



# Introducción a Java Servlets y JSP

Sesión Conceptual 1





# Inicio

{desafío}  
latam\_



15 minutos

- Entender la función de un formulario.
- Introducir a la tecnología JSP y JSTL.
- Instalar librerías JSTL.
- Aplicar envío de variables entre formularios.

## Objetivo



# Desarrollo

{desafío}  
latam\_

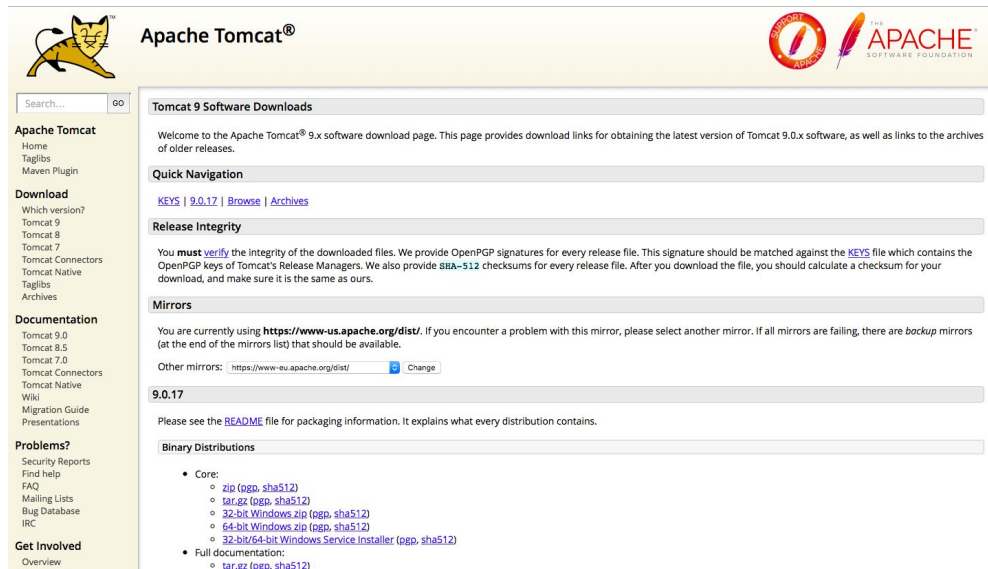


150 minutos

**/\* Configuración del entorno \*/**

# Instalación del contenedor Apache

- El contenedor de aplicaciones es el software que otorga los servicios necesarios a los artefactos que generamos en este curso.
  - Descargar el contenedor.



The screenshot shows the Apache Tomcat 9 Software Downloads page. The page has a yellow header with the Apache Tomcat logo and the Apache Software Foundation logo. Below the header, there is a search bar and a sidebar with navigation links. The main content area is titled "Tomcat 9 Software Downloads" and contains a welcome message, quick navigation links, release integrity information, mirrors, and binary distributions.

**Apache Tomcat®**

Tomcat 9 Software Downloads

Welcome to the Apache Tomcat® 9.x software download page. This page provides download links for obtaining the latest version of Tomcat 9.0.x software, as well as links to the archives of older releases.

**Quick Navigation**

[KEYS](#) | [9.0.17](#) | [Browse](#) | [Archives](#)

**Release Integrity**

You **must** [verify](#) the integrity of the downloaded files. We provide OpenPGP signatures for every release file. This signature should be matched against the [KEYS](#) file which contains the OpenPGP keys of Tomcat's Release Managers. We also provide [SHA-512](#) checksums for every release file. After you download the file, you should calculate a checksum for your download, and make sure it is the same as ours.

**Mirrors**

You are currently using <https://www-us.apache.org/dist/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are [backup](#) mirrors (at the end of the mirrors list) that should be available.

Other mirrors: <https://www-eu.apache.org/dist/> [Change](#)

**9.0.17**

Please see the [README](#) file for packaging information. It explains what every distribution contains.

**Binary Distributions**

- Core:
  - [zip \(pgp, sha512\)](#)
  - [tar.gz \(pgp, sha512\)](#)
  - [32-bit Windows zip \(pgp, sha512\)](#)
  - [64-bit Windows zip \(pgp, sha512\)](#)
  - [32-bit/64-bit Windows Service Installer \(pgp, sha512\)](#)
- Full documentation:
  - [tar.gz \(pgp, sha512\)](#)

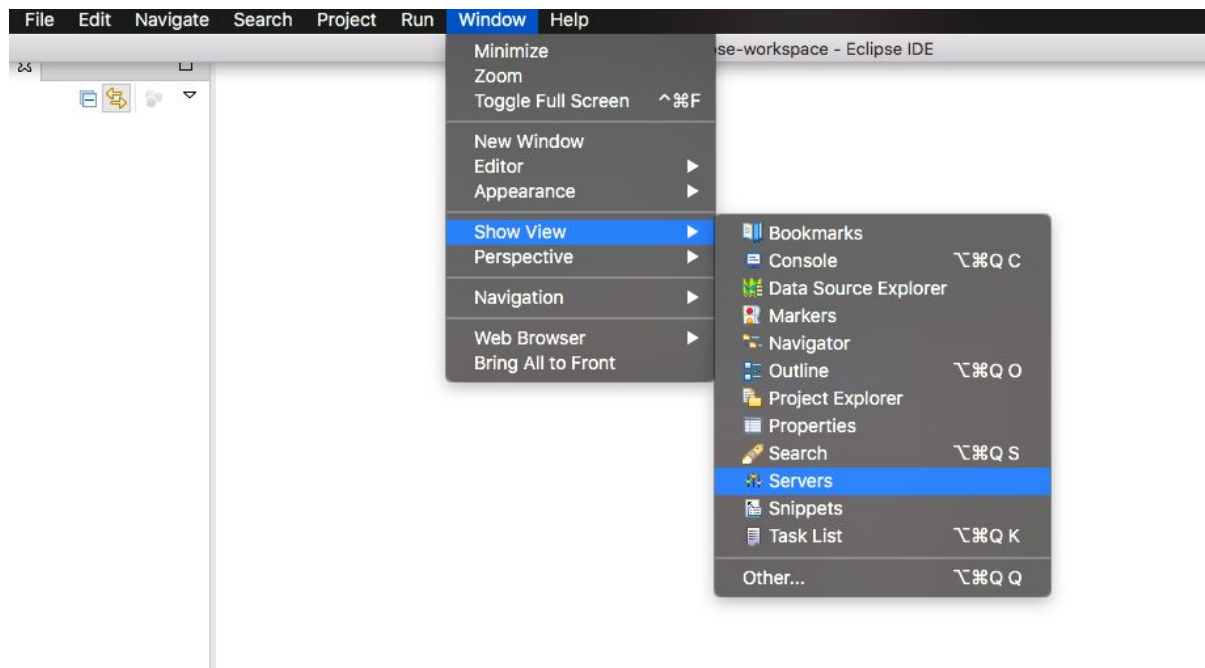


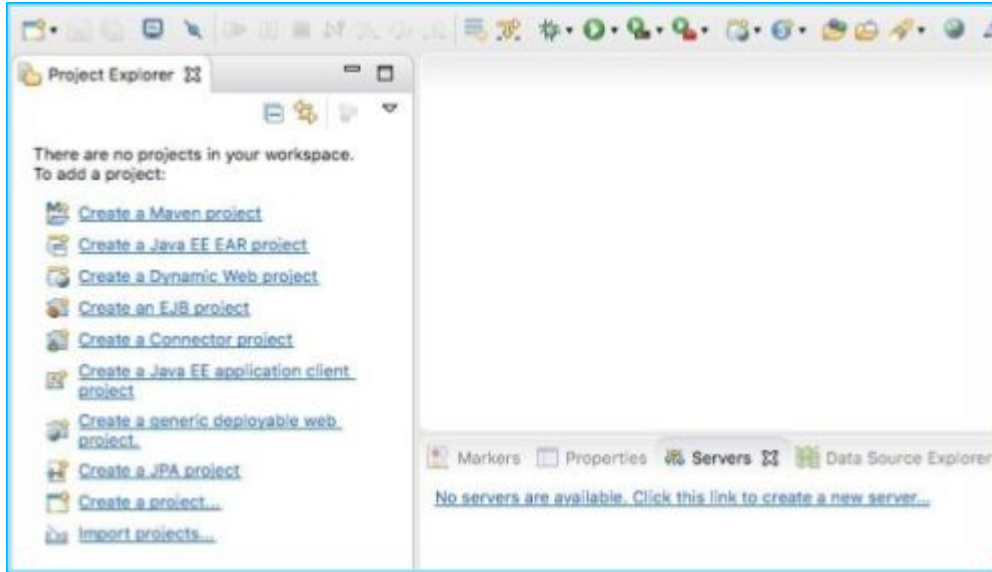
# Estructura de carpetas en Mac

Nombre	Fecha de modificación	Tamaño	Clase
bin	hoy 20:12	--	Carpeta
BUILDING.txt	13-03-2019 15:56	20 KB	Documento de texto sin formato
conf	hoy 20:12	--	Carpeta
CONTRIBUTING.md	13-03-2019 15:56	6 KB	Documento Visual Studio Code
lib	hoy 20:12	--	Carpeta
LICENSE	13-03-2019 15:56	58 KB	Documento TextEdit
logs	13-03-2019 15:55	--	Carpeta
NOTICE	13-03-2019 15:56	2 KB	Documento TextEdit
README.md	13-03-2019 15:56	3 KB	Documento Visual Studio Code
RELEASE-NOTES	13-03-2019 15:56	7 KB	Documento TextEdit
RUNNING.txt	13-03-2019 15:56	17 KB	Documento de texto sin formato
temp	hoy 20:12	--	Carpeta
webapps	13-03-2019 15:56	--	Carpeta
work	13-03-2019 15:55	--	Carpeta



- Configuraremos el contenedor directamente en Eclipse.
  - Windows -> Show View -> Servers.





Ningún servidor  
conectado

New Server

### Define a New Server

Choose the type of server to create

Select the server type:

type filter text

- Apache
  - Tomcat v3.2 Server
  - Tomcat v4.0 Server
  - Tomcat v4.1 Server
  - Tomcat v5.0 Server
  - Tomcat v5.5 Server
  - Tomcat v6.0 Server
  - Tomcat v7.0 Server
  - Tomcat v8.0 Server
  - Tomcat v8.5 Server
  - Tomcat v9.0 Server**

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Tomcat v9.0 Server at localhost

Server runtime environment: Apache Tomcat v9.0 [Add...](#)

[Configure runtime environments...](#)

[?](#) < Back Next > Cancel Finish

@ Seleccionamos la  
versión del  
servidor a utilizar

New Server

### Tomcat Server

Specify the installation directory

Name:  
Apache Tomcat v9.0

Tomcat installation directory:  
/Applications/contenedorPrincipal/apache-tomcat-9.0.17

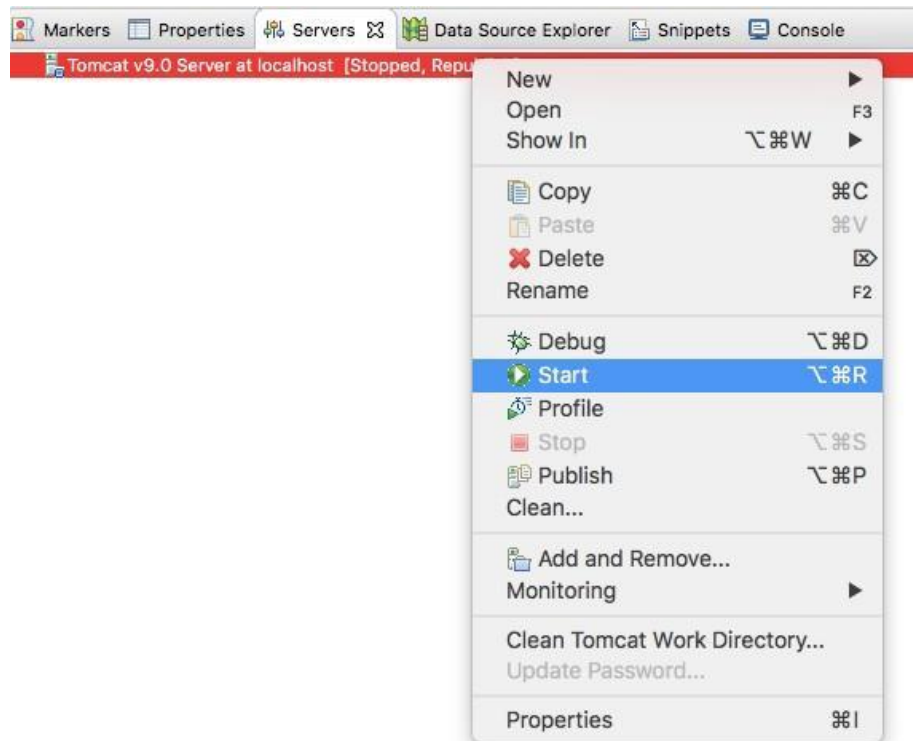
JRE:  
Workbench default JRE

< Back   Next >   Cancel   Finish

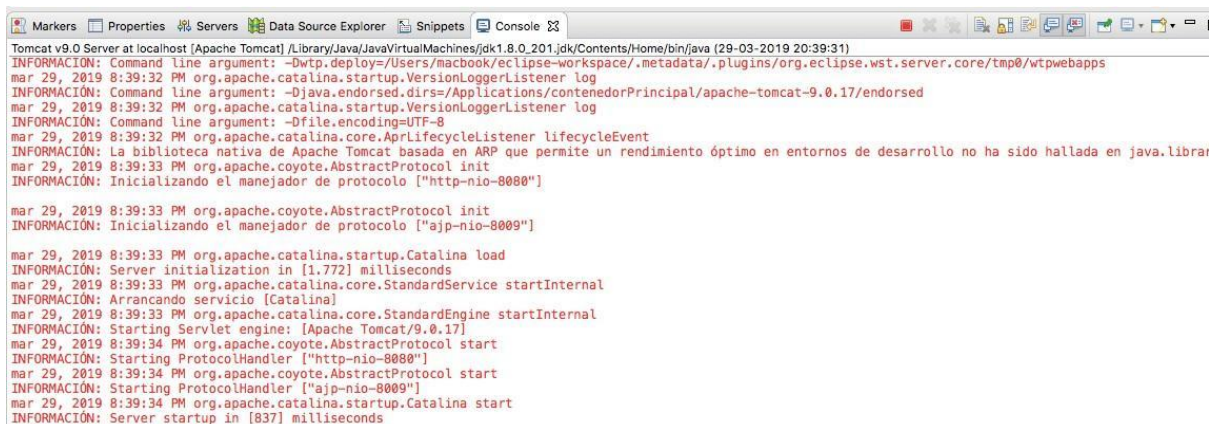
Selección de  
ubicación de  
servidor

# Comprobación de instalación de Tomcat

- Vamos a pinchar sobre el link de Tomcat en la pestaña server, y lo encendemos.



