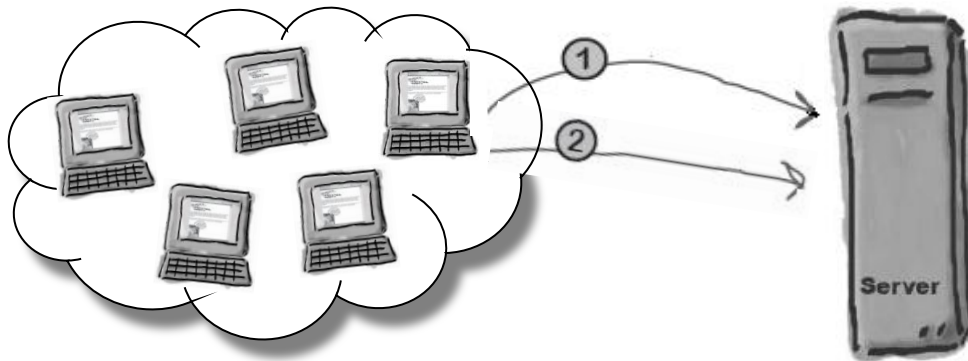


Sesiones en Java

- 1 Cómo mantener el estado con el protocolo HTTP
- 2 Mecanismos de intercambio de ID para manejar sesiones
 - Cookies
 - URL *rewriting*
- 3 La interface **HTTPSession**
 - Ligar y eliminar elementos
 - Invalidar sesión
- 4 Métodos de la interface **HttpServletRequest** para obtener una sesión
- 5 Soporte de sesiones en servlets y JSP: un ejemplo
- 6 Sesiones en ambientes multi-servidores

Soporte de Sesiones

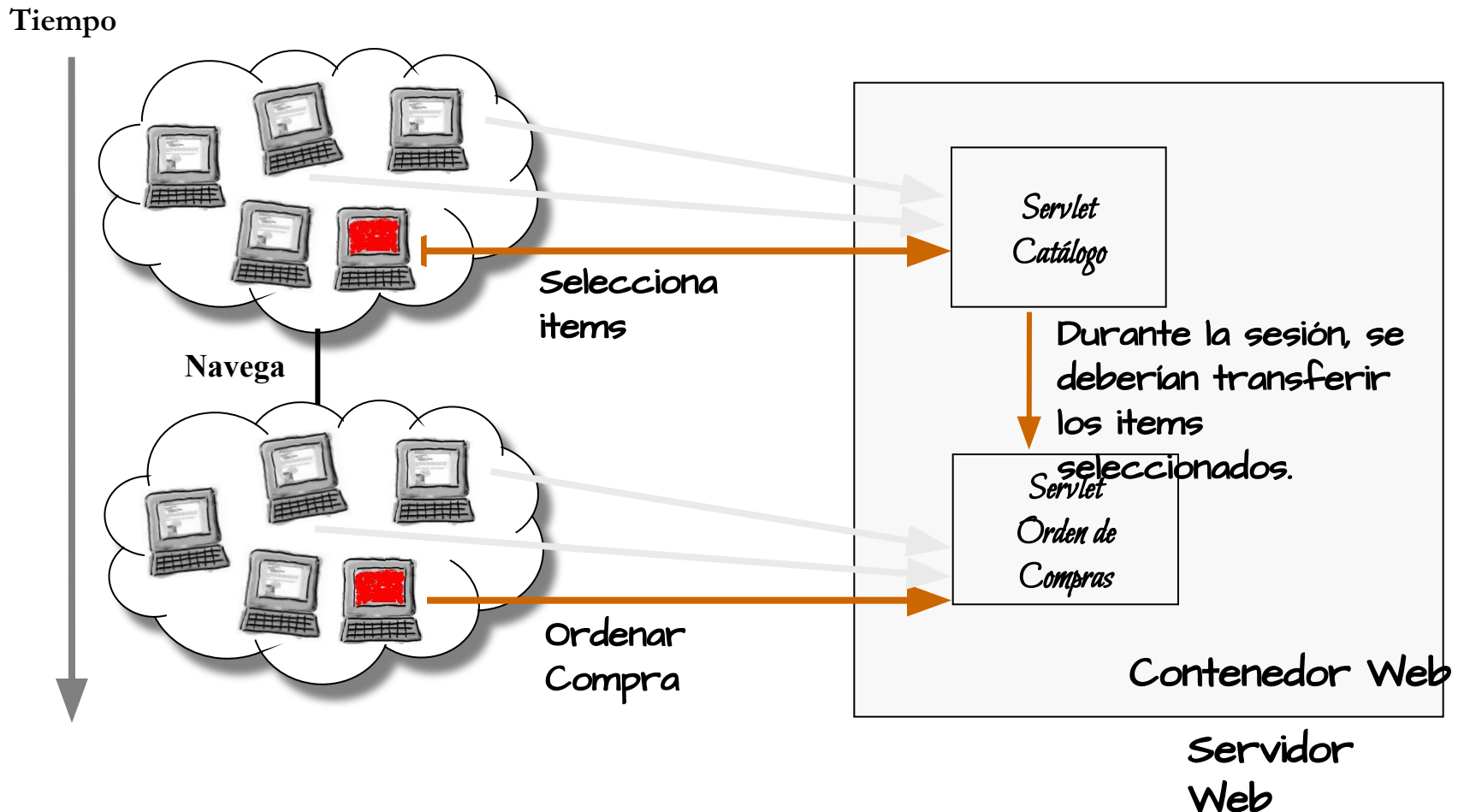
- **HTTP (Hypertext Transfer Protocol)** es un *protocolo sin estado, orientado a conexión*. Los pares requerimiento y respuesta HTTP son independientes unos de otros. El servidor web NO sabe si una serie de requerimientos provienen del mismo o de diferentes clientes y si están además, relacionados entre sí.
- Una **sesión** consiste en una serie de interacciones relacionadas que se realizan entre un mismo navegador web y un servidor web, durante un período de tiempo. Una **sesión** tiene información o estado correspondiente a un cliente particular.



