



# AUT02\_01. Aplicaciones básicas

## Sumario

<b>AUT02_01. APLICACIONES BÁSICAS.....</b>	<b>1</b>
<b>1. OBJETIVOS.....</b>	<b>3</b>
1.1 CREAR APLICACIÓN WEB BÁSICA CON MAVEN Y JAVA EE 5.....	3
1.2 MOSTRAR CABECERAS DE LA PETICIÓN.....	3
1.3 MOSTRAR UNA LISTA DE LINKS.....	3
1.4 CARRUSEL DE IMÁGENES.....	3
<b>2. DESARROLLO DE LA ACTIVIDAD.....</b>	<b>3</b>
2.1 APLICACIÓN WEB BÁSICA.....	3
2.1.1 Examinando el proyecto web de la aplicación básica.....	4
2.1.2 Ejecutando el proyecto Web.....	4
2.1.3 Creando el primer Servlet.....	5
2.1.4 Probando el servlet.....	5
2.1.5 Examinando las welcome files.....	5
2.1.6 Estableciendo un servlet como welcome file.....	6
2.1.7 Implementando el saludo.....	7
2.1.8 Problema.....	7
2.1.9 Solución.....	7
2.1.9.1 Añadimos la dependencia Maven.....	8



2.2 EJERCICIO DE MOSTRAR CABECERAS DE PETICIÓN.....	9
2.2.1 Dando estilo y “color” a la página.....	9
2.3 EJERCICIO 3. LISTA DE LINKS.....	9
2.4 EJERCICIO 4. CARRUSEL DE IMÁGENES.....	10



## 1. Objetivos

Se crearán varias aplicaciones Web para ver los conceptos básicos del desarrollo con la tecnología de Servlets

Estas aplicaciones consistirán en:

- Proyecto IntroServlets

### 1.1 Crear aplicación Web básica con Maven y Java EE 5

1. Crear servlet IndexServlet y establecerlo como welcome-file
2. Mostrar una bienvenida: **Bienvenido/a <nombre>** donde <nombre> será un parámetro de query string de clave *name* o se mostrará el valor “Anónimo/a”.
3. Lo anterior pero mostrando “<Anónimo/a>”

### 1.2 Mostrar cabeceras de la petición

1. Crear un servlet BrowseHeadersServlet donde se mostrarán los valores de las cabeceras de la petición.
2. Se mostrarán como una lista de definiciones <dl>. Los <dt> serán los nombres de las cabeceras y los <dd> sus valores.

### 1.3 Mostrar una lista de links

1. Crear un servlet AgendaServlet que renderizará una lista <ul> con links a:
  - i. <http://developer.mozilla.org>
  - ii. <http://www.w3c.org>
  - iii. <https://docs.oracle.com/javase/tutorial/>