Conectado al servidor.

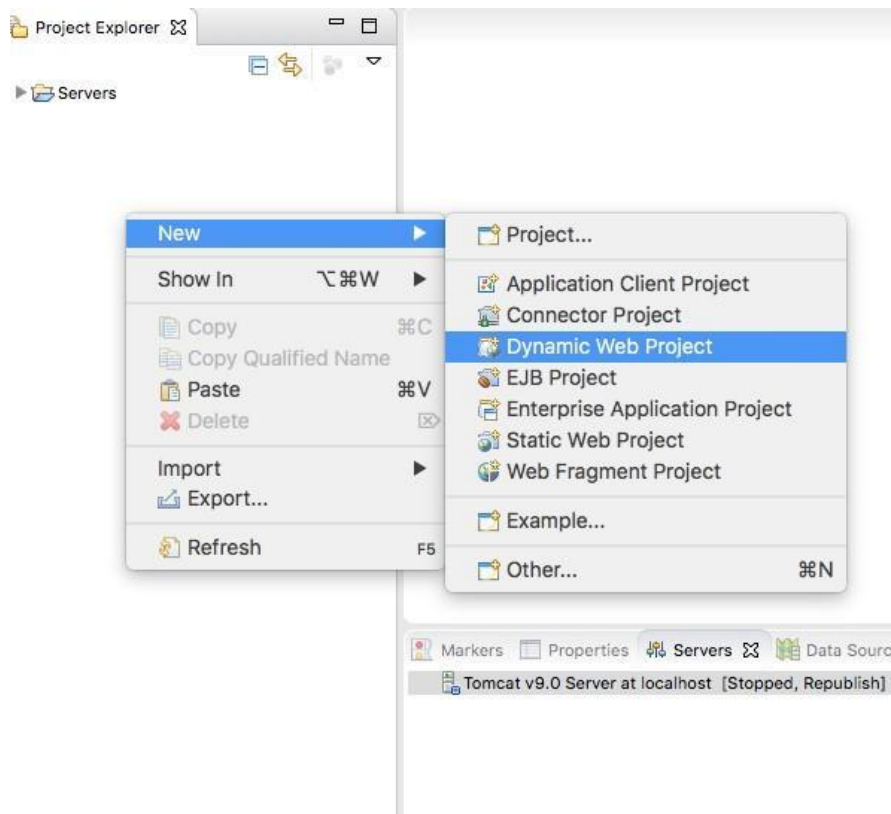


```
Tomcat v9.0 Server at localhost [Apache Tomcat] /Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java (29-03-2019 20:39:31)
INFORMACIÓN: Command line argument: -Dwtp.deploy=/Users/macbook/eclipse-workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/wtpwebapps
mar 29, 2019 8:39:32 PM org.apache.catalina.startup.VersionLoggerListener log
INFORMACIÓN: Command line argument: -Djava.endorsed.dirs=/Applications/contenedorPrincipal/apache-tomcat-9.0.17/endorsed
mar 29, 2019 8:39:32 PM org.apache.catalina.startup.VersionLoggerListener log
INFORMACIÓN: Command line argument: -Dfile.encoding=UTF-8
mar 29, 2019 8:39:32 PM org.apache.catalina.core.AprLifecycleListener lifecycleEvent
INFORMACIÓN: La biblioteca nativa de Apache Tomcat basada en APR que permite un rendimiento óptimo en entornos de desarrollo no ha sido hallada en java.library
mar 29, 2019 8:39:33 PM org.apache.coyote.AbstractProtocol init
INFORMACIÓN: Inicializando el manejador de protocolo ["http-nio-8080"]

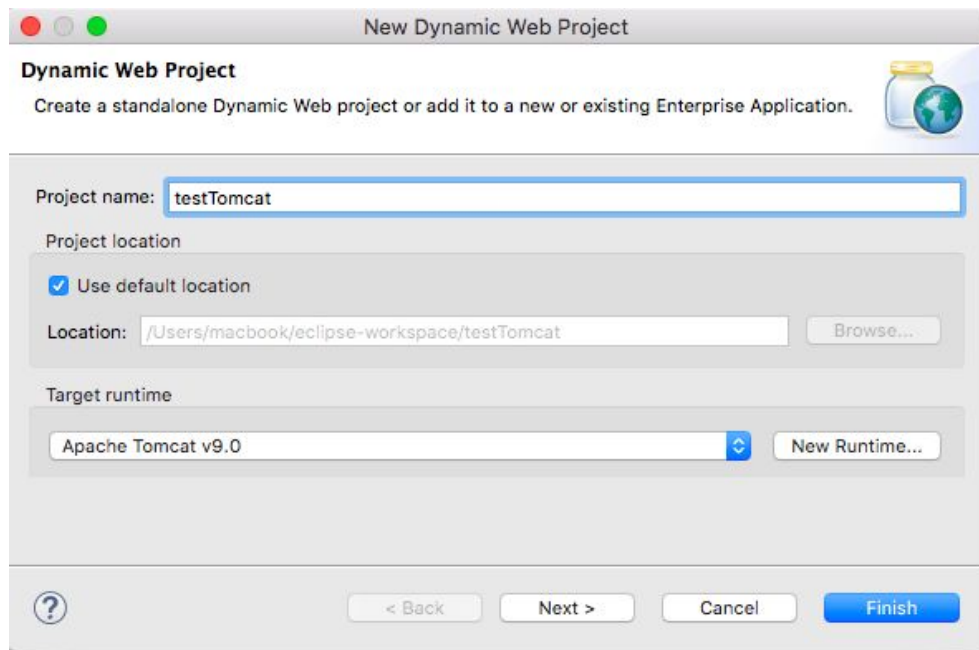
mar 29, 2019 8:39:33 PM org.apache.coyote.AbstractProtocol init
INFORMACIÓN: Inicializando el manejador de protocolo ["ajp-nio-8009"]

mar 29, 2019 8:39:33 PM org.apache.catalina.startup.Catalina load
INFORMACIÓN: Server initialization in [1.772] milliseconds
mar 29, 2019 8:39:33 PM org.apache.catalina.core.StandardService startInternal
INFORMACIÓN: Arrancando servicio [Catalina]
mar 29, 2019 8:39:33 PM org.apache.catalina.core.StandardEngine startInternal
INFORMACIÓN: Starting Servlet engine: [Apache Tomcat/9.0.17]
mar 29, 2019 8:39:34 PM org.apache.coyote.AbstractProtocol start
INFORMACIÓN: Starting ProtocolHandler ["http-nio-8080"]
mar 29, 2019 8:39:34 PM org.apache.coyote.AbstractProtocol start
INFORMACIÓN: Starting ProtocolHandler ["ajp-nio-8009"]
mar 29, 2019 8:39:34 PM org.apache.catalina.startup.Catalina start
INFORMACIÓN: Server startup in [837] milliseconds
```

Para comprobar el correcto funcionamiento del contenedor Tomcat, vamos a probarlo con una pequeña aplicación web.

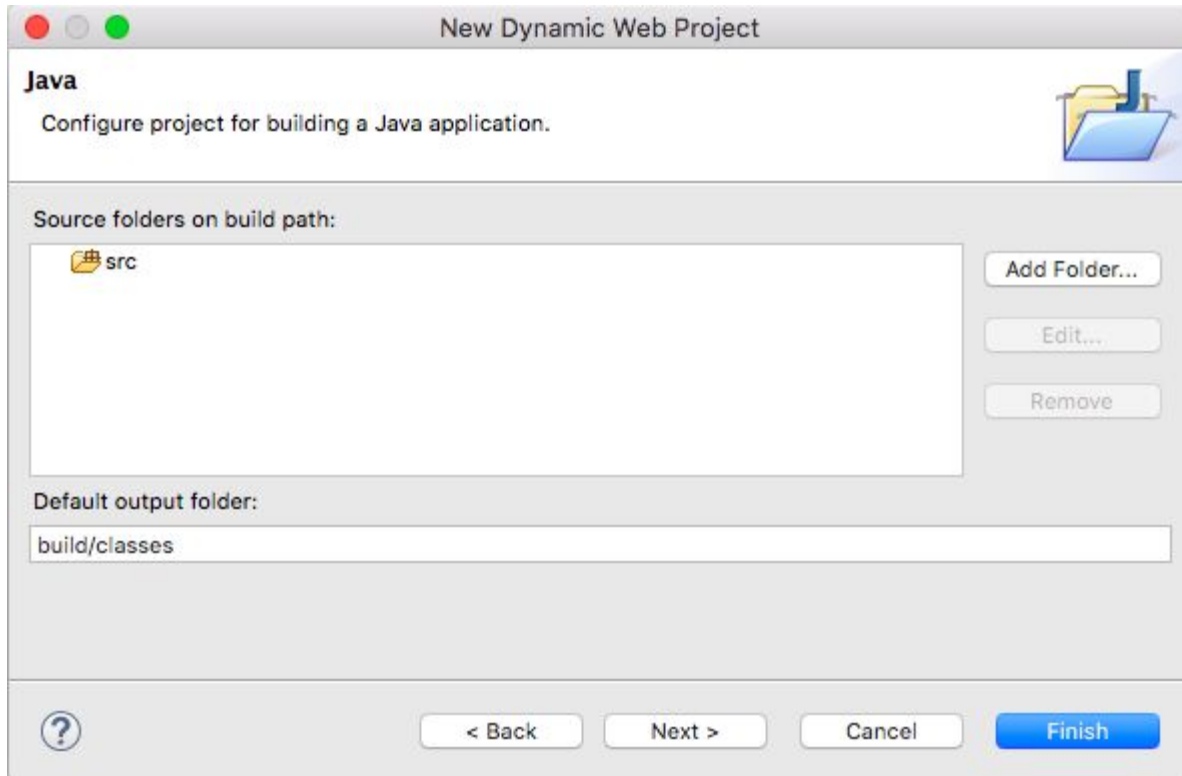


Creando un  
proyecto Web  
Dynamic Project



Creando el  
nombre del  
proyecto





Carpeta raíz  
del proyecto

New Dynamic Web Project

**Web Module**  
Configure web module settings.

Context root:

Content directory:

☐ Generate web.xml deployment descriptor

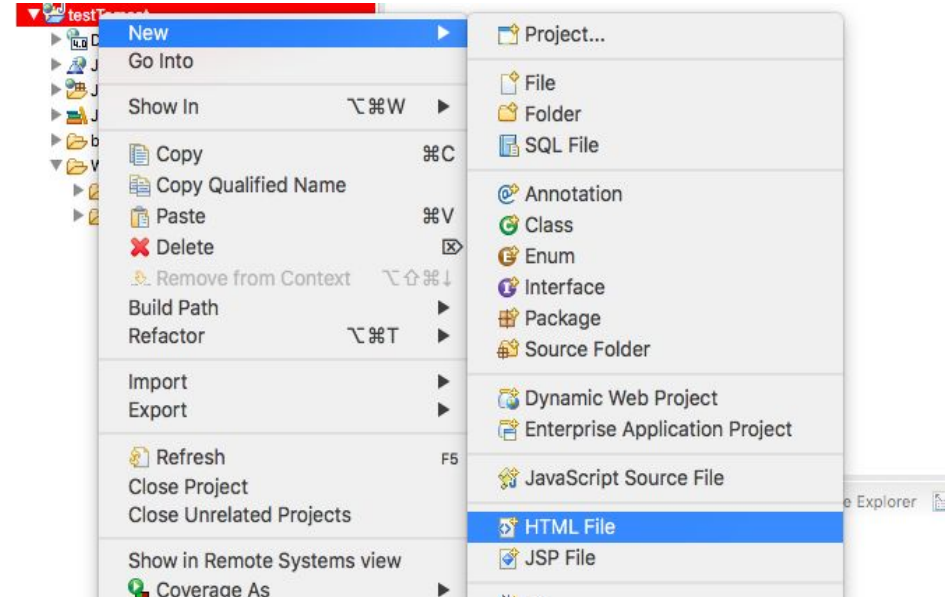
? < Back Next > Cancel Finish

@ Context root y  
content directory  
dejar por defecto

## Estructura de carpetas en el explorador de proyectos

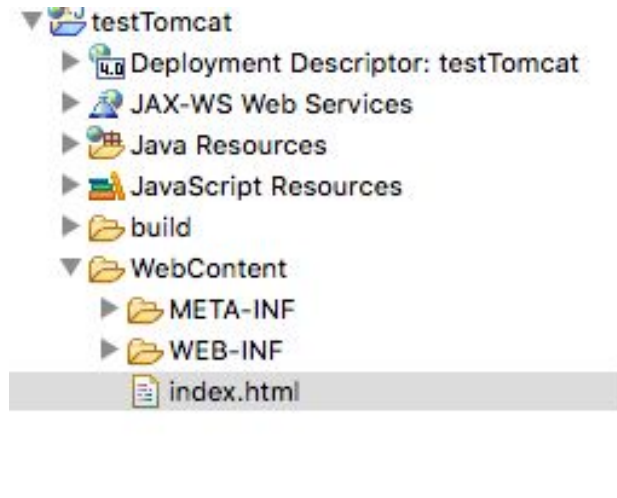
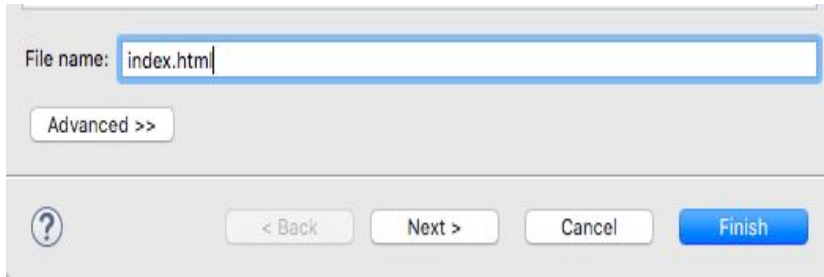