El servidor no recuerda que ambos requerimientos vienen del mismo cliente

- Para mantener el estado de una sesión en el servidor, la clave es identificar todos los requerimientos provenientes de un mismo cliente remoto y mantener el estado entre esos requerimientos.

Soporte de Sesiones



¿Cómo hacemos desde una aplicación web para identificar qué requerimientos provienen de un mismo cliente? ¿Cómo mantenemos el estado entre los diferentes requerimientos?

Soporte de Sesiones

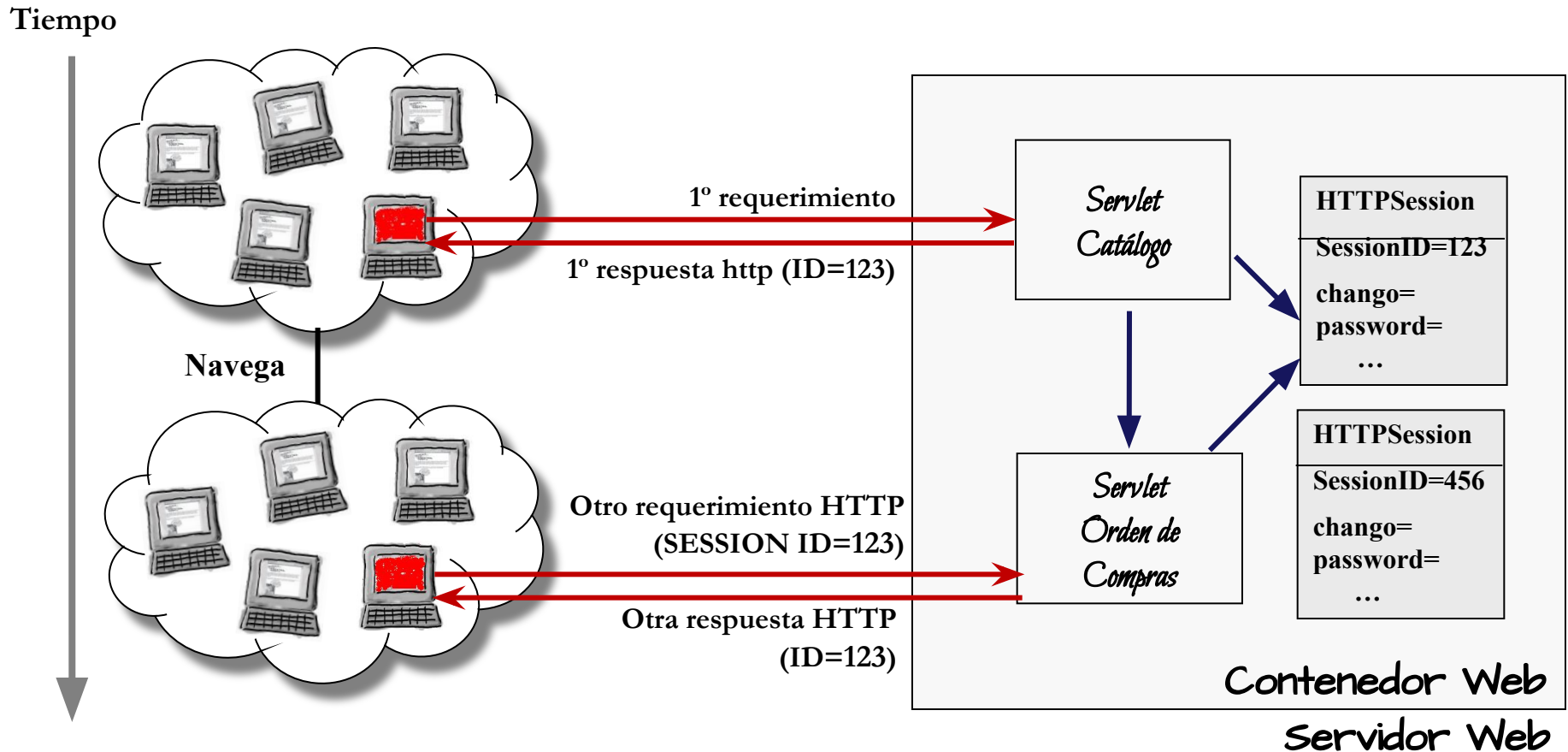
La API Servlet

- Una de las funciones más importantes de la **API de Servlet** es el gerenciamiento (creación, expiración y mantenimiento del estado de la sesión por usuario) de sesiones a través de la interface **javax.servlet.http.HttpSession**.
- Esta API es una interface de alto nivel, construida sobre **cookies y rescritura de URL**. En general, los servidores usan cookies si el browser lo soporta, y automáticamente revierten a rescritura de URL cuando las cookies no son soportadas o están deshabilitadas.
- El programador de servlets no necesita preocuparse por los detalles, no tiene que manipular explícitamente cookies ni agregar información a la URL y además, se le garantiza un lugar en el contenedor para almacenar datos asociados con cada sesión.
- La especificación de Servlets obliga a los contenedores web a proveer soporte para objetos **HttpSession**. Los contenedores usan la interface **HttpSession** para crear una sesión entre un cliente HTTP y un servidor HTTP. Los objetos almacenan atributos que son únicos para un cliente específico, y existen a través de múltiples requerimientos HTTP.

