=\"" +
              response.encodeUrl("/servlet/catalog?Buy=" + bookId)
              + "\">   Add to Cart  </a></td></tr>" +

      }
    }
  }
```

Si el usuario pulsa sobre un enlace con una URL re-escrita, el servlet reconoce y extrae el ID de sesión. Luego el método **getSession** utiliza el ID de sesión para obtener el objeto **HttpSession** del usuario.

Por otro lado, si el navegador del usuario **no soporta cookies** y el usuario pulsa sobre una **URL no re-escrita**, se **pierde la sesión de usuario**. El servlet contactado a través de ese enlace crea una nueva sesión, pero la nueva sesión no tiene datos asociados con la sesión anterior. Una vez que un servlet pierde los datos de una sesión, los datos se pierden para todos los servlets que comparten la sesión. Debemos utilizar la re-escritura de URLs consistentemente para que nuestro servlet soporte clientes que no soportan o aceptan cookies.