## Crear archivo HTML



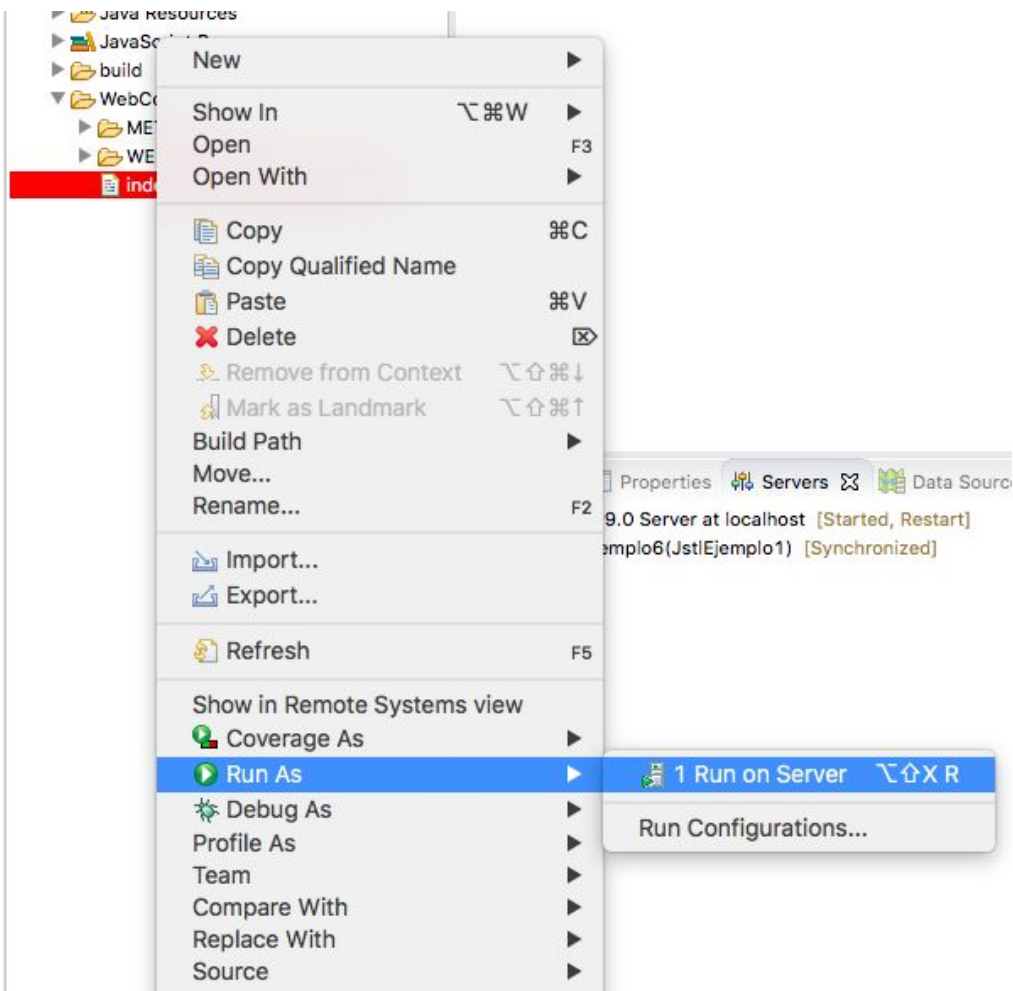
## Asignarle el nombre index.html

## Estructura del proyecto posterior a la creación del archivo index.html

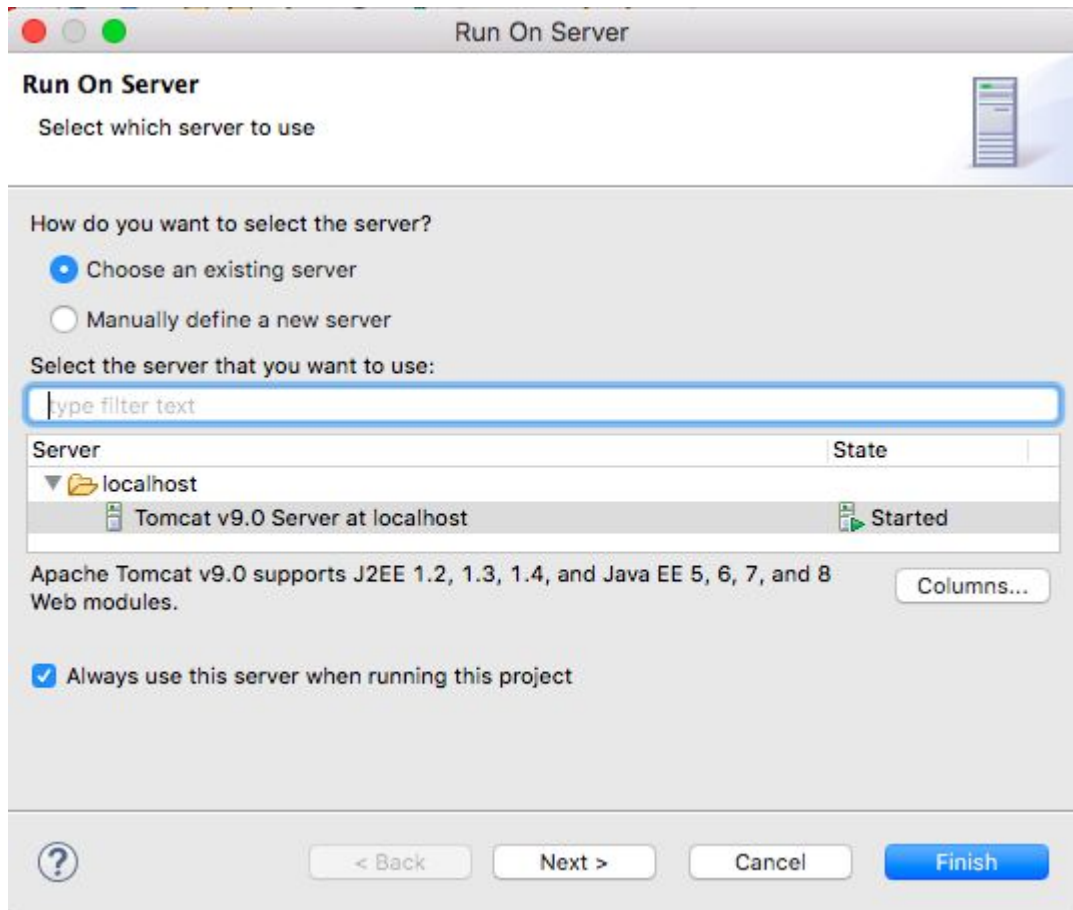


Generamos un título en el HTML.

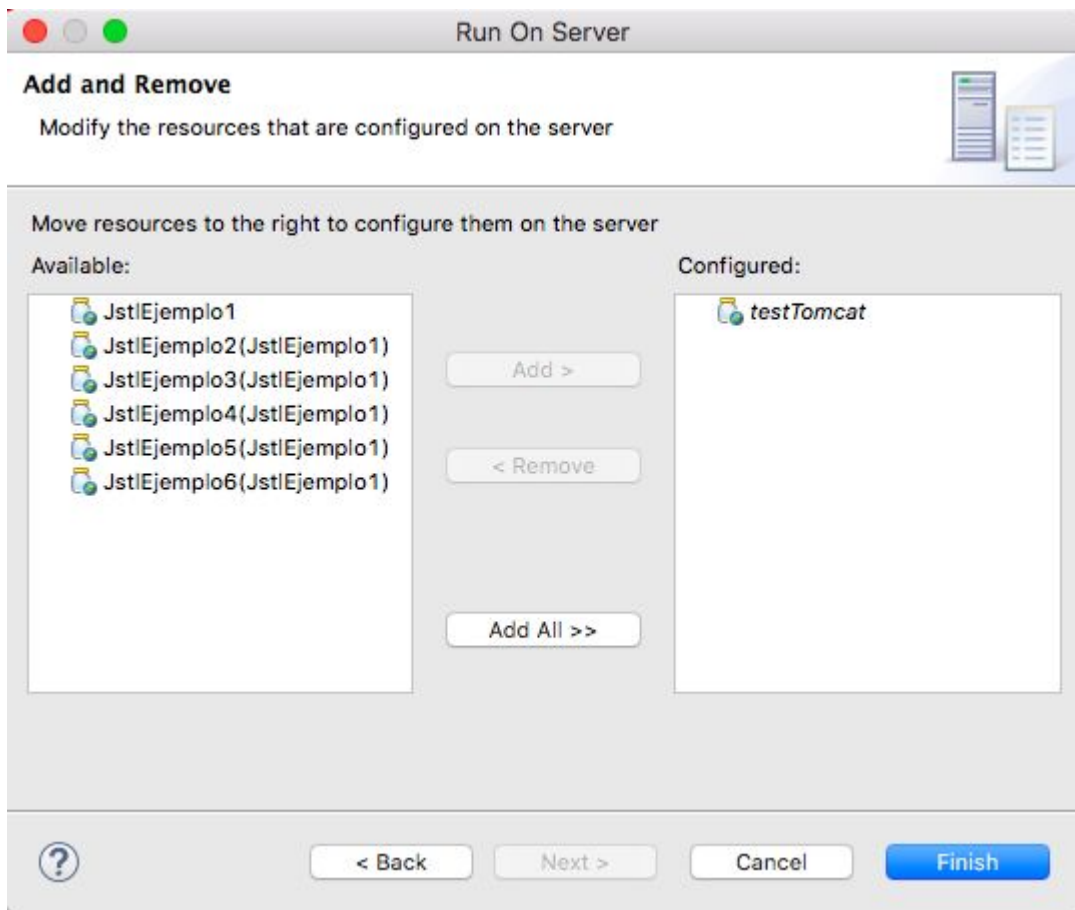
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Prueba Tomcat</title>
</head>
<body>
  <h1>Tomcat funcionando perfectamente</h1>
</body>
</html>
```



Ejecutando la  
aplicación



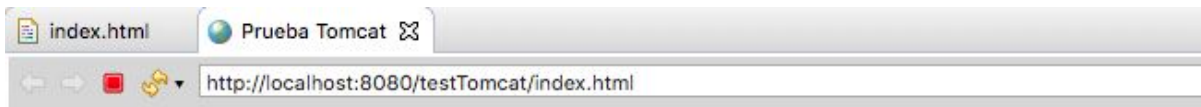
Elegir servidor  
y seleccionar  
siguiente



**Artefactos listos  
para ejecutar**



Ejecutando el primer ejemplo y validando su funcionamiento.



## Tomcat funcionando perfectamente



Si el contenedor inició bien los servicios, abre un navegador chrome, y dirígete a la siguiente dirección: <http://localhost:8080/testTomcat/index.html>.

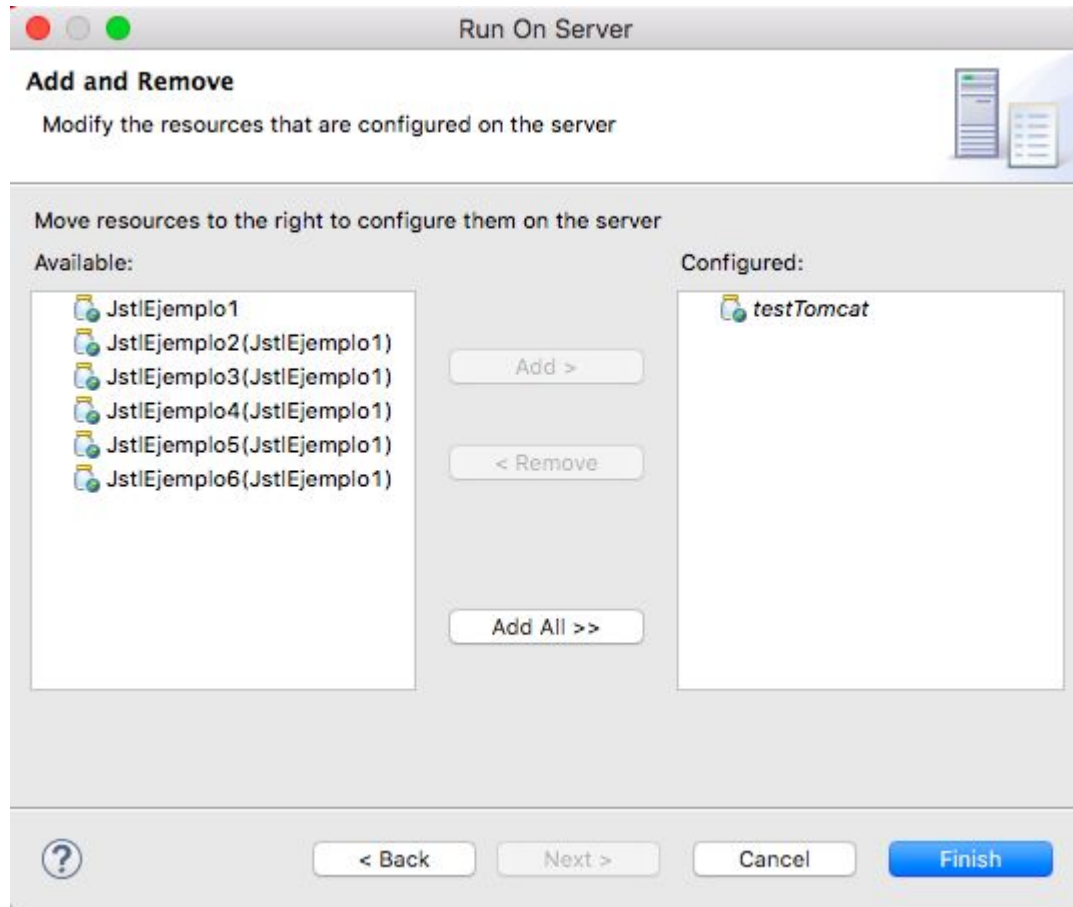
**/\* Contenedor Tomcat \*/**

# Estructura

## Carpetas más importantes de Tomcat

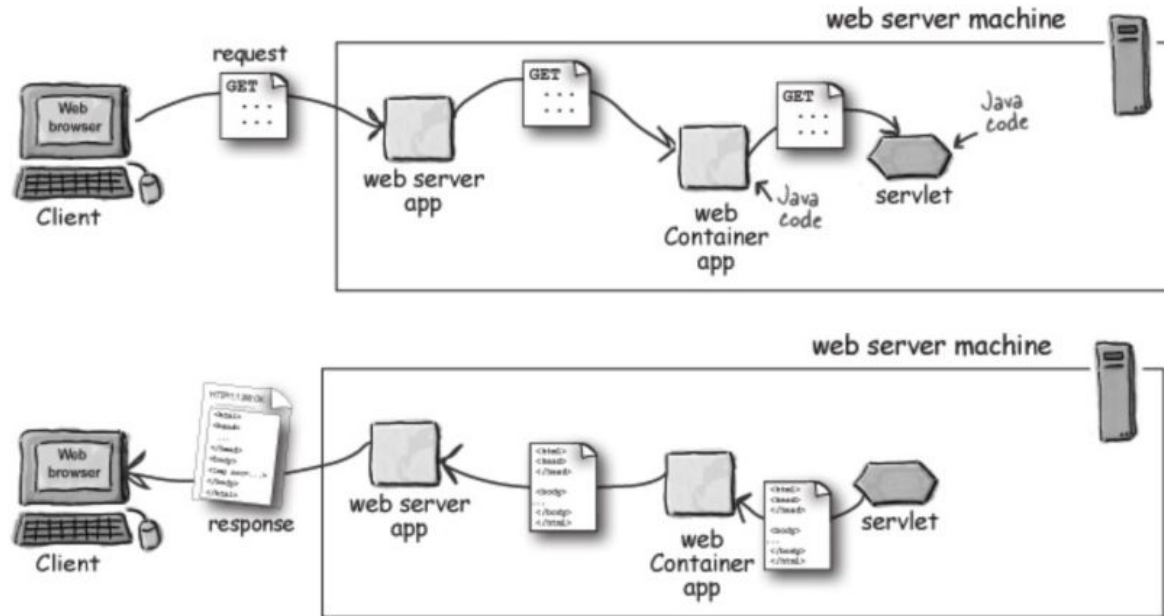
Nombre	Fecha de modificación	Tamaño	Clase
bin	hoy 20:12	--	Carpeta
BUILDING.txt	13-03-2019 15:56	20 KB	Documento de texto sin formato
conf	hoy 20:12	--	Carpeta
CONTRIBUTING.md	13-03-2019 15:56	6 KB	Documento Visual Studio Code
lib	hoy 20:12	--	Carpeta
LICENSE	13-03-2019 15:56	58 KB	Documento TextEdit
logs	13-03-2019 15:55	--	Carpeta
NOTICE	13-03-2019 15:56	2 KB	Documento TextEdit
README.md	13-03-2019 15:56	3 KB	Documento Visual Studio Code
RELEASE-NOTES	13-03-2019 15:56	7 KB	Documento TextEdit
RUNNING.txt	13-03-2019 15:56	17 KB	Documento de texto sin formato
temp	hoy 20:12	--	Carpeta
webapps	13-03-2019 15:56	--	Carpeta
work	13-03-2019 15:55	--	Carpeta

<b>bin</b>	Archivos ejecutables.
<b>common</b>	Clases comunes.
<b>logs</b>	Muestra los logs del servidor.
<b>webapps</b>	Archivo que contiene las aplicaciones web.



**Aplicaciones  
disponibles para  
levantar**

Tomcat nos permite delegar parte de las tareas al recibir peticiones, redireccionando los requisitos a un objeto servlet en específico y ampliando su capacidad de respuesta.



# Cambiar puerto de Tomcat

Hay ocasiones en que el puerto por defecto de Tomcat no es posible utilizarlo, pero existe una alternativa que consta de cambiar el puerto por defecto del contenedor.

## Ejemplo:

Cambiar el puerto por defecto a algún otro.