Soporte de Sesiones

Se necesita un ID de sesión único

- Cuando se crea un objeto **HttpSession**, se le asocia una identificación única, **ID de sesión**, que se le entrega al cliente mediante cookies/rescritura de URL.
- Cada vez que el cliente interactúa con una componente de dicha aplicación web, el cliente provee al servlet (o jsp) el **ID de sesión** (asignado previamente).

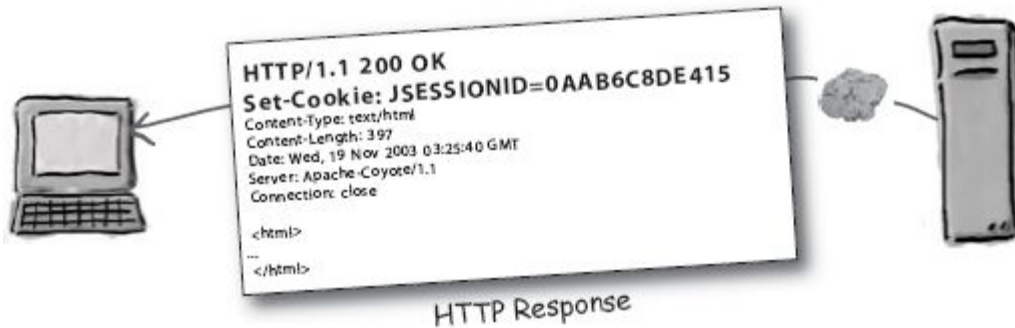


Soporte de Sesiones

¿cómo intercambian ID el cliente y el contenedor?



De alguna manera, el contenedor tiene que crear un ID de sesión y enviarlo como parte de la respuesta, y el cliente tiene que enviar de vuelta el ID de la sesión como parte de los requerimientos subsiguientes. La manera más común y más simple para el intercambio de la información es a través de cookies.



El contenedor envía una cookie con el ID de sesión, "Set-Cookie" es otro header de la respuesta

El cliente envía con el requerimiento la cookie con el ID de sesión.



¿qué pasa si deshabilitan las cookies?