### 1.4 Carrusel de imágenes

1. Crear un carrusel de imágenes, en base a un ejemplo de un sitio Web.



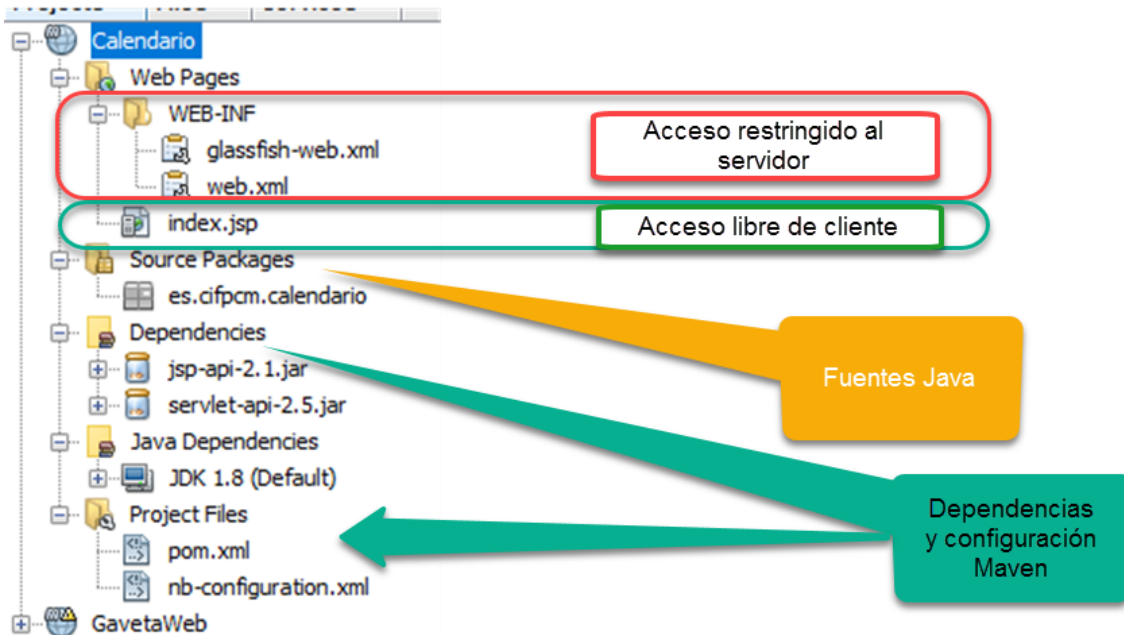
## 2. Desarrollo de la actividad

### 2.1 Aplicación Web básica

1. En Netbeans 8.1 seleccionamos: **File > New project... > Maven > Web Application**
2. En “Name and Location”
  1. Project Name: Calendario
  2. Group Id: es.cifpcm
3. En “Settings”
  1. Server: GlassFish Server 4.1.1 (o 4.1)
  2. Java EE Version: Java EE **5**

#### 2.1.1 Examinando el proyecto web de la aplicación básica

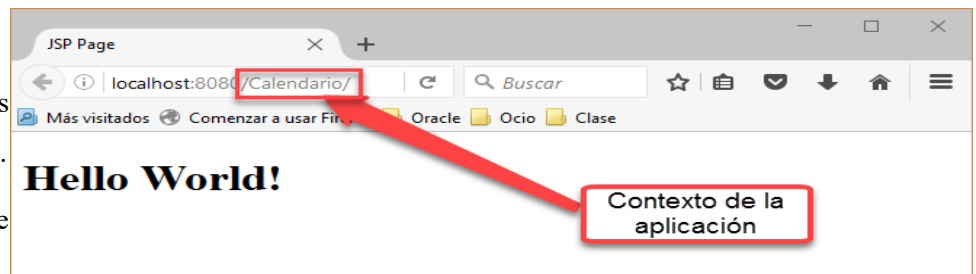
- Web Pages. Aquí se almacenan las páginas HTML, JSP, JSF, recursos estáticos (PNG, GIF, ...) que o bien se envían al cliente o bien se compilan en función de la extensión.
  - WEB-INF. Subcarpeta de la anterior no tiene acceso desde el cliente. Almacena meta-información del proyecto
- Source Packages. Aquí creamos los ficheros fuente de la aplicación (.Java)
- Dependencies. Librerías de terceras partes, normalmente empaquetadas como Jar.
- Project Files. Ficheros del proyecto Maven (herramienta de construcción de la aplicación).



### 2.1.2 Ejecutando el proyecto Web

Dando al botón de play verde (F6) ejecutamos el proyecto en el servidor GlassFish integrado.

Debería abrirse un navegador en la siguiente URL con el texto “Hello World!”



### 2.1.3 Creando el primer Servlet

Vamos a crear un primer *endpoint*, de tipo Servlet, que será invocado cuando desde la aplicación se haga una petición a la URLPattern establecida.

Nos situamos en “Web Pages” o “Source Packages” y: <RMB>+New > Servlet...