```
<Connector port="8080" protocol="HTTP/1.1"  
connectionTimeout="20000" redirectPort="8443">
```

**/\* Protocolo HTTP \*/**

# El protocolo HTTP

Protocolo de  
transferencia  
de hipertexto

Interacción  
entre cliente y  
servidor

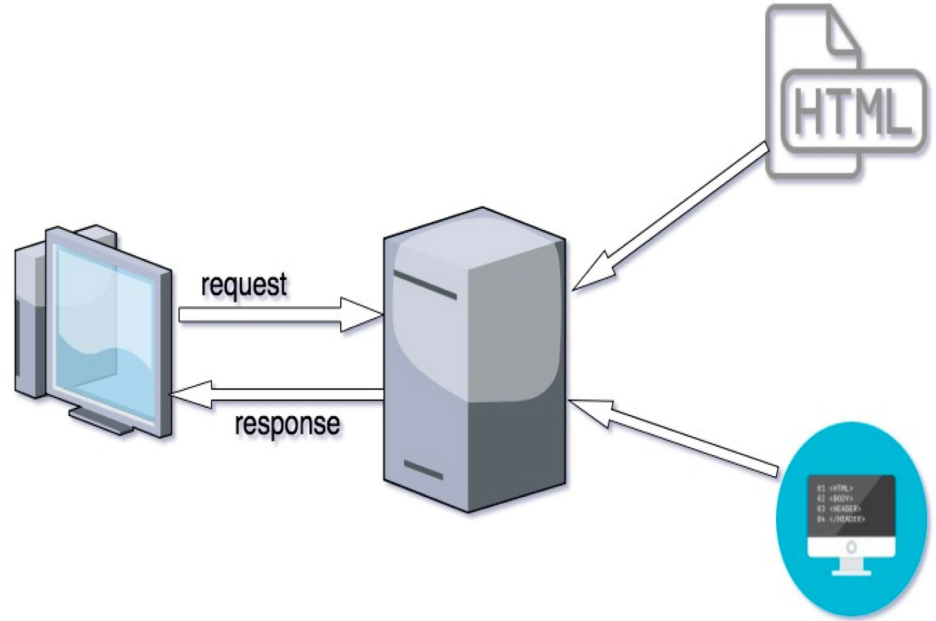
El cliente envía un  
http-request y el  
servidor un  
http-response

Es un  
protocolo  
ampliable



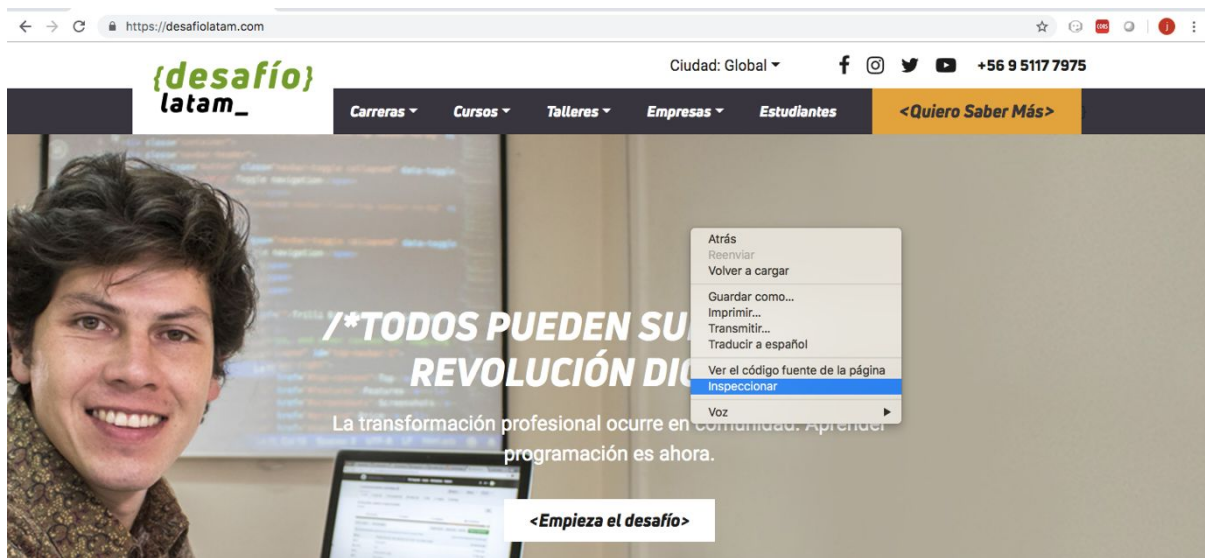
# El modelo request/response en http

- **Cliente:**
  - Un navegador web envía un request.
- **El server recibe el request:**
  - Lo analiza, lo gestiona para luego enviar de vuelta un objeto response.



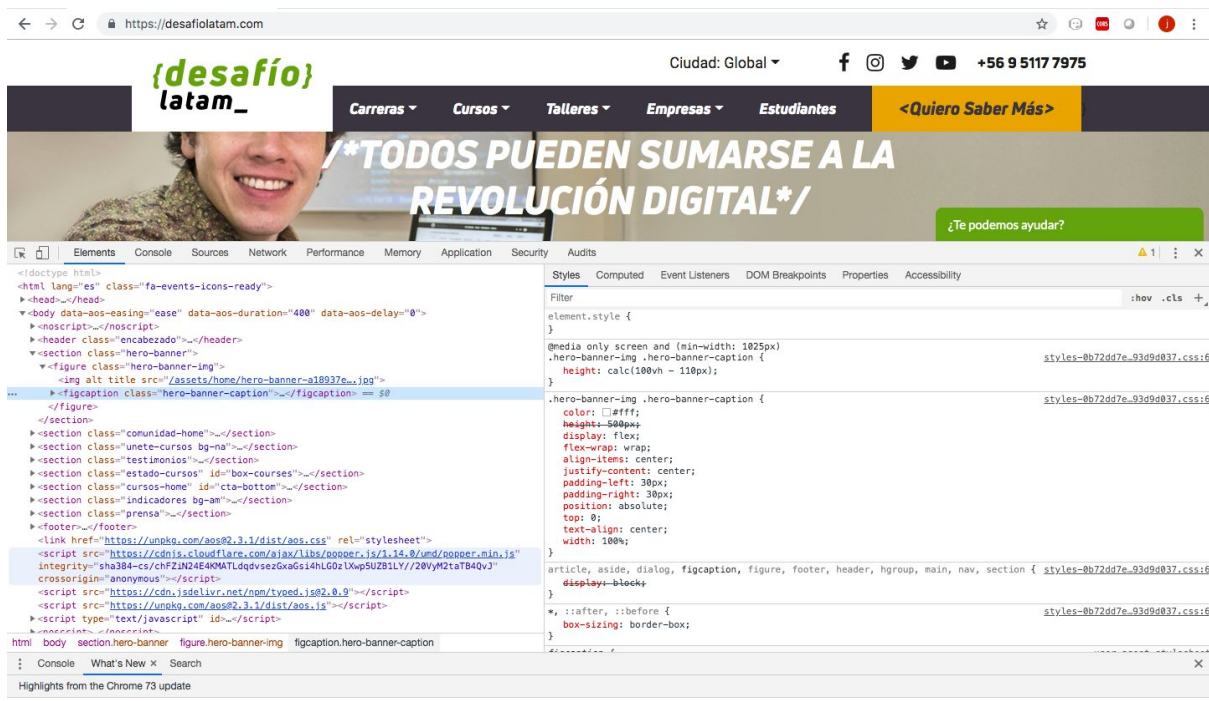
# Mensajes http-Request

Inspeccionando página de Desafío Latam.  
Para verificar cómo se envía la petición mediante http.



# Identificando la consola de desarrollador.

Para poder ver los contenidos de los request y response de la página web, debes pinchar la pestaña Network y en su pantalla principal.



# Revisando la información request.

La sección general, contiene una dirección URL, la cual está formada por:

- El protocolo
- La dirección del servidor
- La ubicación del archivo css solicitado

Ciudad: C

{desafío}  
latam\_

Carreras ▾ Cursos ▾ Talleres ▾ Empresas ▾

¡TODOS PUEDEN SUMAR!

Estableciendo una conexión segura...

Elements Console Sources Network Performance Memory Application Security Audits

View: [Icons] [Icons] Group by frame [ ] Preserve log [ ] Disable cache [ ] Offline Online ▾

Filter [ ] Hide data URLs [x] All XHR JS CSS Img Media Font Doc WS Manifest Other

5000 ms 10000 ms 15000 ms 20000 ms 25000 ms 30000 ms 35000 ms

Name x Headers Preview Response Timing

font-awesome-css.min.css ▾ General

fontawesome-webfont.woff2 [ ]

fontawesome-webfont.woff [ ]

Request URL: https://use.fontawesome.com/releases/v4.7.0/css/font-awesome-css.min.css

Request Method: GET

Status Code: 200

Remote Address: 23.111.9.35:443

Referrer Policy: no-referrer-when-downgrade

# Mensaje http-response

Luego de generar un request a un servidor remoto, este nos devuelve nuestro recurso.

- El request internamente nos entrega bastante información sobre el recurso.
  - Método de respuesta, en este caso GET
  - Content Type, en este caso un archivo .css
  - La fecha.

The screenshot shows a web browser interface with the Network tab selected. The page title is "{desafío} latam\_". The top navigation bar includes links for Carreras, Cursos, Talleres, and Empresas. The main content area displays a banner with the text "/\* TODOS PUEDEN SUMAR". The Network tab shows a list of resources, with the selected resource being "font-awesome-css.min.css". The Response Headers for this resource are displayed, including:

- access-control-allow-methods: GET
- access-control-allow-origin: \*
- access-control-max-age: 3000
- cache-control: max-age=31556926
- content-encoding: gzip
- content-type: text/css
- date: Tue, 26 Mar 2019 23:58:05 GMT
- etag: W/"36082410df2ef7f83932219089dc1443"
- last-modified: Tue, 25 Oct 2016 17:21:58 GMT
- server: NetDNA-cache/2.2
- status: 200

The status bar at the bottom indicates "5 requests | 107 KB transferred | ...".

# Códigos de error http

Los request y los response vienen con un código de respuesta.

Código	Descripción	Detalle
1XX	Respuestas informáticas	Indica una respuesta provisional.
2XX	Peticiones correctas	La acción solicitada por el cliente ha sido recibida, entendida, aceptada y procesada correctamente.
3XX	Redirecciones	El cliente debe tomar una acción adicional para completar el request.
4XX	Errores del cliente	Excepto cuando se responde a un EAD request, el servidor debe incluir una entidad que contiene una explicación del error, y su es temporal o permanente.
5XX	Errores del servidor	El servidor ha fallado al completar una solicitud aparentemente válida.

- Uniform Resource Locators.
  - Identificar las direcciones en donde se alojan los recursos web.

Estructura: <code>https://www.desafiolatam.com:8927/java/inicio</code>	
<code>https</code>	Indica al navegador el protocolo que utiliza en el requerimiento.
<code>www.desafiolatam.com</code>	Ubicación física de la máquina en la cual está alojado el sitio.
<code>8927</code>	Puerto de escucha del servidor.
<code>/java/inicio</code>	ubicación física de los documentos que estamos solicitando.

**`/* Introducción a los servlets */`**



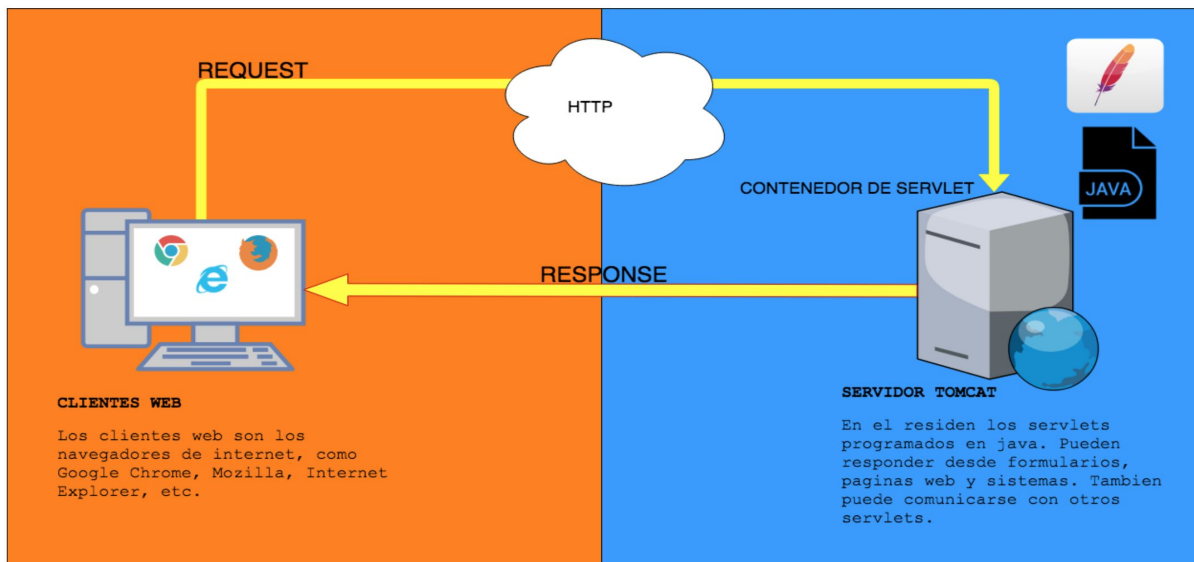
# Los servlets

- Encargado de extender la funcionalidad de la clase y darle más "poder".
- El servlet fuera del contenedor Tomcat pierde todo su poder.

## Ejemplo:

Flujo normal de petición entre clientes y servidor .

Esta es la arquitectura típica de un sistema web, la cual es conocida como cliente-servidor.

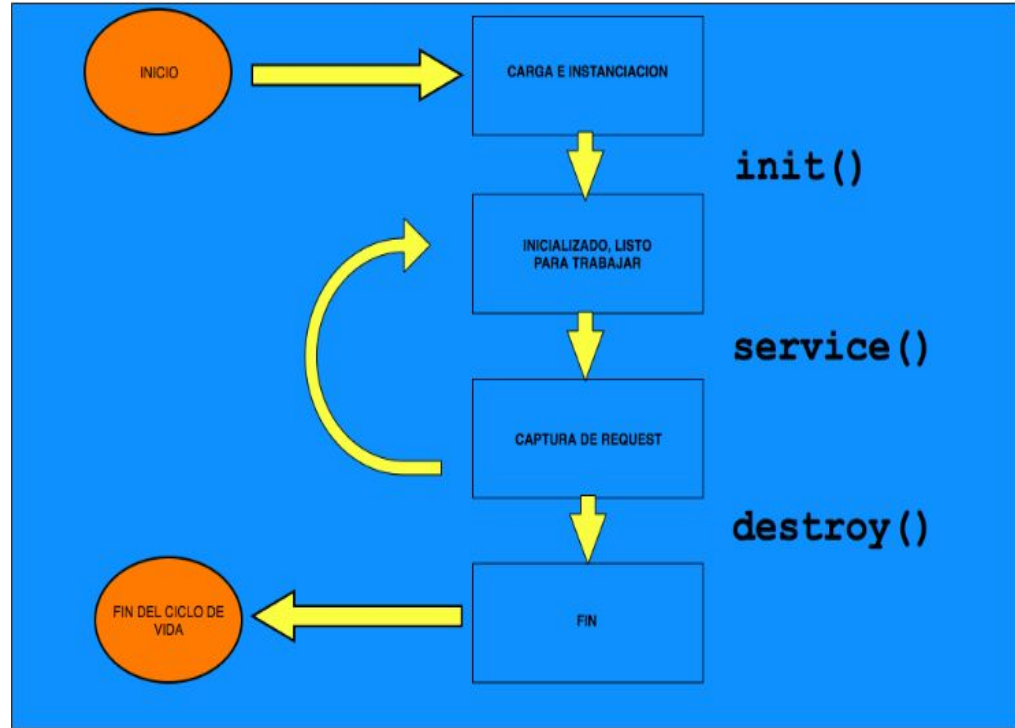


# Ciclo de vida de un Servlet

Se divide en 4 pasos
Carga del servlet
Inicialización del servlet
Captura del request
Destrucción del request

# Carga del servlet

- La primera etapa del ciclo de vida de un servlet parte con su inicialización gracias al servidor Tomcat. El servidor ejecuta dos pasos en esta etapa:
  - **Loading:** Carga de la clase servlet
  - **Instantiation:** Creación de una instancia del servlet.



## Inicialización de Servlets

- Después de que el servlet fue instanciado correctamente, el contenedor se encarga de inicializarlo.

## Captura del Request

- El contenedor de servlet realiza las siguientes operaciones cuando la instancia del servlet está lista:
  - Crea los objetos `ServletRequest` y `ServletResponse`.
  - Se invoca al método `Servlet.service`.

# Destruir el Servlet

Ejecuta las siguientes acciones:

- Permite que todos los subprocesos que se ejecutan actualmente en el método de servicio de la instancia de Servlet completen sus trabajos y sean liberados.
- El contenedor Servlet llama al método `destroy()` en la instancia de Servlet.
- El Servlet container libera todas las referencias de esta instancia de Servlet.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/cicloVida")
public class CicloDeVida extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private String saludo;
    public void init() throws ServletException {
        saludo = ("Hola servlet inicializando...");
    }
    public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        // Seteamos el tipo de response, un html
        response.setContentType("text/html");
        // Declaramos la logica
        PrintWriter out = response.getWriter();
        out.println("<h1>" + saludo + "</h1>");
    }
    public void destroy() {
        // Se destruye el servlet, nada que hacer
    }
}
```

