



Patrones de diseño

Sesión Conceptual 1





Inicio

{desafío}
latam_



15 minutos

- Profundizar en los conceptos de JSP, aplicando sus funcionalidades en una estructura con arquitectura MVC.
- Aproximar al trabajo con bases de datos desde una aplicación web mediante la API de JDBC y se dará forma a un sistema.

Objetivo



Desarrollo

{desafío}
latam_



150 minutos

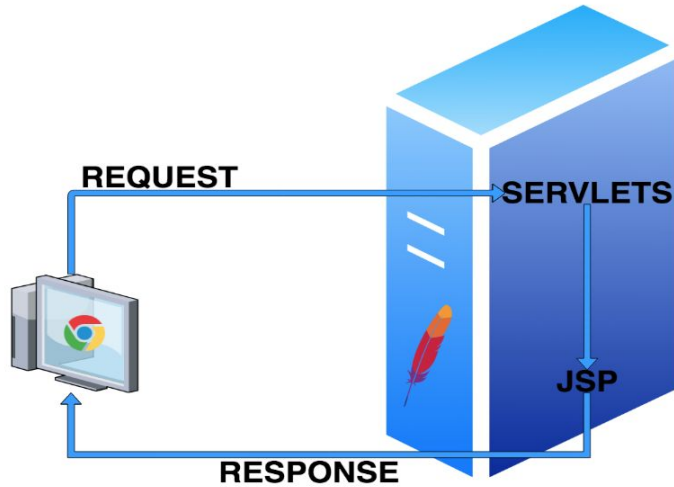
/* Java Server Page */

Conociendo la tecnología JSP

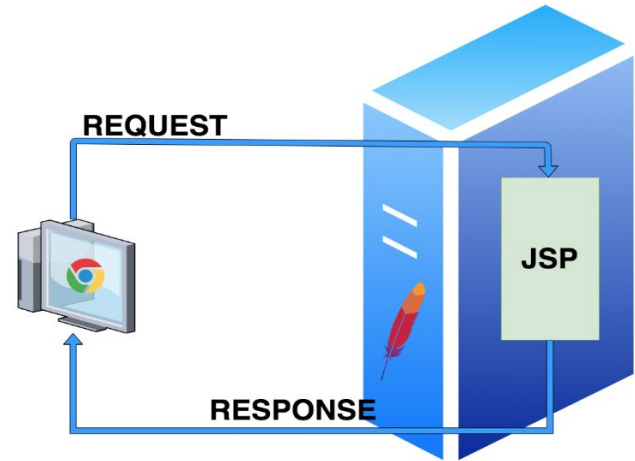
- Tecnología que provee una vía rápida y simple para crear contenido web dinámico.
- JSP está basada en el funcionamiento de los servlets.
- Los JSP se utilizan para generar documentos HTML y XML.

JSP se puede construir utilizando dos mecánicas
Como componente de vista de un diseño del lado del servidor.
Como un componente independiente.

Contexto de trabajo de los JSP

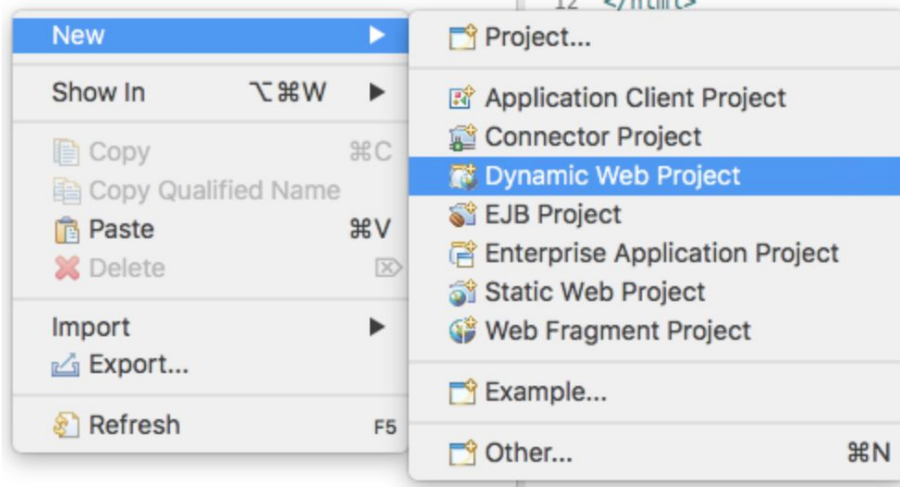


Segundo caso de implementación



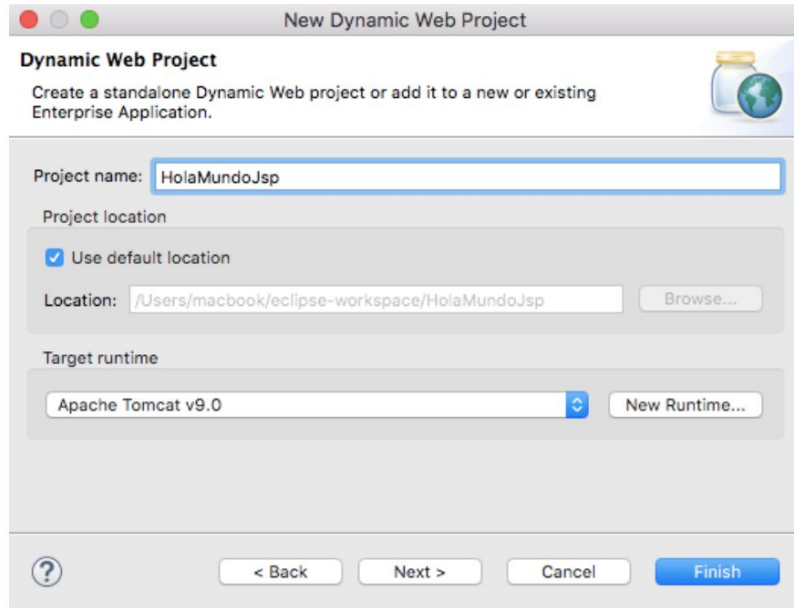
Hola mundo JSP

Creamos en eclipse un nuevo Dynamic Web Project de nombre HolaMundo.jsp.



Crear Dynamic Web Project

Creación de Dynamic Web Project



The screenshot shows the 'New Dynamic Web Project' dialog box. It has a title bar with standard macOS window controls. The main title is 'Dynamic Web Project' with a subtitle 'Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.' and a small icon of a jar and globe. The dialog is divided into three sections: 'Project name' with a text field containing 'HolaMundoJsp'; 'Project location' with a checked 'Use default location' checkbox and a 'Location' text field showing '/Users/macbook/eclipse-workspace/HolaMundoJsp' with a 'Browse...' button; and 'Target runtime' with a dropdown menu showing 'Apache Tomcat v9.0' and a 'New Runtime...' button. At the bottom are buttons for '?', '< Back', 'Next >', 'Cancel', and 'Finish'.

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

Project name: HolaMundoJsp

Project location

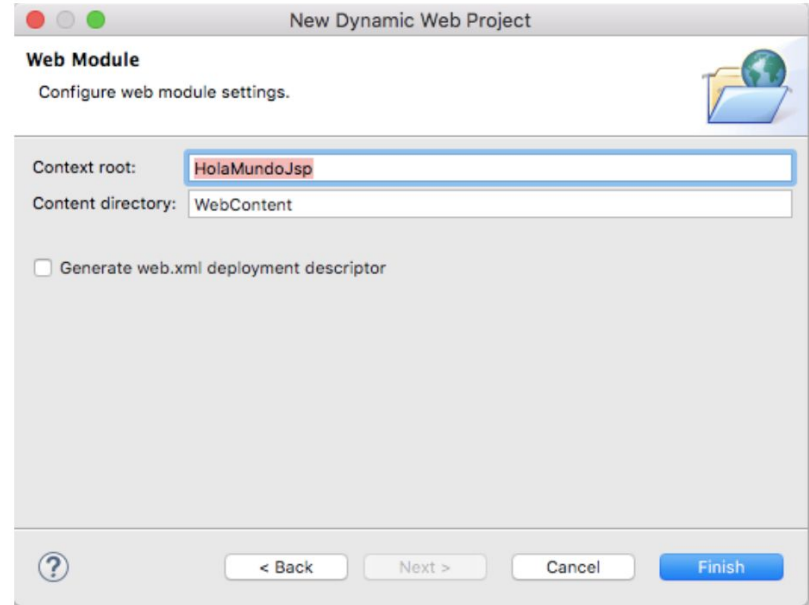
☒ Use default location

Location: /Users/macbook/eclipse-workspace/HolaMundoJsp

Target runtime

Apache Tomcat v9.0

? < Back Next > Cancel Finish



The screenshot shows the 'Web Module' configuration dialog box. It has a title bar with standard macOS window controls. The main title is 'Web Module' with a subtitle 'Configure web module settings.' and a small icon of a folder and globe. The dialog contains two text fields: 'Context root:' with 'HolaMundoJsp' and 'Content directory:' with 'WebContent'. There is an unchecked checkbox for 'Generate web.xml deployment descriptor'. At the bottom are buttons for '?', '< Back', 'Next >', 'Cancel', and 'Finish'.

Web Module
Configure web module settings.

Context root: HolaMundoJsp

Content directory: WebContent

☐ Generate web.xml deployment descriptor

? < Back Next > Cancel Finish



Proyecto 1 JSP Básico

{desafío}
latam_





Proyecto 2 JSP Básico

{desafío}
latam_



Beneficios para los desarrolladores

Ventajas de utilizar JSP en una aplicación empresarial

JSP es una completa implementación de todas las características de un servlet.

