élogos

DESARROLLO DE APLICACIONES WEB CON SERVLETS

élogos

Atributos de petición, sesión y aplicación

ATRIBUTOS DE PETICIÓN

En este capítulo te mostraremos cómo se almacena y recupera un objeto dentro de un atributo de petición.

Mantenimiento del estado de una aplicación

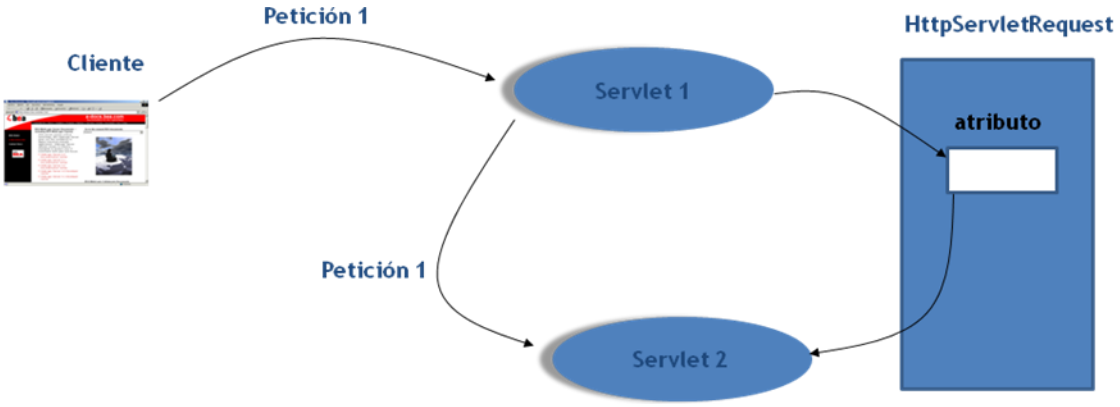
Al proceso mediante el cual la aplicación se encarga de mantener en memoria datos manejados por el usuario, a fin de que estén accesibles durante el tiempo que el usuario utiliza la aplicación, se le conoce como **mantenimiento del estado**.

Existen tres maneras de almacenar información en la memoria del servidor de aplicaciones para uso de la aplicación:

- Atributos de petición
- Atributos de sesión
- Atributos de aplicación

Utilización de atributos de petición

Los atributos de petición permiten compartir datos entre componentes que se ejecutan dentro de una misma petición. Por ejemplo, si Servlet1 genera un dato que tiene que ser utilizado también por Servlet2, el Servlet al que va a trasferir la petición, utilizaríamos este tipo de atributos.



El objeto HttpServletRequest

Los atributos de petición son almacenados en el objeto **HttpServletRequest** que, como sabemos, es compartido por todos los componentes que forman parte de la misma petición.

Para manejar los atributos de petición, la interfaz **HttpServletRequest** proporciona dos métodos:

```
void setAttribute(String nombre, Object valor)
```

Almacena el objeto indicado en el segundo parámetro en un atributo de petición cuyo nombre se indica en el primero.

```
Object getAttribute(String nombre)
```

Recupera el valor del atributo de petición cuyo nombre se indica. En caso de no existir dicho atributo, el método devolverá *null*.

Ejemplo

```
public class Servlet1 extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
         throws ServletExcepervicetion, IOException {
      RequestDispatcher rd=request.getRequestDispatcher("/servlet2");
      //almacena un dato tipo double
      //en un atributo de petición
      Double d=Math.random()*500;
      request.setAttribute("codigo", d);
      //transfiere la petición a servlet2
      rd.forward(request, response);
   }
}

public class Servlet2 extends HttpServlet {
   protected void service(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      //recupera el atributo almacenado por servlet1
      //y genera una página con su valor
      double d=(Double)request.getAttribute("codigo");
      try {
         out.println("<html>");
         out.println("<body>");
         out.println("<h1>El código es " + d + "</h1>");
         out.println("</body>");
         out.println("</html>");
      } finally {
         out.close();
      }
   }
}
```

ATRIBUTOS DE SESIÓN

En este capítulo te mostraremos cómo se almacena y recupera un objeto dentro de un atributo de sesión.