Para graficar  
el ciclo de vida

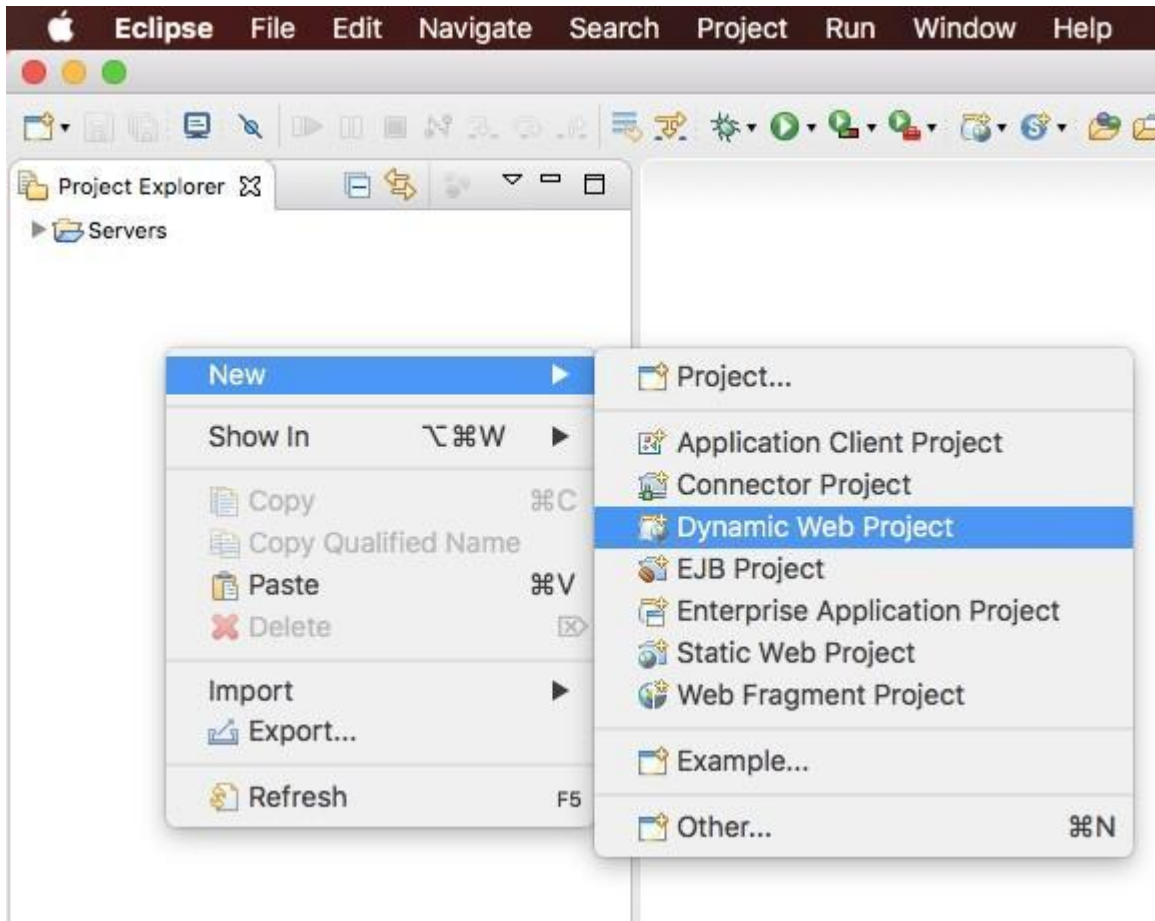
**/\* Creación del primer servlet \*/**

# Composición de un servlet

Se construyen basados en la herencia de su clase padre Http Servlets, clase pública y abstracta.

- La clase que herede de esta clase abstracta, debe sobrescribir al menos:
  - Método doGet
  - Método doPost
  - Método doDelete





Creando un  
nuevo proyecto.

New Dynamic Web Project

**Dynamic Web Project**  
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

☒ Use default location

Location:

Target runtime

Dynamic web module version

Configuration

A good starting point for working with Apache Tomcat v9.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name:

Working sets

☐ Add project to working sets

Working sets:

Lo llamamos  
"PrimerServlet".

# Creación de un servlet

1. Creamos un nuevo proyecto de tipo Dinamic Web Project y lo nombramos.
2. Avanzamos en el wizard con next y finish.
3. Al final Eclipse nos genera una estructura de Proyecto, la cual usaremos como base para nuestro trabajo.

# Java Resources

Se compone de:

<b>Carpeta src</b>	Aloja las clases java y la estructura de paquetes de la aplicación.
<b>Carpeta libraries</b>	Mantiene las librerías utilitarias.
<b>Carpeta build</b>	Mantiene las clases autogeneradas.

New Java Class

**Java Class**  
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

---

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Creamos una  
nueva clase

5. Eclipse nos genera la clase y abre el archivo en el área de código.

```
package com.desafiolatam.web.servlet;  
  
public class GeneradorIndex{  
  
}
```

Vamos a analizar un servlet sencillo que cuenta con todo lo necesario para proveer servicios y generar una página web dinámica.

```
package com.desafiolatam.web.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.logging.Logger;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/inicio")
public class GeneradorIndex extends HttpServlet {

    private static final long serialVersionUID = 1L;
    Logger milog = Logger.getLogger(Saludo.class.getName());
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) {
        try {
            PrintWriter pw = resp.getWriter();
            pw.println("<h1>Hola mundo</h1>");
        } catch (IOException e) {
            milog.severe(e.getMessage());
        }
    }
}
```

Aparece una sentencia en la declaración.

```
@WebServlet("/inicio")
```

Recordemos que un servlet es un componente web que será accedido por internet.

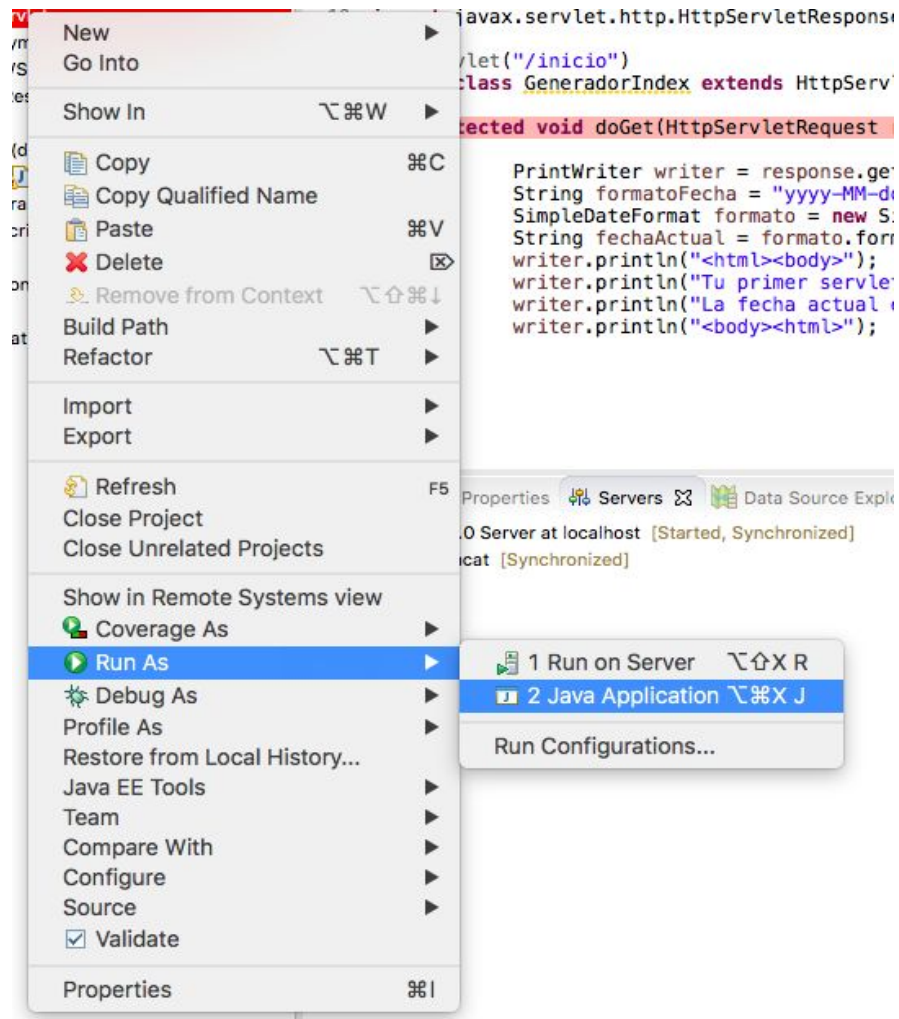
El primer método protected es heredado desde la clase HttpServlet y su nombre es doGet().

```
protected void doGet(HttpServletRequest request,  
HttpServletResponse response)
```

Al imprimir código estamos generando etiquetas html, el navegador puede interpretar directamente.