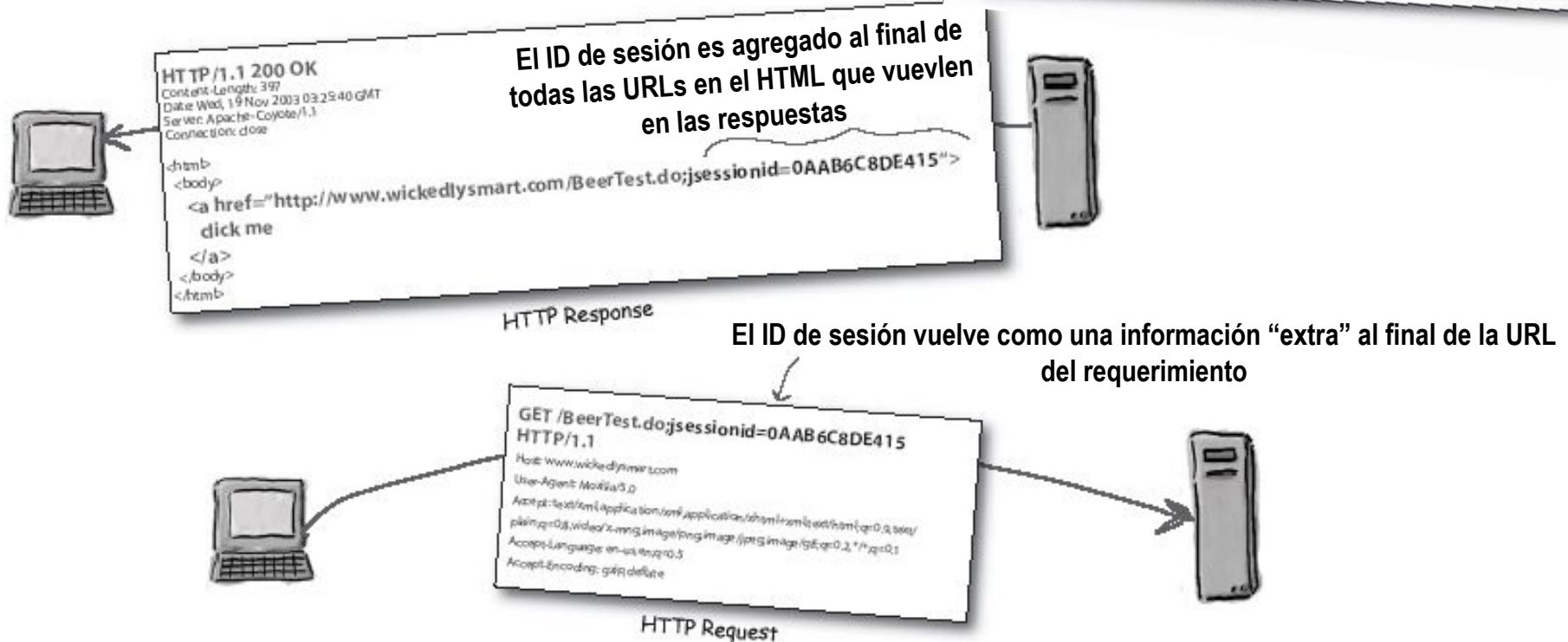
Soporte de Sesiones

Garantizando el funcionamiento de sesiones

Si el cliente no usa cookies, se puede usar **URL rewriting** como alternativa. URL rewriting siempre trabaja.

URL rewriting toma el ID que va en la cookie y lo pega a la derecha de la URL que se envía en la respuesta.

URL + **;jsessionid=0AAB6C8DE415**



El mecanismo usado por defecto por los servidores web para enviar el **identificador**, es cookies y automáticamente (si el programador lo indicó) rewruten a rescritura de URL cuando las cookies no son soportadas o están deshabilitadas.

Soporte de Sesiones

Rescritura de URL (*URL rewriting*)

Si las cookies no funcionan, el contenedor revierte a *URL rewriting*, pero solamente si el programador ha hecho el trabajo extra, de codificar todas las URLs que se envían en la respuesta.

Si no se indica **explícitamente que se codifiquen las URLs** y el cliente no acepta cookies, entonces, no se podrá usar sesiones.

¿cómo se hace?

La API de servlet facilita la rescritura de URL, a través de 2 métodos de la clase `HttpServletResponse`, los cuales incluyen **automáticamente el ID de sesión en la URL**.

- `encodeURL(java.lang.String url)`: toma como parámetro un String que representa una url y restorna un String representando la url con el id de la sesión.
- `encodeRedirectURL(java.lang.String url)`: trabaja de la misma manera pero se utiliza para redirección.

`/applic/MiServlet;jsessionid=0AAB6C8DE415`

Ejemplos de uso:

```
out.println("<a href='/applic/MiServlet'>Seguir</a>");
out.print("<a href='"+response.encodeURL("/applic/MiServlet")+">Seguir</a>");

response.sendRedirect("/applic/MiServlet")
response.sendRedirect(response.encodeRedirectURL("/applic/MiServlet"))
```


Soporte de Sesiones

El objeto HttpSession

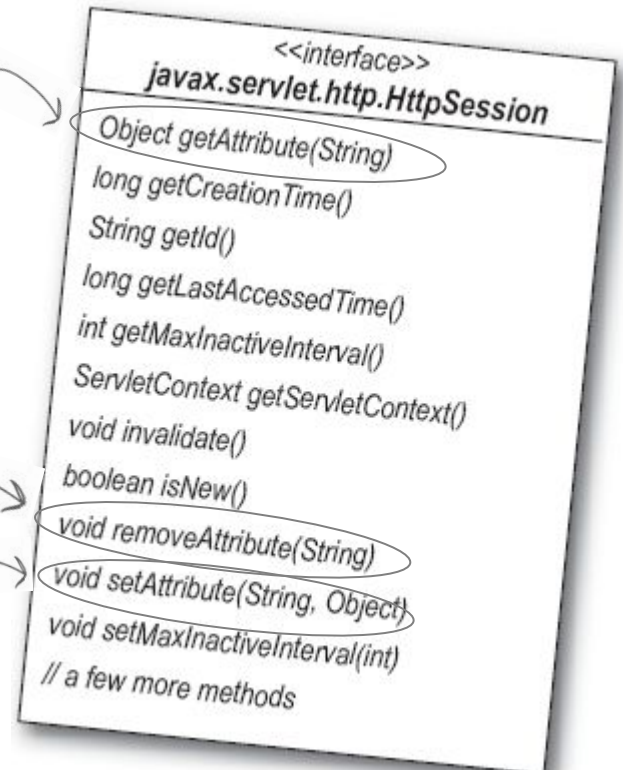
Un objeto **HttpSession** es un objeto que reside en el contenedor web y que mantiene datos para un cliente particular. Cuando se invoca al método **getSession()** se obtiene una instancia de una clase que implementa la interface **HttpSession**. La clase es específica de cada contenedor.

¿Qué podemos hacer con este objeto? Las funcionalidades mas importantes están vinculadas con el guardado y recuperación de *objetos de alcance sesión*.

Devuelve un objeto ligado a la sesión con el nombre pasado como parámetro. El objeto retornado es una instancia de una subclase de Object y generalmente, necesita ser casteado antes de ser usado.

Elimina un objeto de alcance sesión, cuyo nombre coincide con el parámetro. Devuelve null si no existe un objeto ligado con ese nombre.

Liga un objeto a la sesión con el nombre o clave pasado como parámetro. Cualquier objeto ligado previamente con el mismo nombre será reemplazado.