ClassName: IndexServlet

Package: es.cifpcm.calendario.welcome.web

**URL Patterns: /Index**



#### 2.1.4 Probando el servlet

Si ejecutamos de nuevo (F6) vuelve a salir la página anterior. Pero si ahora indicamos la URL: <http://localhost:8080/Calendario/Index> se nos mostrará la salida del Servlet. Esto es porque el servlet está mapeado a la URL Pattern /Index, a partir del contexto de la webapp.

#### 2.1.5 Examinando las welcome files

Las Welcome files permiten especificar un conjunto de ficheros que el contenedor añade a una URL que no “mapee” a un componente Web.

Por ejemplo si indico la dirección del sitio sin especificar un html: <http://localhost:8080>

Si tengo definida una welcome file de nombre **index.html**. El contenedor al no encontrar un componente Web en la dirección original, concatena este fichero a la ruta original y se devuelve el fichero en <http://localhost:8080/index.html>


Abrimos **web.xml**, localizado en Web Pages/WEB-INF. Este fichero es el descriptor de despliegue Web.

 <https://docs.oracle.com/javaee/7/tutorial/webapp005.htm#CHDHGJIA>



El contenido es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.
  <display-name>Calendario</display-name>
  <servlet>
    <servlet-name>IndexServlet</servlet-name>
    <servlet-class>es.cifpcm.calendario.welcome.web.IndexServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>IndexServlet</servlet-name>
    <url-pattern>/Index</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```



### 2.1.6 Estableciendo un servlet como welcome file

De paso conoceremos a BalusC, una de las personas que más sabe de desarrollo Web en java.

 <http://stackoverflow.com/questions/5624568/servlet-as-welcome-file-list-in-tomcat-7>

Vamos a modificar la welcome-file para que utilice nuestro servlet:

```
<welcome-file-list>

  <welcome-file>Index</welcome-file>

</welcome-file-list>
```



### 2.1.7 Implementando el saludo

El método `processRequest` define dos parámetros, la `request` y la `response` que representan la petición y respuesta HTTP respectivamente.

Con `request.getParameter` podemos leer los parámetros de `queryString`.

A través de `PrintWriter` escribimos la salida html (el cuerpo de la respuesta).

Podemos leer y modificar cabeceras de petición y respuesta a través de los métodos de `HttpServletRequest` y `HttpServletResponse`.

```
2.1.8 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet IndexServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet IndexServlet at " + request.getContextPath() + "</h1>");

        String name = request.getParameter("name");
        name = name == null ? "<Anónimo/a>" : name;

        out.println("<p>Bienvenido/a: " + name + "</p>");

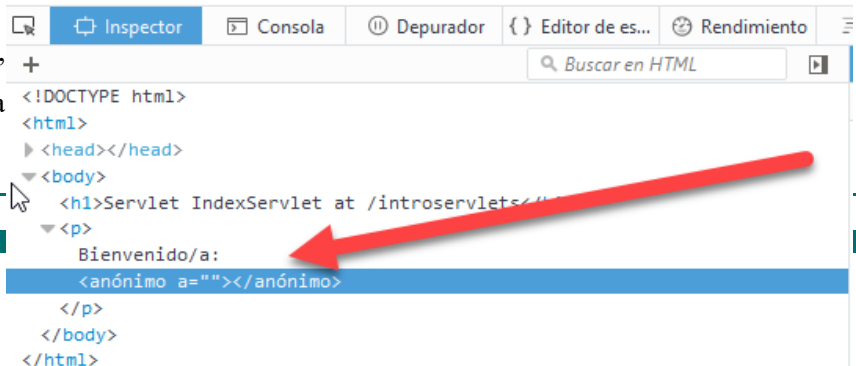
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

#### Problema

El programa no muestra correctamente el código, dado que la salida `<Anónimo/a>` la interpreta

## Servlet IndexServlet at /introservlets

Bienvenido/a:







incorrectamente el navegador. Éste cree que está tratando con un elemento HTML `<...>` y no muestra correctamente lo que queremos.

### 2.1.9 Solución

La solución es “escapar” los caracteres HTML en la salida.

Por ejemplo podemos utilizar las *entities* HTML para reemplazar los símbolos `<` por `&lt;` y `>` por `&gt;`;

El texto sería `&lt;Anónimo/a&gt;`;

Como esto es muy tedioso podemos utilizar la librería Apache Commons Lang que tiene una clase de utilidad para realizar esta labor.

#### 2.1.9.1 Añadimos la dependencia Maven

En dependencias: **<BDR>** + **Add dependency...**

Buscamos **org.apache.commons** y commons-lang3

Modificamos el código del servlet, usando: `StringEscapeUtils.escapeHtml4(name)`

The screenshot shows an IDE interface with a project tree on the left and a code editor on the right. The 'Add Dependency' dialog box is open, and the 'Group ID' is set to 'org.apache.commons', 'Artifact ID' is 'commons-lang3', and 'Version' is '3.4'. The 'Query' field also contains 'org.apache.commons'. The search results list several versions of commons-lang3, with '3.4 [jar] - central' selected. A red arrow points from the 'Dependencies' folder in the project tree to the 'Add Dependency' dialog box. The background shows a project structure with folders like 'Web Pages', 'WEB-INF', and 'Source Packages', and files like 'web.xml', 'index.jsp', and 'IndexServlet.java'.



**NOTA:** Quizá haga que hacer un **clean & build** del proyecto.

## 2.2 Ejercicio de mostrar cabeceras de petición

Es sencillo, basta buscar el método de request que devuelve una enumeración con los nombres de las cabeceras y recorrer con un bucle la enumeración buscando el valor de cada nombre encontrado.

Los pares nombre, valor se representarán mediante una *definition list* de HTML.