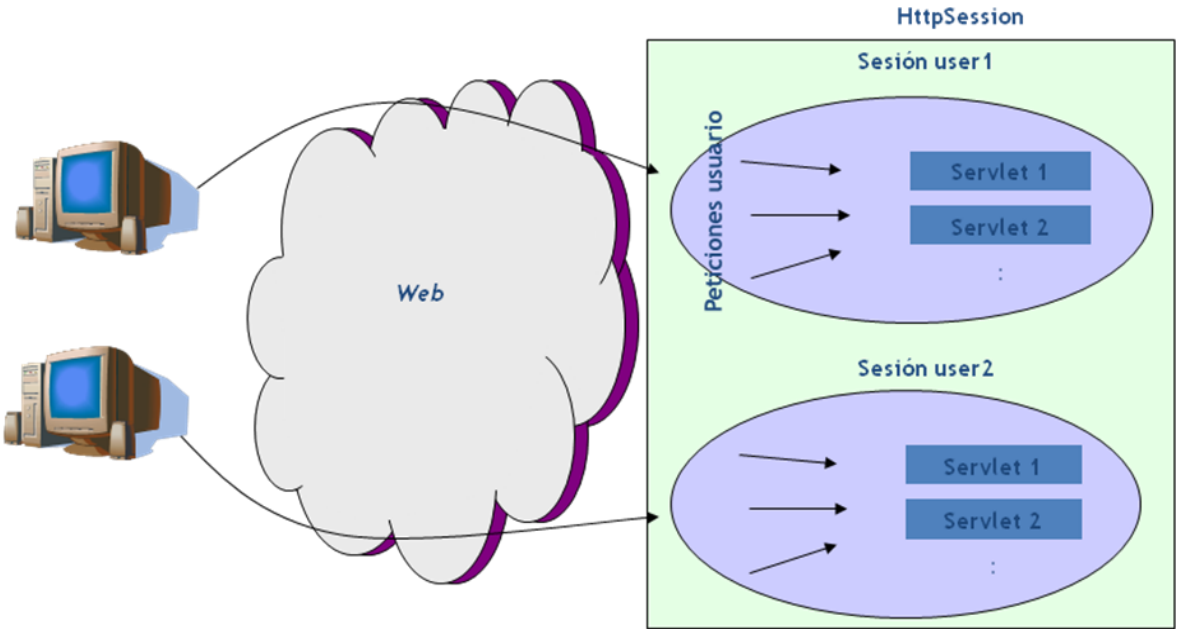
Contexto de utilización

Los atributos de sesión nos permiten mantener datos más allá del tiempo de duración de una petición, permitiendo que los valores de estos atributos sean compartidos por todos los Servlets y páginas JSP de la aplicación.

El tiempo de vida de estos atributos será el de la duración de una **sesión**. Una **sesión** representa el conjunto de peticiones que el usuario realiza a los componentes de la aplicación.

La sesión comienza cuando el usuario realiza la primera petición a uno de los Servlets o páginas JSP y finaliza cuando cierra el navegador, o bien cuando transcurre un determinado periodo de tiempo sin que se realice ninguna petición (Time out).

Cada usuario dispone de sus propios atributos de sesión dentro de la memoria del servidor de aplicaciones.



El objeto HttpSession

Los atributos de sesión se almacenan en el objeto **HttpSession**. Cada usuario de la aplicación dispondrá de su propia instancia del objeto **HttpSession** para manejar sus atributos de sesión.

Para obtener una referencia al objeto HttpSession desde los métodos service() o doGet()/doPost(), utilizaremos el siguiente método del objeto request:

```
HttpSession getSession()
```

Cuando el usuario llama por primera vez a este método durante la sesión se creará una instancia del objeto asociada a ese usuario y se almacenará en la memoria del servidor. Si ya existiera una instancia del objeto para ese usuario, la llamada a getSession() devolverá una referencia al mismo.

HttpSession s = request.getSession();

Cuando se crea el objeto HttpSession para un determinado usuario, el contenedor Web genera un identificador de sesión que es enviado dentro de una cookie en la respuesta generada.

Cuando el usuario lanza una nueva petición y el componente realiza una llmada a getSession(), el contenedor Web utilizará el valor de esta cookie para localizar el objeto HttpSession que corresponde a ese usuario.

Métodos de HttpSession

La interfaz HttpSession dispone de los mismos métodos que HttpServletRequest para el manejo de los atributos de petición. Recordemos que estos método son:

```
void setAttribute(String nombre, Object valor)
```

Almacena el objeto indicado en el segundo parámetro en un atributo de sesión, cuyo nombre se indica en el primero.

Object getAttribute(String nombre)

Recupera el valor del atributo de sesión cuyo nombre se indica. En caso de no existir dicho atributo, el método devolverá *null*.

