Sistemas Distribuidos e Internet
Introducción a JEE – Servlets y JSPs
Sesión 1
Curso 2017/ 2018

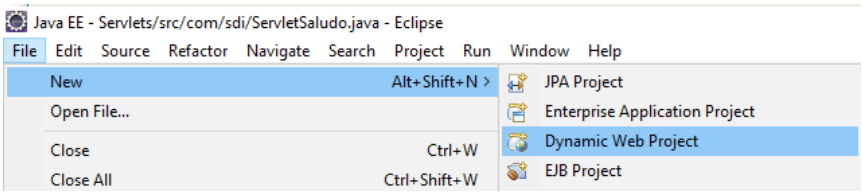
1 Introducción

En esta práctica vamos a implementar una pequeña tienda de la compra empleando de forma progresiva las tecnologías y conceptos base de la tecnología de Servidor JEE. Emplearemos desde Servlets y JSPs hasta JavaBeans y JSTL. Y acabaremos por encajar la aplicación desarrollada en un patrón arquitectónico MVC:

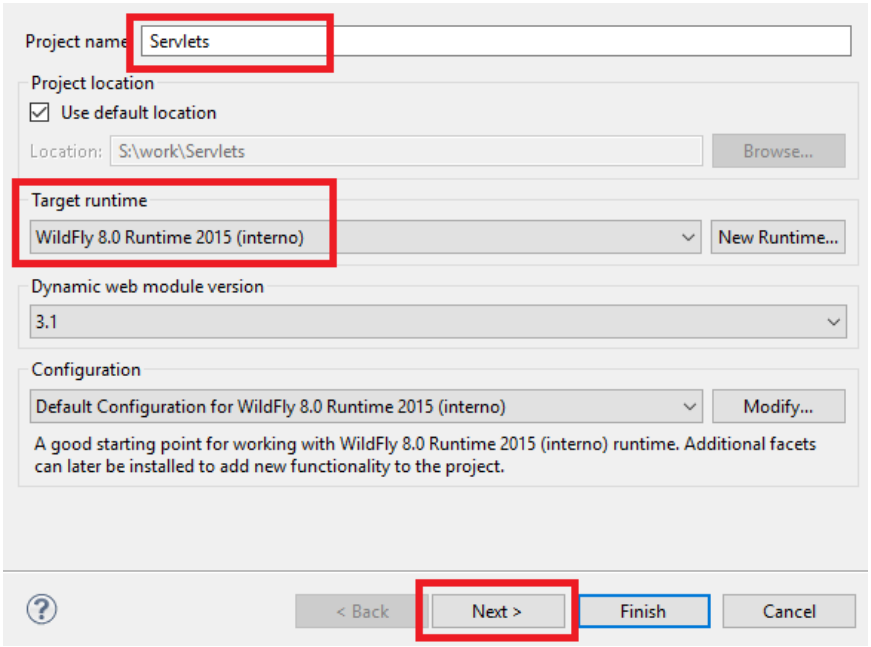
- Servlet: Clase Java "desplegable" que es la base del framework JEE.
- JSP: Combinación de código Java embenido en una página HTML que es traducido a un servlet cuando es desplegado por primera vez.
- JavaBean: Clase Java que debe cumplir ciertas condiciones y que es accesible desde un JSP o código JSTL.
- JSTL: Estructuras de control que pueden ir en una página JSP.

2 Servlets y parámetros

Abrimos el eclipse que se encuentra en el entorno de SDI, para ello ejecutamos el fichero \entorno-sdi\eclipse.bat. Creamos un nuevo proyecto de tipo **Dynamic Web Project**

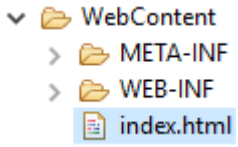


Llamaremos al proyecto **Servlets.** Nos aseguramos de que el Runtime este correctamente configurado con el servidor **WildFly 8.0**



Pulsamos en **Next** y en la última página del asistente marcamos el check **Generate web.xml deployment descriptor**.

Añadimos un fichero **index.html** en la carpeta **/WebContent/** de nuestro proyecto (botón derecho , **New => File**) . Incluimos el siguiente contenido:

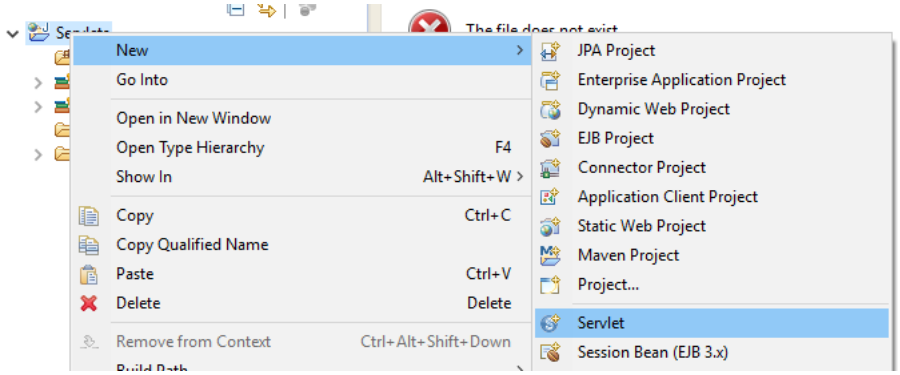


Contendrá dos formularios uno GET y otro POST con un parámetro con clave **nombre** que se envían a la URL **ServletSaludo**.

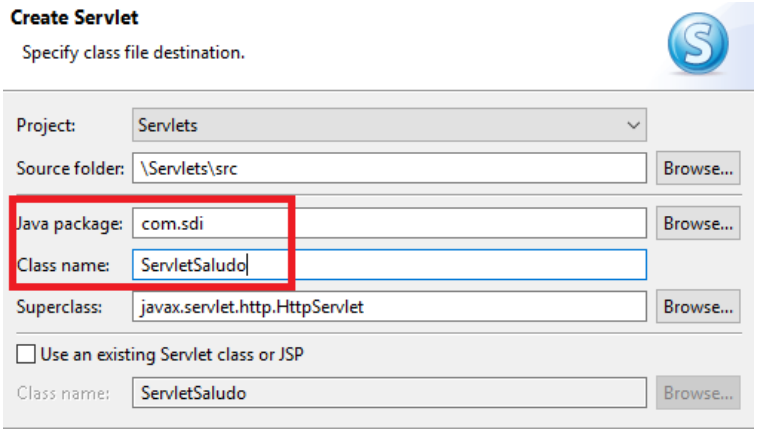
```
<html lang="en">
<head>
  <title>Servlets</title>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<!-- Contenido -->
<div class="container" id="contenedor-principal">

        <h2>Formulario GET Saludar</h2>

        <form action="ServletSaludo" method="get">
          <div class="form-group">
            <label for="nombre-get">Nombre</label>
            <input type="text" class="form-control" name="nombre" id="nombre-get">
          </div>
          <button type="submit" class="btn">Enviar</button>
        </form>

        <h2>Formulario POST</h2>

        <form action="ServletSaludo" method="post">
          <div class="form-group">
            <label for="nombre-post">Nombre</label>
            <input type="text" class="form-control" name="nombre" id="nombre-post">
          </div>
          <button type="submit" class="btn">Enviar</button>
        </form>

</div>

</body>
```

```
</html>
```

Para agregar un nuevo Servlet hacemos click derecho sobre el nombre del proyecto **New -> Servlet.**



Lo almacenaremos en el paquete **com.sdi** y con el nombre **ServletSaludo.**



Abrimos la clase **ServletSaludo**, observamos que implementa los métodos **doGet()** y **doPost()** para responder a peticiones GET y POST respectivamente.