```
<dl>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
  <dt>Milk</dt>
  <dd>White cold drink</dd>
</dl>
```

[http://www.w3schools.com/tags/tag\\_dl.asp](http://www.w3schools.com/tags/tag_dl.asp)

### 2.2.1 Dando estilo y “color” a la página

Añadiremos una hoja de estilos CSS a la página. Para ello:

1. Añadiremos una carpeta **css** en “Web pages” y una hoja de estilos (main.css por ejemplo)
2. En el documento HTML haremos referencia a la hoja de estilos. Elemento **<link>**

[http://www.w3schools.com/tags/tag\\_link.asp](http://www.w3schools.com/tags/tag_link.asp)

**NOTA:** Para referenciar la hoja de estilos, en el atributo **href**, emplearemos una URL absoluta del tipo **/introservlets/css/main.css**

Debemos indicar **‘/’** + el nombre del contexto de nuestra aplicación y, a partir de ahí, la ruta al CSS.



## 2.3 Ejercicio 3. Lista de links

*It's up to you.*



## 2.4 Ejercicio 4. Carrusel de imágenes

Vamos a “fusilar” un ejemplo de W3Schools. La dirección del ejemplo es:

[http://www.w3schools.com/w3css/w3css\\_slideshow.asp](http://www.w3schools.com/w3css/w3css_slideshow.asp)

Tenemos que:

1. Replicar el HTML que podemos ver en el enlace “Try yourself” (fácil con PrintWriter).
2. El código javascript (dentro de `<script>` vamos a sacarlo a un fichero .js, en la ruta `/introservlets/js/carrusel.js`)
3. **Cuidadín.** En el contenido del js no van los elementos `<script> .... </script>`
4. EL código css lo copiamos del sitio y con él creamos un archivo `w3.css` en nuestra ruta de css.
5. En el ejemplo el código javascript se ejecuta cuando se encuentra el tag `<script>` (llamada al método `showDivs`). Como nosotros lo tenemos externalizado en un fichero aparte, tenemos que disparar de algún modo esa llamada. Lo haremos a través del atributo **onload** del elemento `<body>`.
6. Las imágenes podemos “fusilarlas” del sitio, al igual que las css y javascript.

NOTA: Crear un método para crear los elementos `<img>` de la lista, suministrando únicamente el nombre de la imagen como argumento String.