Ejemplo

```
public class Servlet1 extends HttpServlet {
   protected void serviceHttpServletRequest request, HttpServletResponse response)
         throws ServletExcepervicetion, IOException {
      //cuenta el número de veces que el usuario
      //accede al servlet durante la sesión
      HttpSession s=request.getSession();
      int contador=0;
      if(s.getAttribute("contador")!=null){
         contador=(Integer)s.getAttribute("contador");
      }
      contador++;
      s.setAttribute("contador", contador);
   }
}

public class Servlet2 extends HttpServlet {
   protected void service(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      PrintWriter out = response.getWriter();
      //este servlet muestra el valor del contador
      //de visitas
      HttpSession s=request.getSession();
      int contador=0;
      if(s.getAttribute("contador")!=null){
         contador=(Integer)s.getAttribute("contador");
      }
      try {
         out.println("<html>");
         out.println("<body>");
         out.println("<h1>Número de visitas " + contador + "</h1>");
         out.println("</body>");
         out.println("</html>");
      } finally {
         out.close();
      }
   }
}
```

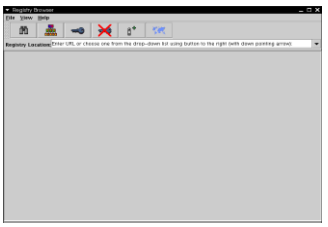
Control de sesiones

Dado que los atributos de sesión pueden estar en memoria una cantidad de tiempo considerable y que cada usuario tiene sus propios atributos, un uso inadecuado de estos puede provocar que la memoria del servidor de aplicaciones se colapse.

Para evitar estas situaciones, la interfaz HttpSession dispone de una serie de métodos para el control de una sesión:

- **removeAttribute(String name):** elimina el atributo cuyo nombre se indica.
- **invalidate():** destruye el objeto HttpSession asociado a la sesión actual, eliminando todos sus atributos.
- **setMaxInactiveInterval(int seg):** establece, en segundos, el periodo máximo de inactividad de la sesión actual.

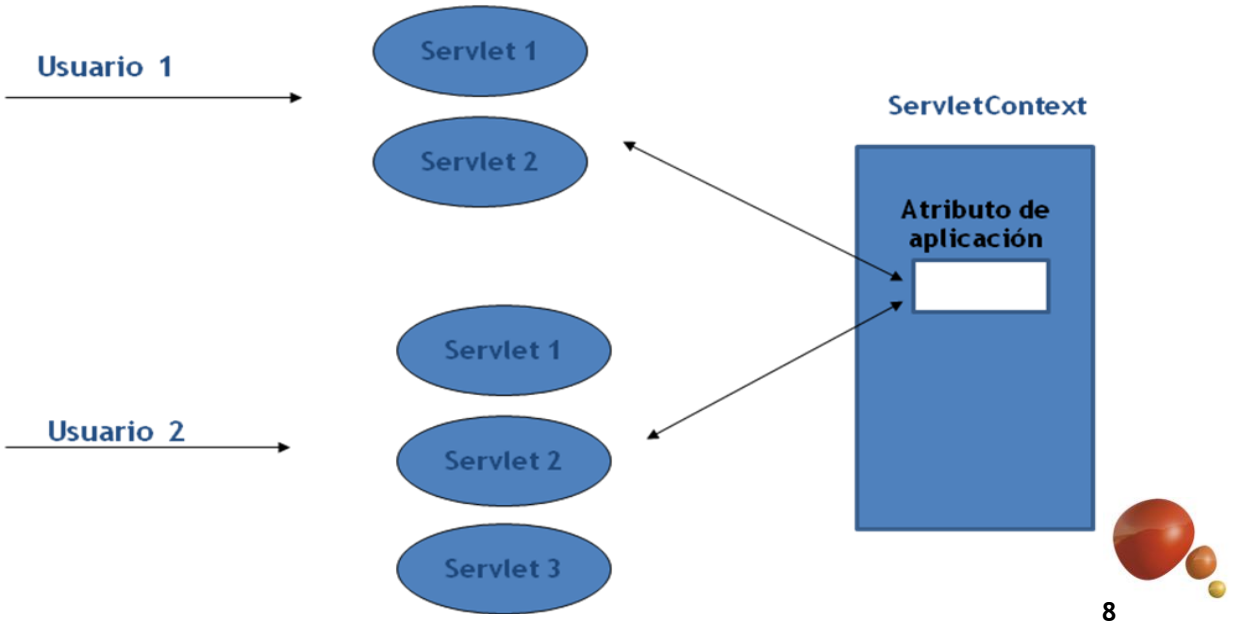
ATRIBUTOS DE APLICACIÓN



En este capítulo te mostraremos cómo se almacena y recupera un objeto dentro de un atributo de aplicación.

Ámbito de utilización

Los atributos de aplicación permiten compartir información entre todos los usuarios de la misma. Los valores de estos atributos están accesibles a todos los componentes de la aplicación.



El objeto ServletContext

Los atributos de aplicación se almacenan en el objeto **ServletContext**. La misma instancia de **ServletContext** es compartida por todos los usuarios de la aplicación.

Para obtener una referencia al objeto **ServletContext**, utilizaremos el siguiente método heredado de HttpServlet:

```
ServletContext getServletContext()
```

Cuando un usuario llama por primera vez a este método se creará una instancia del objeto y se almacenará en la memoria del servidor. Si ya existiera la instancia, la llamada a **geServletContext()** devolverá una referencia a la misma.