Implementamos una respuesta a **doGet(),** obteniendo el parámetro nombre. Esta función ya es capaz de responder a peticiones GET.

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

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hola Mundo!</TITLE></HEAD>");
    out.println("<BODY>");
    String nombre = (String) request.getParameter("nombre");
    if (nombre != null) {
        out.println("Hola " + nombre + "<br>");
    }
    out.println("</BODY></HTML>");
}
```

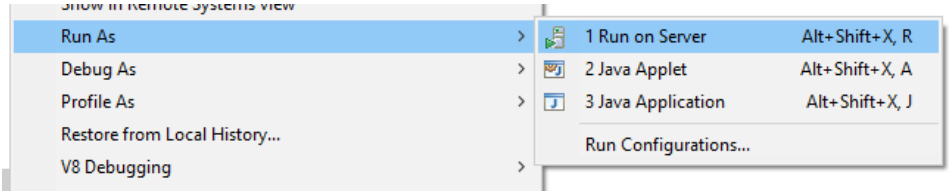
La función **doPost()** se encarga de que el servlet responda a peticiones POST. En este caso la función doPost() simplemente llama a función doGet().

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    doGet( request,response );
}
```

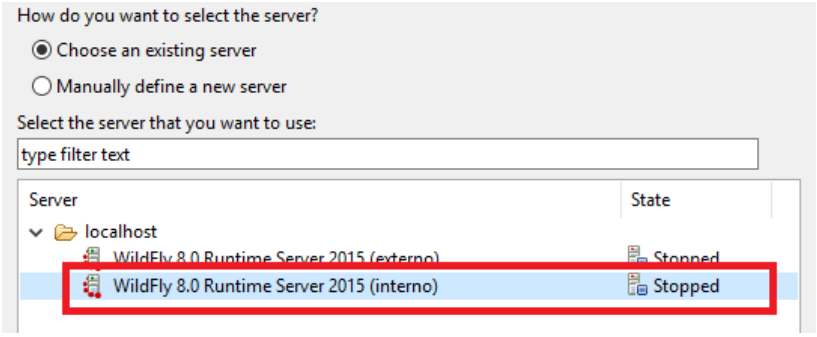
La anotación WebServlet declara en el inicio de la clase nos sirve para definir la URL del servlet, en este caso **/ServletSaludo**, podremos enviarle peticiones a esta URL y el servlet responderá tanto a peticiones GET como POST.

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

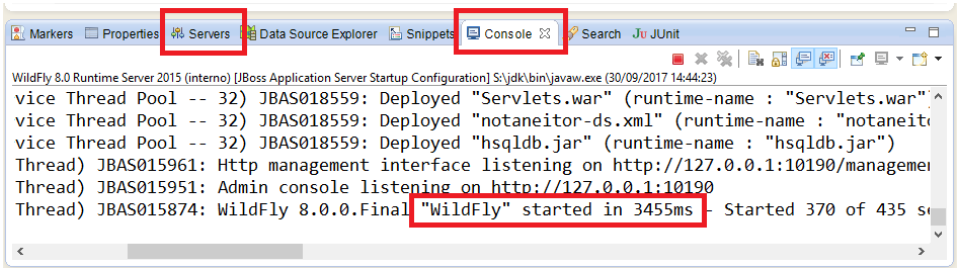
Pulsamos el botón derecho sobre el nombre del proyecto, seleccionamos **Run as -> Run on Server**



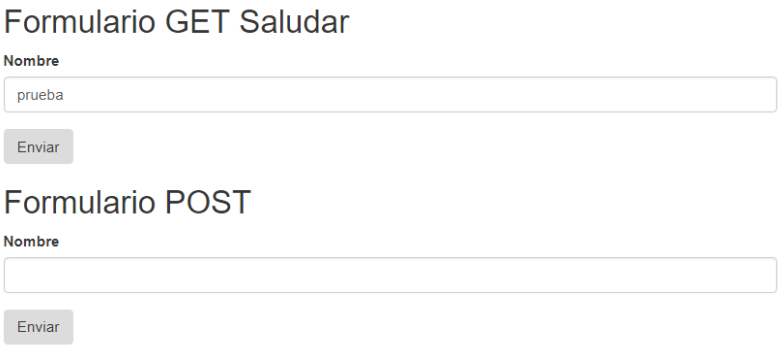
Seleccionamos el servidor **WildFly 8.0 Interno** y pulsamos **Finish.**



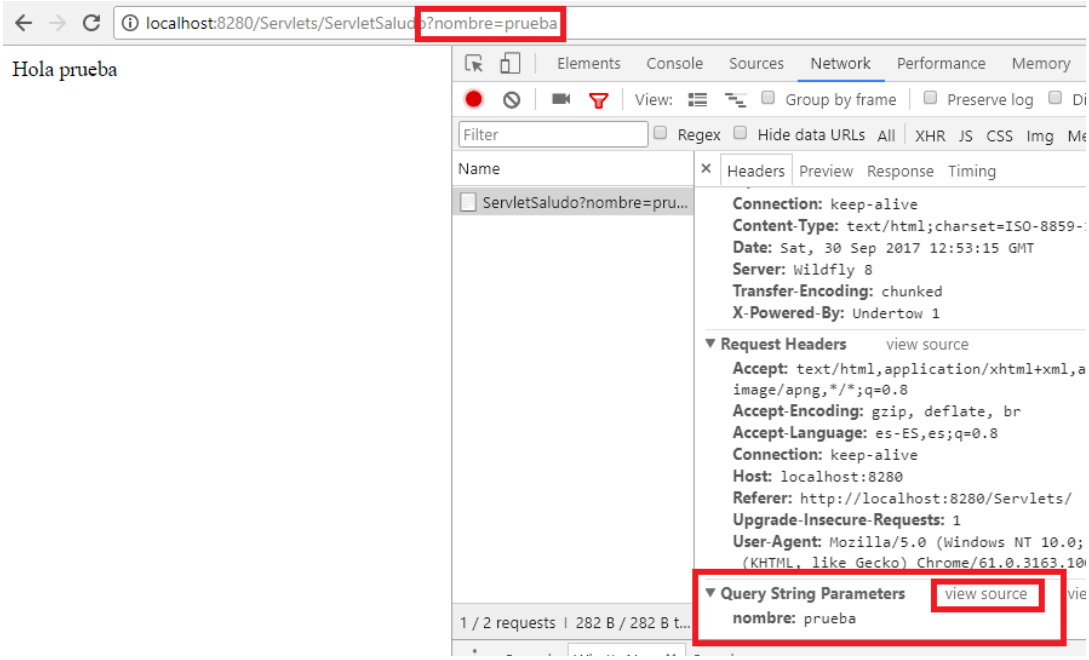
Cuando el servidor este iniciado veremos un mensaje "WildFly started" en la pestaña **Console**, desde la pestaña **Servers** podemos gestionar el servidor.