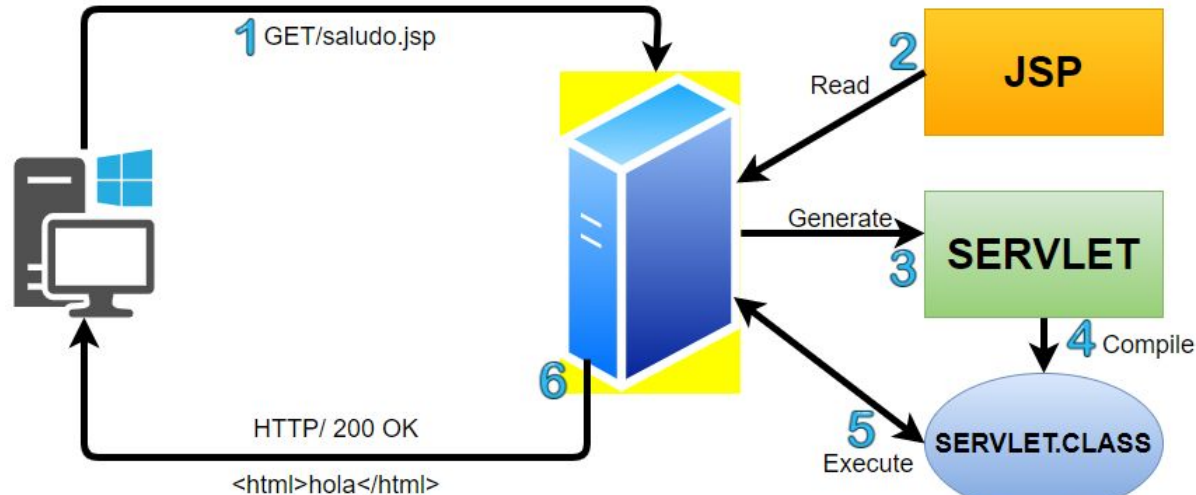
JSP es más fácil hacer la separación entre código de programación java y código de presentación HTML

JSP ya no es necesario recompilar y construir el proyecto cada vez que se haga un cambio.

Con JSP se utiliza menos código para hacer cosas similares que en un servlet.

JSP vs Servlets

El programa java se ejecuta en el lado del servidor y muestra el resultado en el navegador incrustado html que se envía al navegador. El servlet incrusta HTML en el código java a través de las declaraciones `out.println`.



Relación entre JSP y Servlets

JSP se compila primero en un servlet antes de poder hacer cualquier cosa.

- Ventajas de Una página JSP vs un Servlet:
 - JSP mejora sustancialmente la legibilidad del código.
 - JSP no necesita compilación, ni configuración de variables de entorno ni tampoco de empaquetado.
 - En una página JSP, el contenido visual y la lógica están separados
 - Con JSP contamos con la implementación automática.

Ventajas de los JSP frente a otras tecnologías

Otras tecnologías	JSP
Páginas Active Server Pages (ASP)	JSP es una plataforma independiente, es más potente y fácil de usar.
Servlets Puros	El diseñador web puede enfocarse solamente en el html de la página y los programadores pueden insertar código dinámico por separado.
Server Side include (SSI)	Permite usar servlets en lugar de un programa separado para generar esa parte dinámica.
JavaScript	Puede acceder a los recursos del lado del servidor.
HTML estático	html no provee ningún mecanismo de comunicación entre capas, no existe la colaboración ni el envío de mensajes.

- Delimitan el área en que el código java es interpretado.
- Permite que se pueda utilizar lógica utilizando las características del lenguaje.
- Es posible usar y manipular clases java, generar excepciones, conectar a bases de datos, validar variables y utilizar todos los métodos disponibles de la API de java.

Tipo 2 <%= %> Expresiones

Las expresiones son un tipo de scriptlets que se encargan de insertar en la página el resultado de una expresión java encerrada entre los tags <%= %>.

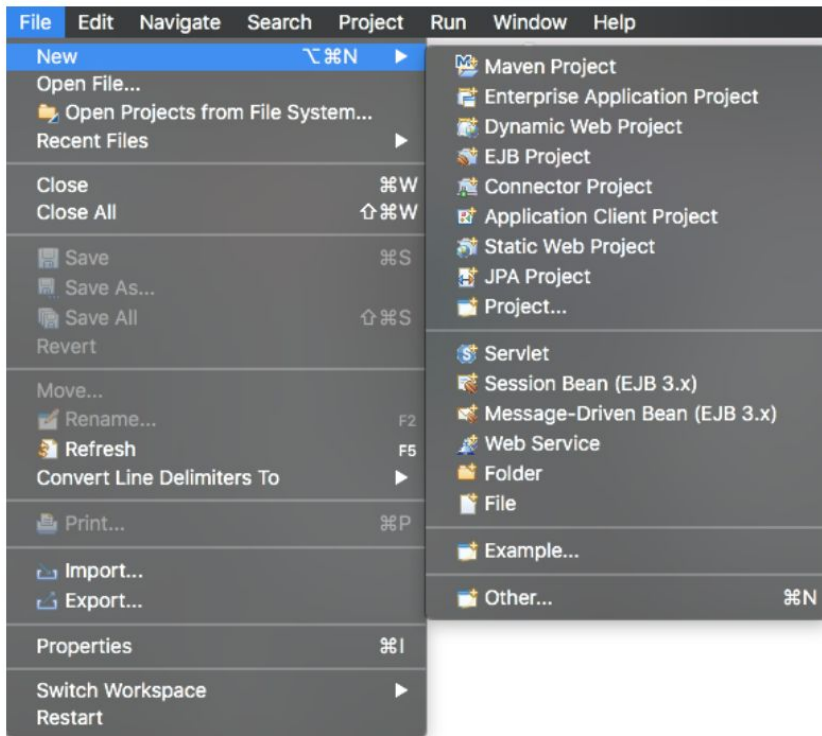
ExpresionEjemplo.jsp

```
<%@ page language="java"
contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@page import="java.util.Date" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Expresion Ejemplo</title>
</head>
<body>
    Fecha actual y tiempo: <%= new
Date() %>
</body>
</html>
```

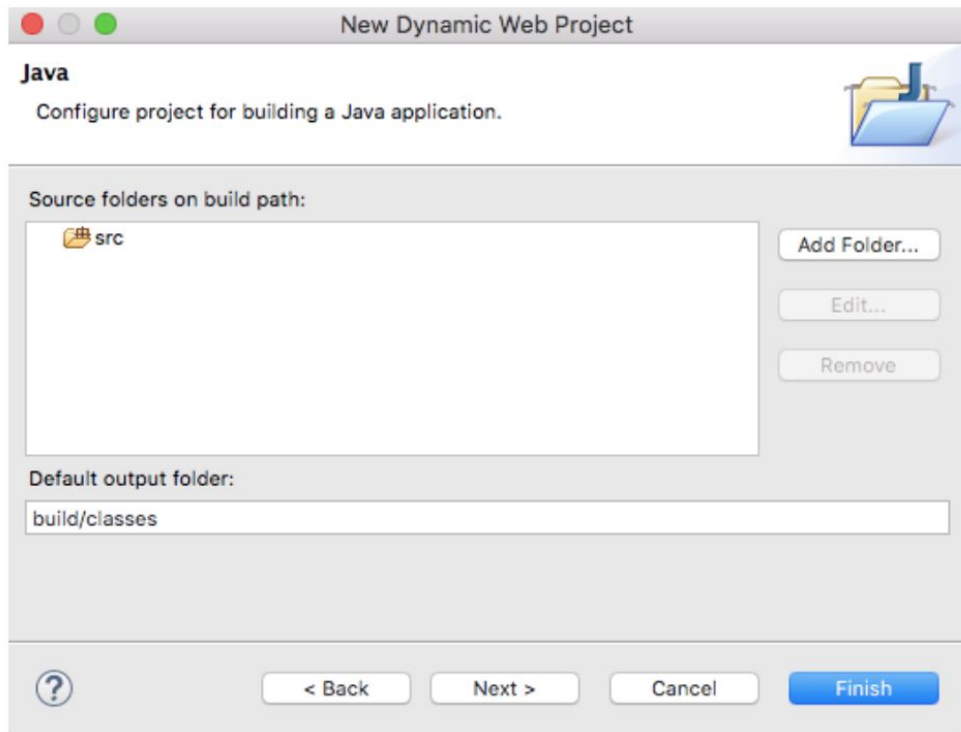
Tipo 3 <%! %> Declaraciones

Son los tags que permiten declarar variables dentro de una página JSP.

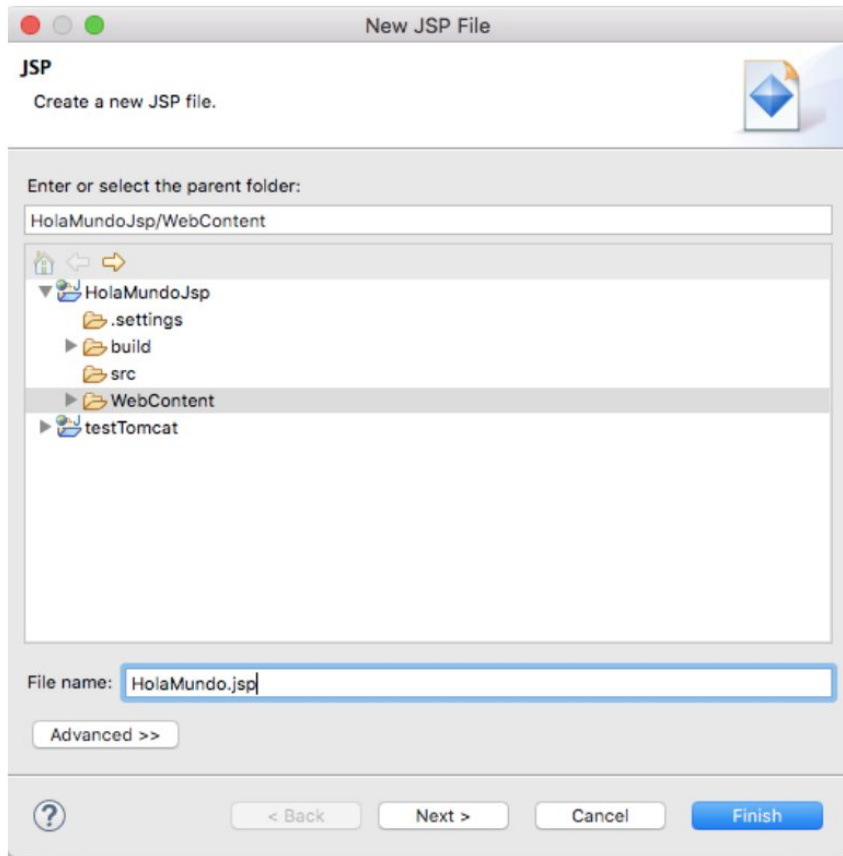
```
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <%! String nombre = "INVITADO"; %>
    <%
        out.println("Bienvenido: " +
nombre);
    %>
</body>
</html>
```