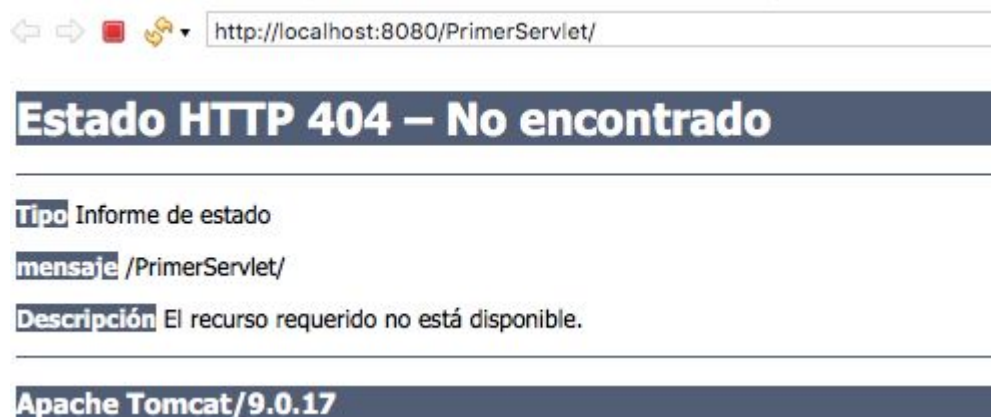
```
PrintWriter pw = resp.getWriter();  
pw.println("<h1>Hola mundo</h1>");
```





Ejecutar  
código.

- Se empezará a levantar el server y, según la configuración de Eclipse, pueden pasar dos cosas:
  - Se abrirá un navegador en el mismo IDE
  - La consola dirá solo que comenzó a trabajar
- Si ocurre lo primero, mostrará una pantalla de error:



¿Por qué pasa esto?

```
@WebServlet("/inicio")
```

Aquí se ve la utilidad de tal sentencia.

```
http://localhost:8080/PrimerServlet
```

Solo estamos llegando hasta la puerta de entrada del servlet.

```
http://localhost:8080/PrimerServlet/inicio
```

**/\* Manejo de información entre servlets \*/**

# Creación de servlet de generación respuesta

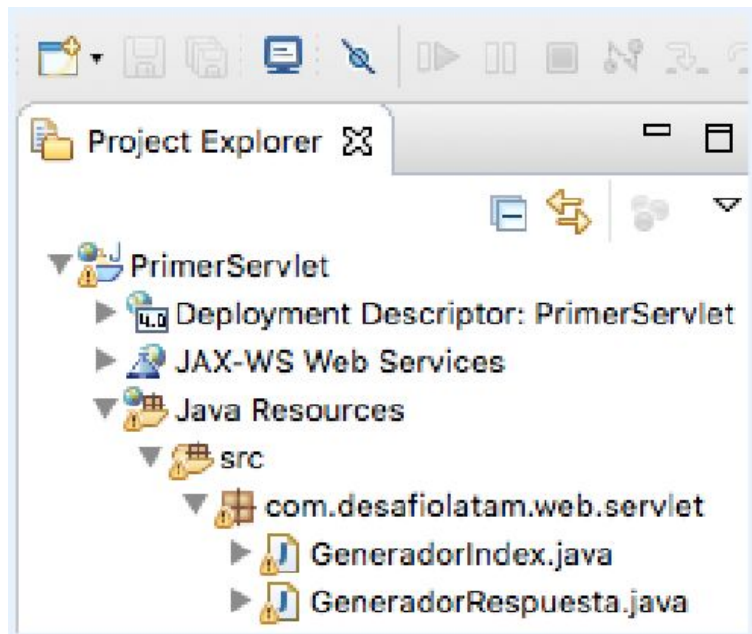
Los servlets tienen la capacidad de recibir parámetros desde los clientes, para luego procesarlos y generar salidas.

Vamos a implementar una nueva funcionalidad en nuestro primer ejemplo.

```
@WebServlet("/generadorRespuesta")
public class GeneradorRespuesta extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {
        String nombre;
        String apellido;
        String fechaActual;
        PrintWriter printWriter = response.getWriter();
        SimpleDateFormat formato = new SimpleDateFormat("dd-MM-yyyy");
        fechaActual = formato.format(new Date());
        nombre = request.getParameter("nombre");
        apellido = request.getParameter("apellido");

        printWriter.println("<html>");
        printWriter.println("<body>");
        printWriter.println("Bienvenido/a " + nombre + " " + apellido);
        printWriter.println("La fecha es: " + fechaActual);
        printWriter.println("</body>");
        printWriter.println("</html>");
    }
}
```



## Estructura del proyecto

En cada variable String guardando el valor del request mediante:

```
nombre = request.getParameter("nombre");  
apellido = request.getParameter("apellido");
```

La variable request, es el parámetro de entrada de nuestro método.

```
protected void doGet(HttpServletRequest request,  
    HttpServletResponse response)
```

Con esas sentencias ya tenemos en nuestro poder el nombre y el apellido que vienen desde el cliente.





# Envío de parámetros desde el cliente

<http://localhost:8080/PrimerServlet/generadorRespuesta?nombre=JUANITO&apellido=PEREZ>

Tenemos la url por la cual estamos probando nuestro servlet.

La url se compone de:

- <http://localhost:8080/>
  - Corresponde a la dirección del servidor que aloja al servlet.
  - La dirección ip va acompañado del puerto de conexión 8080.

- Dirección del servidor remoto que queremos consultar.
  - **/PrimerServlet**: Es el nombre de nuestro proyecto.
  - **/generadorRespuesta**: Es el nombre de nuestro servicio.
- **nombre=JUANITO&apellido=PEREZ**
  - Siempre que se envían parámetros por url tenemos que indicar el nombre de la variable.

```
request.getParameter("nombre");
```

# Aplicar el envío de información entre Servlets

- Hay ocasiones en que se requiere que en la capa de negocio un servlet pueda comunicarse con otro servlet.
  - java nos provee de los métodos:

```
getRequestDispatcher();  
getServletContext();
```

- **getServletContext:** obtiene el contexto del servlet que queremos utilizar.
- **getRequestDispatcher:** le enviamos la dirección del servlet.

Vamos a modificar nuestro recién creado método Saludo.java.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
    PrintWriter printWriter = response.getWriter();
    SimpleDateFormat formato = new SimpleDateFormat("dd-MM-yyyy");

    printWriter.println("<html><body>");
    printWriter.println("Tu primer servlet");
    printWriter.println("La fecha actual es: " + fechaActual);
    printWriter.println("</body></html>");
    request.getRequestDispatcher("/inicio").forward(request,
response);
}
}
```

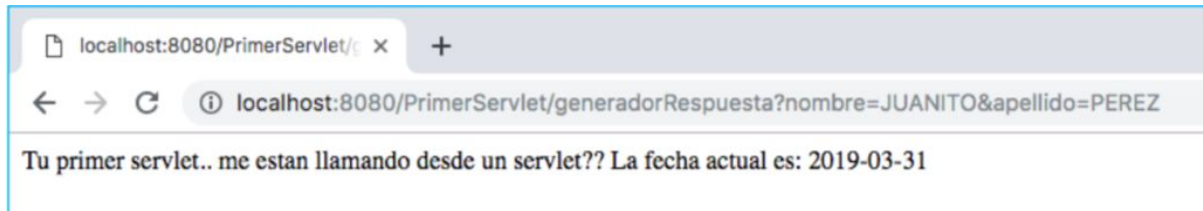
Si ejecutas la llamada:

```
http://localhost:8080/PrimerServlet/generadorRespuesta?nombre=JUANITO&apellido=PEREZ
```

En la clase `GeneradorIndex.java`, agrega un mensaje representativo para que sepamos que estamos llamando al servlet.

```
writer.println("Tu primer servlet.. me estan llamando desde un servlet?? ");
```

Probando el Servlet.



# **`/* Introducción a los Formularios y JTSL con JSP */`**

# Introducción a JSP

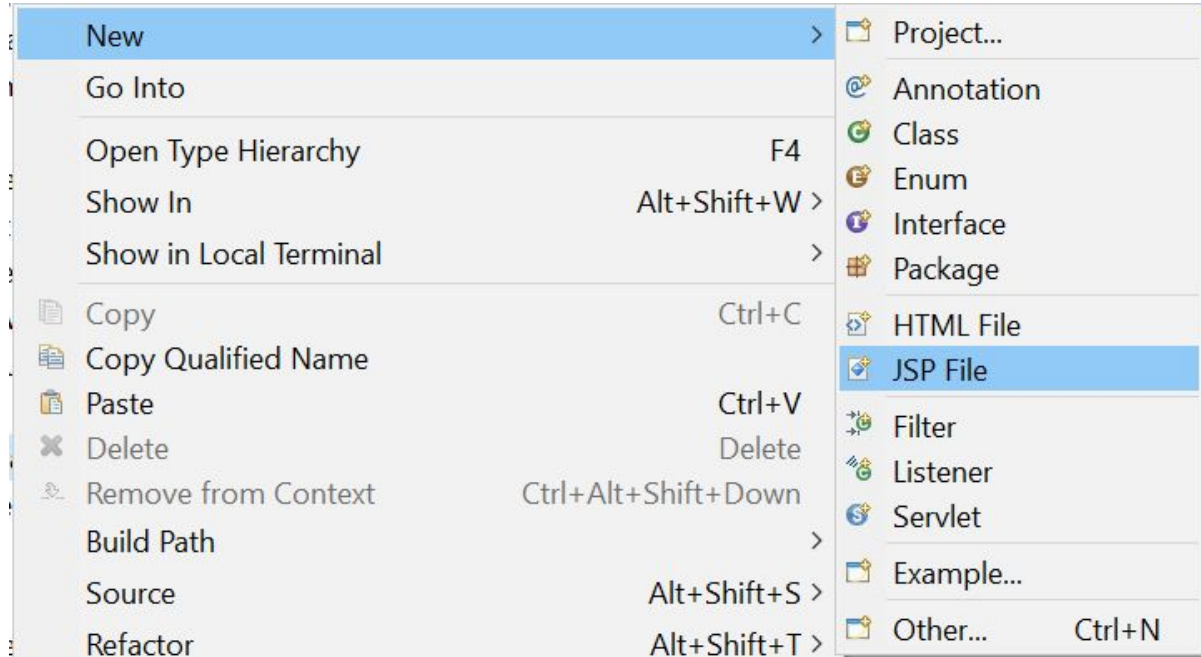
Tecnología de  
Java JEE

Provee  
herramientas  
para construir  
páginas web  
dinámicas

basadas en el  
lenguaje de  
marcado HTML,  
XML u otro

# Primer acercamiento a JSP

Ejemplo JSP:





## Caracteres <%@ %>:

```
<%@ page language="java" contentType="text/html;  
charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>  
<!DOCTYPE html>  
<html>  
<head>  
  <meta charset="ISO-8859-1">  
  <title>Insert title here</title>  
</head>  
<body>  
</body>  
</html>
```

Esas son las llamadas directivas y le indican al contenedor que el lenguaje que estamos utilizando es Java.

Atributos de uso habitual en las directivas.	
language	Indica qué lenguaje de programación estamos utilizando.
contentType	Indica qué tipo de contenido posee la página.
isErrorPage	Indica si la página que estamos cargando es una página de error.
errorPage	Define la página a la que debe dirigirse si ocurre una excepción.


```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Mi primer JSP</title>
</head>
<body>
  <H1>Algunas expresiones en Java</H1>
  <!-- Mostrar texto en mayúscula -->
  <%= "Hola mundo".toUpperCase() %><br>

  <!-- Imprimir una variable -->
  <% String variable = "La hora actual:"; %>
  <%= variable %><br>

  <!-- Mostrar la fecha y hora -->
  <%= new java.util.Date().toString() %><br>

  <!-- Suma de dos enteros -->
  <% int suma = 1+1; %>
  <%= "1+1 = "+suma %><br>

</body>
</html>
```



**Volvamos a  
nuestro archivo y  
ejecutamos lo  
siguiente:**

Resultado de este código Java incrustado.

← → ↻ 🌐 localhost:8080/PrimerServlet/Expresiones.jsp

## Algunas expresiones en Java

HOLA MUNDO

La hora actual:

Mon Apr 13 23:39:12 CLT 2020

$1+1 = 2$

```
<html>
```

```
  <head>
```

```
    <meta charset="UTF-8">
```

```
    <title>Mi primer JSP</title>
```

```
  </head>
```

```
... <body> == $0
```

```
  <h1>Algunas expresiones en Java</h1>
```

```
  "HOLA MUNDO"
```

```
  <br>
```

```
  "La hora actual:"
```

```
  <br>
```

```
  " Mon Apr 13 23:39:12 CLT 2020"
```

```
  <br>
```

```
  " 1+1 = 2"
```

```
  <br>
```

```
</body>
```

```
</html>
```

html

body

## Inspector de elementos

# Comentarios

La forma de comentar es un tanto distinta a los que hemos visto en HTML o incluso en Java.

