



Recibe una cálida:

# ¡Bienvenida!

---

Te estábamos esperando 😊 

# » Java Server Pages

---

**Plan formativo:** Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

# HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



# REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Configuración de Tomcat dentro del proyecto
- ✓ Ejecución de un proyecto web dinámico



# LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

5.2

Start! 🏁

Java Server Pages

Vistas JSP  
JSTL (Java Servlet Tag Libs)

Java Server Pages

JSP

Java Servlet Tag Libs

JSTL

# OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



*Comprender el concepto e implementación de las Java Server Pages*



*Aprender a utilizar los tags JSTL*





# Rompehielo 🧊

**Respondan en el chat o levantando la mano 🙋**

¿Creen que es posible escribir código Java en archivos HTML? ¿Por qué?

¿Qué les parece el siguiente código? ¿Podría funcionar? ¿Por qué?

```
3 <html>
4 <body>
5 <%
9 <p>Hola <%= nombre %>, veo que tienes <%= edad %> años.</p>
10
11 </body>
12 </html>
```



# ➤ Java Server Pages (JSP)

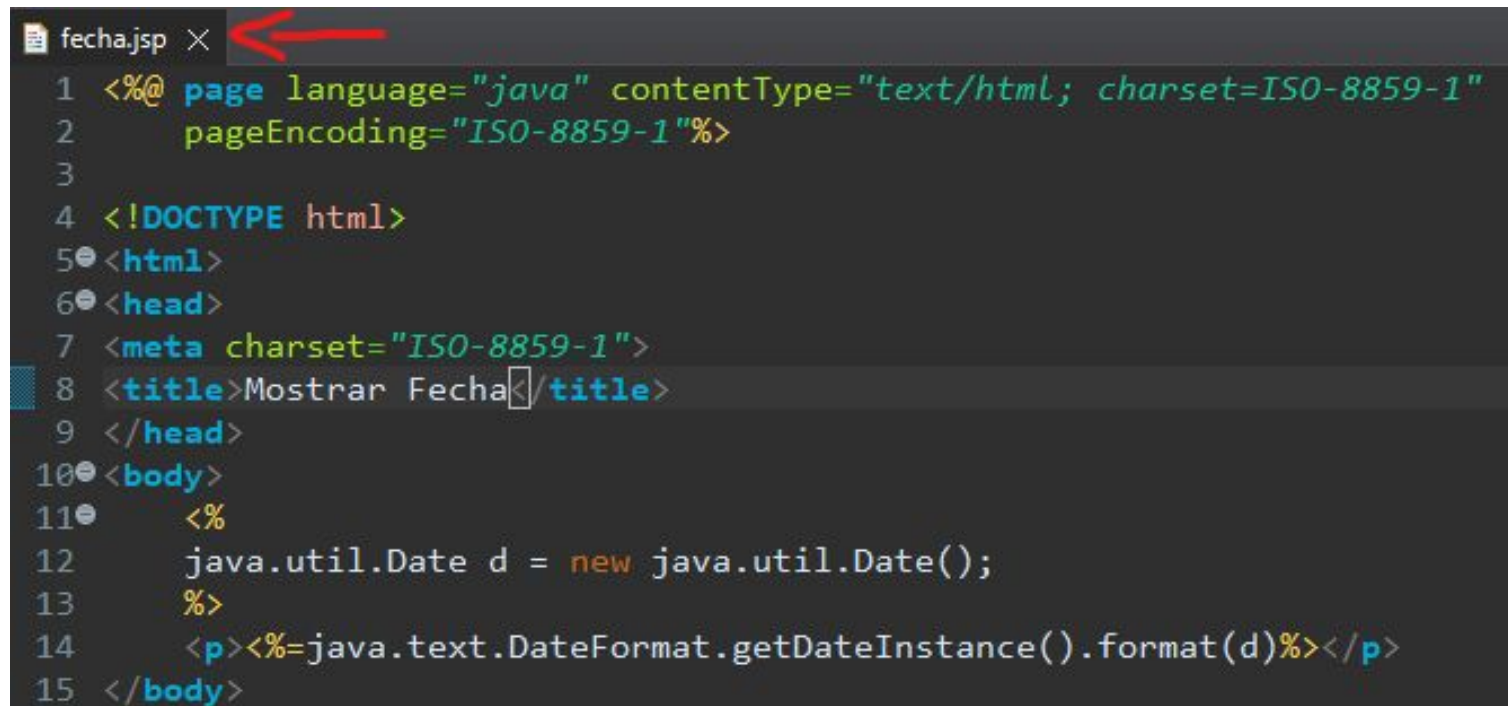




# Java Server Pages (JSP)

JSP (JavaServer Pages) es una tecnología que permite incluir código Java en páginas web.

El denominado contenedor JSP (que sería un componente del servidor web) es el encargado de tomar la página, sustituir el código Java que contiene por el resultado de su ejecución, y enviarla al cliente. Así, se pueden diseñar fácilmente páginas con partes fijas y partes variables.



```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="ISO-8859-1">
8   <title>Mostrar Fecha</title>
9 </head>
10 <body>
11   <%
12     java.util.Date d = new java.util.Date();
13   %>
14   <p><%=java.text.DateFormat.getDateInstance().format(d)%></p>
15 </body>
```



# Java Server Pages (JSP)

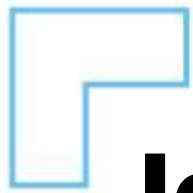
## ¿Cómo funciona internamente JSP?

1. Los .jsp generan .java (servlets). Los servlets compilados son .class.

Es el contenedor de aplicaciones (tomcat, etc) el que convertirá los jsp en servlets.

2. El servidor leerá el archivo jsp. Luego, va a buscar el jsp compilado (el .class), si no lo encuentra, lo compila.
  - **Traducción:** pasar de jsp a java.
  - **Compilación:** pasar de java a .class





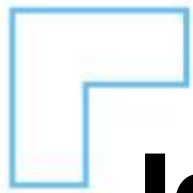
# Java Server Pages (JSP)



Existen tres tipos de elementos JSP que podemos insertar en una página web:

- **Código:** podemos "incrustar" código Java de distintos tipos (declaraciones de variables y/o métodos, expresiones, sentencias) para que lo ejecute el contenedor JSP.
- **Directivas:** permiten controlar distintos parámetros del servlet resultante de la traducción automática del JSP
- **Acciones:** normalmente sirven para alterar el flujo normal de ejecución de la página (p.ej. redirecciones), aunque tienen usos variados.

Se pueden poner **comentarios** en una página JSP entre los símbolos **<%-- y --%>**. El contenedor JSP ignorará todo lo contenido entre ambos. Dentro de los fragmentos de código Java también se pueden colocar comentarios siguiendo la sintaxis habitual del lenguaje.



# Java Server Pages (JSP)



## Algunas ventajas de JSP:

1. **Fácil de mantener:** se puede administrar fácilmente porque podemos separar nuestra lógica de negocios con la lógica de presentación.
2. **Desarrollo rápido:** Si se modifica la página JSP, no es necesario volver a compilar ni implementar el proyecto.
3. **Menos código que Servlet:** En JSP, podemos usar muchas etiquetas, como etiquetas de acción, JSTL, etiquetas personalizadas, etc., que reducen el código.



# Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro...

## Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase.



# LIVE CODING

Ejemplo en vivo

## Creando un archivo .jsp:

*Vamos a crear una página JSP llamada '**index.jsp**'. La página debe estar dentro de la carpeta '**webapp**' del proyecto.*

*Esta página debe tener en la etiqueta `<body>` debe incluir lo siguiente:*

```
<body>  
    <% java.util.Date d = new java.util.Date(); %>  
    <p><%=java.text.DateFormat.getDateInstance().format(d)%></p>  
</body>
```



# LIVE CODING

Ejemplo en vivo

*A continuación, vamos a correr el proyecto en el servidor para que Tomcat muestre en el localhost lo que acabamos de codificar.*

*Deberíamos poder ver la fecha del día de hoy en el navegador determinado!*

 **Tiempo: 20 minutos**



# Ejercicio N° 1

# JSP





## Manos a la obra: 🙌

Ahora es tu turno de agregar un archivo 'index.jsp' al proyecto AlkeWallet.

## Consigna: 📝

- 1- Crear el archivo index.jsp en la carpeta webapp
- 2- Colocar 'Inicio' como título de pestaña
- 3- Colocar un párrafo que diga 'En este momento la hora es: '
- 4- Colocar una etiqueta jsp con el código Java necesario para mostrar la hora actual.



**Tiempo** 🕒: 20 minutos

# › Java Standard Tag Libs

# JSTL



## ¿Qué es?

Es un conjunto de etiquetas (**tags**) standard que encapsulan funcionalidades de uso común para muchas aplicaciones con JSPs.

Debido a que las etiquetas JSTL son XML, se integran uniformemente con las etiquetas HTML y serán fáciles de usar por alguien que conozca html.

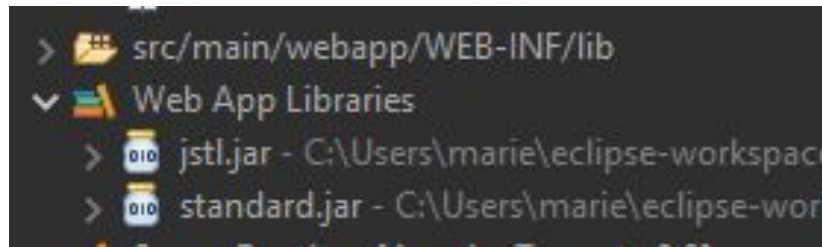
Las etiquetas JSTL están específicamente preparadas para realizar las tareas que van a tener lugar en la lógica de la presentación.



# JSTL

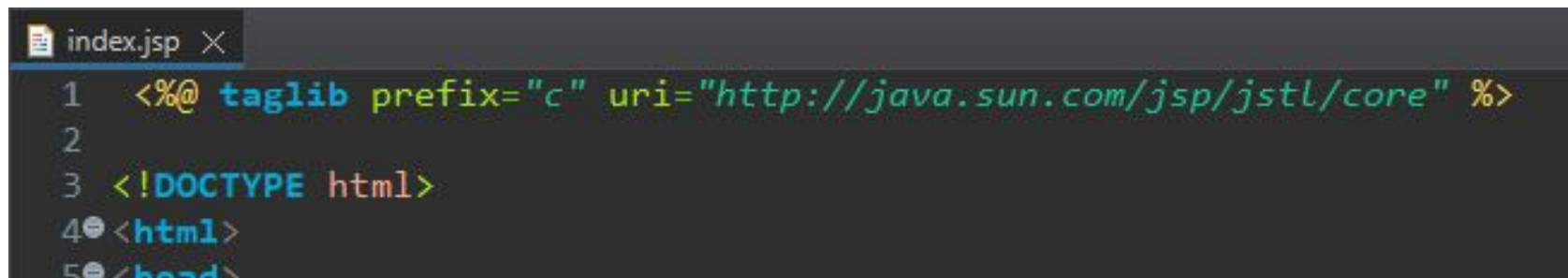
## Para utilizar las tags de JSTL es necesario cargar JSTL en nuestro proyecto:

Cargamos los archivos **jstl.jar** y **standard.jar** en la carpeta **lib** de nuestro proyecto. De esta forma no hará falta añadirlo al Java Build Path.



Luego, incluimos la siguiente etiqueta en la cabecera de nuestro jsp, como vemos en la imagen

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





# Ejercicio N° 2

# JSTL

# JSTL



## Manos a la obra: 🙌

Vamos a incluir JSTL en nuestro proyecto AlkeWallet

## Consigna: ✍️

- 1- Incluir los archivos **jstl.jar** y **standard.jar** en la carpeta **lib** del proyecto
- 2- Incluir el taglib necesario en la página 'index.jsp'

¡Es hora de poner a prueba lo aprendido!

**Tiempo🕒:** 15 minutos

○

# ¿Alguna consulta?

+



# RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender el concepto e implementación de JSP**
- ✓ **Reconocer el uso y configuración de JSTL**





# #WorkingTime

Continuemos ejercitando

**¡Antes de cerrar la clase!** Te invitamos a: 🙌🙌🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
  - a. *Lectura Modulo 5, Lección 2: páginas 1 - 2*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

Momento: ✚

# Time-out!

🕒 5 min.