ServletContexts = this.getServletContext();

También se puede utilizar el objeto ServletContext en el método init() del servlet, de cara a inicializar el valor de algún atributo. En este caso, se deberá sobrescribir la siguiente versión de init():

void init(ServletConfig config)

Y utilizar el método getServletContext() de ServletConfig para obtener la referencia al objeto.

Métodos de ServletContext

La interfaz ServletContext dispone de los mismos métodos que **HttpServletRequest** y **HttpSession** para el manejo de los atributos de petición. Recordemos que estos método son:

```
void setAttribute(String nombre, Object valor)
```

Almacena el objeto indicado en el segundo parámetro en un atributo de aplicación, cuyo nombre se indica en el primero.

Object getAttribute(String nombre)

Recupera el valor del atributo de aplicacióncuyo nombre se indica. En caso de no existir dicho atributo, el método devolverá *null*.

Ejemplo

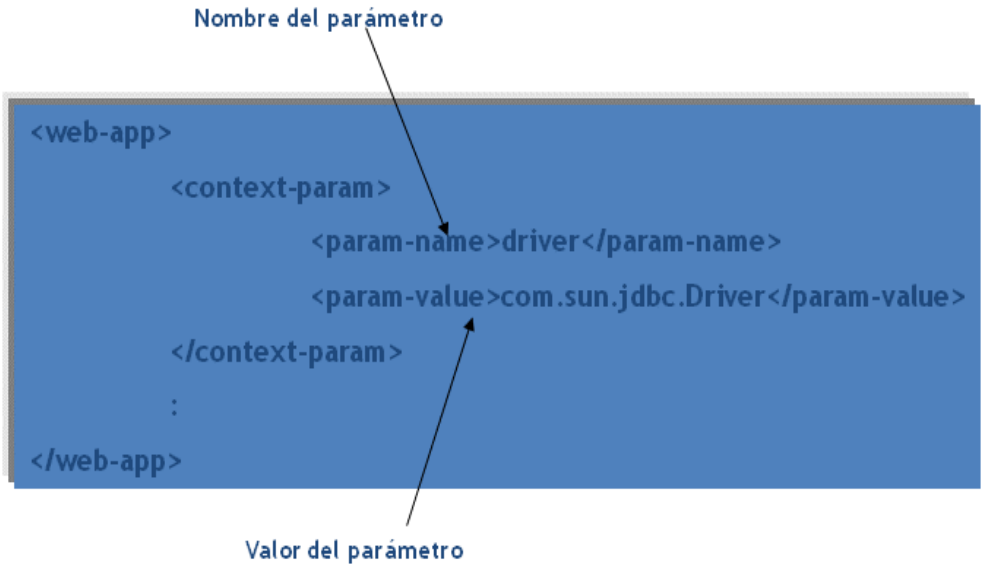
```
public class Servlet1 extends HttpServlet {
   @Override
   public void init(ServletConfig config) throws ServletException {
      //inicializa el contador de visitas globales a 0
      //esto sucederá con la primera petición
      //realizada al servlet
      ServletContext sc=config.getServletContext();
      sc.setAttribute("global",0);
      super.init(config);
   }
   protected void service(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      //incrementa el contador cada vez que un usuario
      //solicita el servlet
      ServletContext sc=this.getServletContext();
      int contador=(Integer)sc.getAttribute("global");
      contador++;
      sc.setAttribute("global", contador);
   }
}

public class Servlet2 extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      //recupera el valor del contador global para
      //mostrarlo
      ServletContext sc=this.getServletContext();
      int contador=(Integer)sc.getAttribute("contador");
```

```
            try {
                out.println("<html>");
                out.println("<body>");
                out.println("<h1>Número de visitas globales" + contador + "</h1>");
                out.println("</body>");
                out.println("</html>");
            } finally {
                out.close();
            }
        }
    }
```

Parámetros de contexto

Los parámetros de contexto no son exactamente una técnica para el mantenimiento del estado, pero nos permiten definir variables (parámetros) dentro del archivo de configuración web.xml para luego poder recuperar sus valores en tiempo de ejecución:



La utilización de este tipo de parámetros permite personalizar la aplicación en tiempo de ejecución, ya que pueden modificarse los valores de los parámetros sin necesidad de alterar el código de la misma. Resultan útiles para almacenar datos como cadenas de conexión a bases de datos, drivers, etc.

Para recuperar los valores de estos parámetros, utilizaremos los siguiente métodos de ServletContext:

- **getInitParameter(String name):** devuelve el valor del parámetro cuyo nombre se indica.
- **getInitParameterNames():** devuelve una enumeración con los nombres de todos los parámetros.