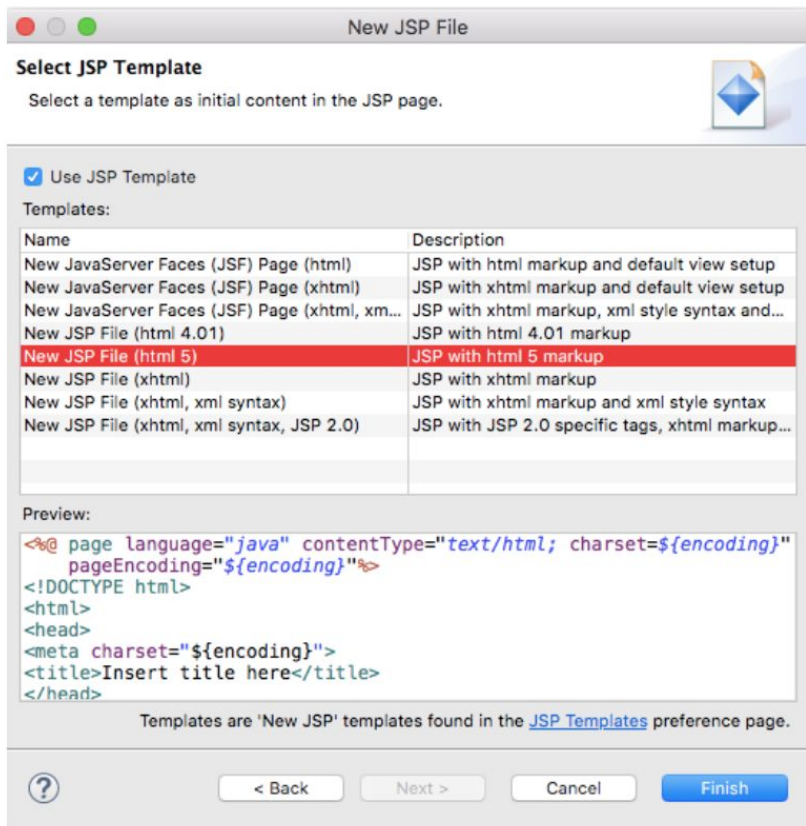
**Crear un nuevo
proyecto
mediante menú**



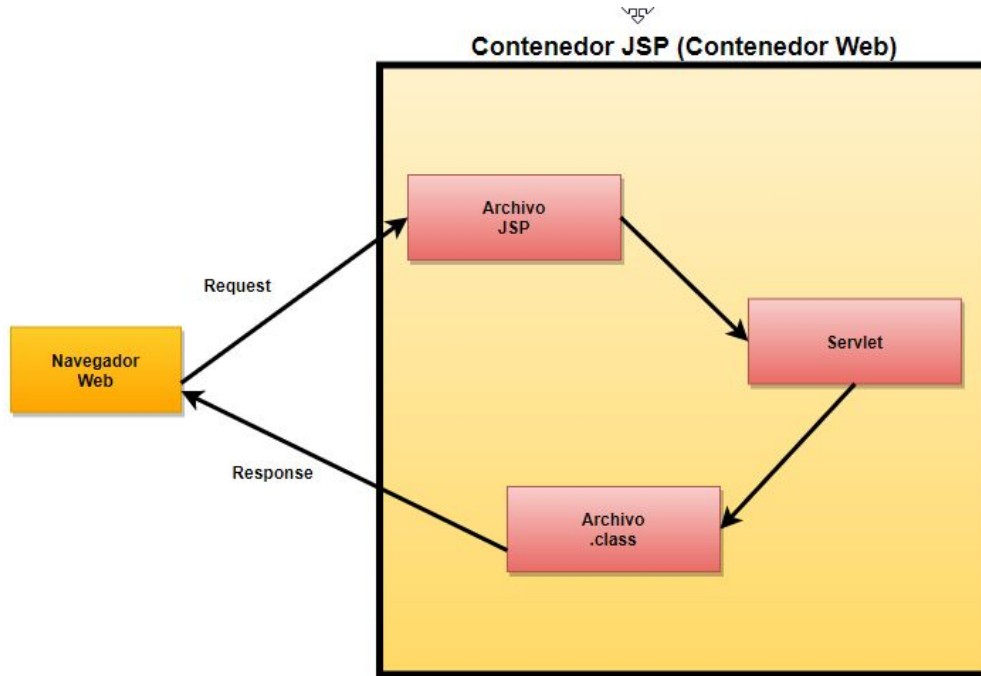
@ Configuración Proyecto Dynamic Web Project