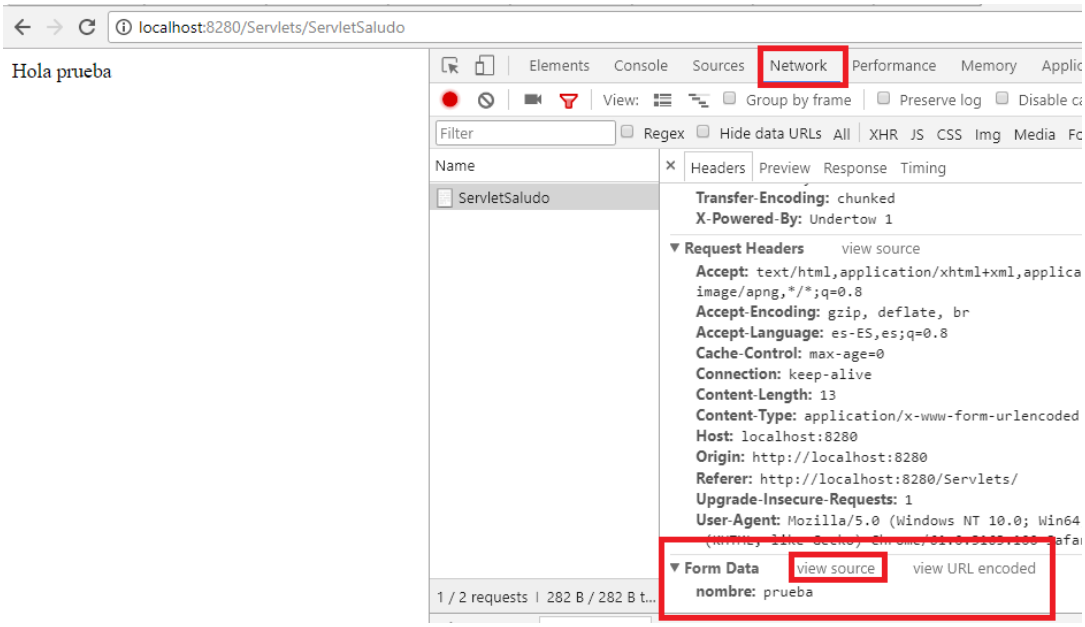
Una vez desplegado abrimos la aplicación en **Chrome** http://localhost:8280/Servlets/ .



Analizamos las peticiones de ambos formularios con la herramienta **Network del Chrome** (F12 para entrar). Los parámetros GET se envían en la propia URL



En cambio, los de la POST se envían en el cuerpo.



2.1 Los Servlets son multihilo

Cada vez que el servidor accede al servlet este se ejecuta, si se reciben varias peticiones estas se ejecutaran de forma simultánea en hilos diferentes no obstante ambos hilos se ejecutan sobre la misma instancia del Servlet, para verificar este funcionamiento podemos incluir un "Sleep" e imprimir la ID del hilo, si realizamos varias peticiones veremos que se trata de hilos diferentes. En el método ***doGet()*** del **servlelSaludo** añadimos el siguiente bloque de código.

```
@WebServlet("/ServletSaludo")
public class ServletSaludo extends HttpServlet {
    private static final long serialVersionUID = 1L;
    int contador = 0;

    public ServletSaludo() {
        super();
    }

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

 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
   out.println("<HTML>");
  out.println("<HEAD><TITLE>Hola Mundo!</TITLE></HEAD>");
  out.println("<BODY>");
  String nombre = (String) request.getParameter("nombre");
  if (nombre != null) {
      out.println("Hola " + nombre + "<br>");
  }
 try {
Thread.sleep(15000);
} catch (InterruptedException e) {}
out.println("ID del hilo:"+Thread.currentThread().getId()+"<br>");
contador++;
out.println("Visitas:"+contador+"<br>");
out.println("</BODY></HTML>");
```

```
}
```

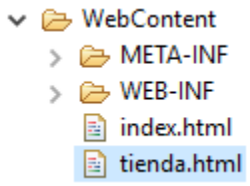
Sí volvemos a hacer un **Run** sobre la aplicación se desplegará una nueva versión con los cambios realizados. Accedemos desde dos pestañas a la URL http://localhost:8280/Servlets/ServletSaludo?nombre=prueba y comprobamos que los servlets se ejecutan en diferentes hilos.

Hola prueba
ID del hilo:232
Visitas:2

2.2 Manejo de sesión

Los servlets proporcionan una solución simple para **el seguimiento de sesiones de usuario** basada en la clase HttpSession de la API JEE . Empleando esta clase podremos identificar de manera diferenciada las sesiones de usuarios y tener la posibilidad de guardar información (objetos) asociada a cada usuario de forma independiente.

Para ilustrar el uso de la clase HttpSession vamos a ejemplo un pequeño ejemplo consistente en la típica tienda de la compra. Creamos un nuevo fichero **tienda.html** en la carpeta **/WebContent** agregando el siguiente contenido:




```
<html lang="en">
<head>
  <title>Servlets</title>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<!-- Contenido -->
<div class="container" id="contenedor-principal">
    <h2>Productos</h2>
    <div class="row ">
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
                <img src="https://s1.postimg.org/1hmgm474yn/manzana.png"    />
                <div>Manzanas</div>
                <a href="incluirEnCarrito?producto=manzanas" class="btn btn-default" >
                    2.05 €
                </a>
            </div>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
```

```
                <img src="https://s1.postimg.org/8rxjx64xpr/fresa.png"  />
                <div>Fresas</div>
                <a href="incluirEnCarrito?producto=fresas" class="btn btn-default" >
                    2.20 €
                </a>
            </div>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
                <img src="https://s1.postimg.org/2vlr4bi9rz/naranja.png" />
                <div>Naranjas</div>
                <a href="incluirEnCarrito?producto=naranjas" class="btn btn-default" >
                    2.10 €
                </a>
            </div>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
                <img src="https://s1.postimg.org/41sayqzdsv/pan.png"/>
                <div>Pan</div>
                <a href="incluirEnCarrito?producto=pan" class="btn btn-default" >
                    0.80 €
                </a>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

Cada producto lanza una petición **/incluirEnCarrito** (de tipo GET ya que se trata de un enlace) enviándole un parámetro de clave producto con el identificador único del producto.

Para recibir esta petición URL vamos a crear un nuevo Servlet, **ServletCarrito**, al que le asociaremos la URL "**incluirEnCarrito**".

