TABLA DE CONTENIDO

[TABLA DE CONTENIDO 1](#_Toc495182288)

[Crear API Pais 2](#_Toc495182289)

[Objetivo: 2](#_Toc495182290)

[Crear Interface País API. 2](#_Toc495182291)

[Crear API Servicio 3](#_Toc495182292)

[Objetivo: 3](#_Toc495182293)

[Crear proyecto para implementar los servicios a exponer 3](#_Toc495182294)

[Consumir los Servicios en el proyecto pais-web. 4](#_Toc495182295)

[Clase para leer archivo con los datos de los países 7](#_Toc495182296)

[Implementar los Servicios a exponer por el API. 10](#_Toc495182297)

[Notas 14](#_Toc495182298)

[bnd.bnd 14](#_Toc495182299)

Crear API Pais

### Objetivo:

Crear una interface (contrato) con los servicios a exponer para listar, agregar, modificar y borrar países.

### Crear Interface País API.

1. Crear proyecto “pais-api”
2. Crear bnd.bnd.

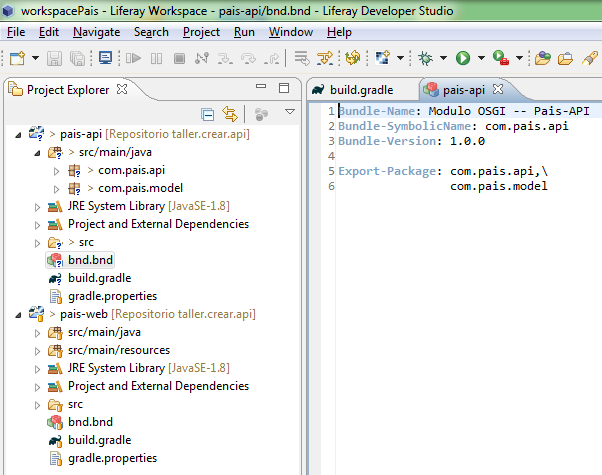
**Bundle-Name**: Modulo OSGI -- Pais-API

**Bundle-SymbolicName**: com.pais.api

**Bundle-Version**: 1.0.0

**Export-Package**: com.pais.api,\

com.pais.model



* Bundle-SymbolicName: Paquete del proyecto (com.pais.api )
* Bundle-version: Versión de la librería a exponer.
* Export-Package: Paquetes a exponer por el proyecto API (En este caso se exponen los servicios a través de una interface y el modelo de datos del negocio)
* com.pais.api-1.0.0

Crear API Servicio

### Objetivo:

Implementar los metodos o servicios expuestos en la interface

* + Servicio para consultar lista de paises originales, sin modificaciones, para ello se utiliza una clase (JSONRead) creada con el objetivo de consultar un archivo JSON que contiene los datos de los paises.
  + Servicio para consulatar la lista de paises modificados (agregados, modificados o borrados) simulada a través un objeto **List<pais>**  que contiene los datos actualizados en cada iteración de los servicios.
  + Servicio para agregar un país a la lista de países originales.
  + Servicio para modificar un país de la lista de países originales.
  + Servicio para eliminar un país de la lista de países originales.

### Crear proyecto para implementar los servicios a exponer

1. Crear proyecto “pais-service”
2. Crear bnd.bnd.

**Bundle-Name**: Modulo OSGI -- Pais-Service

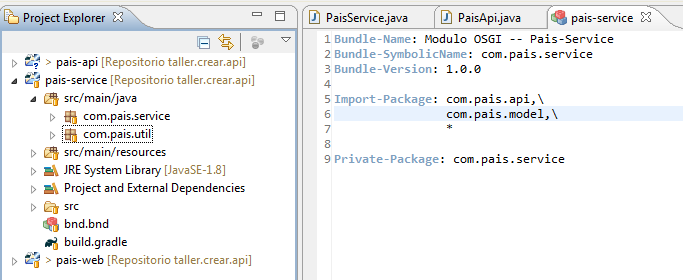
**Bundle-SymbolicName**: com.pais.service

**Bundle-Version**: 1.0.0

**Import-Package**: com.pais.api,\

com.pais.model,\

\*



* Bundle-SymbolicName: Paquete del proyecto (com.pais.service )
* Bundle-version: Versión de la librería a exponer.
* Import-Package: Paquetes a consumir por el proyecto Service (Se implementan los servicios a exponer por el api)
* com.pais.service-1.0.0

### Consumir los Servicios en el proyecto pais-web.

1. Actualizar el archivo bnd.bnd para que pueda consumir la librería de servicios api

**Bundle-Name**: Modulo OSGI -- Pais-WEB

**Bundle-SymbolicName**: com.pais.web

**Bundle-Version**: 1.0.0

**Import-Package**: com.pais.api,\

com.pais.model,\

\*

1. Actualizar el archivo build.gradle para que pueda tenr acceso a la librería de servicios api (pais.api.jar)

buildscript {

dependencies {

classpath group: "com.liferay", name: "com.liferay.gradle.plugins", version: "$gradlePluginsVersion"

}

repositories {

mavenLocal()

maven {

url "https://cdn.lfrs.sl/repository.liferay.com/nexus/content/groups/public"

}

}

}

apply plugin: "com.liferay.plugin"

dependencies {

//Dependencias del proyecto

compileOnly "javax.ws.rs:javax.ws.rs-api:$jaxWsRsVersion"

compile "org.jodd:jodd-json:$jsonVersion"

compileOnly group: "com.liferay.portal", name: "com.liferay.portal.kernel", version: "$kernelVersion"

compileOnly group: "com.liferay.portal", name: "com.liferay.util.taglib", version: "$taglibVersion"

compileOnly group: "javax.portlet", name: "portlet-api", version: "$portletApiVersion"

compileOnly group: "javax.servlet", name: "servlet-api", version: "$servletApiVersion"

compileOnly group: "jstl", name: "jstl", version: "$jstlVersion"

compileOnly group: "org.osgi", name: "org.osgi.compendium", version: "$compendiumVersion"

// Servicios APIs

compileOnly group: "com.pais", name: "com.pais.api", version: "1.0.0"

}

repositories {

mavenLocal()

maven {

url "https://cdn.lfrs.sl/repository.liferay.com/nexus/content/groups/public"

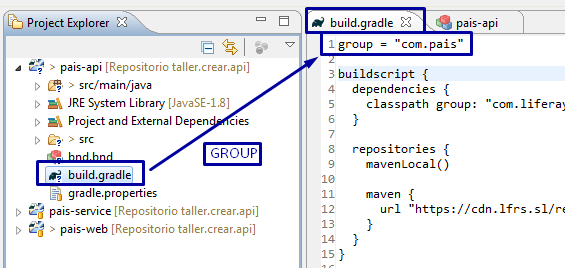
}

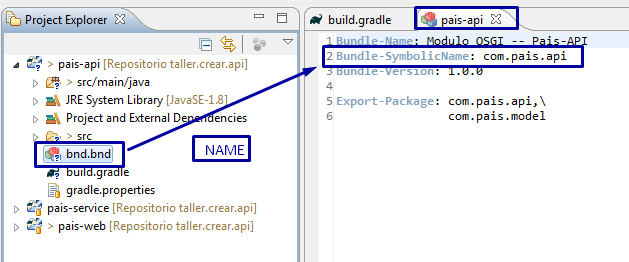
}

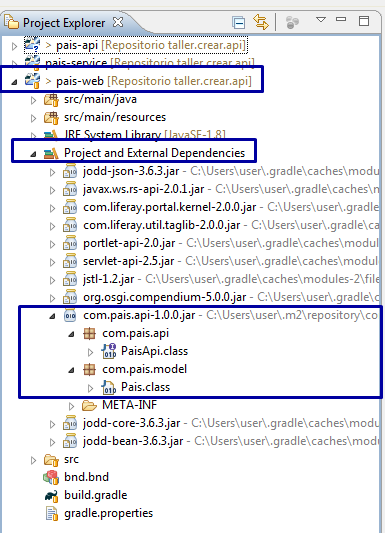
**Nota:**

compileOnly group: "com.pais", name: "com.pais.api", version: "1.0.0"

* grupo: com.pais
* nombre: com.pais.api



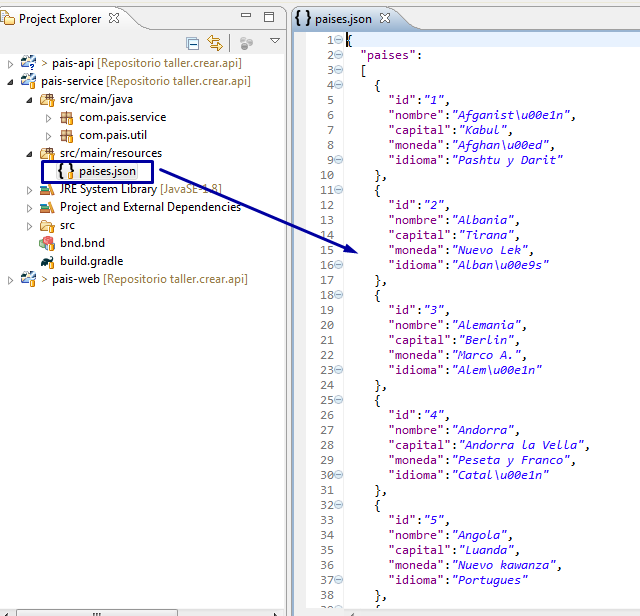




**Gradle refresh del proyecto pais-web:**

### Clase para leer archivo con los datos de los países

1. Crear un archivo países.json con el listado JSON de los paises



1. Crear una clase que permite leer el archivo paises.json, con los datos de los paises. Esta clase simula la consulta de países que podría ser cambiada por una clase que implemente a la interface api y que consulte los datos de paises en una base de datos, a través de un servicio WEB, un listado DUMMY, etc.

/\*\*

\* Clase que permite leer los datos de un archivo json.

\*

\* En este ejercicio se consulta la lista de paises de un archivo paises.json

\*

\* @author lopera

\*/

package com.pais.util;

import com.pais.model.Pais;

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.List;

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

public class JSONRead {

public List<Pais> getDataPaises() {

JSONParser parser = new JSONParser();

try {

List<Pais> paises = new ArrayList<Pais>();

InputStream is = JSONRead.class.getResourceAsStream("/paises.json");

BufferedReader reader = new BufferedReader(new InputStreamReader(is));

StringBuffer sb = new StringBuffer();

String line;

while ((line = reader.readLine()) != null) {

sb.append(line);

}

Object obj = parser.parse(sb.toString());

JSONObject jsonObject = (JSONObject) obj;

JSONArray tags = (JSONArray) jsonObject.get("paises");

Pais pais;

for (int i = 0; i < tags.size(); i++) {

JSONObject jsonObj = (JSONObject) tags.get(i);

pais = new Pais();

pais.setId(jsonObj.get("id").toString());

pais.setNombre(jsonObj.get("nombre").toString());

pais.setCapital(jsonObj.get("capital").toString());

pais.setMoneda(jsonObj.get("moneda").toString());

pais.setIdioma(jsonObj.get("idioma").toString());

paises.add(pais);

}

return paises;

} catch (Exception e) {

System.out.println("getDataPaises - error: " + e.getMessage());

}

return null;

}

public static void main(String[] args) {

JSONRead jsonRead = new JSONRead();

List<Pais> paises = jsonRead.getDataPaises();

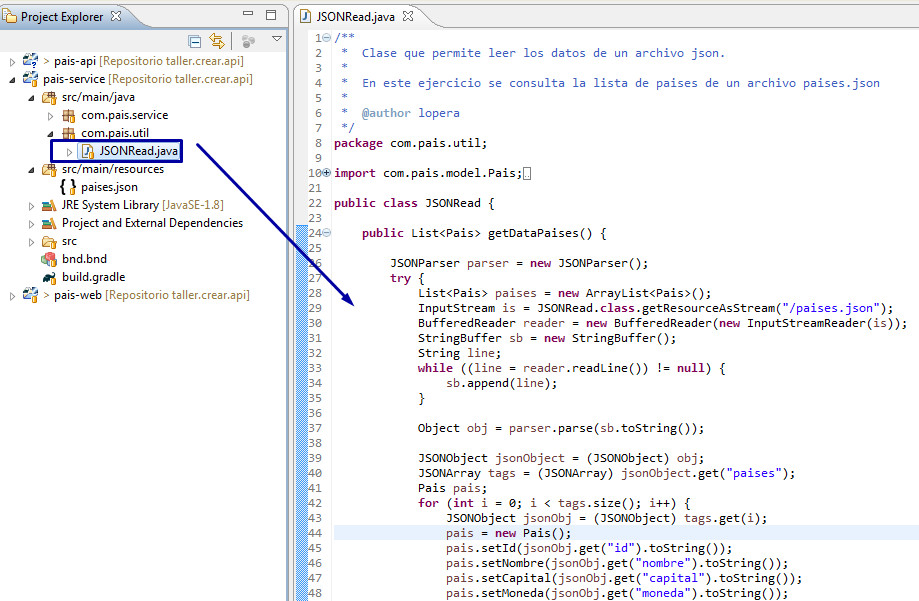
for (Pais pais : paises) {

System.out.println(pais);

}

}

}



### Implementar los Servicios a exponer por el API.

1. Implementar los métodos de cada servicio a exponer

/\*\*

\* Clase que permite a modo de ejercicio:

\* - Consultar lista de paises originales

\* - Agregar un pais a la lista de paises

\* - Modificar un pais a la lista de paises

\* - Borrar un pais d ela lista de la lista de paises.

\* - Reiniciar el listado de paises

\*

\* Nota: la lista de paises se simula a traves de una clase que permite

\* consultar un json con la lista de paises originales.

\*

\* **@author** wlopera

\*/

**package** com.pais.service;

**import** com.pais.api.PaisApi;

**import** com.pais.model.Pais;

**import** com.pais.util.JSONRead;

**import** java.util.Iterator;

**import** java.util.List;

**import** org.osgi.service.component.annotations.Component;

@Component(immediate = **true**, property = {}, service = PaisApi.**class**)

**public** **class** PaisService **implements** PaisApi {

**private** List<Pais> paises;

/\*\*

\* Permite obtener la lista de paises actuales

\*

\* **@author** wlopera

\* **@return** Lista de paises con o sin cambios

\*/

@Override

**public** List<Pais> obtenerlistaPaises() {

**if** (paises == **null** || paises.size() == 0) {

paises = **new** JSONRead().getDataPaises();

}

**return** paises;

}

/\*\*

\* Permite obtener la lista de paises originales

\*

\* **@author** wlopera

\* **@return** Lista de paises sin cambios

\*/

@Override

**public** List<Pais> obtenerListaPaisesOriginales() {

**return** **new** JSONRead().getDataPaises();

}

/\*\*

\* Permite agregar un pais a la lista de paises actuales

\*

\* **@author** wlopera

\*

\* **@param** pais

\* Pais a agregar

\* **@return** Lista de paises actualizada

\*/

@Override

**public** List<Pais> agregarPais(Pais pais) {

pais.setId(String.*valueOf*(paises.size() + 1));

paises.add(pais);

**return** paises;

}

/\*\*

\* Permite modificar un pais de la lista de paises actuales

\*

\* **@author** wlopera

\*

\* **@param** pais

\* Pais a modificar

\* **@return** Lista de paises actualizada

\*/

@Override

**public** List<Pais> modificarPais(Pais pais) {

**for** (Pais paisTemp : paises) {

**if** (pais.getId().equals(paisTemp.getId())) {

paisTemp.setNombre(pais.getNombre());

paisTemp.setCapital(pais.getCapital());

paisTemp.setMoneda(pais.getMoneda());

paisTemp.setIdioma(pais.getIdioma());

**break**;

}

}

**return** paises;

}

/\*\*

\* Permite borrar un pais de la lista de paises actuales

\*

\* **@author** wlopera

\*

\* **@param** pais

\* Pais a borrar

\* **@return** Lista de paises actualizada

\*/

@Override

**public** List<Pais> borrarPais(Pais pais) {

Iterator<Pais> iterator = paises.iterator();

**while** (iterator.hasNext()) {

Pais itPais = iterator.next();

**if** (itPais.getId().equals(pais.getId())) {

iterator.remove();

**break**;

}

}

**return** paises;

}

}

**Notas:**

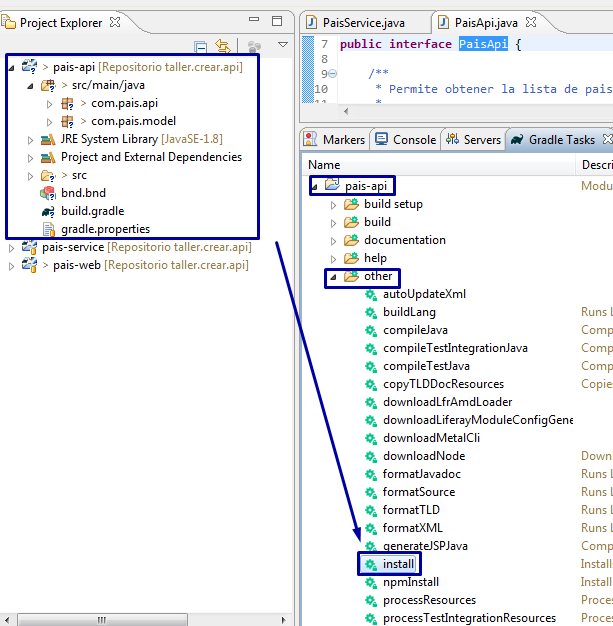
@Component(immediate = **true**, property = {}, service = PaisApi.**class**)

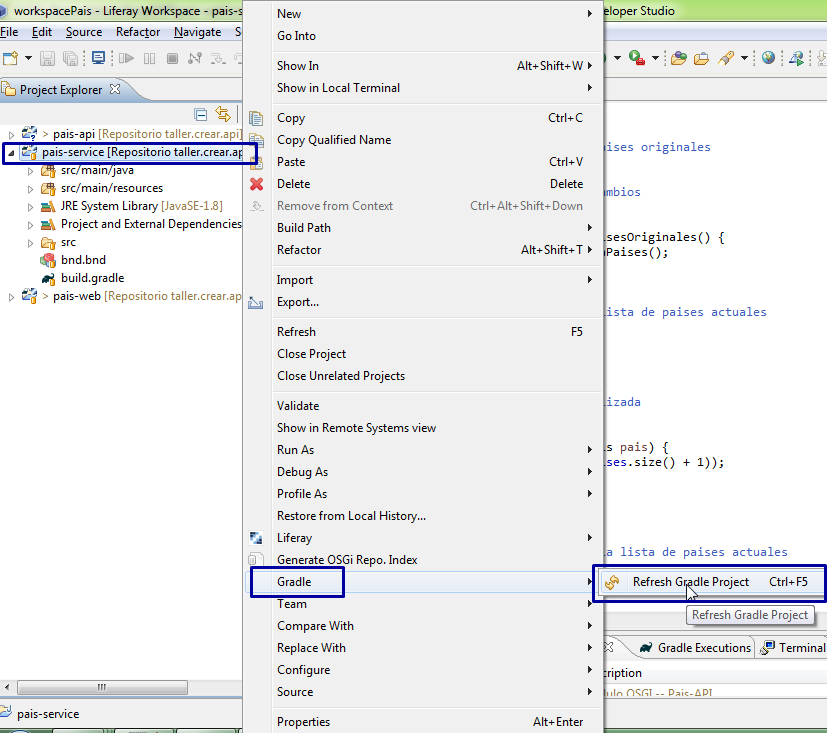
@Component: En el módelo OSGI se implementa la interface PaisAPI (PaisService) y genera el servicio PaisApi.

List<Pais> paises

Objeto creado para simular la modificación de la lista de países originales.

Importante: Cada vez que se modifique el proyecto pais-api se debe compilar (*gradle install*) y se debe refrescar el proyecto pais-service.





### Implementar controlador y servicio del lado del cliente (frontend)

1. Implementar javascript-angular de servicio (pais-service.js): Encargada de consumir los servicios REST expuestos en el controlador de JAVA (PaisController.java)

(**function**(win) {

"use strict";

**var** WS\_PATH = "/o/api/pais";

**var** paisService = **function**($log, serviceInvoker) {

$log.info("Consumiendo servicio paisService");

// Consultar de Paises

**function** getPaises() {

**return** serviceInvoker.invoke("GET", WS\_PATH + "/getPaises");

}

// Reiniciar Paises

**function** reiniciarPaises() {

**return** serviceInvoker.invoke("GET", WS\_PATH + "/reiniciarPaises");

}

// Agregar Pais

**function** agregarPais(pais) {

**return** serviceInvoker.invoke("POST", WS\_PATH + "/agregarPais", pais);

}

// Modificar Pais

**function** modificarPais(pais) {

**return** serviceInvoker.invoke("POST", WS\_PATH + "/modificarPais", pais);

}

// Borrar Pais

**function** borrarPais(pais) {

**return** serviceInvoker.invoke("POST", WS\_PATH + "/borrarPais", pais);

}

**return** {

getPaises: getPaises,

reiniciarPaises: reiniciarPaises,

agregarPais: agregarPais,

modificarPais: modificarPais,

borrarPais: borrarPais

};

};

paisService.$inject = ["$log", "serviceInvoker"];

win.MainApp.Services.service("paisService", paisService);

### }(window));

1. Implementar controlador de angular (pais-controller.js): Expone los servicios javascript a la vista (jsp) y a través de pais-services.js consume los servicios REST.

(**function**(win) {

"use strict";

**var** PaisController = **function**($log, $window, paisService) {

$log.info("Consumir Controller PaisController");

/\*

==============

VALUES

==============

\*/

// VM

**var** vm = **this**;

vm.paises = {};

vm.pais = '';

/\*

==============

METHODS

==============

\*/

// Consultar lista de paises

vm.getPaises = **function**() {

**var** xhr = paisService.getPaises();

xhr.then(**function**(response) {

console.log(response);

vm.paises = response.data;

});

xhr.**catch**(**function**(exception) {

$log.error('error getPaises: '+exception);

});

xhr.**finally**(**function**() {

$log.log("Fin getPaises");

});

};

// Reiniciar lista de paises originales

vm.reiniciarPaises = **function**() {

console.log("##=> Reiniciar Paises: ");

**var** xhr = paisService.reiniciarPaises();

xhr.then(**function**(response) {

vm.paises = response.data;

vm.pais= "";

console.log(response);

});

xhr.**catch**(**function**(exception) {

$log.error('error reiniciando paises: '+exception);

});

}

// Agregar pais a la lista

vm.agregarPais = **function**() {

console.log("##=> Agregar Pais: ", vm.pais);

**var** xhr = paisService.agregarPais(vm.pais);

xhr.then(**function**(response) {

vm.paises = response.data;

vm.pais= "";

console.log(response);

});

xhr.**catch**(**function**(exception) {

$log.error('error agregar pais: '+exception);

});

}

// Modificar pais seleccionado

vm.modificarPais = **function**() {

console.log("##=> Modificar Pais: ", vm.pais);

**var** xhr = paisService.modificarPais(vm.pais);

xhr.then(**function**(response) {

vm.paises = response.data;

vm.pais= "";

console.log(response);

});

xhr.**catch**(**function**(exception) {

$log.error('error modificar pais: '+exception);

});

}

// Borrar pais seleccionado

vm.borrarPais = **function**() {

console.log("##=> Borrar Pais: ", vm.pais);

**var** xhr = paisService.borrarPais(vm.pais);

xhr.then(**function**(response) {

vm.paises = response.data;

vm.pais= "";

console.log(response);

});

xhr.**catch**(**function**(exception) {

$log.error('error borrar pais: '+exception);

});

}

// limpiar los campos de pais selecionado

vm.limpiarCampos = **function**() {

console.log("##=> Limpiar campos Pais");

vm.pais= "";

}

// Llena los campos de pais seleccionado

vm.seleccionarPais = **function**(item) {

vm.pais = {

"id":item.id,

"nombre":item.nombre,

"capital":item.capital,

"moneda":item.moneda,

"idioma":item.idioma

};

}

/\*

==============

SETUP

==============

\*/

vm.setup = **function**() {

vm.reiniciarPaises();

};

vm.setup();

};

/\*

==============

CONFIGURATION

==============

\*/

PaisController.$inject = [

"$log",

"$window",

"paisService"

];

win.MainApp.Controllers

.controller("PaisController",

PaisController);

}(window));

Nota: Uso de Angular JS mejores prácticas ‘John Papas’

1. Implementar archivo de estilos (estilo.css). Como ejercicio para ver el consumo de archivo .css en e portlet.

*.tabla* {

height: *250px*;

overflow: *auto*;

width:*400px*;

overflow-y: *scroll*;

}

*.table-responsive* {height:*180px*;}

**body**{

background-color: *#bdc3c7*;

}

*.table-fixed*{

width: *100%*;

background-color: *#f3f3f3*;

tbody{

height:*200px*;

overflow-y:*auto*;

width: *100%*;

}

**thead,tbody,tr,td,th**{

display:*block*;

}

**tbody**{

td{

float:*left*;

}

}

**thead** {

tr{

th{

float:*left*;

background-color: *#f39c12*;

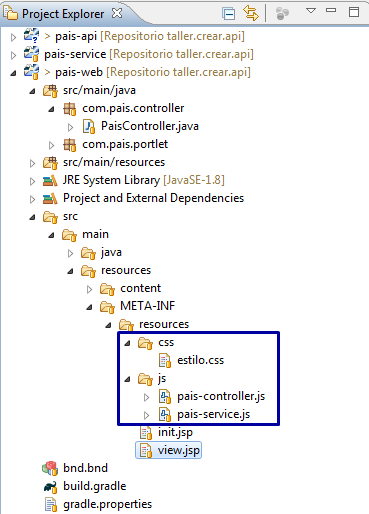
border-color:*#e67e22*;

}

}

}

}



### Ajustar Portlet y vista JSP

1. Modificar el archivo PaisPortlet.java para que el Portlet tenga acceso a librerías servicios propios y remotos, controladores, estilos propios y remotos; y servicios REST requeridos por el aplicativo.

*package com.pais.portlet;*

*import com.liferay.portal.kernel.portlet.bridges.mvc.MVCPortlet;*

*import javax.portlet.Portlet;*

*import org.osgi.service.component.annotations.Component;*

*@Component(immediate = true, property = {*

*"com.liferay.portlet.display-category=categoria.pais",*

*"com.liferay.portlet.instanceable=true",*

*"javax.portlet.display-name=Modulo OSGI. Pais-WEB Portlet",*

*"javax.portlet.init-param.template-path=/",*

*"javax.portlet.init-param.view-template=/view.jsp",*

*"javax.portlet.resource-bundle=content.Language",*

*"javax.portlet.security-role-ref=power-user,user",*

*"com.liferay.portlet.footer-portlet-javascript=/js/pais-service.js",*

*"com.liferay.portlet.footer-portlet-javascript=/js/pais-controller.js",*

*"com.liferay.portlet.header-portlet-css=https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css",*

*"com.liferay.portlet.footer-portlet-javascript=https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js",*

*"com.liferay.portlet.header-portlet-css=/css/estilo.css"*

*}, service = Portlet.class)*

*public class PaisPortlet extends MVCPortlet {*

**Notas:**

***"com.liferay.portlet.footer-portlet-javascript=/js/pais-service.js"***

* Acceso del Portlet a la librería de servicios ***pais-service.js***

***"com.liferay.portlet.footer-portlet-javascript=/js/pais-controller.js",***

* Acceso del Portlet al controlador de l aplicativo ***pais-controller.js***

***"com.liferay.portlet.header-portlet-css=https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css",***

* Acceso del Portlet a los estilos ***online*** de ***Bootstrap***

***"com.liferay.portlet.footer-portlet-javascript=https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js",***

* Acceso del Portlet a librería ***online*** de ***Bootstrap***

***"com.liferay.portlet.header-portlet-css=/css/estilo.css"***

* Acceso del Portlet al archivo de estilos propios del aplicativo

1. Modificar la vista view.jsp para consumir los servicios y para modifciar los estilos del aplicativo.

<%@ include file=*"/init.jsp"*%>

<div class=*'row'* ng-controller=*'PaisController as ctrl'*>

<div class=*"col-sm-6"*>

<p style="padding-left: *1cm*">

<button type=*"button"* class=*"btn btn-success btn-small"*

ng-click=*"ctrl.getPaises()"*>

<span class=*"glyphicon glyphicon-search"*></span> Países

</button>

<a id=*"reiniciar"* href ng-click=*"ctrl.reiniciarPaises()"*>Reiniciar</a>

</p>

<hr>

<div class=*"table table-fixed tabla"*>

<table class=*"table table-striped"*>

<thead>

<tr>

<th align=*"center"*>NOMBRE</th>

<th align=*"center"*>CAPITAL</th>

<th align=*"center"*>MONEDA</th>

<th align=*"center"*>IDIOMA</th>

</tr>

</thead>

<tbody>

<tr ng-repeat=*"pais in ctrl.paises"* ng-click=*"ctrl.seleccionarPais(pais);"*>

<td>{{pais.nombre}}</td>

<td>{{pais.capital}}</td>

<td>{{pais.moneda}}</td>

<td>{{pais.idioma}}</td>

</tr>

</tbody>

</table>

</div>

</div>

<div class=*"col-sm-6"*>

<h2>Modificar o agregar país</h2>

<hr>

<div class=*"form-group"*>

<label class=*"control-label col-sm-3"* for=*"pwd"*>País:</label>

<div class=*"col-sm-9"*>

<input type=*"input"* class=*"form-control"* placeholder=*"Pais"*

name=*"pais"* ng-model=*"ctrl.pais.nombre"*>

</div>

<br><br>

<label class=*"control-label col-sm-3"* for=*"pwd"*>Capital:</label>

<div class=*"col-sm-9"*>

<input type=*"input"* class=*"form-control"* placeholder=*"Capital"*

name=*"capital"* ng-model=*"ctrl.pais.capital"*>

</div>

<br><br>

<label class=*"control-label col-sm-3"* for=*"pwd"*>Moneda:</label>

<div class=*"col-sm-9"*>

<input type=*"input"* class=*"form-control"* placeholder=*"Moneda"*

name=*"modeda"* ng-model=*"ctrl.pais.moneda"*>

</div>

<br><br>

<label class=*"control-label col-sm-3"* for=*"pwd"*>Idioma:</label>

<div class=*"col-sm-9"*>

<input type=*"input"* class=*"form-control"* placeholder=*"Idioma"*

name=*"idioma"* ng-model=*"ctrl.pais.idioma"*>

</div>

<br><br>

</div>

<button type=*"button"* class=*"btn btn-success btn-small"*

ng-click=*"ctrl.agregarPais();"*>

<span class=*"glyphicon glyphicon-search"*></span> Agregar

</button>

<button type=*"button"* class=*"btn btn-success btn-small"*

ng-click=*"ctrl.modificarPais();"*>

<span class=*"glyphicon glyphicon-search"*></span> Modificar

</button>

<button type=*"button"* class=*"btn btn-success btn-small"*

ng-click=*"ctrl.borrarPais();"*>

<span class=*"glyphicon glyphicon-search"*></span> Borrar

</button>

<button type=*"button"* class=*"btn btn-success btn-small"*

ng-click=*"ctrl.limpiarCampos();"*>

<span class=*"glyphicon glyphicon-search"*></span> Limpiar

</button>

</div>

</div>

**Notas:**

<div class=*'row'* ng-controller=*'PaisController as ctrl'*>

* Uso de buenas practicas angular – JS.

<tr ng-repeat="pais in ctrl.paises" ng-click="ctrl.seleccionarPais(pais);">

* Uso de buenas practicas angular – JS.

<div class="table table-fixed tabla">

* Uso de estilos personalizados propios del aplicativo (estilo.css)

<div class="form-group">

<label class="control-label col-sm-3" for="pwd">País:</label>

<div class="col-sm-9">

* Uso de estilos bootstrap

Notas

### bnd.bnd

La definición de bnd permite que la especificación se haga usando "patrones", una expresión regular modificada. Todos los patrones de la definición se comparan con cada paquete de la [ruta de clase] [# CLASSPATH]. Si el patrón es un patrón de negación (comienza con!) Y se empareja, entonces el paquete está completamente excluido. Los patrones normales hacen que el paquete se incluya en el paquete resultante. Los patrones pueden incluir directivas y atributos, estos elementos se copiarán en la salida. La lista se ordena, los patrones anteriores tienen efecto antes de los patrones posteriores. Los ejemplos siguientes copian todo en la ruta de clase excepto para los paquetes que empiezan por com. El valor predeterminado para Export-Package es "\*", lo que puede resultar en paquetes bastante grandes. Si los paquetes fuente tienen una versión asociada (de su manifiesto de archivo de paquete), esta versión se agrega automáticamente a las cláusulas.