```
<!-- comentario HTML -->  
<%-- comentario JSP --%>
```

Los comentarios que realicemos en las páginas JSP deben indicarse entre las etiquetas `<%-- --%>`.

# Expresiones

Debemos utilizar las etiquetas <%= %>.

```
<%= "Hola mundo".toUpperCase() %><br>
```

- Tener en consideración tres aspectos relevantes:
  - El resultado de la expresión debe ser válida para el lenguaje de la página.
  - El resultado de la expresión será convertido a String una vez resuelto.
  - No se debe terminar la sentencia en punto y coma.

# Scriptlets

- Puede manejar declaraciones, expresiones o cualquier otro tipo de fragmento de código válido en el lenguaje script de la página.
- las etiquetas son <% %>

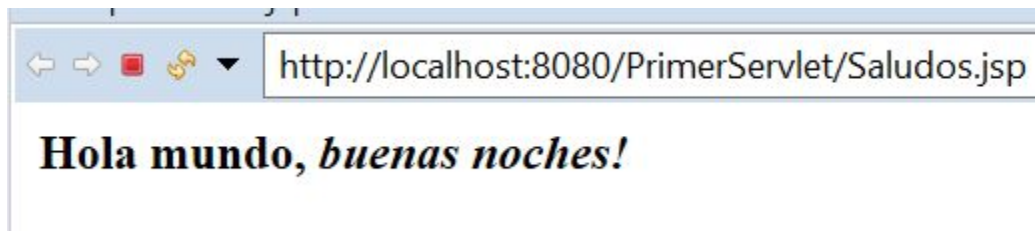
```
<% int suma = 1+1; %>
```



¿se te ocurre cómo hacer esta página dinámica de acuerdo al horario en que el usuario la visite?:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Ejemplo de Scriplets</title>
</head>
<body>
  <%
    java.util.Calendar now = java.util.Calendar.getInstance();
    int hora = now.get(java.util.Calendar.HOUR_OF_DAY);
  %>
  <b>Hola mundo,
  //si son más de las 6 y menos de las 12, dirá "buenos días"
  //si son más de las 12 y menos de las 20, dirá "buenas tardes"
  //para el resto de los horarios, dirá "buenas noches"
  </b>
</body>
</html>
```

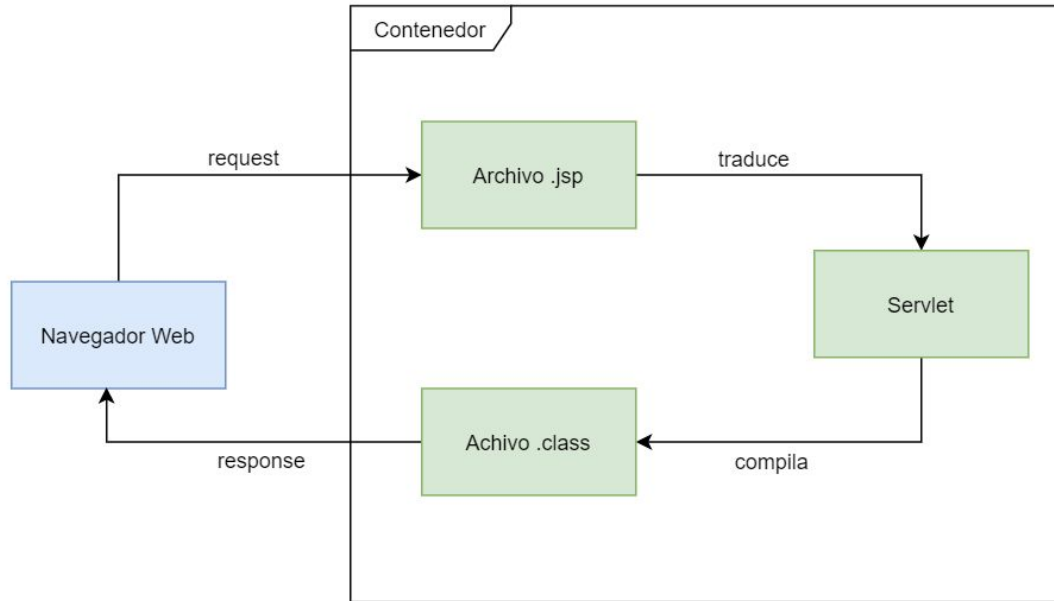
Saludando con scriptlets.



```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Ejemplo de Scriptlets</title>
</head>
<body>
  <%
    java.util.Calendar now = java.util.Calendar.getInstance();
    int hora = now.get(java.util.Calendar.HOUR_OF_DAY);
  %>
  <b>Hola mundo,
    <% if ((hora >= 6) && (hora <= 12)) {%>
      buenos días!
    <% } else if ((hora > 12) && (hora < 20)) {%>
      buenas tardes!
    <% } else {%>
      buenas noches!
    <% };%>
  </b>
</body>
</html>
```

# Un poco de arquitectura

Con JSP podemos incrustar código java y ciertas acciones predefinidas como inicialización de variables e importaciones en una página web con HTML.



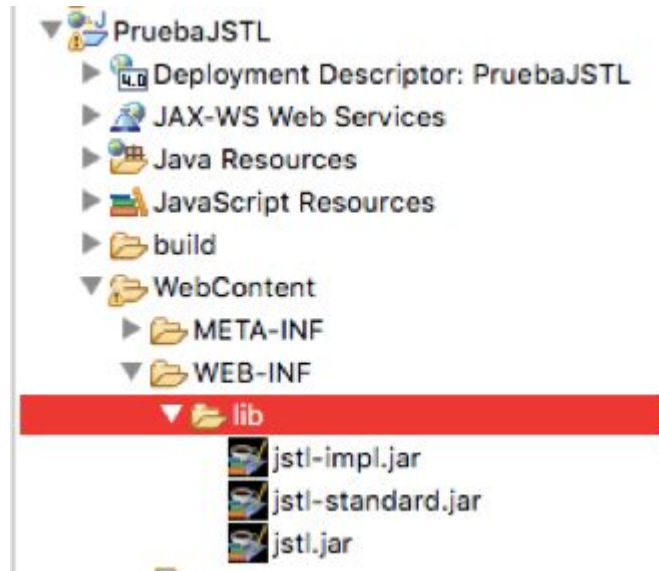
- Proporcionan una manera fácil de mantener los componentes de una página JSP.

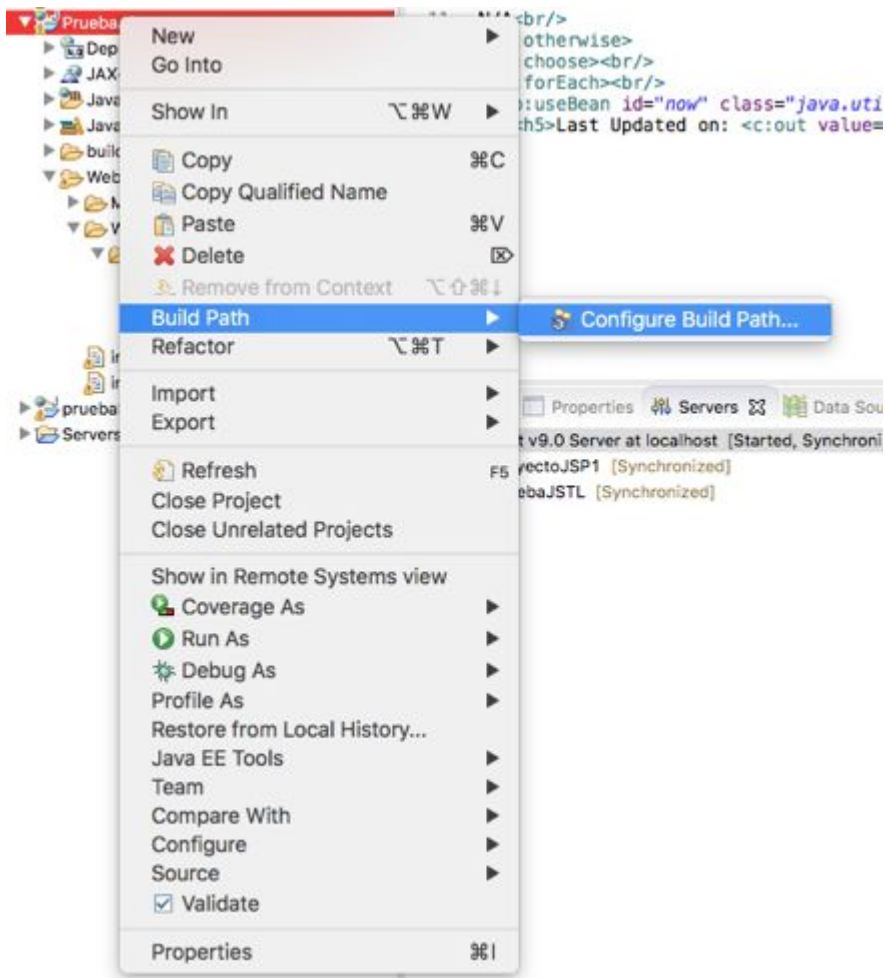
Las características más significativas de los JSTL
JSTL también es JSP, siendo un conjunto complementario de este.
Utiliza 4 librerías estándar: SQL, XML, CORE, INTERNALIZACIÓN
JSTL define un nuevo lenguaje de expresiones llamado EL.
Al usar una etiqueta JSTL, lo que hacemos es añadir una acción.
Una etiqueta JSTL está delimitada por <code>\${ }</code>

# Instalación de JSTL en un proyecto web

La librería JSTL está compuesta por una serie de clases empaquetadas en forma de jar.

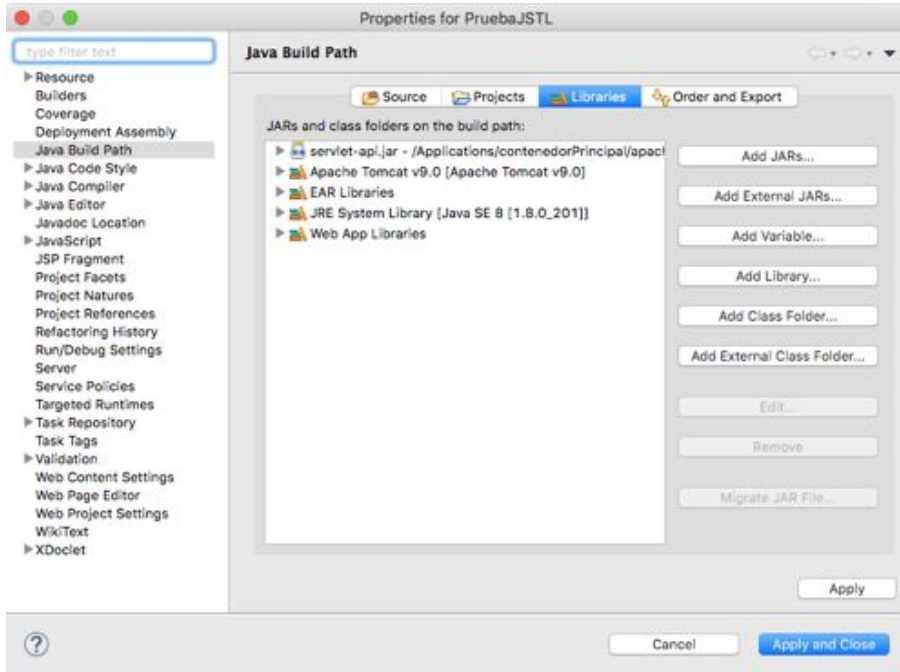
## Incorporando JSTL.