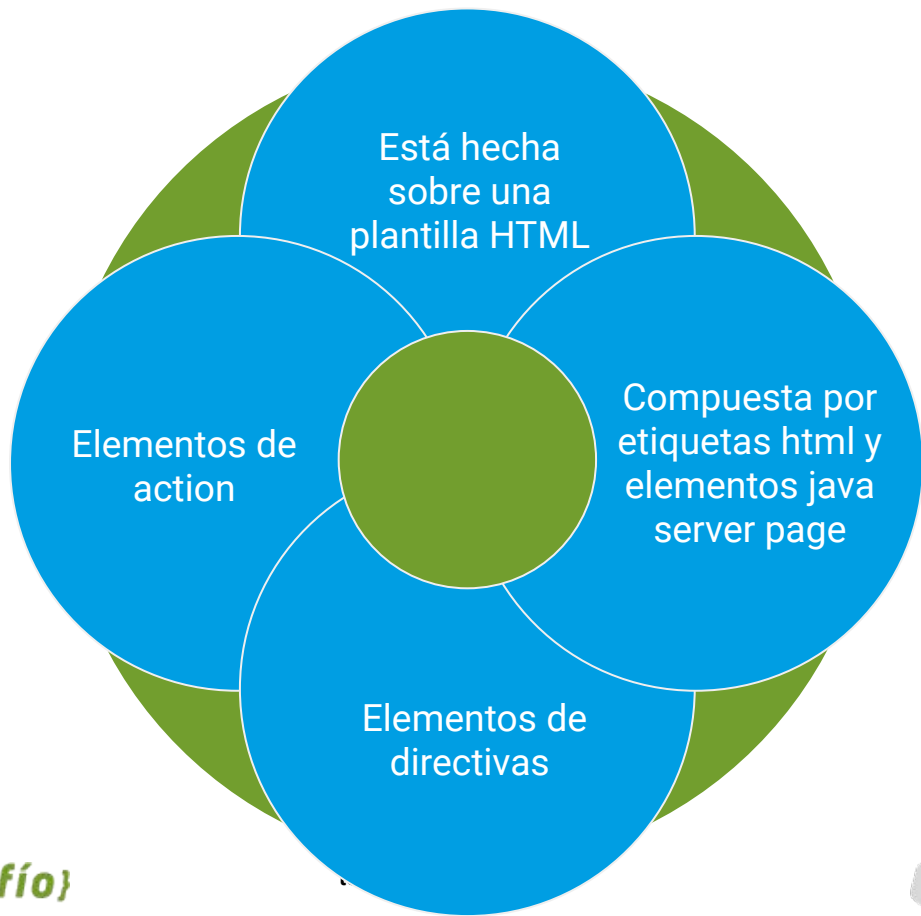
Creación del
nuevo archivo
JSP



Creación de archivo JSP



Arquitectura JSP



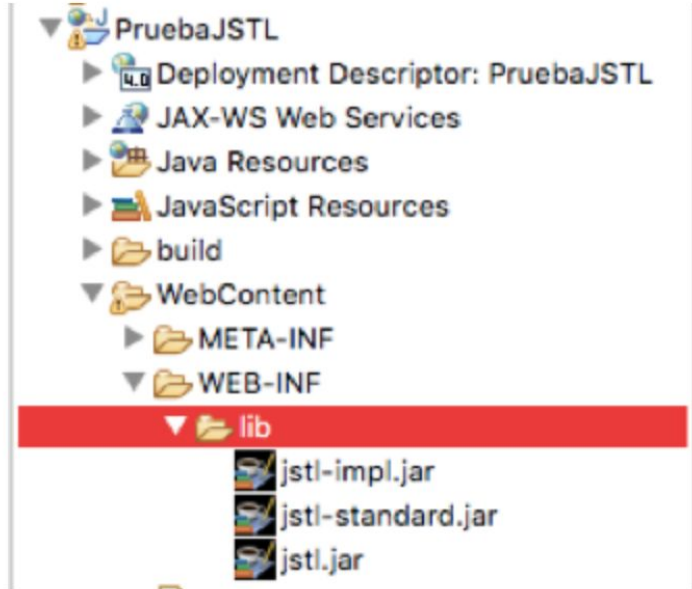
Elementos de los JSP

/* JSTL Taglib */

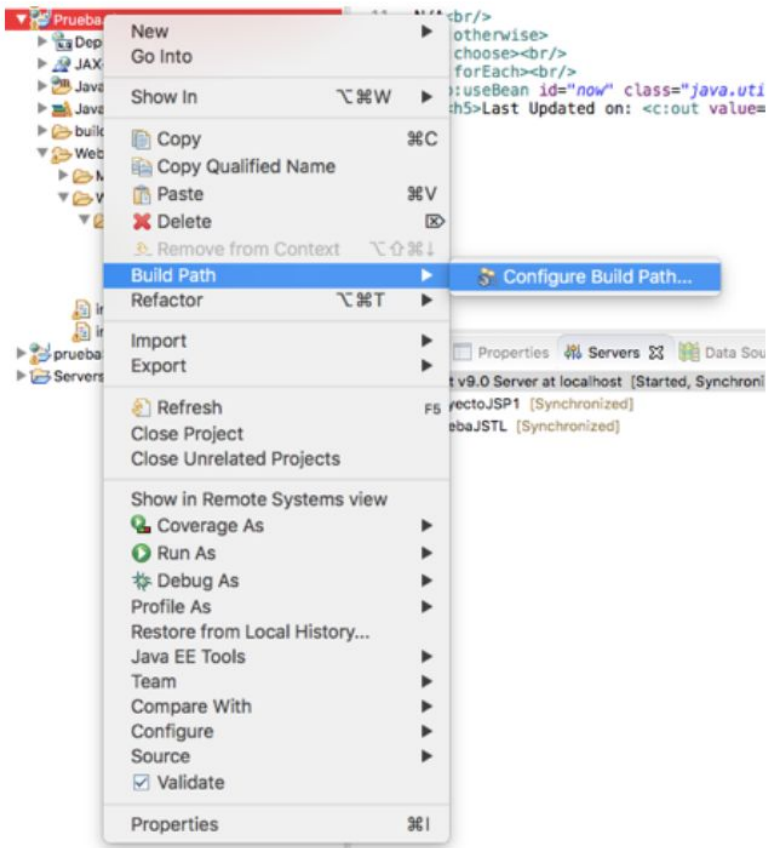
Qué son los JSTL



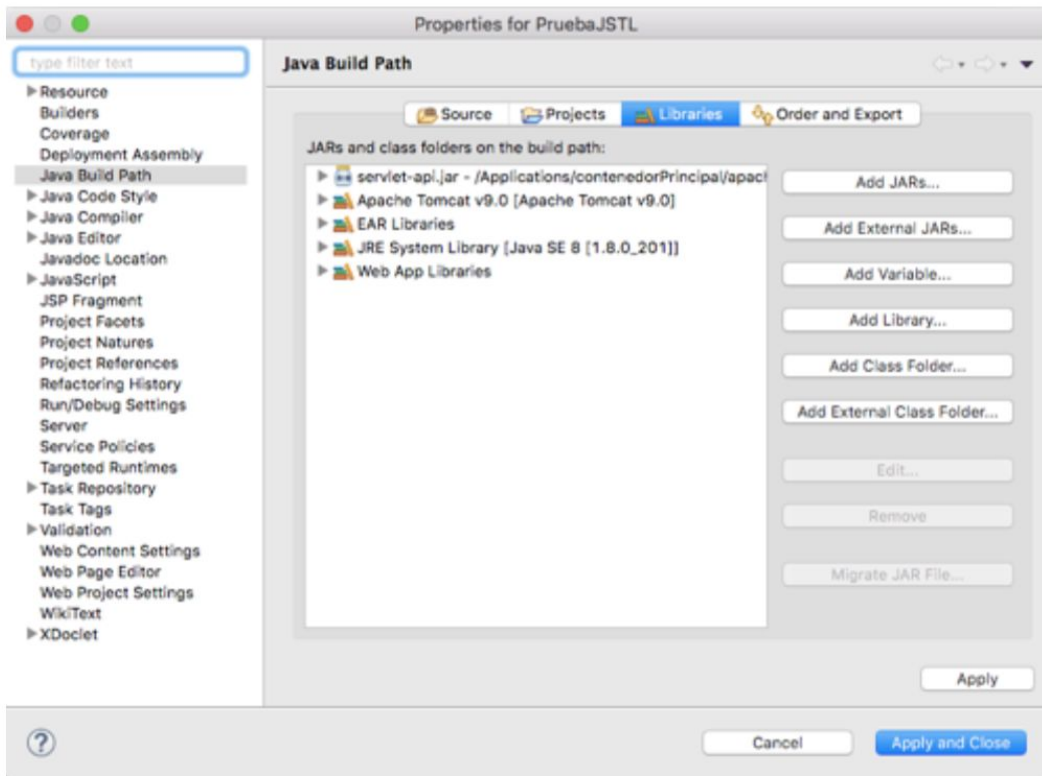
Instalar el JAR en el classpath y además importar las librerías a la carpeta lib del proyecto.