```
package com.sdi;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession
@WebServlet("/incluirEnCarrito")
public class ServletCarrito extends HttpServlet {
        private static final long serialVersionUID = 1L;
```

Implementamos la función **doGet() en donde seguiremos los siguientes pasos:**

- Obtenemos el carrito guardado en sesión
- Sí no hay carrito - se trata de un nuevo usuario, instanciamos un nuevo carrito y lo insertamos en sesión. El carrito será un HashMap<String,Integer> donde guardaremos como clave (String) el nombre del producto y como valor (Integer) el número de unidades compradas.

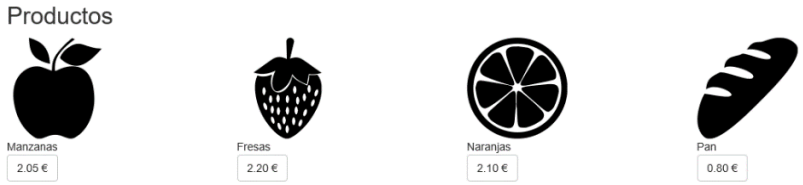
Insertamos el producto dentro del carrito y mostramos el contenido del carrito.

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

        HttpSession session=request.getSession();

        HashMap<String,Integer> carrito =
                 (HashMap<String,Integer>) request.getSession().getAttribute("carrito");

        // No hay carrito, creamos uno y lo insertamos en sesión
        if (carrito == null) {
                carrito = new HashMap<String,Integer>();
            request.getSession().setAttribute("carrito", carrito);
        }

        String producto = request.getParameter("producto");
        if ( producto != null){
                insertarEnCarrito(carrito, producto);
        }

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Tienda SDI: carrito</TITLE></HEAD>");
        out.println("<BODY>");
        out.println(carritoEnHTML(carrito)+"<br>");
        out.println("<a href=\"tienda.html\">Volver</a></BODY></HTML>");
}
```

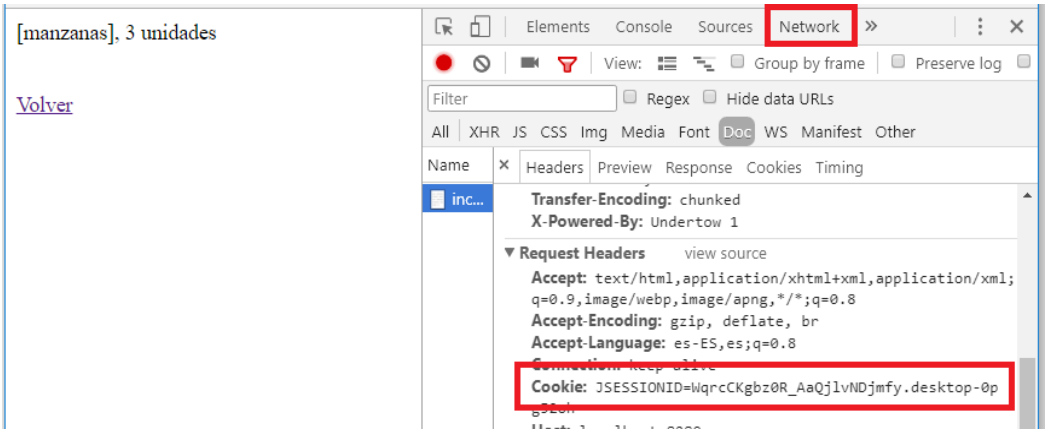
Los métodos auxiliares que completan el servlet son:

```
private void insertarEnCarrito(Map<String,Integer> carrito, String claveProducto) {
        if (carrito.get(claveProducto)==null)
                carrito.put(claveProducto, new Integer(1));
        else {
                int numeroArticulos=(Integer)carrito.get(claveProducto).intValue();
                carrito.put(claveProducto,new Integer(numeroArticulos+1));
        }
}

private String carritoEnHTML(Map<String,Integer> carrito) {
        String carritoEnHTML="";

        for (String key:carrito.keySet())
                carritoEnHTML+="<p>["+key+"], "+carrito.get(key)+" unidades</p>";
        return carritoEnHTML;
}
```

Ejecutamos la aplicación y comprobamos el correcto funcionamiento del carrito (accedemos a la web desde dos navegadores diferentes) http://localhost:8280/Servlets/tienda.html



Desde el analizador de peticiones de Chrome comprobamos que todas las peticiones contienen el ID de sesión que identifica al usuario.



En el código anterior se pueden producir **problemas de sincronización** entre diferentes hilos correspondientes a peticiones del mismo usuario (realizadas desde varias instancias del mismo navegador) en la manipulación de los objetos que se extraen, se actualizan y se insertan en el objeto sesión. Es necesario determinar qué problemas son estos y solucionarlos sincronizando las porciones de código problemáticas utilizando o bien:

```
synchronized(session) { … }
```

o bien una estructura sincronizada tal como:

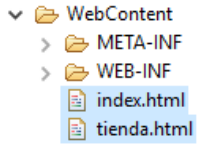
```
ConcurrentHashMap o SyncronizedMap[1]
```

teniendo en cuenta que las porciones de código sincronizadas deben tener el tamaño mínimo (sincronización de granularidad lo más fina posible) imprescindible para maximizar el rendimiento de la aplicación.

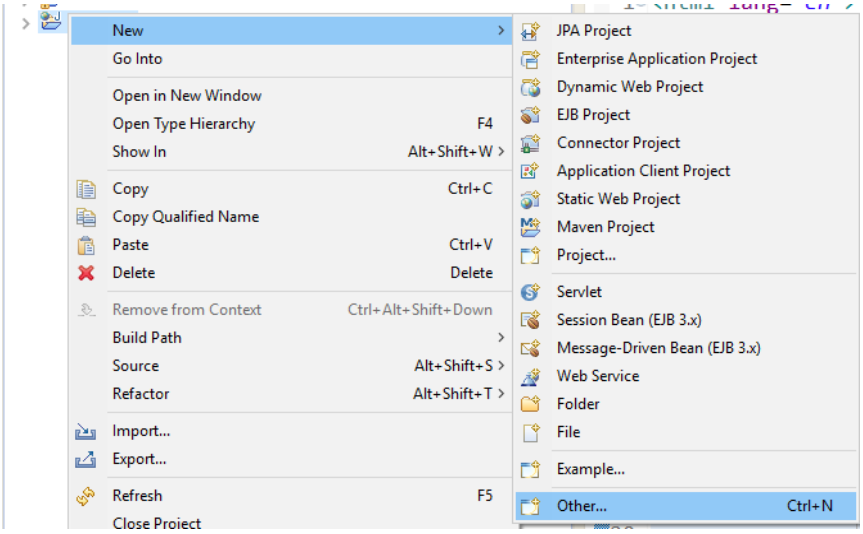
3 JSP Java Server Pages

Vamos a implementar ahora la misma idea que hemos realizado con Servlets pero con JSPs. Sobre el mismo proyecto, eliminamos los dos ficheros **WebConcent/WEB-INF index.html** y **tienda.html** .

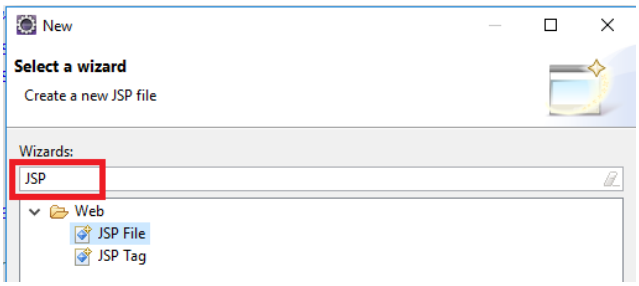
[1] https://dzone.com/articles/java-7-hashmap-vs



A continuación creamos un nuevo fichero JSP al que llamaremos **index.jsp**. Desde **New -> Other…**



Buscamos el fichero de tipo **JSP File.**



Llamaremos al fichero **index.jsp** que por defecto se guardará dentro de la carpeta **/WebContent**

Abrimos el fichero **index.jsp** y copiamos el siguiente contenido HTML, es igual al que utilizamos en la Web anterior, pero cuenta con la directiva JSP en la cual hemos especificado utf-8 como formato (por defecto al crear ficheros JSP utiliza otro formato), y las peticiones se realizan contra el servlet **ServletCarrito**.

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
  <title>JSP</title>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
```

```
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<!-- Contenido -->
<div class="container" id="contenedor-principal">
    <h2>Productos</h2>
    <div class="row ">
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
                <img src="https://s1.postimg.org/1hmgm474yn/manzana.png"   />
                <div>Manzanas</div>
                <a href="incluirEnCarrito?producto=manzanas" class="btn btn-default" >
                    2.05 €
                </a>
            </div>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
                <img src="https://s1.postimg.org/8rxjx64xpr/fresa.png"  />
                <div>Fresas</div>
                <a href="incluirEnCarrito?producto=fresas" class="btn btn-default" >
                    2.20 €
                </a>
            </div>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
                <img src="https://s1.postimg.org/2vlr4bi9rz/naranja.png" />
                <div>Naranjas</div>
                <a href="incluirEnCarrito?producto=naranjas" class="btn btn-default" >
                    2.10 €
                </a>
            </div>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
            <div>
                <img src="https://s1.postimg.org/41sayqzdsv/pan.png"/>
                <div>Pan</div>
                <a href="incluirEnCarrito?producto=pan" class="btn btn-default" >
                    0.80 €
                </a>
            </div>
        </div>
    </div>
</div>

</body>
</html>
```

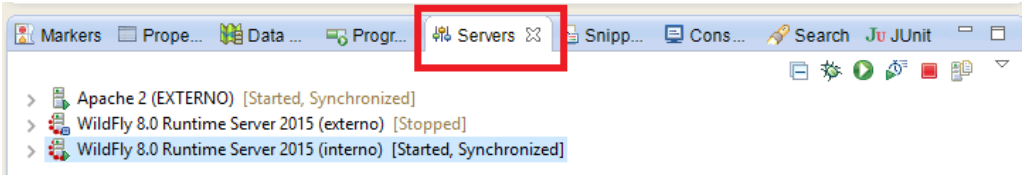
Modificamos la redirección final de la función **doGet()** en **ServletCarrito** para que nos retorne a **index.jsp** .

```
        out.println(carritoEnHTML(carrito)+"<br>");
        out.println("<a href=\"index.jsp\">Volver</a></BODY></HTML>");
}
```

Accedemos a http://localhost:8280/Servlets/ o http://localhost:8280/Servlets/index.jsp para probar la modificación.

Sí se sigue cargando la versión anterior debido a la cache del navegador Actualiza la página o abre una "Ventana de incognito"

Desplegamos la aplicación y comprobamos que el funcionamiento es correcto. Se recomienda utilizar más de un navegador para probar diferentes sesiones. Podemos restablecer todas las sesiones parando el servidor y volviendo a arrancarlo desde la pestaña Servers (la sesión también se puede hacer expirar desde código o eliminando la cookie correspondiente a la sesión en la opción "Privacidad" del navegador)



3.1 Manejo de sesión (2)

Un uso muy común de la sesión es la portabilidad de datos de un usuario de una página a otra. Un ejemplo típico es la identificación de usuarios en diferentes páginas. Creamos un nuevo fichero **login.jsp**, donde vamos a implementar un formulario que solicite un nombre y password al usuario.

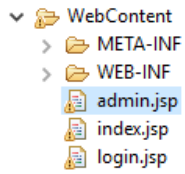
```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
  <title>JSP</title>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<!-- Contenido -->
<div class="container" id="contenedor-principal">
        <h2>Identificación de usuario</h2>

        <form class="form-horizontal" method="post" action="login.jsp">
          <div class="form-group">
            <label class="control-label col-sm-2" for="nombre">Nombre:</label>
            <div class="col-sm-10">
              <input type="text" class="form-control" name="nombre" required="true"/>
            </div>
          </div>
          <div class="form-group">
```

```
            <label class="control-label col-sm-2" for="password">Password:</label>
            <div class="col-sm-10">
              <input type="password" class="form-control" name="password"
                required="true"/>
            </div>
          </div>
          <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
              <button type="submit" class="btn btn-primary">Agregar</button>
            </div>
          </div>
        </form>
</div>

</body>
</html>
```

Utilizando las etiquetas <% %> podemos introducir código Java en la JSP. Como el formulario se envía contra la propia página **login.jsp** comprobamos, si los parámetros **nombre** y **password** coinciden con ***"admin",*** en ese caso introducimos un atributo en sesión con la clave **usuario**.

```
<body>

<%
        String nombre = request.getParameter("nombre");
        String password = request.getParameter("password");

        if ( nombre != null && nombre.equals("admin") &&
                        password != null && password.equals("admin")){

                // Credencial valido, lo guardo en sesión
                request.getSession().setAttribute("usuario", "admin");
                response.sendRedirect("admin.jsp");
        } else {
                // Credencial invalido, lo elimino de sesion (opcional)
                request.getSession().setAttribute("usuario", null);
        }
%>

<!-- Contenido -->
<div class="container" id="contenedor-principal">
```

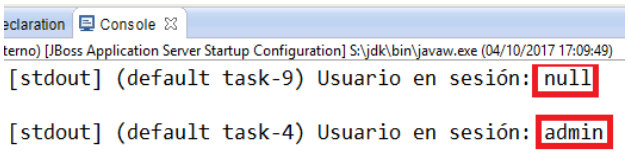
Cuando el usuario se identifica correctamente lo enviamos a **admin.jsp**. Vamos a crear un fichero **admin.jsp** , donde vamos a comprobar que la sesión tiene un atributo usuario con valor admin, si no es así retornamos al usuario al **login.jsp**.



```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
  <title>JSP</title>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<%
        String usuario = (String) request.getSession().getAttribute("usuario");
        System.out.println("Usuario en sesión: "+usuario);
        if ( usuario == null || usuario.equals("admin") == false ){
                // No hay usuario o no es admin
                response.sendRedirect("login.jsp");
        }
%>
<!-- Contenido -->
<div class="container" id="contenedor-principal">
        <h2>Administrar</h2>
</div>

</body>
</html>
```

Ejecutamos la aplicación y probamos a acceder directamente a http://localhost:8280/Servlets/admin.jsp que debería redireccionarnos a la página principal.

En cambio, si entramos en http://localhost:8280/Servlets/login.jsp y nos identificamos correctamente se guardará un usuario en sesión y nos dejará acceder a **admin.jsp** sin problemas. La salida por consola nos muestra el usuario identificado.



3.2 Contexto de la aplicación

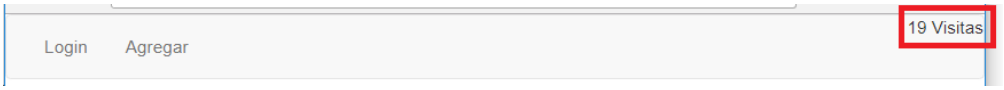
Vamos a incluir un contador que muestre el número de visitas totales, para ello utilizaremos la variable **application** (nos permite compartir datos en la aplicación con todos los usuarios)

Funciona de forma muy similar a la sesión, ya que para gestionar los atributos utiliza los métodos: **application.getAttribute(clave)** y **application.setAttribute(clave,valor).**

Incluimos el siguiente fragmento (en amarillo) en **index.jsp** antes de la definición del .

```
<%
        Integer contador = (Integer) application.getAttribute("contador");

        if (contador == null) {
                contador = new Integer(0);
        }
        application.setAttribute("contador", contador.intValue() + 1);
%>

<!-- Barra de Navegación superior -->
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <ul class="nav navbar-nav">
      <li><a href="incluirEnCarrito">Carrito</a></li>
      <li><a href="login.jsp">Login</a></li>
      <li><a href="admin.jsp">Administrar productos</a></li>
    </ul>
    <div class="nav navbar-right">
      <%=contador%> Visitas
    </div>
  </div>
</nav>

<!-- Contenido -->
<div class="container" id="contenedor-principal">

    <h2>Productos</h2>
    <div class="row ">
…….
```

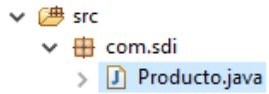
Ejecutamos la aplicación y comprobamos (desde varios navegadores) que el contador funciona correctamente.



3.3 Listado dinámico de productos

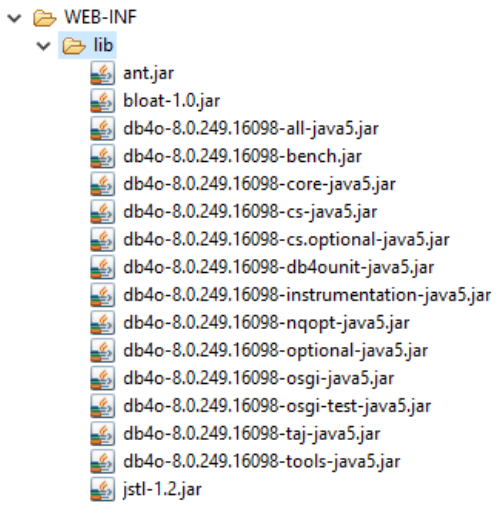
En lugar de listar los productos con código HTML estático vamos a obtenerlos de una base de datos e insertarlos dinámicamente en el código HTML de **index.jsp**.

Creamos la clase **Producto** en el paquete **com.sdi.**



```
public class Producto {
	private String nombre;
	private String imagen;
	private float precio;

	public Producto(String nombre, String imagen, float precio) {
		this.nombre = nombre;
		this.imagen = imagen;
		this.precio = precio;
	}

	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getImagen() {
		return imagen;
	}
	public void setImagen(String imagen) {
		this.imagen = imagen;
	}
	public float getPrecio() {
		return precio;
	}
	public void setPrecio(float precio) {
		this.precio = precio;
	}

}
```

Descargamos el fichero **PL-SDI-Material1.zip y copiamos todos los archivos jar** en el directorio **/WebContent/WEB-INF/lib/**



Creamos la clase **ProductosService** dentro de ella dos métodos, uno para retornar una lista con todos los productos en la base de datos (**getProductos**) y otro para agregar un nuevo producto (**setNuevoProducto**). Usamos funciones con nombres **get/set** para que en el futuro puedan ser utilizadas desde un JSP Bean.

```
import java.util.LinkedList;
import java.util.List;

import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;

public class ProductosService {

    public List<Producto> getProductos(){
        List<Producto> productos = new LinkedList<Producto>();

        ObjectContainer db = null;
        try {
            db = Db4oEmbedded.openFile("bdProductos");
            List<Producto> respuesta = db.queryByExample(Producto.class);
            // NO RETORNAR LA MISMA LISTA DE LA RESPUESTA
            productos.addAll(respuesta);

        } finally {
            db.close();
        }

        return productos;

    }

    public void setNuevoProducto(Producto nuevoProducto){
        ObjectContainer db = null;
        try {
            db = Db4oEmbedded.openFile("bdProductos");
            db.store(nuevoProducto);

        } finally {
            db.close();
        }
    }

}
```

Volvemos a **index.jsp** y eliminamos todo el anterior en el que la lista de productos se especificaba en el propio HTML. Ahora obtendremos una instancia de **ProductosService** y recorremos la lista que nos retorna (aunque la base de datos está actualmente vacía). Intercalamos el código Java con el código HTML.

```
<!-- Contenido -->
<div class="container" id="contenedor-principal">

    <h2>Productos</h2>
    <div class="row ">

        <%
                List<Producto> listaProductos = new ProductosService().getProductos();
                for(Producto producto : listaProductos){
        %>
      <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
          <div>
              <img src="<%=producto.getImagen() %>"   />
              <div><%=producto.getNombre() %></div>
              <a href="incluirEnCarrito?producto=<%=producto.getNombre() %>" class="btn btn-
default" >
                  <%=producto.getPrecio() %> €
              </a>
```

```
            </div>
        </div>
          <%
                }
          %>
      </div>

</div>
```

Incluimos los imports necesarios en **index.jsp**

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ page language="java" import="com.sdi.* , java.util.List"%>
```

Antes de probar esta funcionalidad incluiremos un mecanismo para poder agregar productos a la tienda.

3.4 Agregar un producto a la tienda

Modificaremos el contenido de **admin.jsp** incluyendo un formulario que solicite el nombre, imagen (URL) y precio de un producto, estos datos se enviarán contra **POST /admin.jsp**.

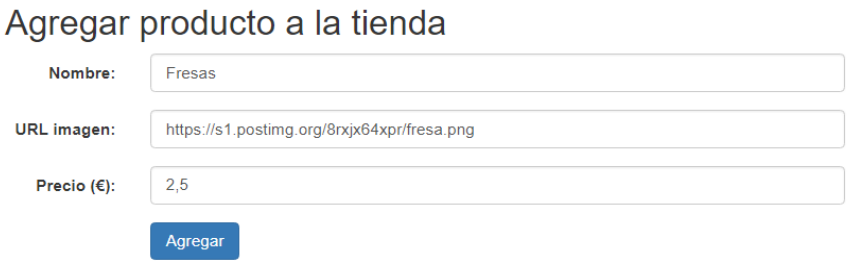
```
<!-- Contenido -->
<div class="container" id="contenedor-principal">

        <h2>Agregar producto a la tienda</h2>
        <form class="form-horizontal" method="post" action="admin.jsp">
          <div class="form-group">
            <label class="control-label col-sm-2" for="nombre">Nombre:</label>
            <div class="col-sm-10">
              <input type="text" class="form-control" name="nombre" required="true"/>
            </div>
          </div>
          <div class="form-group">
            <label class="control-label col-sm-2" for="imagen">URL imagen:</label>
            <div class="col-sm-10">
              <input type="text" class="form-control" name="imagen" required="true"/>
            </div>
          </div>
          <div class="form-group">
            <label class="control-label col-sm-2" for="precio">Precio (€):</label>
            <div class="col-sm-10">
              <input type="number" step="0.01" class="form-control" name="precio"
                required="true"/>
            </div>
          </div>
          <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
              <button type="submit" class="btn btn-primary">Agregar</button>
            </div>
          </div>
        </form>

</div>
```

Vamos a incluir dentro de **admin.jsp** un segundo script de código Java que obtenga los parámetros de la petición, construya un objeto Producto y lo agregue a través de **ProductosService (en amarillo)**.

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ page language="java" import="com.sdi.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"><html lang="en">
<head>
  <title>JSP</title>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<%
        String usuario = (String) request.getSession().getAttribute("usuario");
        System.out.println("Usuario en sesión: "+usuario);
        if ( usuario == null || usuario.equals("admin") == false ){
                // No hay usuario o no es admin
                response.sendRedirect("login.jsp");
        }
%>
<%
        if ( request.getParameter("nombre") != null &&
                        request.getParameter("imagen") != null &&
                        request.getParameter("precio") != null){

                String nombre = (String) request.getParameter("nombre");
                String imagen = (String) request.getParameter("imagen");
                float precio = Float.parseFloat(request.getParameter("precio"));

                Producto producto = new Producto(nombre, imagen, precio);
                new ProductosService().setNuevoProducto(producto);
                request.getRequestDispatcher("index.jsp").forward(request, response);
        }

%>
```

Ejecutamos la aplicación, accedemos a http://localhost:8280/Servlets/admin.jsp y agregamos algunos de los siguientes productos.

- Manzanas , https://s1.postimg.org/1hmgm474yn/manzana.png , 3
- Fresas , https://s1.postimg.org/8rxjx64xpr/fresa.png , 2.5
- Naranjas , https://s1.postimg.org/2vlr4bi9rz/naranja.png , 2.10
- Pan , https://s1.postimg.org/41sayqzdsv/pan.png , 0.80



El fichero correspondiente a la base de datos se genera en la primera ejecución, en **\entorno-sdi\wildfly\bin\bdProductos**

JSP Beans

Son clases Java construidas en base a unas especificaciones:

- Constructor por defecto sin argumentos
- Tienen "propiedades" (atributos) que pueden ser: leídas y/o escritas.
- Se manejan a través de los métodos **get** y **set** de sus propiedades.

Un uso muy común de los **Beans** en JSP es la recuperación de datos de formularios, es decir formar un Objeto a partir de los parámetros recibidos.

Para que producto pueda ser utilizado como un Bean le tenemos que agregar un constructor sin argumentos.

```
public class Producto {
        private String nombre;
        private String imagen;
        private float precio;

        public Producto(){

        }
```

Modificamos **admin.jsp,** con la etiqueta **jsp:useBean** se crea un Bean nuevo de tipo Producto, con el nombre de variable "producto" .

Con la etiqueta **setProperty (con property=*)** analizar todos los parámetros de la request y los guarda en las propiedades del Bean (las que tengan el mismo nombre, se hace de forma automática). Sustituimos el código anterior por el Bean.

El **producto** siempre va a contener un objeto != null ya que se crea automáticamente con el jsp:useBean, por lo tanto no vale comprobar *producto != null,* debemos comprobar si sus propiedades.

```
<jsp:useBean id="producto" class="com.sdi.Producto" />
<jsp:setProperty name="producto" property="*"/>
<%
        if( producto.getNombre() != null){
                new ProductosService().setNuevoProducto(producto);
                request.getRequestDispatcher("index.jsp").forward(request, response);
        }
%>
<%
        if ( request.getParameter("nombre") != null &&
                        request.getParameter("imagen") != null &&
                        request.getParameter("precio") != null){

                String nombre = (String) request.getParameter("nombre");
                String imagen = (String) request.getParameter("imagen");
                float precio = Float.parseFloat(request.getParameter("precio"));

                Producto productoNuevo = new Producto(nombre, imagen, precio);
                new ProductosService().agregarProducto(productoNuevo);
                request.getRequestDispatcher("index.jsp").forward(request, response);
        }

%>
<!-- Contenido -->
<div class="container" id="contenedor-principal">
```

Con estos cambios la aplicación debería funcionar de la misma forma.

JavaBeans y ámbitos

Las directivas de JSP para manejar Beans nos permite crear/obtener y modificar de forma sencilla las propiedades de un objeto desde una JSP. Vamos a crear un nuevo contador y a utilizarlo como un Bean.

En primer lugar, creamos la clase **Contador** en el paquete **com.sdi.**

```
public class Contador {
        private int total;

        public int getTotal() {
                return total;
        }

        public void setIncremento(int incremento) {
                total+=incremento;
        }
}
```

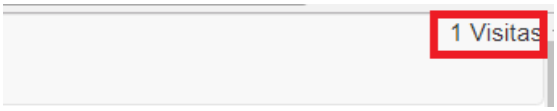
Sustituimos el anterior script Java por el nuevo Bean.

- Incluimos el Bean en la página **index.jsp**
- Mostramos el valor del **total** - **getProperty(property)**
- Establecemos el **incremento** de 1. – **setProperty(property , value)**

```
<%
        Integer contador = (Integer) application.getAttribute("contador");

        if (contador == null) {
                contador = new Integer(0);
```

```
        }
        application.setAttribute("contador", contador.intValue() + 1);
%>

<jsp:useBean id="contador" class="com.sdi.Contador"/>
<jsp:setProperty name="contador" property="incremento" value="1"/>

<!-- Barra de Navegación superior -->
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <ul class="nav navbar-nav">
      <li><a href="carrito.jsp">Carrito</a></li>
      <li><a href="login.jsp">Login</a></li>
      <li><a href="administrar.jsp">Administrar productos</a></li>
    </ul>
    <div class="nav navbar-right">
      <div class="center-block"> <%=contador%> <jsp:getProperty name="contador"
property="total"/> Visitas </div>
    </div>
  </div>
</nav>
```

Ejecutamos el proyecto y comprobamos que el contador siempre marca 1 **¿Qué está sucediendo?**



Por defecto el Bean tiene un ámbito (scope) de página, cada vez que se abre la página se crea el Bean (y el objeto Contador). Podemos modificar el ámbito (scope) de los Bean. Los posibles valores del atributo scope son: **page|request|session|application**

Si queremos hacer un contador para todos los usuarios el ámbito del Bean sería "**application**".

Añadimos el atributo **scope** a la etiqueta **useBean**.