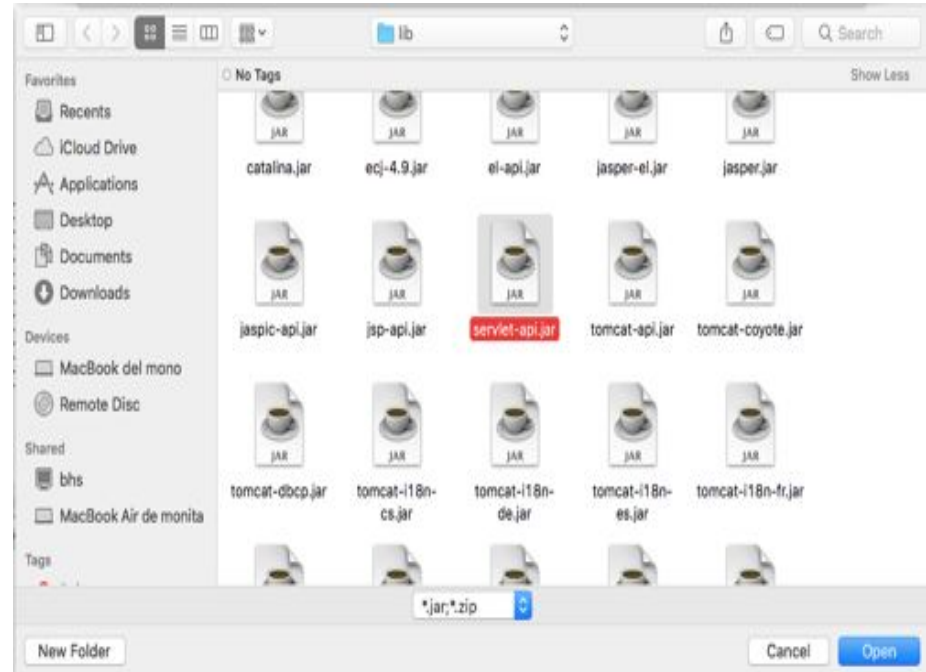


**Agregando la  
librería JSTL**

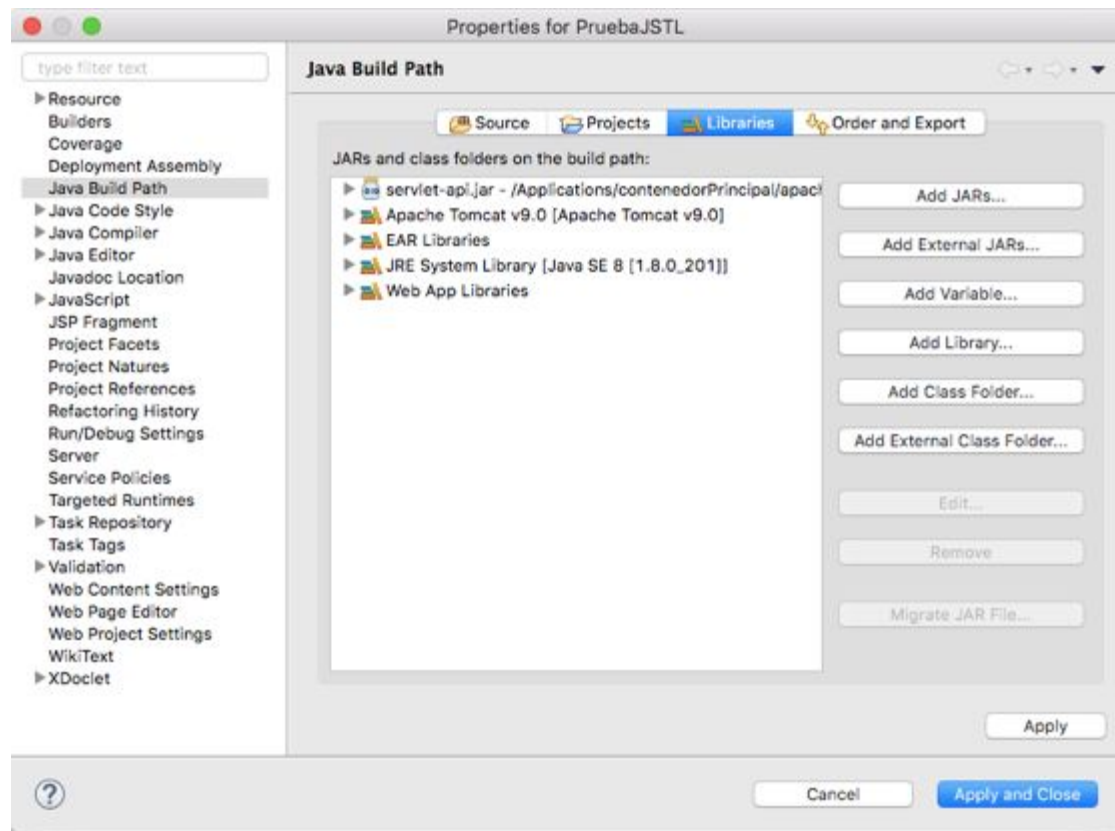
# Add External JARs



# Agregar la librería







Aplicar los  
cambios.

# Descargar las librerías necesarias

Crear una carpeta con un nombre, y dentro de ella debes alojar los siguientes jar:



jstl-impl.jar



jstl-standard.jar



jstl.jar

40. [Download jstl-impl.jar](#)

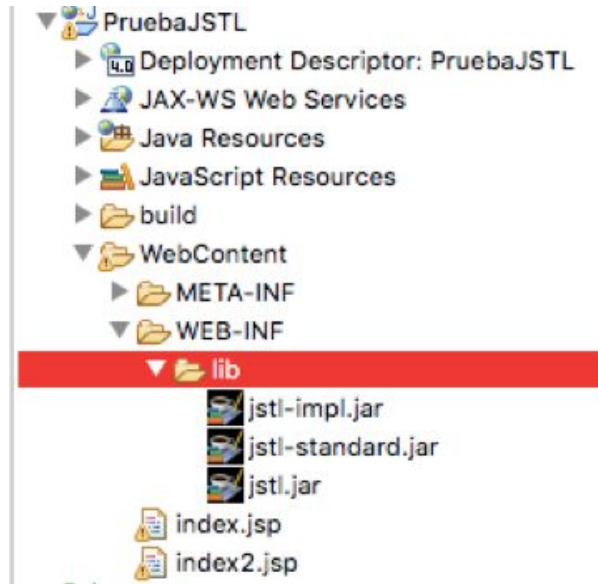
41. [Download jstl-standard.jar](#)

42. [Download jstl.jar](#)



**Descarga de  
librerías**

Agregar los .jar a la carpeta lib.



Exportamos las 4 bibliotecas básicas en el archivo jsp.

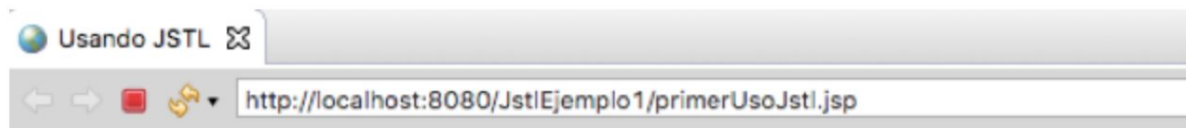
```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
```

# Usando la librería

Podemos utilizar los tags que queramos.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Usando JSTL</title>
</head>
<body>
  <p>Cadena de caracteres: <strong><c:out value="1+2+3"/> </strong></p>
</body>
</html>
```

Resultado usando la librería.



Cadena de caracteres: **1+2+3**

# Tags JSTL

Para referenciar la librería JSTL Core en una página JSP.

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

Tabla con los múltiples tags disponibles.

Area	Subfunction	Prefix
Core	Variable support	c
	Flow control	
	URL management	
	Miscellaneous	
XML	Core	x
	Flow control	
	Transformation	
I18N	Locale	fmt
	Message formatting	
	Number and date formatting	
Database	SQL	sql
Functions	Collection length	fn
	String manipulation	

# Librerías CORE

- Incluyen variables para el control de flujo, manejo de variables, administración de URLs, y algunas funciones misceláneas.
- El tag de salida estándar de código es el c:out.



# ¿Qué ventaja tiene JSTL sobre los scriptlets?

## Ejemplo:

```
<html>
<head>
  <title>Contando de 1 a 10 en JSP con
Scriptlet </title>
</head>
<body>
  <%
    for (int i = 1; i <= 10; i ++){%>
    <% = i%> <br/>
    <%}%>
  </body>
</html>
```

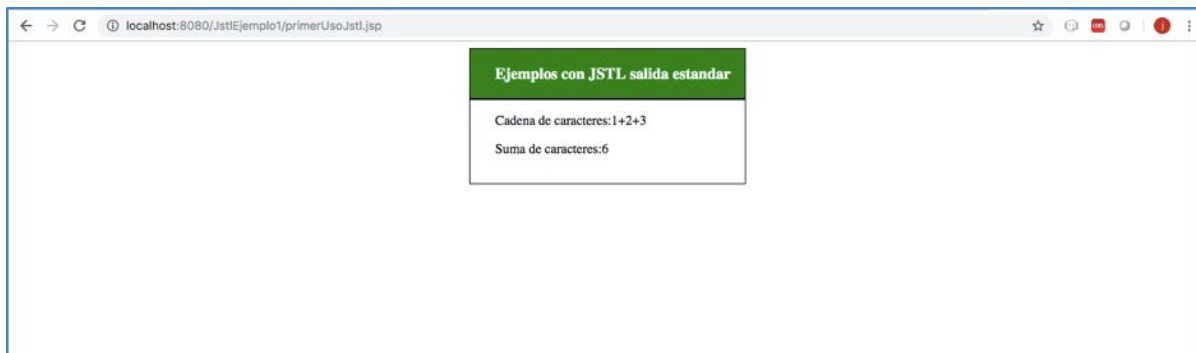
{desafío}  
latam\_

## Desarrollo del desafío con JSTL:

```
<% @ taglib uri =
"http://java.sun.com/jstl/core" prefix = "c"%>
<html>
  <head>
    <title>Contando de 1 a 10 en JSP con
JSTL</title>
  </head>
  <body>
    <c:forEach var = "i" begin = "1" end = "10"
step = "1">
      <c:out value="${i}" />
    <br />
    </c:forEach>
  </body>
</html>
```

- **Ejercicio: Utilización de tag <c: out>**

- Por un lado el texto "1+2+3" y debajo de él implementaremos la suma de esos valores.




## Archivo primerUsoJstl.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Usando JSTL</title>
<style>
#texto{
    margin:0 auto;
    width:300px;
    height:100px;
    border:solid 1px black;
    padding-left:30px;
}
```

```
#cabecera{
  margin:0 auto;
  width:300px;
  border:solid 1px black;
  padding-left:30px;
  background-color:green;
  color:white;
}
</style>
</head>
<body>
  <div id="cabecera">
    <h3>Ejemplos con JSTL salida estandar </h3>
  </div>
  <div id="texto">
    <p>Cadena de caracteres:<c:out value="1+2+3"/></p>
    <p>Suma de caracteres:<c:out value="\${1+2+3}"/></p>
  </div>

</body>
</html>
```

# Que es un formulario web



Puente entre el  
usuario y las  
bases de datos

lógica real de  
la empresa.

Componentes  
que capturan  
los datos del  
usuario

# Ejercicio: Envío de valores entre jsp

Crearemos un formulario con la entrada de dos valores, que luego de ser enviados serán recibidos por un jsp con las etiquetas c: out.

**Resultado envio de valores entre jsp.**

**Resultado al recibir los parámetros.**



FORMULARIO DE ENVÍO

Nombre:  Apellido:



Ejemplos con JSTL salida estandar

Nombre: Desafio  
Apellido: Latam

El código es el siguiente para el archivo formulario.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Usando JSTL</title>
<style>
//codigo de estilo
</style>
</head>
<body>
<div id="cabecera">
<h3>FORMULARIO DE ENVIO</h3>
```

```
</div>
<div id="texto">
  <form action="primerUsoJstl.jsp" method="POST">
    <label>Nombre</label>
    <input type="text" placeholder="Ingrese nombre" id="nombre" name="nombre">
    <label>Apellido</label>
    <input type="text" placeholder="Ingrese apellido" id="apellido"
name="apellido">
    <input type="submit" value="enviar">
  </form>
</div>
</body>
</html>
```



## Ejercicio 3: Envío de datos entre formularios

Vamos a generar un pequeño formulario en un archivo jsp, el cual simplemente enviará sus valores a otro formulario jsp y los desplegará en la pantalla.

**Para utilizar bootstrap, dentro de la etiqueta <head></head> pegar las siguientes líneas.**

```
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"

integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFf/njGzIxFDsf4x0xIM+B07j
RM" crossorigin="anonymous"></script>

<link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw
1T" crossorigin="anonymous">
```

# Formulario de datos

# Resultado del envío de datos

← → ↻ ⓘ localhost:8080/JstEjemplo1/formulario.jsp

Dirección de correo

Nombre Completo

Dirección

Password

No le des la contraseña de tu correo a alguien más

http://localhost:8080/JstEjemplo1/procesaForm.jsp

Hemos recibido datos

Nombre enviado desde formulario: desafio latam

Dirección enviada desde formulario: santiago de chile

Correo enviado desde formulario: desafio@desafio.cl

Password enviado desde formulario: no puedes verlo

## El código del formulario.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Usando JSTL</title>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07j
RM" crossorigin="anonymous"></script>
<link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw
1T" crossorigin="anonymous">
<style>
.
```

```

container{
    margin-top: 39px;
    border: solid 1px;
    padding-bottom: 30px;
}
</style>
</head>
<body>
<div class="container">
    <div class="row">
        <div class="col">
            <form action="procesaForm.jsp" method="POST">
                <div class="form-group">
                    <label for="exampleInputEmail">Direccion de
correo</label>
                    <input type="email" class="form-control" id="correo"
name="correo" aria-describedby="emailHelp" placeholder="Enter email" required>
                </div>
                <div class="form-group">
                    <label for="exampleInputEmail">Nombre
Completo</label>

```

```

        <input type="text" class="form-control" id="nombre"
name="nombre" placeholder="Ingresa Nombre" required>
    </div>
    <div class="form-group">
        <label for="exampleInputEmail">Direccion</label>
        <input type="text" class="form-control"
id="direccion" name="direccion" placeholder="Ingresa Direccion" required>
    </div>
    <div class="form-group">
        <label for="exampleInputPassword">Password</label>
        <input type="password" class="form-control" id="pass"
name="pass" placeholder="Password" required>
        No le des la contraseña de tu correo a alguien mas
    </div>
    <button type="submit" class="btn btn-primary">Enviar </button>
</form>
</div>
</div>
</body>
</html>

```

Código del archivo procesa, donde se reciben los datos.

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
4 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
5 <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <meta charset="UTF-8">
10 <title>Usando JSTL</title>
11 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60Q6VrjIEa"
12 <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-ggOyR0iXCbMQv3Xipma3
13 <style>
14 .container{
15     margin-top:39px;
16     border:solid 1px; black;
17     padding-top : 30px;
18     padding-bottom: 30px;'
19 }
20 #datos{
21     padding-left:30px;
22 }
23 </style>
24 </head>
25 <body>
26 <div class="container">
27 <div id="datos">
28 <h1>Hemos recibido datos</h1>
29 <br>
30 <h2>Nombre enviado desde formulario: <c:out value="${param.nombre}"/></h2>
31 <br>
32 <h2>Direccion enviada desde formulario: <c:out value="${param.direccion}"/></h2>
33 <br>
34 <h2>Correo enviado desde formulario: <c:out value="${param.correo}"/></h2>
35 <br>
36 <h2>Password enviado desde formulario: no puedes verlo</h2>
37 <br>
38 <a href="formulario.jsp"><button type="button" class="btn btn-primary">Volver</button></a>
39 </div>
40 </div>
41 </body>
42 </html>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Usando JSTL</title>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEeAff/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
<link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
<style>
.container{
margin-top: 39px;
```

```

        border: solid 1px;
        padding-bottom: 30px;
    }
    #datos{
        padding-left:30px;
    }
</style>
</head>
<body>
<div class="container">
    <div id="datos">
        <h1>Hemos recibido datos</h1>
        <br>
        <h2>Nombre enviado desde formulario: <c:out value="\${param.nombre}"/></h2>
        <br>
        <h2>Direccion enviada desde formulario: <c:out
value="\${param.direccion}"/></h2>
        <br>
        <h2>Correo enviado desde formulario: <c:out value="\${param.correo}"/></h2>
        <br>
        <h2>Password enviado desde formulario: no puedes verlo</h2>
        <br>
        <a href="formulario.jsp"><button type="button" class="btn

```



```
    btn-primary">Volver</button></a>  
  </div>  
</div>  
</body>  
</html>
```



# Cierre

{desafío}  
latam\_



15 minutos



*Academia de  
talentos digitales*

[www.desafiolatam.com](http://www.desafiolatam.com)



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam