

## Codificación MVC de la capa de presentación

<b>Codificación MVC de la capa de presentación</b>	<b>1</b>
¿Qué aprenderás?	2
Introducción	2
Construcción Inscripción Servlet	3
Construcción Inscripción.jsp	5
Servlet de post-procesamiento	7
Servlet Pre-Confirmación	9



**¡Comencemos!**

## ¿Qué aprenderás?

- Construir Inscripción Servlet e Inscripción JSP.
- Construir Servlet Post-Procesamiento y Servlet Pre-Confirmación.

## Introducción

De cara al usuario, esta capa es el “sistema” completo, ya que es en donde interactúa y obtiene su información. En este sector de la aplicación es donde interactúan las páginas JSP, los servlets y toda su funcionalidad.

En este nuevo esquema, el servlet (desde ahora será conocido como el controlador) es el encargado de llevar y traer información entre las páginas JSP (llamadas Vistas) y el backend (llamado modelo).

En esta arquitectura existirán dos servlets por cada página JSP, un servlet de preprocesamiento que primero se encargará de obtener los datos desde el backend para suministrarlos a la página y un servlet de post procesamiento que procesa la acción que realice el usuario sobre la página web.

¿Por qué es tan necesaria esta capa? La respuesta está en el usuario mismo, el cual quiere que su sistema sea fluido, rápido, estéticamente atractivo y con capacidades responsivas por lo que siguiendo la estructuración en capas podemos desacoplar esta capa de la lógica permitiendo, por ejemplo, cambiar el framework de front end que se esté utilizando. Nada impide que desconectando las llamadas a las capas de negocio se pueda migrar todo el front-end con jsp a por ejemplo una estructura que solamente utilice html con jquery, o nodejs con comunicación mediante rest a los servlets, o siguiendo en la línea java implementar frameworks más avanzados basados en jsp como Primefaces o Icefaces. El bajo acoplamiento es una gran ventaja que permite facilitar estas tareas de migración de tecnologías.

## Construcción Inscripción Servlet

Esta página le permite al usuario ingresar su nombre y un número de celular y además muestra dos listas desplegables, una que despliega los cursos disponibles y otra que muestra los distintos tipos de pagos. Cabe mencionar que para mostrar tales valores en las listas primero se deben rescatar los valores desde el backend y una base de datos.

En este ejemplo, si bien tenemos programadas las clases *daos* que se comunican con una base de datos, aún no contamos con una, así que se simularán los datos mediante unas simples estructuras java.

Una de las ideas de estas arquitecturas es justamente esa, como todo está separado por capas, da igual el motor de base de datos que utilicemos, ya que fácilmente podemos integrar nuestro sistema a algún otro sin mucho esfuerzo.

En el siguiente diagrama podemos ver el mapa completo de la construcción.

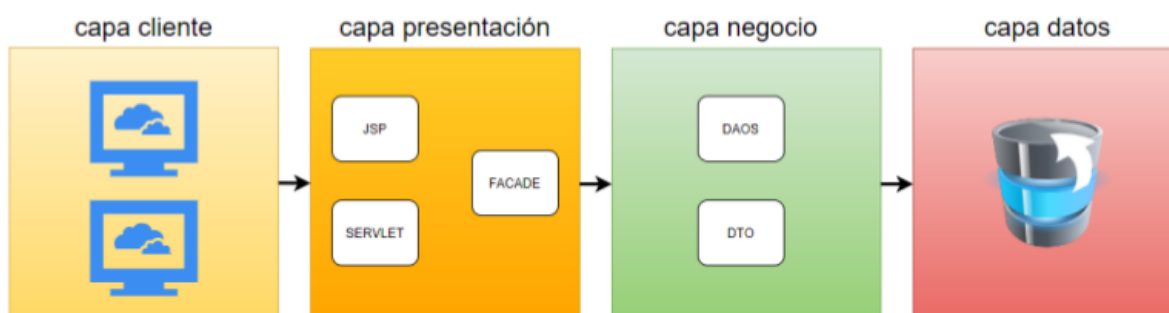


Imagen 1. Presentación del entorno.

Fuente: Desafío Latam.

### servlets de preinscripción

```
package com.desafiolatam.servlets;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.desafiolatam.entidadesCursoDTO;
import com.desafiolatam.entidadesFormaDePagoDTO;
```

```
import com.desafiolatam.facade.Facade;

@WebServlet("/preInscripcion")
public class PreInscripcion extends HttpServlet {
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        //obtenemos el facade
        Facade facade = new Facade();
        try {
            //obtenemos las listas
            List<CursoDTO> listaCursos = facade.obtenerCursos();
            List<FormaDePagoDTO> listaFormasPago =
        facade.obtenerFormasDePago();

            //guardo las listas al request
            request.setAttribute("cursos", listaCursos);
            request.setAttribute("formasPago", listaFormasPago);

            //mandamos el request a la pagina jsp
            request.getRequestDispatcher("inscripcion.jsp").forward(request,
            response);
        } catch (SQLException | ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

El servlet de preinscripción se encarga de la comunicación con la clase FACADE el cual proporciona los métodos para obtener la lista de cursos y la lista de formas de pago. El punto importante de este servlet, además de la recopilación de datos con el facade, está en la sentencia donde se utiliza el objeto `request.getRequestDispatcher`, que se encarga de redireccionar a la página `jsp inscripcion.jsp`.

## Construcción Inscripción.jsp

Gracias al servlet preinscripción es posible redireccionar y enviar los valores mediante el objeto request a cualquier otro recurso web, en este caso a la página jsp inscripción, que recibirá los valores que se rescataron desde la base de datos.

### JSP de inscripción

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="com.desafiolatam.entidades.CursoDTO" %>
<%@ page import="com.desafiolatam.entidades.FormaDePagoDTO" %>
<%@ page import="java.util.List"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Formulario Inscripción</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.
css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"><
/script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js
"></script>
</head>
<%
List<CursoDTO> cursos;
List<FormaDePagoDTO> formasPago;
// capturamos informacion que viene desde el request
cursos = (List)request.getAttribute("cursos");
formasPago = (List)request.getAttribute("formasPago");
%>
<body>

<nav class="navbar navbar-default">
<div class="container-fluid">
<div class="navbar-header">
```

```
<a class="navbar-brand" href="#">Mantenedor De Cursos</a>
</div>
<ul class="nav navbar-nav">
  <li class="active"><a
href="preListarInscripciones">Home</a></li>
  <li><a href="preInscripcion">Inscribir Cursos</a></li>
</ul>
</div>
</nav>

<div class="container">
  <form action="posInscripcion">
    <div class="form-group">
      <label for="nombre">Nombre:</label>
      <input type="text" class="form-control" id="nombre"
name="nombre">
    </div>
    <div class="form-group">
      <label for="telefono">Telefono:</label>
      <input type="text" class="form-control" id="telefono"
name="telefono">
    </div>
    <div class="form-group">
      <label for="cursos">Cursos:</label>
      <select name="idCurso">
        <%
          for(CursoDTO dto: cursos){
            %>
            <option value="<%=dto.getIdCurso()%>">
              <%=dto.getDescripcion()%>
            </option>
            <%
              }
            %>
          %>
        </select><br>
      </div>
      <div class="form-group">
        <label for="formasPago">Formas de Pago:</label>
        <select name="idFormaPago">
          <%
            for(FormaDePagoDTO dto : formasPago ){
              %>
              <option
```

```
value="<%=dto.getIdFormaDePago()%>">
                                <%=dto.getDescripcion()%>
                                </option>
                                <%
                                }
                                %>
                                </select><br>
        </div>

        <button type="submit" class="btn
btn-default">Enviar</button>
    </form>
    <div class="container">
</body>
</html>
```

## Servlet de post-procesamiento

El servlet de preprocesamiento es el encargado de procesar la acción que ejecuta el usuario. En este caso la acción es completar el formulario y presionar el botón de “Enviar”.

### Servlet de preconfirmación

```
package com.desafiolatam.servlets;

import java.io.IOException;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.desafiolatam.entidades.InscripcionDTO;
import com.desafiolatam.facade.Facade;
@WebServlet("/posInscripcion")
public class PosInscripcion extends HttpServlet
{
    protected void doGet(HttpServletRequest request
                        ,HttpServletResponse response)
                        throws ServletException, IOException
```

```
{
    // obtengo los datos del formulario
    String nombre = request.getParameter("nombre");
    String celular = request.getParameter("telefono");

    // recordemos que los parametros siempre son cadenas
    String idCurso = request.getParameter("idCurso");
    String idFormaDePago = request.getParameter("idFormaPago");

    // instancio el DTO y le asigno los datos
    InscripcionDTO dto = new InscripcionDTO();
    dto.setNombre(nombre);
    dto.setCelular(celular);
    dto.setIdCurso(idCurso);
    dto.setIdFormaDePago(idFormaDePago);

    // invoco al facade para procesar la inscripcion
    Facade facade = new Facade();
    int idInsc = 0;
    try {
        idInsc = facade.registrarInscripcion(dto);
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    // el resultado lo adjunto como atributo en el request
    request.setAttribute("idInsc", idInsc);

    // redirecciono el control hacia la siguiente vista,
    // es decir: hacia su servlet de pre-procesamiento

    request.getRequestDispatcher("/preConfirmacion").forward(request,
    response);
}
```



## Servlet Pre-Confirmación

El servlet de preConfirmación es muy fácil de entender, ya que solamente redirecciona al jsp confirmación.jsp y le envía el request y el response para desplegar el resultado de la ejecución del registro.

### Página jsp confirmación

```
<%
    int idInsc = (Integer)request.getAttribute("idInsc");
%>

<html>
<body>
    <h1>Solicitud Generada Correctamente</h1>
    Su codigo de inscripción es: <%=idInsc%>
</body>
</html>
```

Con los servlets estamos finalizando el ejemplo de sistema en capas. El diagrama de clases ya está completamente cubierto y se refleja la capa de servlets.

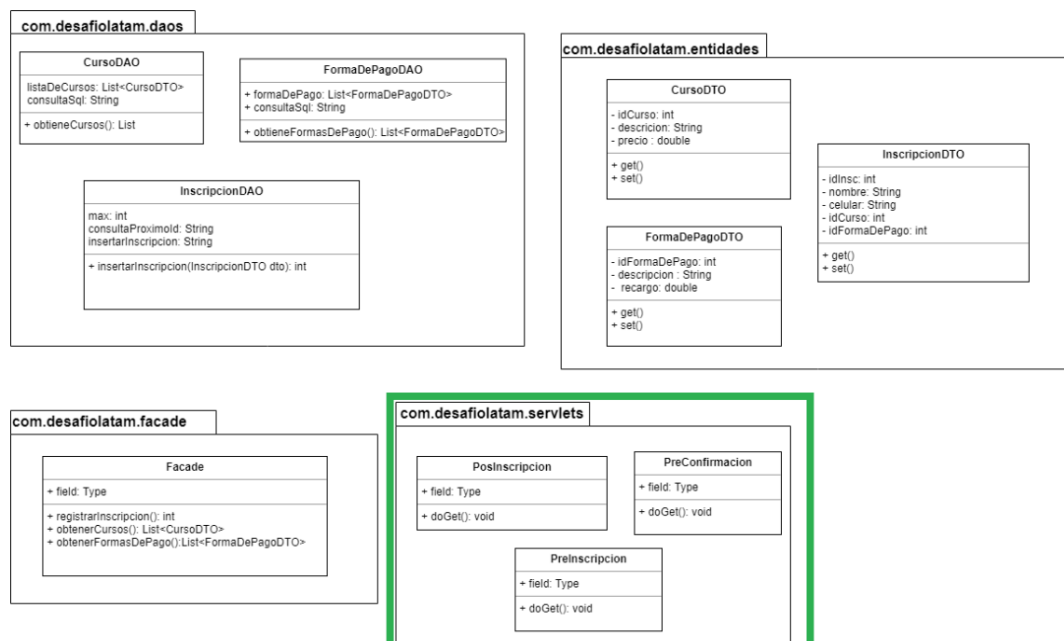


Imagen 2. Diagrama de clases: Servlets.

Fuente: Desafío Latam.

Para ver funcionando la aplicación debemos ingresar a la url:

- <http://localhost:8081/MantenedorCursosDesafioLatam/preInscripcion>

La cual desplegará el formulario. Si bien es un formulario estéticamente feo lo importante aquí es el despliegue de los combobox con los cursos y las formas de pago, los cuales son rescatados desde la base de datos.

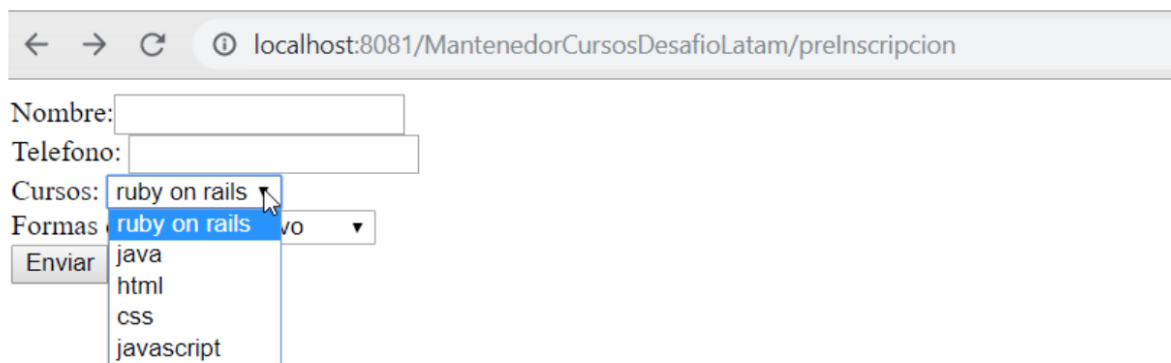


Imagen 3. Despliegue de formulario cursos.  
Fuente: Desafío Latam.

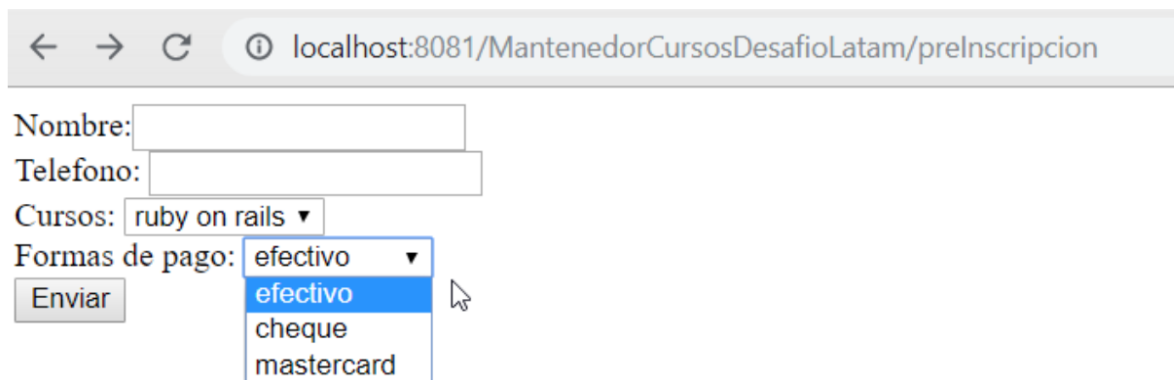


Imagen 4. Despliegue de formulario de formas de pago.  
Fuente: Desafío Latam.

Cuando el usuario llene el formulario y presione el botón enviar, el flujo hará su recorrido para al final guardar el registro en la tabla de inscripciones. Al finalizar simplemente se redirige al jsp que muestra solo el resultado del número de id generado.



Imagen 5. Resultado de la solicitud.  
Fuente: Desafío Latam.

Si se generó el id, es porque se guardó en la base de datos el registro específicamente en la tabla registro:

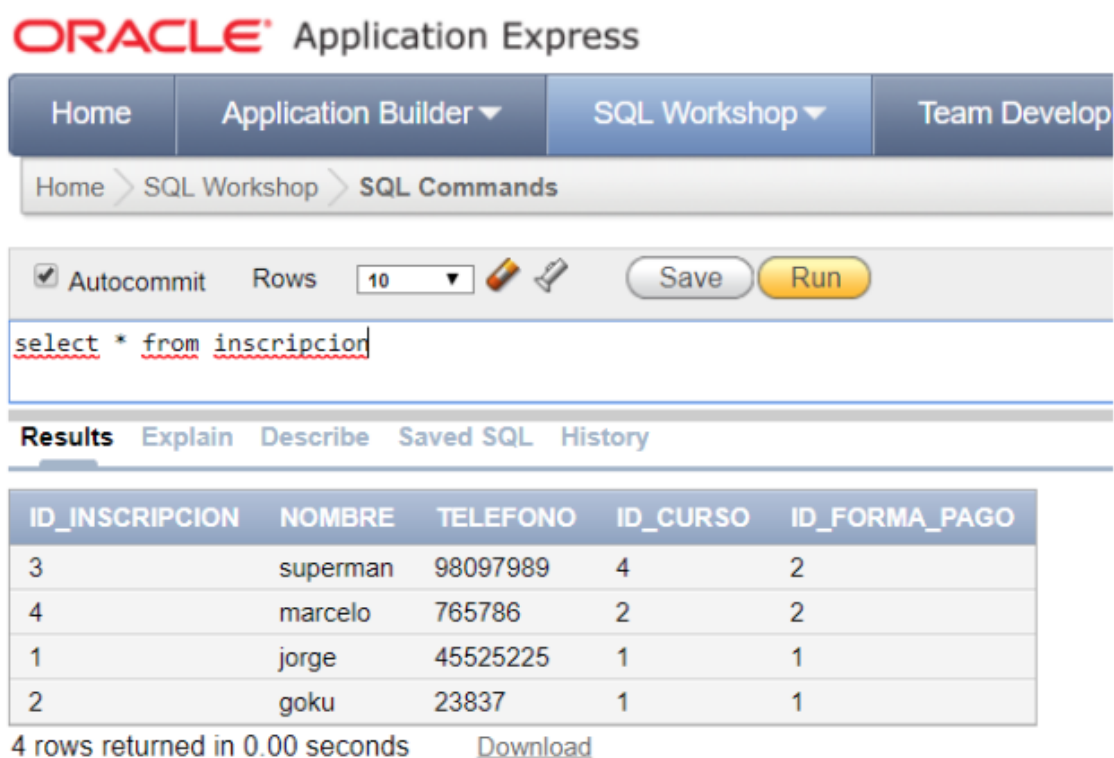


Imagen 6. Revisando la base de datos.  
Fuente: Desafío Latam.