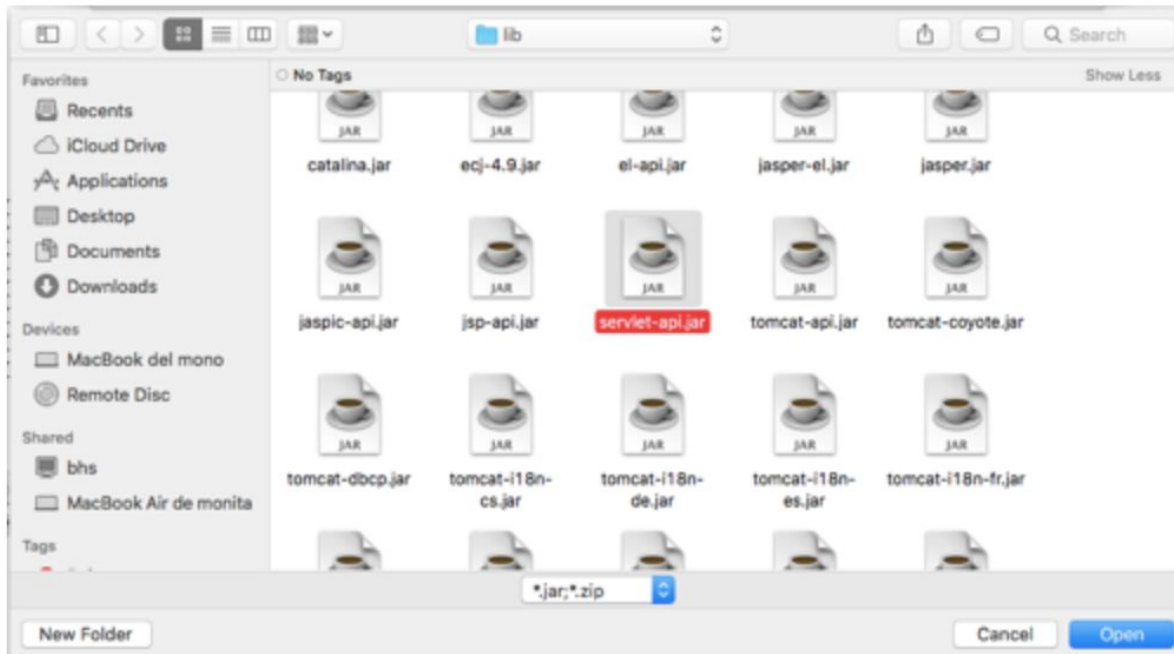
Instalación de
la librería JSTL



Configure
build path

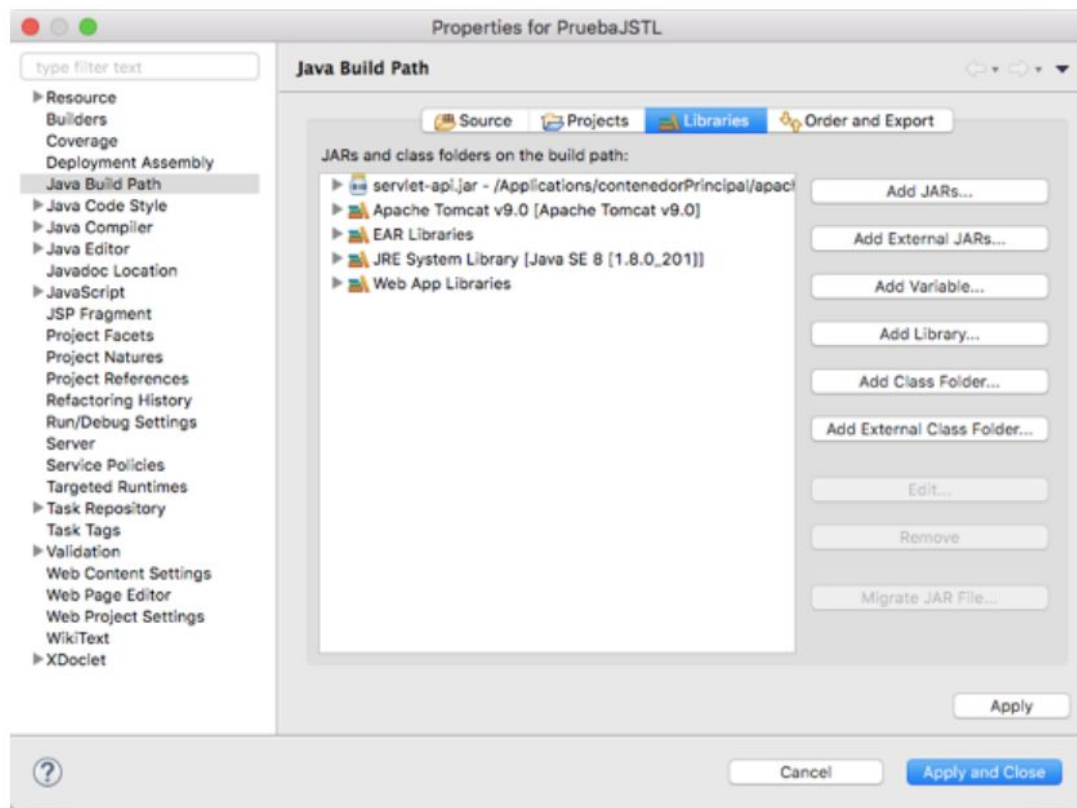


Java Build Path



Librería
servlet-api.jar

{desafío}
latam_



Resultado después
de importar la
librería

Librerías JSTL



jstl-impl.jar



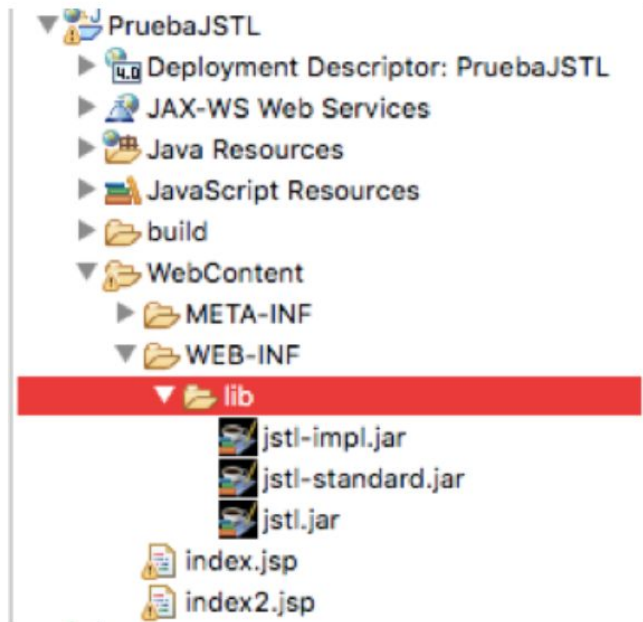
jstl-standard.jar



jstl.jar

- Para descargarlos, dirígete a la dirección:
 - <http://www.java2s.com/Code/Jar/j/Downloadjstlstandardjar.htm>

Librerías JSTL



Funcionalidades JSTL

Set de funcionalidades comunes que están basadas en jsp 1.2 y servlets 2.3.

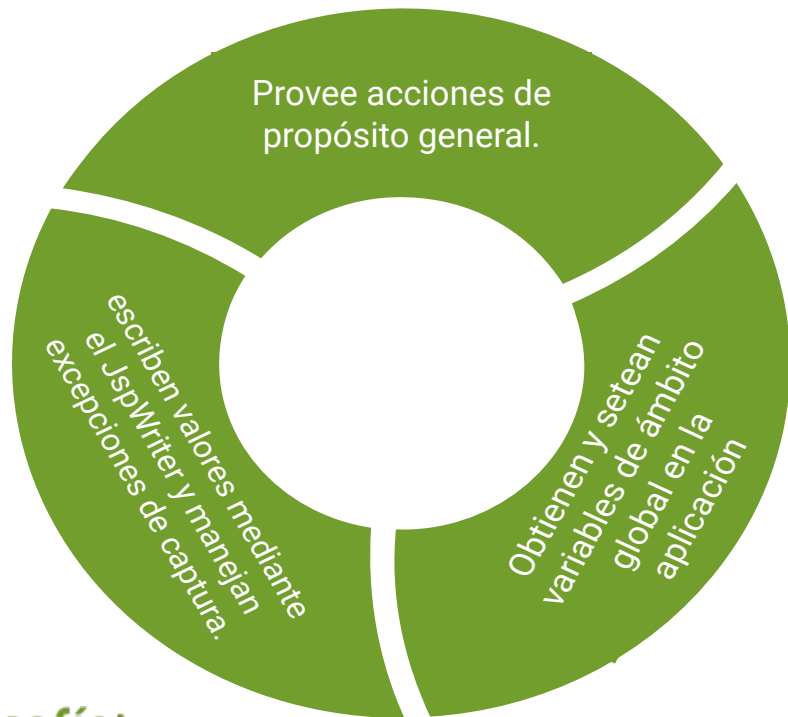
Las librerías de tags jstl se dividen en	
Core	http://java.sun.com/jsp/jstl/core
XML	http://java.sun.com/jsp/jstl/xml
SQL	http://java.sun.com/jsp/jstl/sql
Internacionalización	http://java.sun.com/jsp/jstl/fmt
Functions	http://java.sun.com/jsp/jstl/functions

Para utilizar estas funcionalidades es necesario añadir a la página jsp la directiva correspondiente al tag que se desea utilizar.

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

Core JSTL

funcionalidades que componen el TagLib Core



Soporte a variables	tags: remove,set.
Control de flujo	tags: choose,when, otherwise,forEach, forTokens, if.
Administración de url	tags: import,param,redirect,url.
Misceláneos	tags: catch, out.

Código utilizando el prefijo

```
< c:out value="Hola, estoy usando  
jstl" />
```

Código de archivo vista.jsp

```
<%@ page language="java"  
contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
    <%@ taglib  
uri="http://java.sun.com/jsp/jstl/core"  
prefix="c"%>  
    <!DOCTYPE html>  
    <html>  
    <head>  
    <meta charset="ISO-8859-1">  
    <title>Uso de forEach con JSTL</title>  
    </head>  
    <body>  
        <c:forEach var="elementos"  
items="${requestScope.listaDeCompras}">  
            <c:out value="${elementos}" />  
        </c:forEach>  
    </body>  
    </html>
```

```
/**
 * Servlet implementation class ListadoGenericoServlet
 */
@WebServlet("/ListadoGenericoServlet")
public class ListadoGenericoServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException
    {
        List<String> listaDeCompras = new ArrayList<String>();
        listaDeCompras.add("Manzanas");
        listaDeCompras.add("Peras");
        listaDeCompras.add("Uvas");
        listaDeCompras.add("Mandarinas");
        listaDeCompras.add("Platanos");
        listaDeCompras.add("Lechugas");
        listaDeCompras.add("Mangos");

        request.setAttribute("listaDeCompras", listaDeCompras);
        RequestDispatcher reenviador =
        request.getRequestDispatcher("vista.jsp");
        reenviador.forward(request, response);
    }
}
```

@ Código de archivo
ListadoGenericoS
ervlet.java

- Otorga las herramientas necesarias para manipular este tipo de documentos.
- Permiten trabajar con el análisis de sintaxis y escritura de documentos xml.

Hay que importarlas mediante los tags:

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


Lista de tags disponibles para el módulo xml

TAG XML	Descripción
x:out	Tag similar a <%= %>, pero para xml.
x:parse	Se utiliza para analizar los datos xml especificados en el cuerpo de la etiqueta o en un atributo.
x:choose	Tag condicional.
x:when	Subtag en caso de que la primera condición no se cumpla.
x:otherwise	Sub subtag en caso de que when no se cumpla.
x:if	Tag condicional.
x:transform	Transforma xml a xls.
x:params	Se utiliza junto con la etiqueta de transformación para configurar el parámetro en la hoja de estilo XSLT.
x:set	Usado para setear valores.

Provee la capacidad para interactuar con bases de datos.
Incluye el manejo de dataSources, querys, updates y transacciones.

Crear un proyecto Dynamic Web Project de nombre Patrones_Diseño_Sesion1_ejemplo009 y un jsp de nombre index.jsp.

```
<sql:query var="listaLibros" dataSource="${datasource}">
SELECT * FROM libros WHERE title = 'JSTL' ORDER BY autor
</sql:query>
<table>
<c:forEach var="libro" items="${listaLibros.row}">
<tr>
<td><c:out value="${libro.titulo}" />
</td><td><c:out value="${libro.autor}" />
</td></tr></c:forEach></table>
```

Internacionalización

- Prefijo fmt.
 - Da funcionalidad de formateo de mensajes, de números y fechas e internacionalización.

Importar las librerías:

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

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"
%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"
%>
<%@ page isELIgnored="false" %>

<fmt:setLocale value="zh"/>
<fmt:setBundle basename="messages" />

<html>
<head>
    <title>Distintos Idiomas</title>
</head>
<body>
    <h2>
        <fmt:message key="label.welcome" />
    </h2>
</body>
</html>
```

**Agregamos la
instrucción para
llamar a alguno de
estos archivos
.properties**

**/* Uso de formularios para
captura de información */**

Formulario JSP a formulario JSP

En la tecnología JSP es posible transferir información desde dos páginas java server page.

Ejemplos de los mecanismos para empezar la comunicación.

- **Composición:**
 - Un formulario está compuesto por componentes HTML tienen como misión capturar información ingresada por el usuario.

Input Text

Input Text de password

Elemento más utilizado en los formularios.

Los navegadores ocultan el texto utilizando asteriscos o círculos.

Nombre

Contraseña

CheckBox

Radio Button

Son controles de formulario que permiten al usuario seleccionar y deseleccionar opciones individualmente.

Puestos de trabajo buscados

- ☐ Dirección
- ☐ Técnico
- ☐ Empleado

Solamente se puede escoger una opción entre las distintas opciones relacionadas que se le presentan.

☒ RadioButton1

☐ RadioButton2

☐ RadioButton3

Boton de envio de formulario

Botón para enviar al servidor los datos introducidos por el usuario.

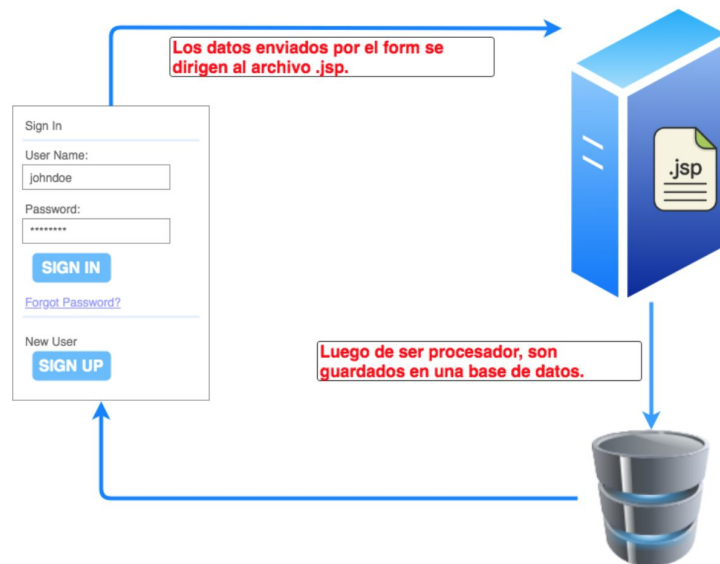


Estructura de un formulario:

```
<form action="proceso.jsp"
method="POST"></form>
```

La etiqueta comienza con la palabra form la cual declara el inicio de un formulario en lenguaje HTML (No olvidar que una página JSP es también una página con html).

Flujo entre formulario y servidor.



Comunicación de JSP a JSP

Se crearán dos archivos Java Server Page, uno encargado de recopilar los datos que ingrese el usuario y otro que los capture y despliegue en la página.

Formulario Contacto.jsp.



A diagram of a web form titled 'Formulario Contacto.jsp'. The form is enclosed in a light gray rounded rectangle. It contains two input fields: a single-line text box labeled 'Usuario' and a multi-line text area labeled 'Comentario'. Below the text area is a blue button with the text 'Enviar' in white. The text area has a small icon in its bottom right corner, indicating it can be expanded.

Formulario Contacto.jsp con datos ingresados

Formulario validación.jsp con datos rescatados desde el request

Usuario

Comentario

Recepcion de datos desde Contacto.jsp

Usuario

Comentario

[Volver](#)

- Los servlets pueden acceder a los valores enviados mediante los request al igual que las páginas JSP.
- Además de la variable request existen un conjunto de ellas las cuales son:

Request	Indica la acción que se desea realizar para un recurso determinado.
Response	Respuesta generada por el servidor.
out	Flujo de salida del cuerpo de la respuesta HTTP.
session	Permite mantener una sesión para cada uno de los usuarios conectados.
config	Información relativa a la configuración del servlet generado.
exception	Excepción que se ha producido en una página JSP.

El objeto request

- Es quien representa este mensaje que se envía desde el navegador y puede contener la información de los valores enviados desde el cliente.
- Cuando un navegador envía información al servidor lo hace a través del objeto request.

Transferencia de request mediante la url	
Url encoded parameters	Los valores que viajan desde el navegador al servidor van incrustados en la misma url en formato de parámetros explícitamente visibles.
Form encoded parameters	Enviados como resultado de la ejecución del envío de un formulario mediante un submit.

- <http://www.miservidor.com/carpetaproyecto/proyecto?var1=nombre1&var2=nombre2>
- Utilización del objeto implícito request.

<%