Soporte de Sesiones

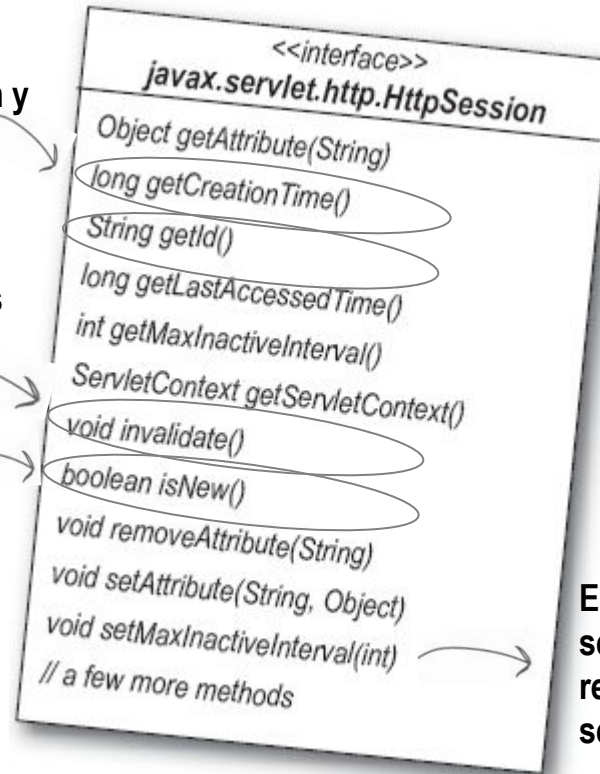
El objeto HttpSession

Un objeto **HttpSession** también nos permite manejar tiempos de la sesión, conocer los últimos accesos e invalidarla.

Devuelven la fecha de creación y el ID de sesión asignado

Invalida la sesión y elimina todos los objetos ligados a ella.

Devuelve un valor boolean que indica si la sesión es nueva, esto es, no se ha recibido del cliente un requerimiento con ID todavía



¿Cuándo invalidar una sesión?

Soporte de Sesiones invalidando una sesión

Los objetos **HttpSession** tienen un tiempo de vida finito. Una sesión puede expirar (*time-out*) por dos causas:

- 1 porque permaneció inactiva un determinado período de tiempo, prefijado como máximo período de inactividad.

El período de tiempo que una sesión puede estar ociosa antes de que expire, se puede configurar en el archivo **web.xml**, para todas las sesiones:

```
<session-config>  
  <session-timeout>20</session-timeout>  
</session-config>
```

minutos

Los contenedores establecen un tiempo de vida por defecto.

También se puede establecer un período máximo de inactividad a una sesión particular:

```
. . .  
sesion.setMaxInactiveInterval(3*600);
```

segundos

Cambia el time-out de una sesión, sin afectar el time-out de restos de las sesiones de la aplicación web

- 2 porque fue invalidada por la aplicación. Una aplicación puede invalidar una sesión en cualquier momento, invocando el método **invalidate()** del objeto **HttpSession**. Si una sesión no es explícitamente invalidada, será invalidada automáticamente después de un cierto período de tiempo.

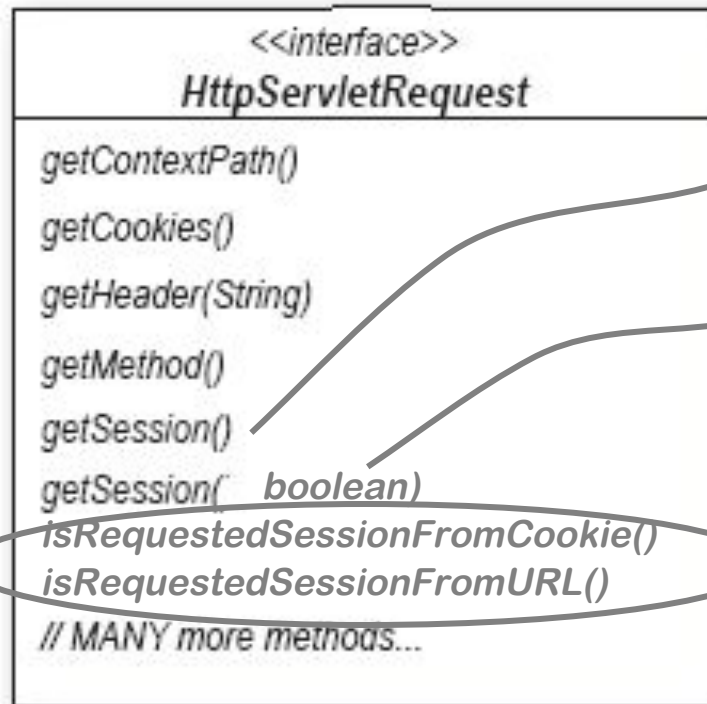
Soporte de Sesiones

Métodos útiles de la interface `HttpServletRequest`

La interface **`HttpServletRequest`**, además de permitir recuperar parámetros, atributos, información del cliente, tiene métodos útiles para crear y recuperar sesiones.

`HttpServletRequest` interface

(`javax.servlet.http.HttpServletRequest`)



Devuelve la sesión actual asociada al requerimiento, o si el requerimiento no tiene sesión asociada, crea una nueva.

Devuelve la sesión actual asociada al requerimiento, o crea una nueva sólo si `boolean` es `true`.

Chequea si el ID de sesión enviado con el requerimiento proviene de una Cookie o de rescritura de URL respectivamente. Ambos métodos retornan `true/false`


Soporte de Sesiones

¿cómo se obtiene una sesión?

Usando el método **getSession()** de un objeto **HttpServletRequest**, se le indica al contenedor que se quiere crear o usar una sesión y el **contendor web hace casi todo el trabajo!**

Cualquier componente web puede contener el siguiente código:

```
HttpSession sesion = request.getSession();
```



Le preguntamos al objeto **request** por una sesión y el contenedor hace todo. Este método, la 1º vez, además de crear la sesión, causa que una cookie sea enviada con la respuesta

if (el request incluye ID de sesión)

Se recupera el ID sesión y se busca la sesión (objeto **HttpSession**)

else

Crea una nueva sesión

- Crea un objeto **HttpSession**
- Genera un ID de sesión
- Crea un objeto cookie y le asocia el ID de sesión
- Setea la cookie en la respuesta (set-cookie en el header)


Guardar y recuperar datos en la sesión

Escribir datos en un objeto HttpSession

Todo objeto que se agrega a un objeto **HttpSession**, tiene un nombre único asociado (un String) que lo identifica.

```
HttpSession sesion = request.getSession(true);  
CarritoCompras compras = new CarritoCompras();  
sesion.setAttribute("carrito", compras);
```

Si ya existia en la sesión un objeto con nombre "carrito", lo sobrescribe.



Recuperar datos desde un objeto HttpSession

Para recuperar un objeto de la sesión hay que hacer **casting** explícito a un tipo de dato java.

```
HttpSession sesion = request.getSession(true);  
CarritoCompras compras=(CarritoCompras)sesion.getAttribute("carrito");
```

Retorna *null* si no encuentra en la sesión un objeto con clave "carrito".
Si no se sabe si existe en la sesión preguntar si es null antes de castear



Soporte de sesiones un ejemplo

Los siguientes códigos pertenecen a una aplicación web simple, que permite seleccionar libros para comprar. Los libros seleccionados se van guardando en la sesión del usuario.

`catalogo.jsp`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Libreria</title>
</head>
<body>
<H1>Libreria RAYUELA</H1>
<FORM action="ComprarLibros" method="post">
  La Isla sin tesoro <input type="checkbox" name="item" value="La Isla sin tesoro"><br>
  Las aventuras de Tom Sawyer <input type="checkbox" name="item" value="Las aventuras de Tom Sawyer"><br>
  Nadie te creería <input type="checkbox" name="item" value="Nadie te creería"><br>
  Las mil y una Noche <input type="checkbox" name="item" value="Las mil y una Noche"><br>
  El Quijote <input type="checkbox" name="item" value="El Quijote"><br>
  <input type="submit" value="Comprar">
</FORM>

</body>
</html>
```



Soporte de sesiones un ejemplo

```
package libreria;
// imports
@WebServlet("/ComprarLibros")
public class ComprarLibros extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

        String nombre=null;
        Integer cant_items=null;
        HttpSession session = request.getSession(true);
        Object cant = session.getAttribute("cant_items");
        if (cant == null)
            cant_items=0;
        else
            cant_items=(Integer)cant;

        String[] itemsSeleccionados = request.getParameterValues("item");
        if (itemsSeleccionados != null){
            for (int i=0; i<itemsSeleccionados.length; i++) {
                nombre = itemsSeleccionados[i];    //Isla, Sawyer
                cant_items=cant_items+1;
                session.setAttribute("Item"+cant_items,nombre);
            }
            session.setAttribute("cant_items", cant_items);
        }
        response.sendRedirect("MostrarCompras");
    }
}
```

Se obtiene la cantidad de ítems seleccionados en la sesión

Se recuperan en un **arreglo** los libros seleccionados en formulario

Se guardan en el objeto sesión cada ítem seleccionado bajo la clave item# y la **cantidad total de ítems seleccionados**

Transfiere el control al servlet MostrarCompras para que genere la respuesta.

Soporte de sesiones un ejemplo

El *servler* **MostrarCompras** toma los *items* guardados en la sesión y los muestra. Si no hay sesión, redireccióna a la página **catalogo.html**

```
@WebServlet("/MostrarCompras")
public class MostrarCompras extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        HttpSession session = request.getSession();
        if (session.isNew()) {
            response.sendRedirect("catalogo.html");
        } else {
            Integer cant_items = (Integer) session.getAttribute("cant_items");
            out.println("<H1>Libreria RAYUELA</H1>");
            for (int i = 1; i <= cant_items.intValue(); i++) {
                out.write("<P>");
                out.write("Item" + i + ": " + session.getAttribute("Item" + i));
                out.write("<br>");
            }
            out.write("<a href='catalogo.html'>Comprar Más Libros</a>");
            out.write("<a href='Salir'>Salir</a>");
            Out.close()
        }
    }
}
```



Se obtiene la cantidad de
items seleccionados en la
sesión

Se recupera el valor de
cada uno de los items de
la sesión y se imprimen

Soporte de sesiones un ejemplo

```
package misServlets;