```
<jsp:useBean id="contador" class="com.sdi.Contador" scope="application"/>
```

Volvemos a desplegar la aplicación y la probamos de nuevo.

3.5 Uso de tags JSTL Core

Las etiquetas JSTL encapsulan gran parte de funcionalidad común que se suele requerir en las páginas web JSP. Usando estas etiquetas se evita incluir scripts de código Java propios. http://docs.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/ Para usar JSTL necesitamos incluir la librería **jstl.jar** en el directorio **WebContent/WEB-INF/lib ,** en nuestro caso ya

la habíamos movido a ese directorio al copiar las librerías de la base de datos (la versión 1.2 de jstl).[2]

Para utilizar las etiquetas de JSTL debemos declarar el uso de JSTL, agregamos la directiva al inicio de **index.jsp.**

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" import="com.sdi.* , java.util.List"%>
```

Vamos a sustituir el código Java que recorría la lista de productos mediante etiquetas JSTL, <c:forEach> permite recorrer elementos de una lista. Para obtener la lista utilizamos **productosService** como un Bean, obtendremos los artículos a través de productos (se invoca por detrás a getProductos).

El elemento que se está recorriendo actualmente se guarda en la variable declarada en el atributo "var" en este caso **producto; y** podemos acceder a sus atributos **#{producto.<nombre_propiedad>}** para imprimir por pantalla el valor de la variable podemos utilizar <c:out>

Tras estos cambios el nuevo código del pasará a ser el siguiente.

```
<!-- Contenido -->
<div class="container" id="contenedor-principal">

<h2>Productos</h2>
<div class="row ">

    <jsp:useBean id="productosService" class="com.sdi.ProductosService"/>
    <c:forEach var="producto" begin="0" items="${productosService.productos}">
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-3">
           <div>
                <img src="<c:out value="${producto.imagen}"/>"   />
                <div><c:out value="${producto.nombre}"/></div>
                <a href="incluirEnCarrito?producto=<c:out value="${producto.nombre}"/>"
                    class="btn btn-default" >
                    <c:out value="${producto.precio}"/> €
                </a>
            </div>
        </div>
    </c:forEach>
</div>
</div>
```

Sí utilizamos Beans y JSTL podríamos llegar a prescindir de los scripts java tradicionales. Por ejemplo, en **admin.jsp** agregamos la directiva (taglib prefix="c") para poder usar JSTL mediante el prefijo "c:".

```
<%@ page language="java" contentType="text/html; charset=utf-8"
```

[2] Además para servidores que cumplen JEE 1.7 la librería jstl.jar ya está incluida en las librerías del servidor.

```
    pageEncoding="utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

Cambiamos el antiguo script utilizado para agregar el producto por uno que utilice JSTL y los JavaBean, la **directiva <c:if test="exp">** permite ejecución condicional.

```
<%
        String usuario = (String) request.getSession().getAttribute("usuario");
        System.out.println("Usuario en sesión: "+usuario);
        if ( usuario == null || usuario.equals("admin") == false ){
                // No hay usuario o no es admin
                response.sendRedirect("login.jsp");
        }
%>
<c:if test = "${sessionScope.usuario != 'admin'}">
        <c:redirect url="/login.jsp"/>
</c:if>

<jsp:useBean id="producto" class="com.sdi.Producto"/>
<jsp:setProperty name="producto" property="*"/>
<%
        if( producto.getNombre() != null){
                new ProductosService().setNuevoProducto(producto);
        request.getRequestDispatcher("index.jsp").forward(request, response);
        }
%>
<c:if test = "${producto.nombre != null}">
        <jsp:useBean id="productosService" class="com.sdi.ProductosService"/>
        <jsp:setProperty name="productosService" property="nuevoProducto"
value="${producto}"/>
        <c:redirect url="/index.jsp"/>
</c:if>
```

También podemos crear nuestras propias librerías de Tags, con funcionalidades personalizadas: http://docs.oracle.com/cd/E11035_01/wls100/taglib/quickstart.html

Sí quisiéramos completar la arquitectura de la aplicación el Carrito podría ser otro Bean y la lista de productos incluidos en el carrito se podría recorrer con JSTL

4 MVC, Modelo Vista Controlador

Vamos a incluir una implementación muy simple de una arquitectura MVC (Modelo-Vista - Controlador) para obtener los productos que hay en el carrito. **ServletCarrito** va a continuar respondiendo a la petición **/incluirEnCarrito** desde la función **doGet().** Este Servlet hará el papel de **controlador** una vez recibida la petición:

- Igual que antes -> Comprueba si hay carrito en sesión, comprueba si la petición contiene un producto
- Nuevo -> introduce el carrito como attributo en la request con la clave "paresCarrito"

- Nuevo -> redirije la petición a la vista **vista-carrito.jsp** (la vista puede utilizar la variable "paresCarrito" que acabamos de definir)

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

        HttpSession session=request.getSession();

        HashMap<String,Integer> carrito =
             (HashMap<String,Integer>) request.getSession().getAttribute("carrito");

        // No hay carrito, creamos uno y lo insertamos en sesión
        if (carrito == null) {
                carrito = new HashMap<String,Integer>();
            request.getSession().setAttribute("carrito", carrito);
        }

        String producto = request.getParameter("producto");
        if ( producto != null){
                insertarEnCarrito(carrito, producto);
        }

        // Retornar la vista con parámetro "carrito"
        request.setAttribute("paresCarrito", carrito);
        getServletContext().getRequestDispatcher("/vista-carrito.jsp").forward(request,
response);

}
```

Creamos la vista **vista-carrito.jsp**, en ella hacemos uso de las etiquetas de JSTL para recorrer los elementos de la hashmap "**paresCarrito**", como es una hashmap donde cada objeto tiene una **key** y un **value**.

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
  <title>JSP</title>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<!-- Contenido -->
<div class="container" id="contenedor-principal">
    <h2>Vista-Carrito</h2>
    <ul>
                <c:forEach var="par" items="${paresCarrito}">
                        <tr>
                                <li>${par.key} - ${par.value} </li>
                        </tr>
                </c:forEach>
     </ul>
</div>

</body>
</html>
```

5 Tareas

- Aplica la arquitectura MVC a la obtención de productos. Un servlet debe responder a la petición /productos, obtener todos los productos de la base de datos y almacenarlos en una variable “productosTienda” y abrir una vista vista-productos.jsp . Dentro de la vista se debe recorrer la lista productosTienda con etiquetas de JSTL.

- Implementa una aplicación web nueva similar a un blog, donde los usuarios puedan dejar comentarios (su nombre + comentario).