```
String nombre = request.getParameter("user");  
String comentario = request.getParameter("comentario");
```

%>

- No se muestran los valores enviados.

localhost:8080/ComunicacionEntreJsp/validacion.jsp

- Si cambiamos el método de envío a GET, veremos como los parámetros son expuestos.

localhost:8080/ComunicacionEntreJsp/validacion.jsp?user=visible&comentario=totalmente

/* Modelo MVC, Arquitectura en Capas */

Conocer el patrón de diseño MVC

División de responsabilidades y funcionalidades.

Considera tres roles:

El modelo

Objeto que representa información sobre el dominio.

La vista

Visualización del modelo en la interfaz de usuario

El controlador

Tomar la entrada del usuario, manipular el modelo y hace que la vista se actualice.

Diferencias entre el modelo de capas y MVC

Arquitectura en capas	Desacopla las responsabilidades de un sistema en capas lógicas, identificando 1 capa o N.
Arquitectura MVC	Separar y asigna una responsabilidad única a cada capa permitiendo que la comunicación entre ellas no sea unidireccional.

El modelo de una capa se basa en contener la lógica de negocio y la lógica de presentación en un solo proyecto.

Componentes del modelo MVC

Son divididos en tres categorías:

Modelo

Contiene las clases e instancias de clases que representan el dominio del sistema u aplicación.

Vista

Se ubica todo lo que se mostrará al usuario.

Controlador

Conoce cómo conversar con el front end y conoce todos los métodos del modelo.

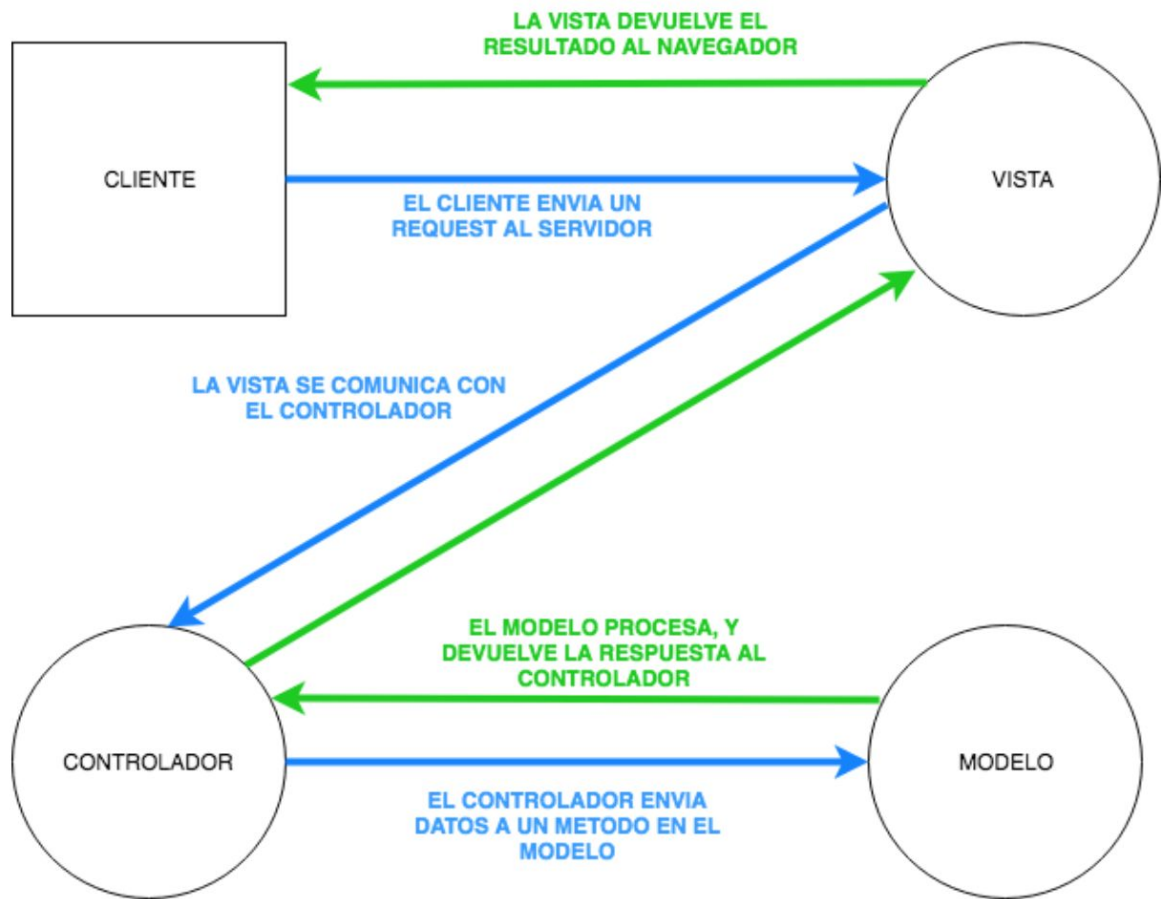


Diagrama de
alto nivel
arquitectura
MVC

Arquitectura en capas

Capa de datos

Trabaja con los datos del sistema, comunicando directamente con algún sistema de base de datos.

Capa de acceso a datos

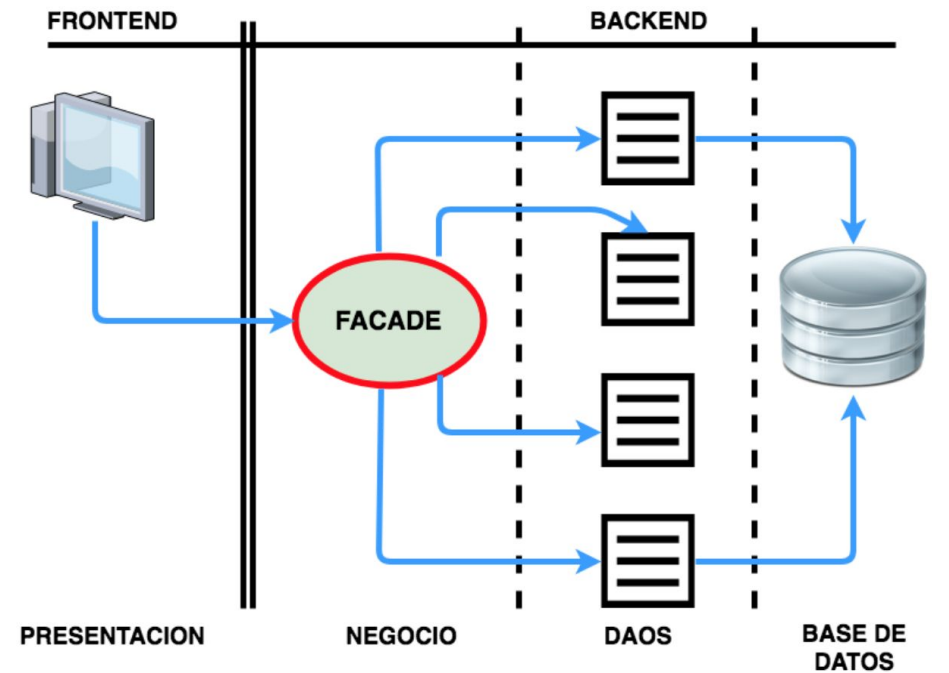
Entidades de negocios en objetos perfectamente reproducibles en lenguaje java.

Capa de aplicación

Se programa la lógica de negocio de la aplicación.

Capa web

Capa de presentación.



Arquitectura en capas

Definición de acuerdo a su responsabilidad

01	Capa de datos	Proveen los mecanismos de persistencia.
02	La capa de acceso a datos DAO	Interactúa directamente con la capa de datos y se encarga de acceder a ella.
03	DTO	Proveer los objetos que son un fiel reflejo de las entidades de base de datos.
04	La capa de negocios	Procesar los datos y de otorgar la lógica de negocios al sistema.

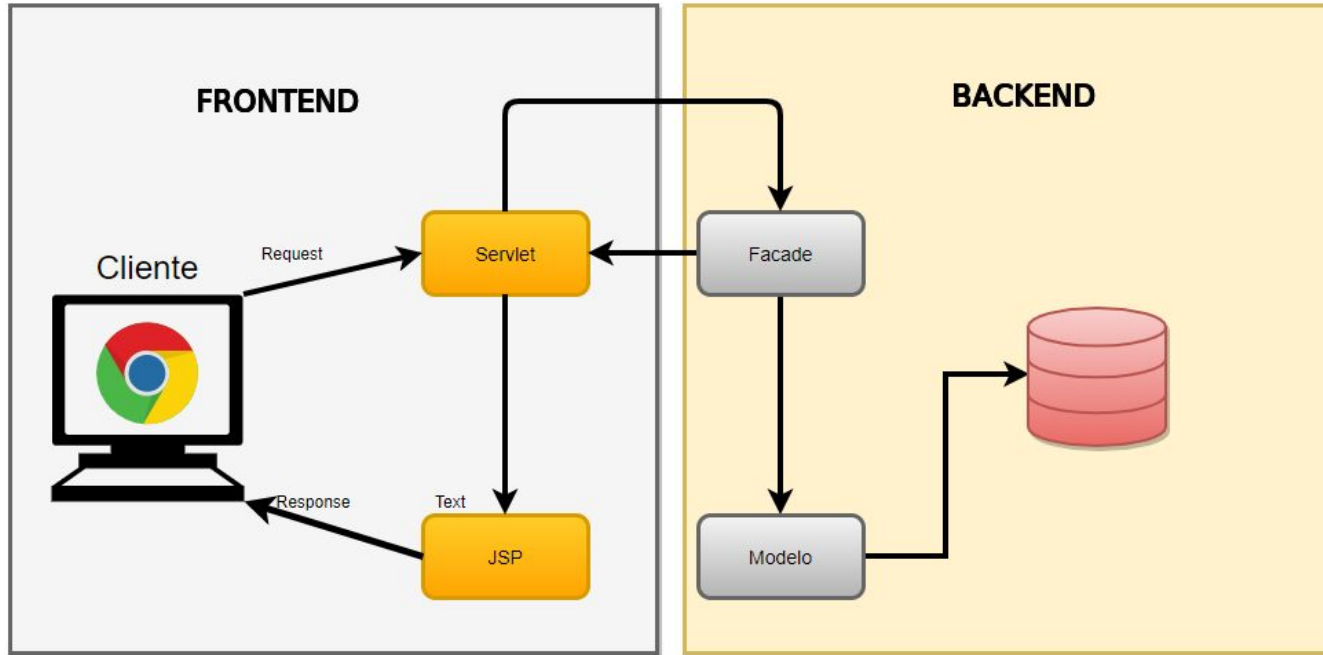


Facade

El backend y el frontend de una aplicación

- Front end:
 - Capas que tienen interacción directa con el usuario final.
 - Capas que le dan soporte a las primeras.
- backend:
 - Capas que están entre la capa de negocio y la capa de datos.

Arquitectura backend y frontend



/* Aplicación en capas */

Análisis de la solución

Al momento de plasmar los requerimientos de un sistema se utiliza un artefacto llamado casos de uso.

- **Pasos**

- El usuario solicita una página de inscripción.
- El sistema la arma y se la entrega al usuario.
- El usuario selecciona un curso y sus datos para que al momento de confirmar, se genere una transacción a la base de datos.

Implementación del sistema

Crear un proyecto Dynamic Web Project de nombre
Patrones_Diseño_MantenedorCursos.

Estructura de carpetas.

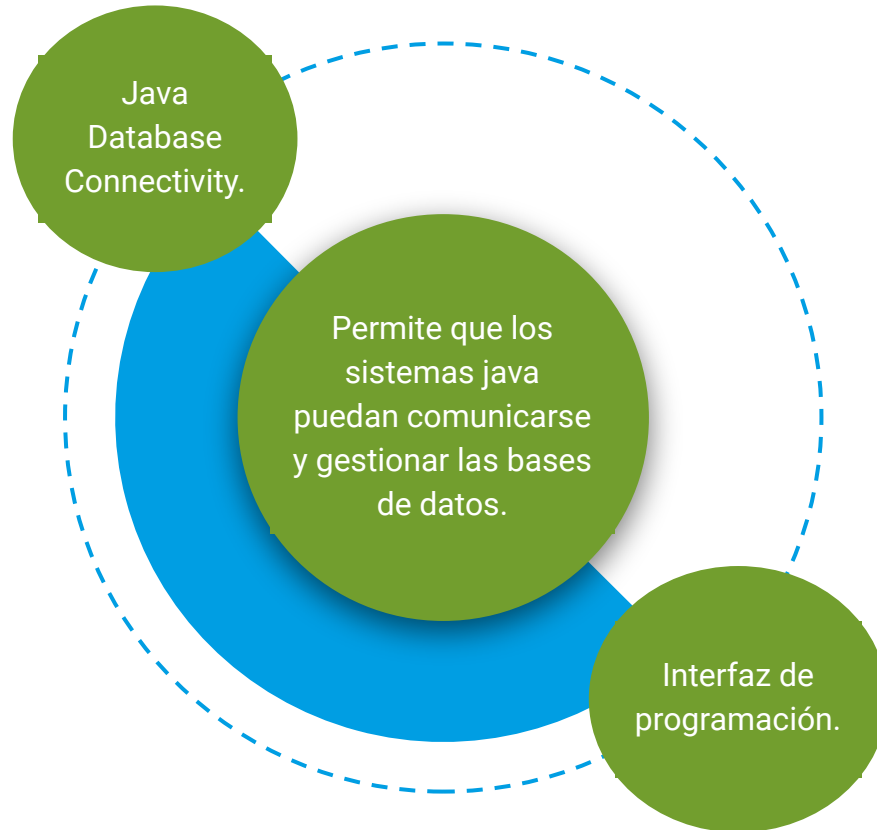


- Describen a las clases java puras con una estructura estándar.
- No dependen de ningún framework.

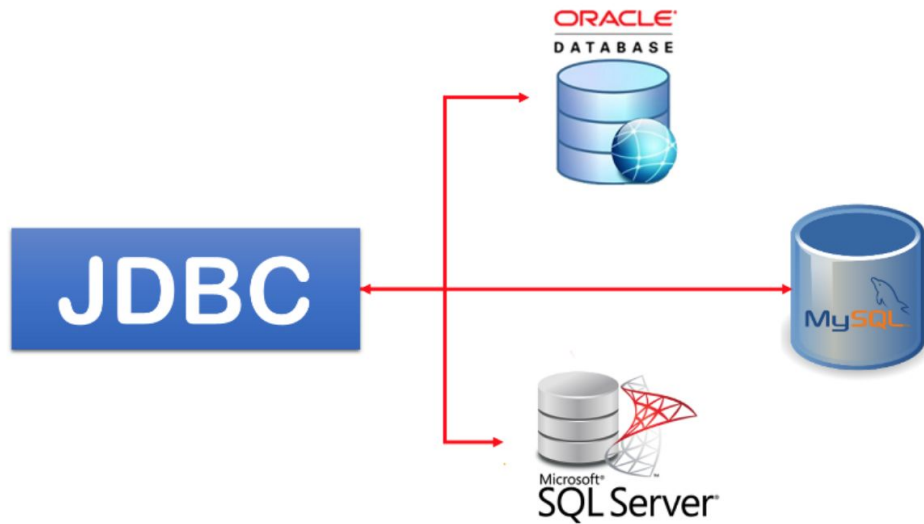
DAO

- Interlocutor entre el sistema y la base de datos.

API JDBC



Con jdbc es posible utilizar cualquier motor de BD



Pasos para generar una conexión

1. Descargar OJDBC6.
2. Importar .jar al proyecto.
3. Registrar el driver jdbc.
4. Abrir la conexión
5. Establecer la conexión recibida.

getConnection

```
Connection conn = DriverManager.getConnection  
("jdbc:oracle:thin:@localhost:1521:xe", "Empresa_DB",  
"empresa");
```

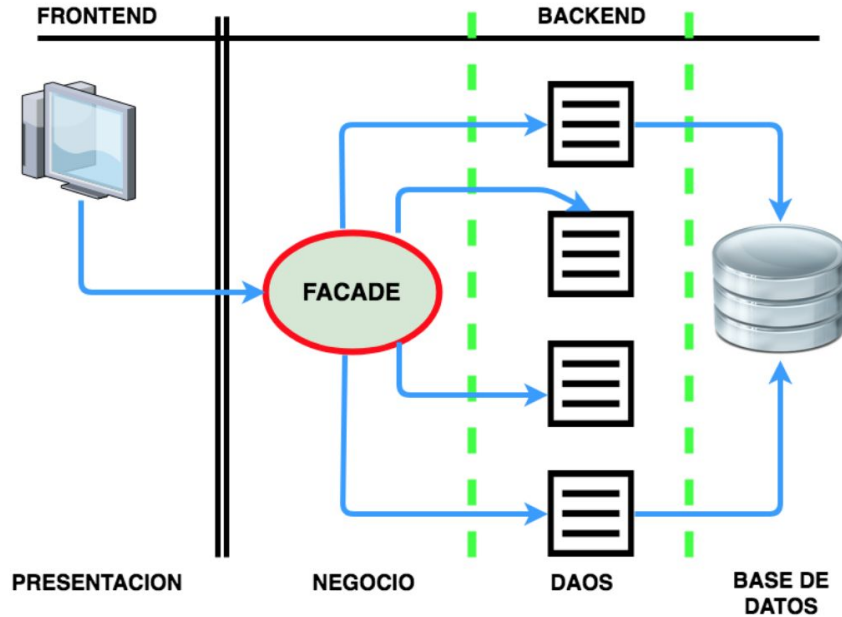
/* Capa de acceso a datos y objetos DAO */

DAO

- Data Access Object.
- objeto de acceso a datos.
- Acceso directo al motor de base de datos.

DTO

- Data Transfer Object.
- Transporta los datos desde el motor de datos hasta la capa lógica de negocio.



Capa cubierta
con clases
DAOS y DTO