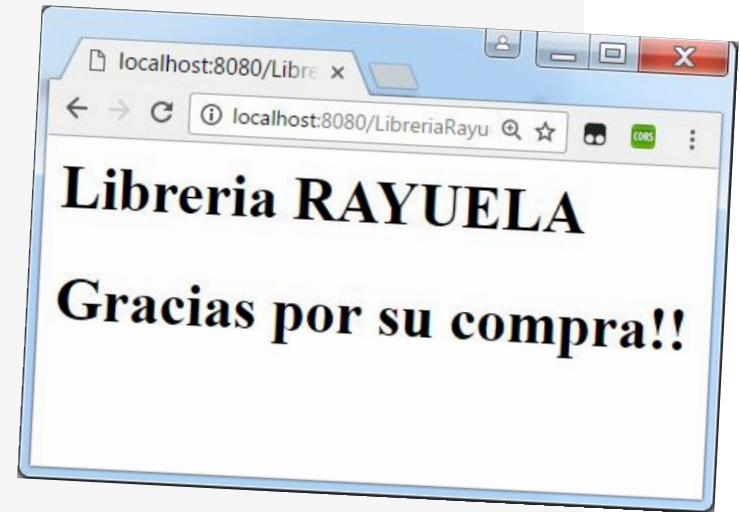
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet("/Salir")
public class Salir extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws . . . {
        PrintWriter out=resp.getWriter();
        resp.setContentType("text/html");
        out.print("<html>");
        out.println("<H1>Libreria RAYUELA</H1>");
        out.print("<body>");

        HttpSession ses = req.getSession(false);
        if (ses!=null){
            out.print("<H1>Gracias por su compra!!</H1>");
            ses.invalidate();
        }

        out.print("</body>");
        out.print("<html>");
        out.close();
    }
}
```



Soporte de sesiones

La interface HttpSessionListener

En la API de Servlets existen listeners relacionados con sesiones. Una clase que implementa la interface **javax.servlet.http.HttpSessionListener** es notificada de los cambios que se producen en la lista de sesiones activas de la aplicación web. Asociada con esta interface *listener* está la clase **javax.servlet.http.HttpSessionEvent** que representa el evento ocurrido y permite acceder al objeto **HttpSession**.

```
package libreria;  
import javax.servlet.*;  
import javax.servlet.http.*;
```

```
@WebListener
```

```
public class ContadorSesionesActivas implements HttpSessionListener{  
    private static int sesionesActivas;
```

```
    public static int getSesionesActivas() {  
        return sesionesActivas;  
    }
```

```
    public void sessionCreated(HttpSessionEvent arg0) {  
        sesionesActivas++;  
    }
```

```
    public void sessionDestroyed(HttpSessionEvent arg0) {  
        sesionesActivas--;  
    }
```

```
}
```

Esta clase es deployed en **WEB-INF/classes** como todas las clases por lo tanto el método de clase **getSesionesActivas()** estará accesible para cualquier recurso de la aplicación.

Estos métodos reciben el evento **HttpSessionEvent** a partir del cual se puede recuperar la sesión.

También existe la interface **javax.servlet.http.HttpSessionAttributeListener** que permite detectar cuando se agregan, reemplazan o eliminan atributos de la sesión.

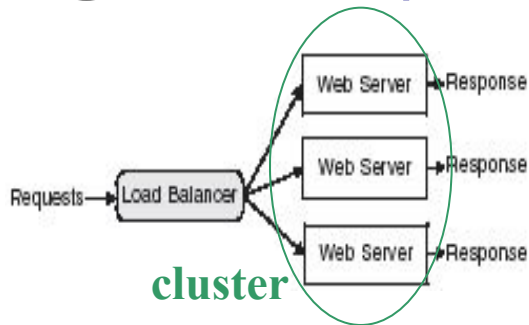
Soporte de sesiones

Persistencia de la sesión

Las sesiones mantienen el estado conversacional de los usuarios. El modelo que hemos discutido hasta ahora, es uno en donde toda esta información o estado está en la memoria de un servidor web.

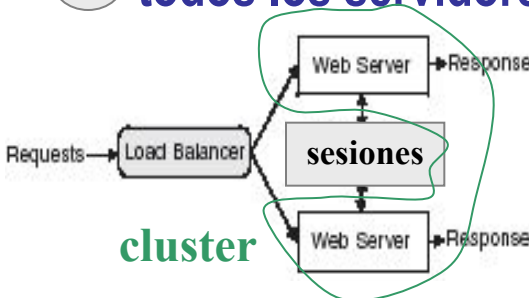
¿Cómo manejar el estado en un ambiente multi-servidor ?

1 todos los requerimientos del mismo usuario van al mismo servidor:



- **balanceo de carga:** si toda la información de una sesión está toda en un servidor y se fuerza a que todos requerimientos vayan al mismo servidor, se arruina el balanceo.
- **respaldo del estado:** ¿Qué pasa si el servidor se cae? Si el estado de la sesión está guardado en la memoria de un solo Servidor, la información se pierde.

2 todos los servidores comparten la misma información:



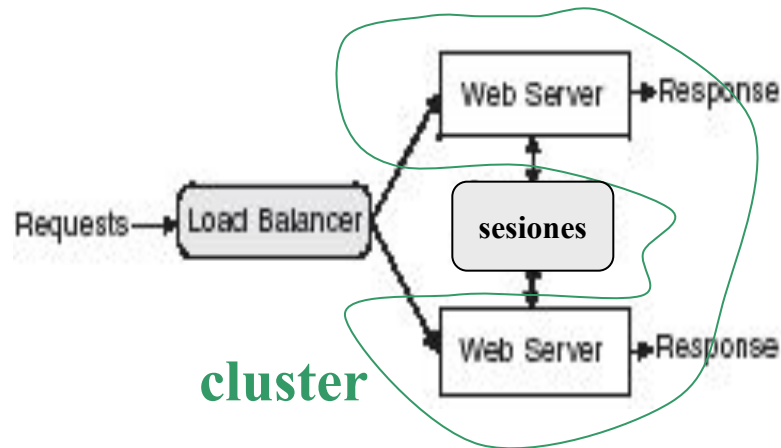
- **Esta es la solución preferida!!**. En este modelo, la información de la sesión del usuario, esta disponible para cualquier servidor que la necesite. El estado puede estar en memoria o en archivos.

Soporte de sesiones

Persistencia de la sesión

Todos los servidores comparten la misma información

- Los servidores web, pueden replicar el estado de la sesión en memoria. En general, el servidor que recibe el primer requerimiento, crea la sesión (**HTTPSession**) y la replica al resto de los servidores. El proceso de copiar el estado de una sesión, desde un servidor a otro es llamado réplica en memoria. Todos los servidores mantienen actualizados los estados de las sesiones.
- Los servidores también podrían mantener el estado de las sesiones HTTP usando persistencia basada en archivos o JDBC.



Soporte de sesiones

Persistencia de la sesión

Todos los servidores comparten la misma información

(A) Tener la sesión replicada en todos los servidores que forman el cluster

La especificación de Servlet, define el elemento `<distributable>`, el cual debe incluirse en el `web.xml`, para indicar que la aplicación web puede ser distribuida, a través de múltiples servidores (cluster).

Para que una sesión pueda persistir entre distintos servidores, se debe cumplir con un requerimiento: los objetos de alcance sesión deben implementar la interface `java.io.Serializable`. Los contenedores que soportan ambientes multi-server, invocarán los métodos `writeObject()` y `readObject()` para guardar y recuperar información de la sesión. El método `setAttribute(String k, Object o)` no chequea que se guarden objetos serializables, pero es una buena práctica hacerlo.

Implementar la interface `java.io.Serializable` es generalmente lo único que tiene que hacer un programador para tener a todos los objetos de alcance sesión, distribuidos en un ambiente *multi-servidor*.

Soporte de sesiones

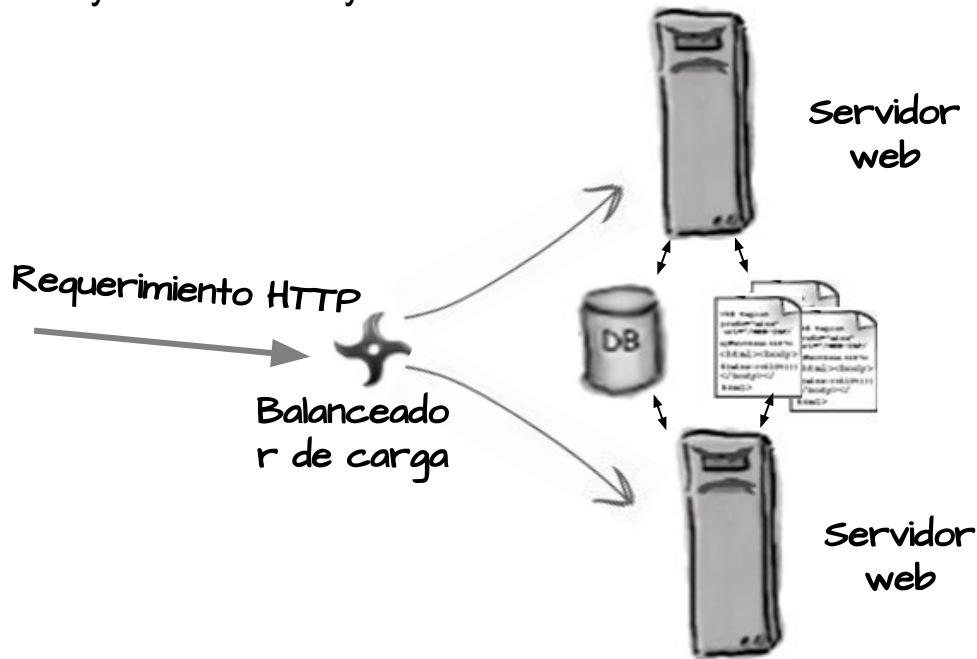
Persistencia de la sesión

Todos los servidores comparten la misma información

(B) Compartir el estado de las sesiones usando una base de datos/archivos

Consiste en usar un servidor centralizado para manejar la información de las sesiones. Esta solución le permite a los programadores tener un soporte completo sobre la persistencia del estado. También podrían guardarse en archivos.

La API de servlets provee listeners para ayudar a construir la base de datos/archivos como **HttpSessionListener** y **HttpSessionAttributeListener** que permiten detectar cuando se crea o destruye una sesión y cuando se cambian atributos de una sesión.



En cuanto a esta solución hay que tener en cuenta que los accesos a base de datos son casi siempre los módulos que bajan la *performance* de una aplicación.