```
package com.desafiolatam.facade;
```

```
import java.sql.SQLException;
```

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

```
import com.desafiolatam.daos.CursorDao;
```

```
import com.desafiolatam.daos.FormaDePagoDAO;
```

```
import com.desafiolatam.daos.InscripcionDAO;
```

```
import com.desafiolatam.entidades.CursorDTO;
```

```
import com.desafiolatam.entidades.FormaDePagoDTO;
```

```
import com.desafiolatam.entidades.InscripcionDTO;
```

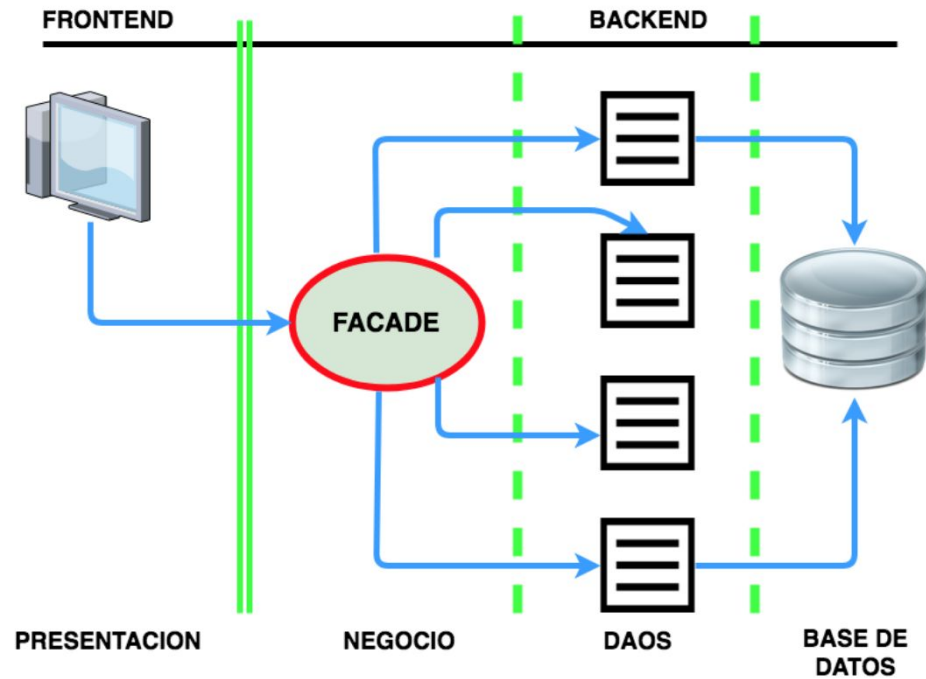
```
public class Facade {
```

```
    public int registrarInscripcion(InscripcionDTO dto) throws  
SQLException, ClassNotFoundException {  
        InscripcionDAO dao = new InscripcionDAO();  
        return dao.insertarInscripcion(dto);  
    }
```

Clase Facade

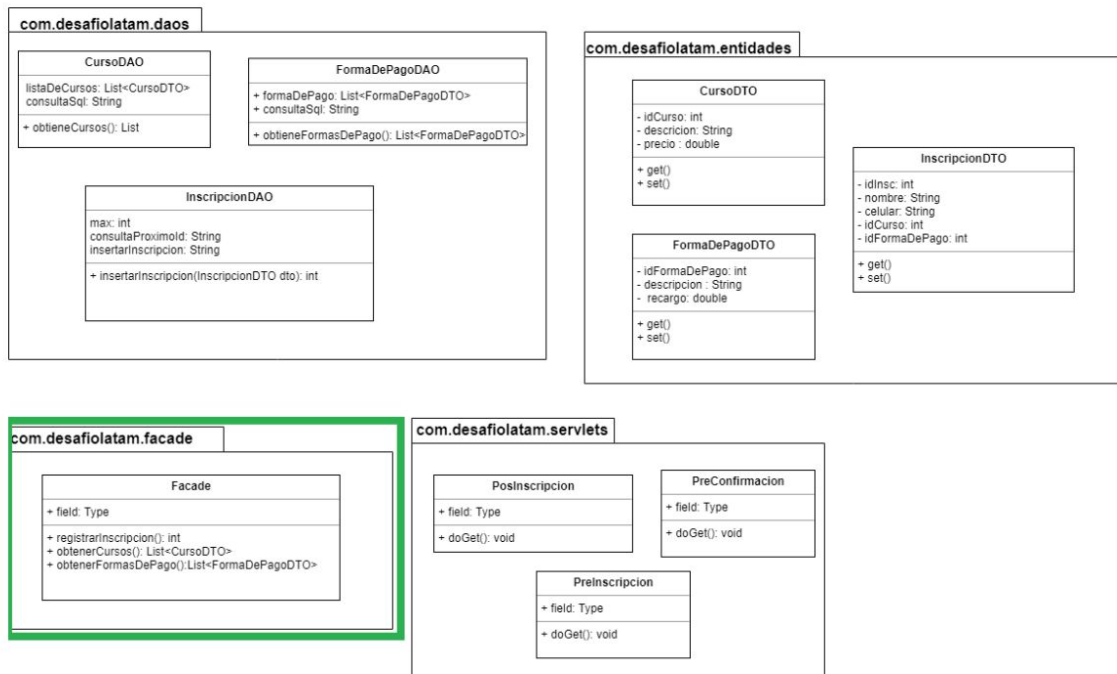
```
    public List<CursoDTO> obtenerCursos() throws  
SQLException, ClassNotFoundException{  
        CursoDao dao = new CursoDao();  
        return dao.obtieneCursos();  
    }  
  
    public List<FormaDePagoDTO> obtenerFormasDePago()  
throws SQLException, ClassNotFoundException{  
        FormaDePagoDAO dao = new FormaDePagoDAO();  
        return dao.obtieneFormasDePago();  
    }  
}
```

Clase Facade



Gracias al
facade
podemos
comunicarnos
con el backend

Diagrama de clases - Facade



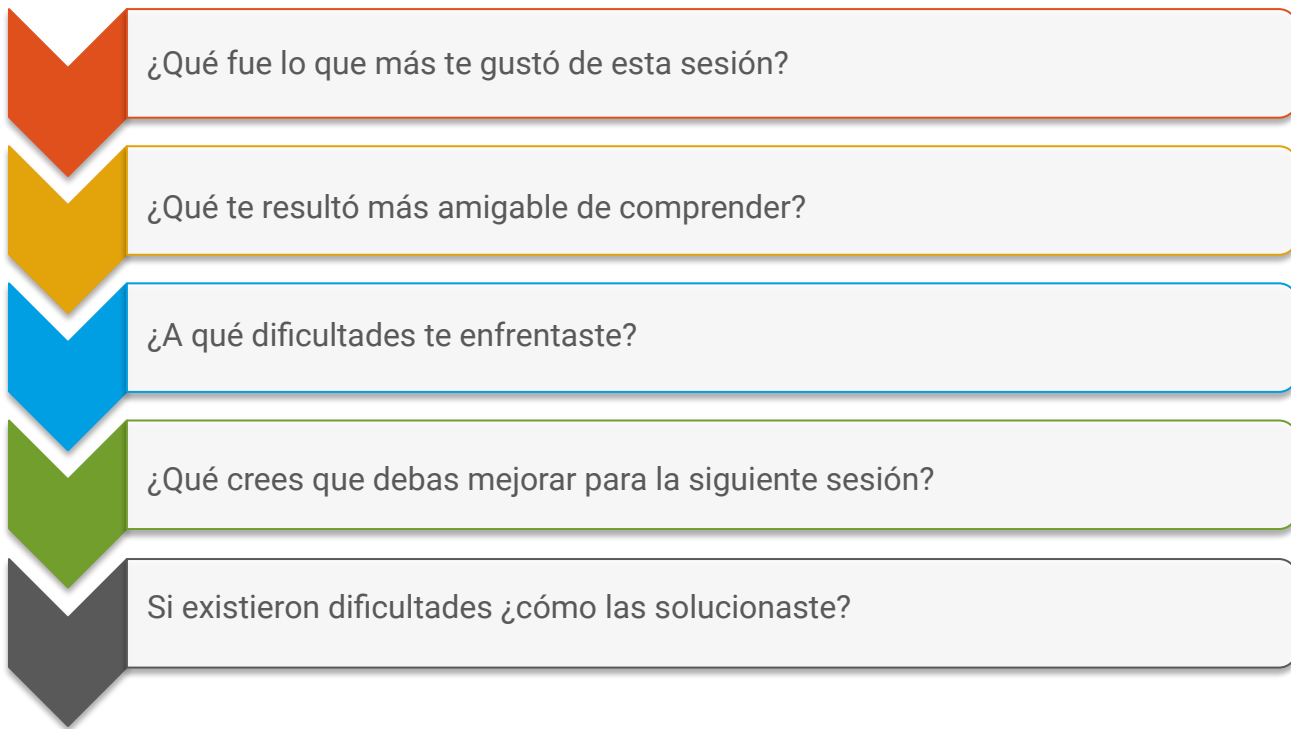


Cierre

{desafío}
latam_



15 minutos



¿Qué fue lo que más te gustó de esta sesión?

¿Qué te resultó más amigable de comprender?

¿A qué dificultades te enfrentaste?

¿Qué crees que debas mejorar para la siguiente sesión?

Si existieron dificultades ¿cómo las solucionaste?



*Academia de
talentos digitales*

www.desafiolatam.com



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam