TABLA DE CONTENIDO

[TABLA DE CONTENIDO 1](#_Toc494576287)

[Crear Modulo OSGI – Portlet WEB 2](#_Toc494576288)

[Objetivo: 2](#_Toc494576289)

[Preparar el proyecto 2](#_Toc494576290)

[Crear el Proyecto Gradle en LiferayDeveloperStudio (Importar) 3](#_Toc494576291)

[MVCPortlet 7](#_Toc494576292)

[JSP 7](#_Toc494576293)

[Language.properties 7](#_Toc494576294)

[Leyenda 15](#_Toc494576295)

[Protocolo http 15](#_Toc494576296)

[bnd.bnd 16](#_Toc494576297)

[build.gradle 16](#_Toc494576298)

[Clase: 16](#_Toc494576299)

[Herencia - Clases en Java: 16](#_Toc494576300)

[Clases Abstractas 17](#_Toc494576301)

[Interfaces 18](#_Toc494576302)

[Notas 19](#_Toc494576303)

[Mostrar todas las tareas de Gradle 19](#_Toc494576304)

[Gogo Shell 19](#_Toc494576305)

[Revisar Modulo Generado 20](#_Toc494576306)

Crear Modulo OSGI – Portlet WEB

### Objetivo:

Crear un módulo OSGI (Portlet), paso a paso, que permita desplegar un portlet en Liferay DXP donde se despliegue una lista de paises en una tabla estatica.

### Preparar el proyecto

Construir un proyecto Liferay 7, para lo cual utilizaremos Gradle en Liferay Developer Studio.

En el Explorador de windows:

1. Crear repositorio: Unidad: \Repositorio: Repositorio del proyecto
2. Crear Carpeta pais-web : Proyecto WEB (Modulo OSGI)

En cualquier editor de texto:

1. Crear archivo: bnd.bnd: Configuración del paquete (Nombre, paquete y versión). Permite identificar un paquete único del módulo.

Bundle-Name:Modulo OSGI -- Pais-WEB

Bundle-SymbolicName: com.pais.web

Bundle-Version: 1.0.0

1. Crear archivo build.gradle: Configuración del gradle.

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:$sonVersion"

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"

}

repositories {

mavenLocal()

maven {

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

}

}

1. Crear archivo gradle.properties: Propiedades del proyecto.

# Versiones dependencias

gradlePluginsVersion=3.0

jaxWsRsVersion=2.0.1

jsonVersion=3.6.3

kernelVersion=2.0.0

taglibVersion=2.0.0

portletApiVersion=2.0

servletApiVersion=2.5

jstlVersion=1.2

compendiumVersion=5.0.0

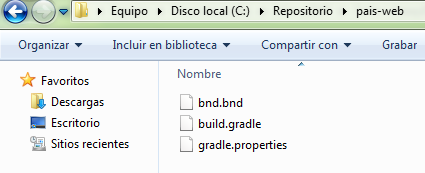
# Valores por defecto - workspace properties:

#

liferay.workspace.environment=dev

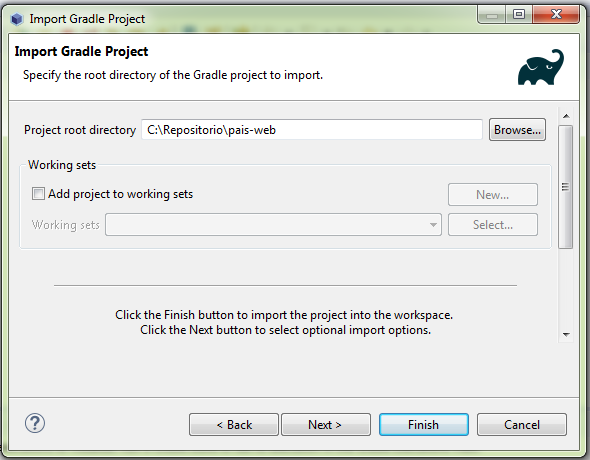
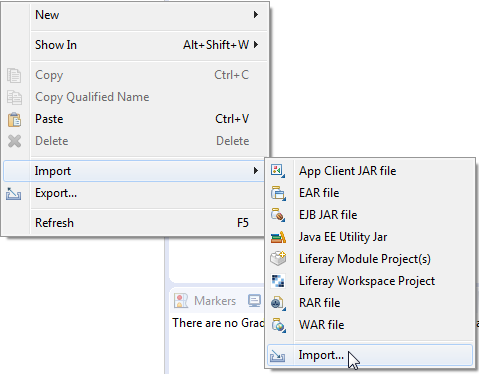
liferay.workspace.bundle.artifact.name=tomcat

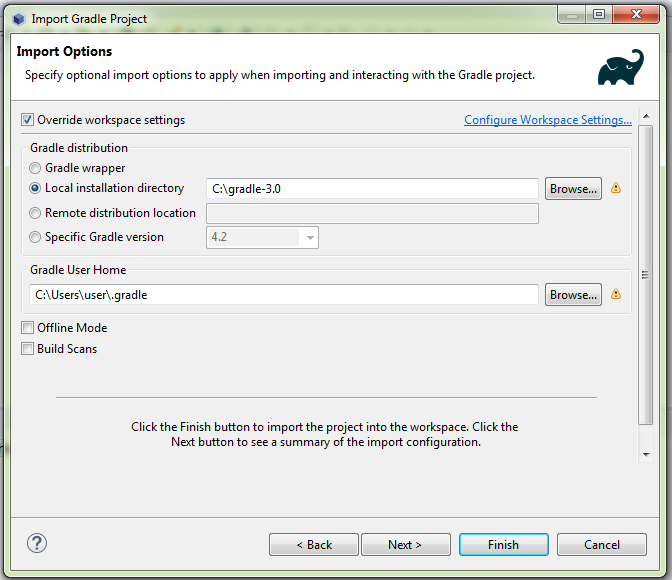
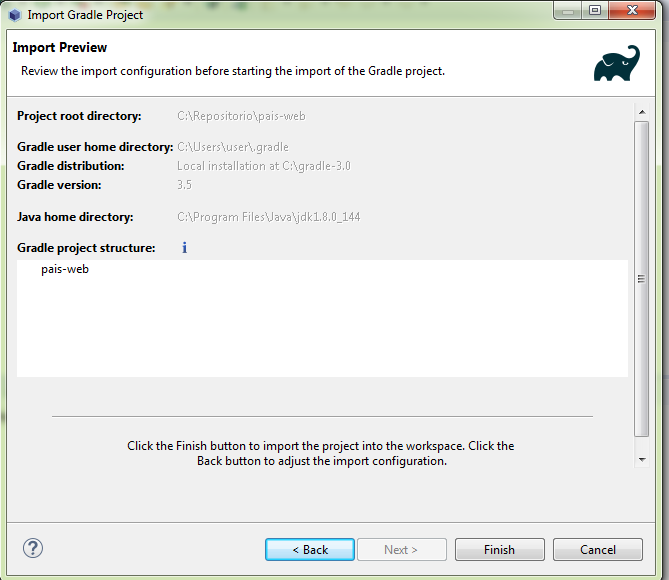
org.gradle.daemon=false



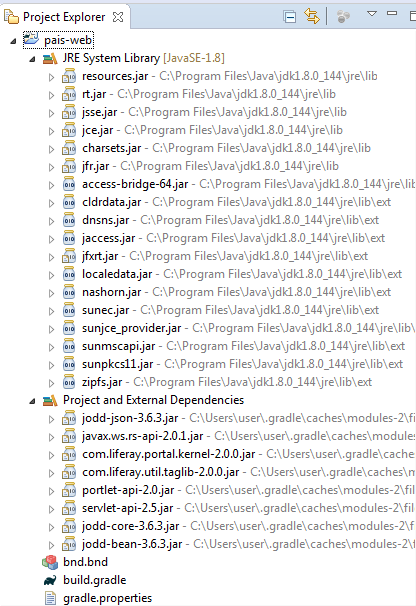
### Crear el proyecto gradle en LiferayDeveloperStudio (Importar)

Importar el repositorio pais-web creado previamente, en Liferay Developer Studio.

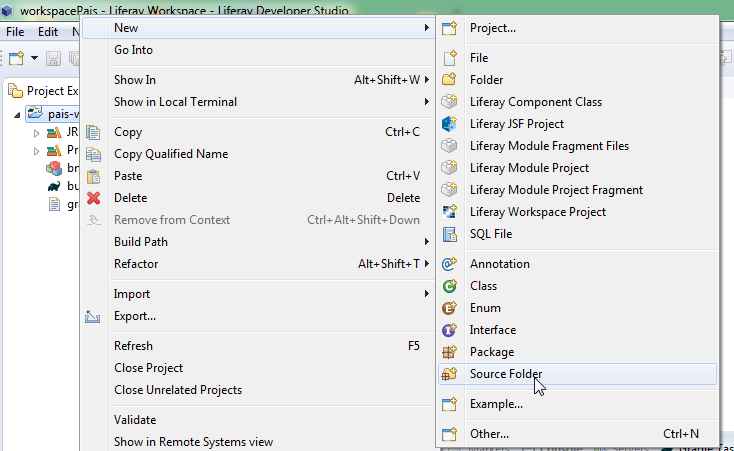


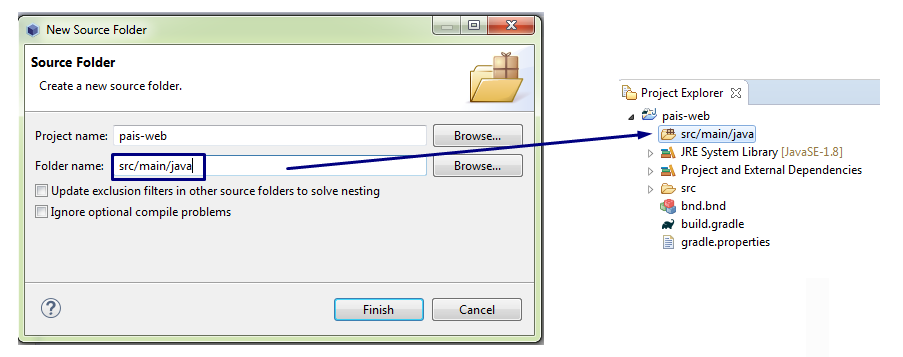
****

Se agrega el proyecto al espacio de trabajo. Con las librerias de JRE y las dependencias solicitadas en el archivo build.gradle

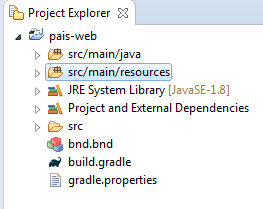
**Crear las carpetas de trabajo y paquetes.**

* Crear carpeta de archivos fuentes (src/main/java). Requerida para agregar paquetes y archivos fuentes (classPath).
  + Click derecho del mouse, sobre el nombre del proyecto

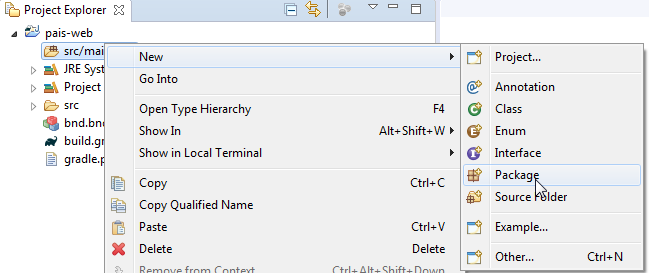
****

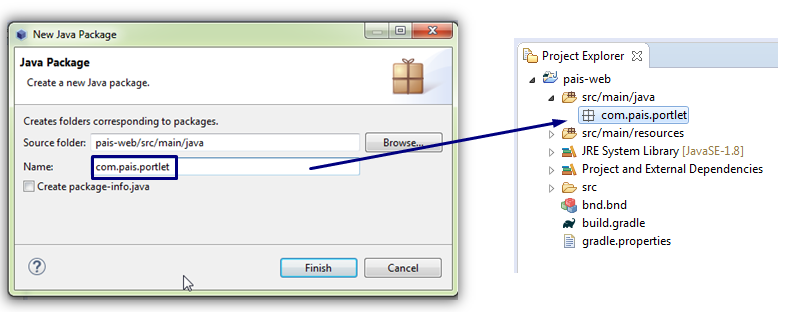
Se agrega la carpeta de fuentes al proyecto

Agregar carpeta de recursos del proyecto (resources)

****

* Crear paquete para archivos fuentes del proyecto
* Click derecho del mouse, sobre la carpeta de fuentes creada previamente





**Crear el portlet del proyecto.**

### MVCPortlet

*PaisPortlet* extiende *MVCPortlet*, clase propia de Liferay anotada con @*Component* para indicar que es un servicio declarativo de *OSGi*. Sus elementos son:

* immediate = true. Indica que el componente se active inmediatamente después de ser instalado.
* service = Portlet.class. Especifica el tipo bajo el cual registrar este componente como servicio. En nuestro caso será javax.portlet.Portlet.
* property = {...}. Conjunto de propiedades del componente.

### JSP

Los archivos JSP formarán nuestra vista. En Liferay es una buena práctica que el archivo init.jsp contenga todos nuestros import de Java, declaraciones taglib e inicialización de variables.

init.jsp esta incluido en view.jsp, que es el JSP que representa la vista del portlet, esto se establece en la clase *PaisPortlet* (javax.portlet.init-param.view-template=/view.jsp)

### Language.properties

El archivo *Language.properties* está compuesto de pares nombre=valor y sirve para definir los textos que aparecen en nuestro portlet.

Permite internacionalizar nuestro portlet, esto se estable en la clase PaisPortlet

(javax.portlet.resource-bundle=content.Language)

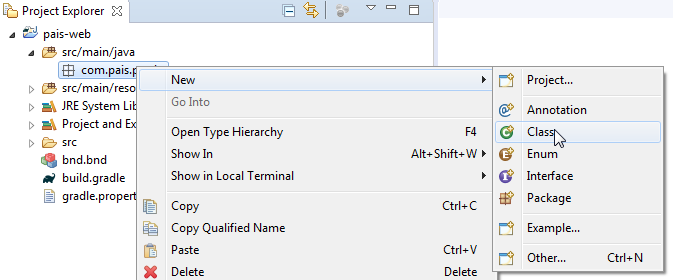
**Notas:**

Entre otras:

* com.liferay.portlet.display-category=categoria.pais => Define la categoria donde se ubica el portlet en Liferay.
* javax.portlet.security-role-ref=power-user,user => Rol

En la versiones anteriores de Liferay, estas propiedades se definian a través de archivos XML.

* Crear clase PaisPortlet.java.



**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",

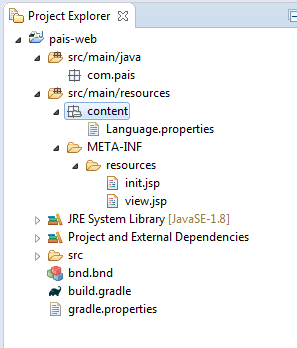
"javax.portlet.resource-bundle=content.Language",

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

}, service = Portlet.**class**)

**public** **class** PaisPortlet **extends** MVCPortlet {}

* Crear paquetes y contenido de los recursos. (Este se puede copiar y modificar y ajustar de otro proyecto)
  + Content -> Language.properties
  + META-INF -> resources -> init.jsp y view.jsp



* Crear view.jsp. Información del portlet y listado de países

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

<div>

<h1><liferay-ui:message key=*"pais.web.portlet.title"*/></h1>

<hr>

<h3><liferay-ui:message key=*"pais.web.portlet.caption"*/></h3>

<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>

<td>Argentina</td>

<td>Buenos Aires</td>

<td>Peso</td>

<td>Espa\u00f1ol</td>

</tr>

<tr>

<td>Brasil</td>

<td>Brasilia</td>

<td>Cruzeiro Real</td>

<td>Portugues</td>

</tr>

<tr>

<td>Colombia</td>

<td>Bogot\u00e1</td>

<td>Peso</td>

<td>Espa\u00f1ol</td>

</tr>

<tr>

<td>Panam\u00e1</td>

<td>Panam\u00e1</td>

<td>Balb\u00f3a</td>

<td>Espa\u00f1ol</td>

</tr>

<tr>

<td>Canad\u00e1</td>

<td>Otawa</td>

<td>D\u00f3lar</td>

<td>Ingles</td>

</tr>

<tr>

<td>Italia</td>

<td>Roma</td>

<td>Lira</td>

<td>Italiano</td>

</tr>

<tr>

<td>Jap\u00f3n</td>

<td>Tokio</td>

<td>Yen</td>

<td>Japon\u00e9s</td>

</tr>

<tr>

<td>Venezuela</td>

<td>Caracas</td>

<td>Bol\u00edvar</td>

<td>Espa\u00f1ol</td>

</tr>

</tbody>

</table>

</div>

</div>

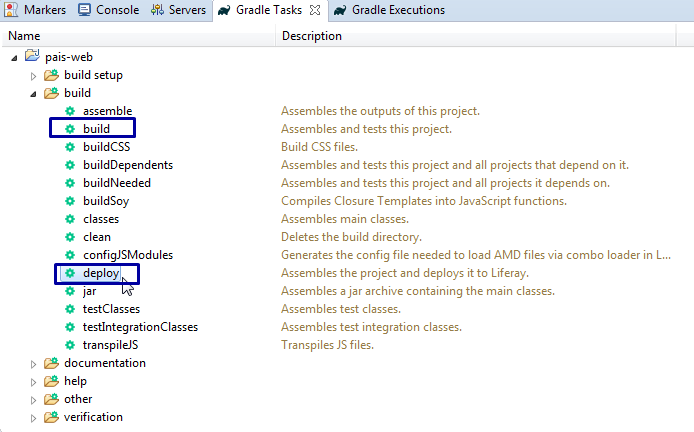
* Crear archivo Languages.properties

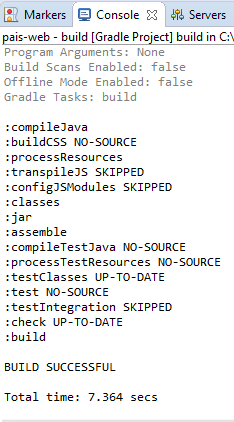
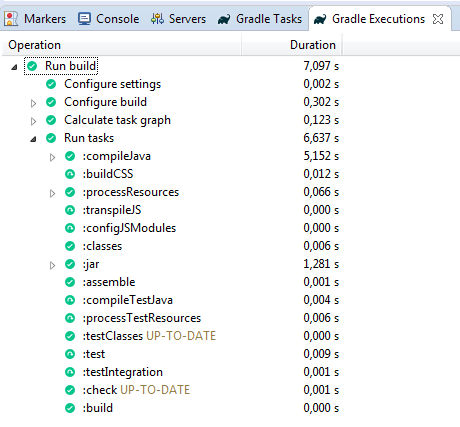
pais.web.portlet.title=Modulo OSGI. Pais-WEB Portlet

pais.web.portlet.caption=Bienvenidos a Paises del Mundo

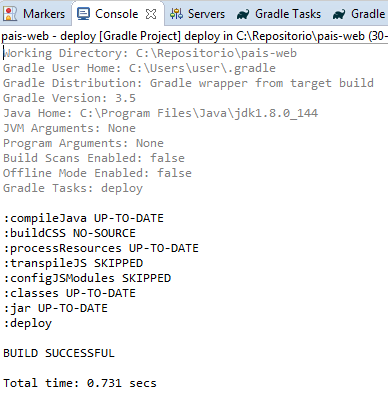
**Levantar el servidor, publicar y probar el portlet**

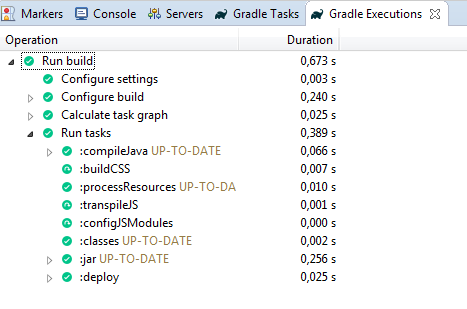
1. Compilar y publicar el proyecto.
   * Build: Compilar el proyecto
   * Deploy: Publicar el proyecto

****

****

**Compilar (build)**

**Publicar (deploy)**

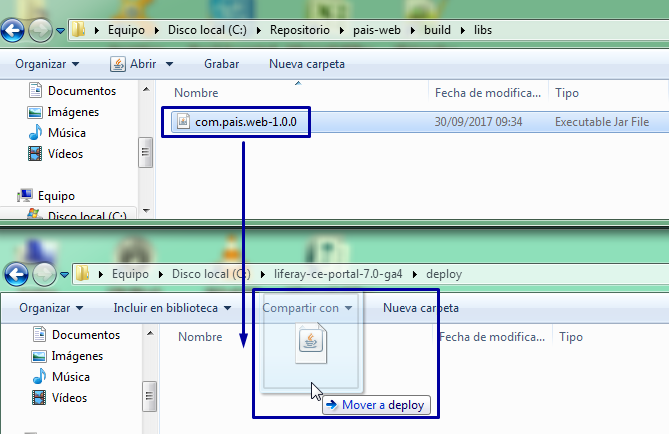


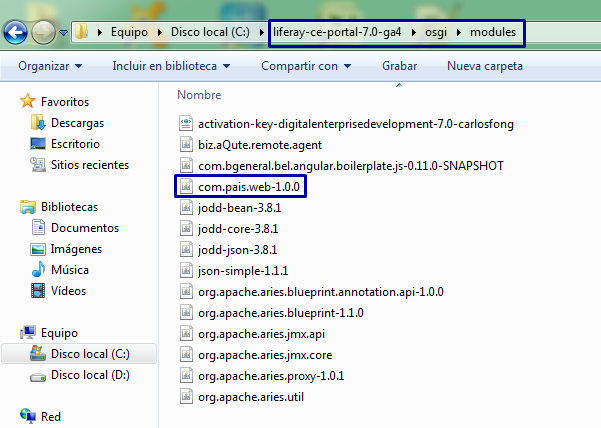
**Nota:**

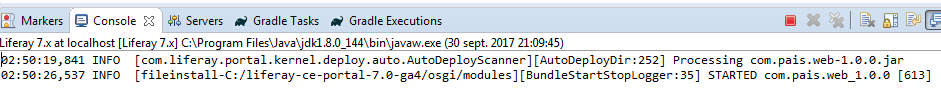
Si no esta bien configurado el ambiente no se publica el proyecto directamente al servidor. En este caso buscar el proyecto en el repositorio y copiar el archivo .jar directamente en el directorio deploy del servidor para que se publique.

Unidad:\Repositorio\pais-web\build\libs\ com.pais.web-1.0.0

* Unidad:\liferay-ce-portal-7.0-ga4\deploy

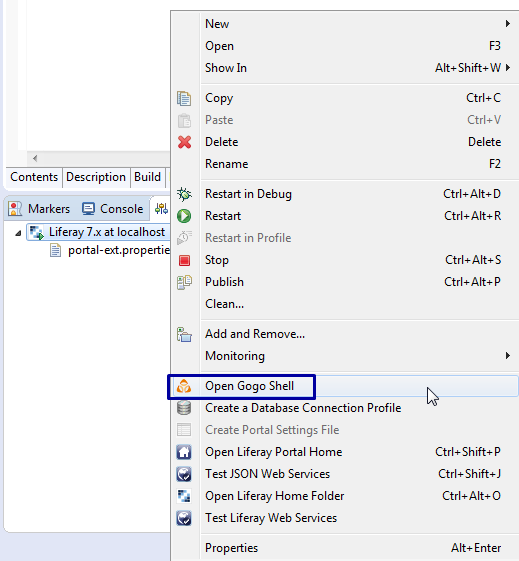


El Proyecto se agrega al servidor de Liferay.

En la consola del IDE se refleja el cambio:

Nota: STARTED: Implica que el proyecto se publico correctamente. Sin errores.

1. Validar el despliegue del proyecto a través de a Herramienta Gogo Shell del IDE.



**Comandos Gogo Shell:**

**Comando: lb pais:**

**ID**: Identificador del modulo o proyecto en el servidor

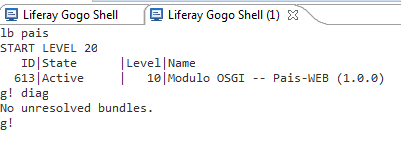
**State**: Esado actual del Proyecto o modulo.

**Level**: Nivel

**Name**: Nombre del proyecto. (Agregado en el bnd.bnd)

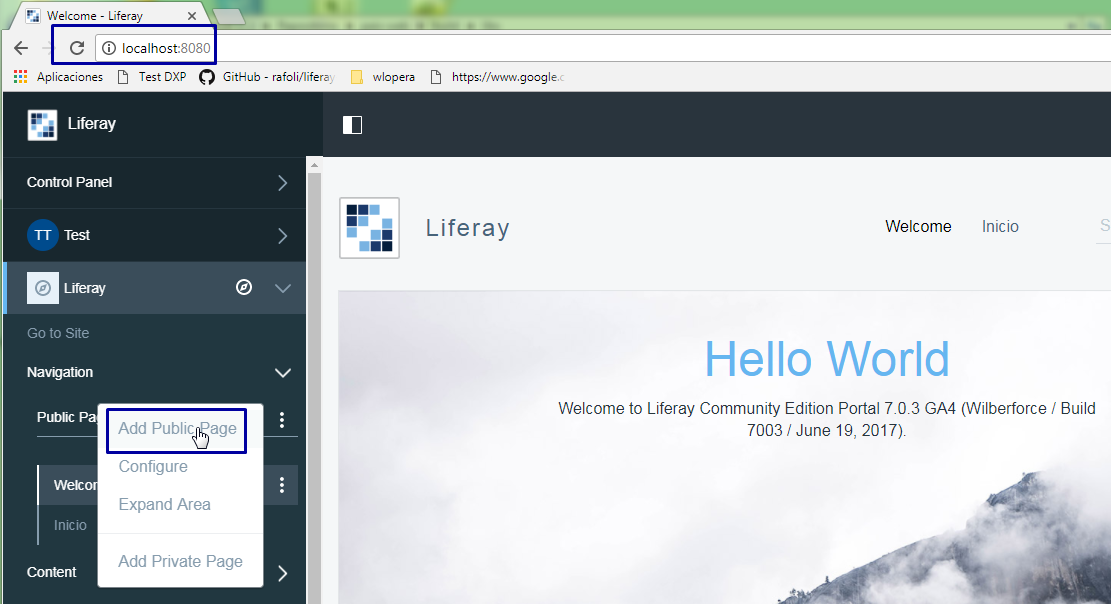
**Comando: diag:**

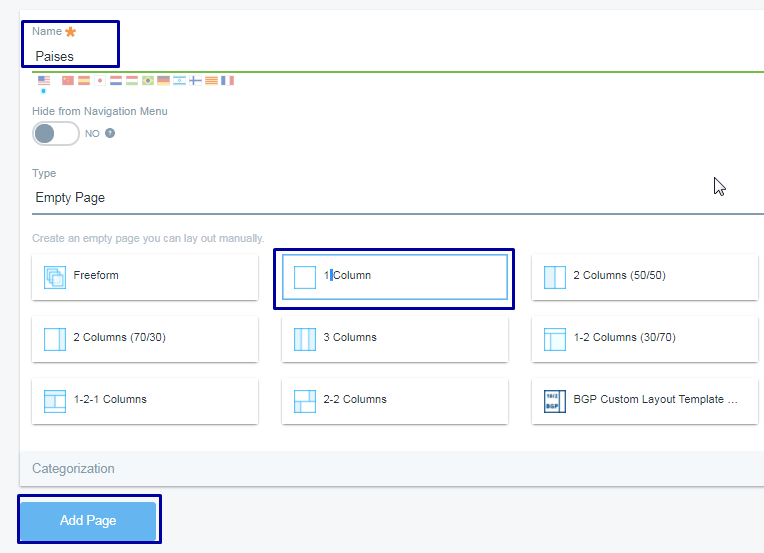
Valida estado actual de los proyectos. Paquetes con problemas o correctos.



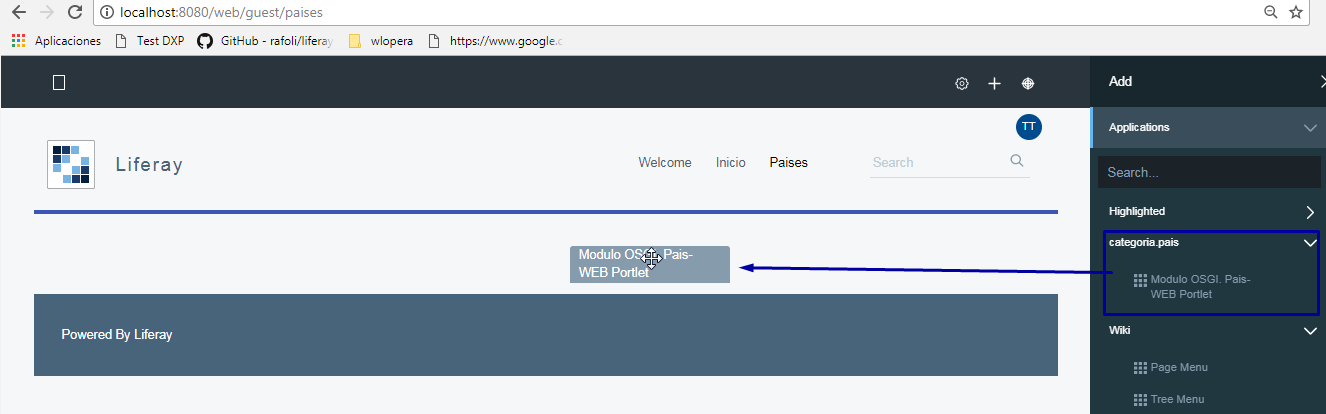
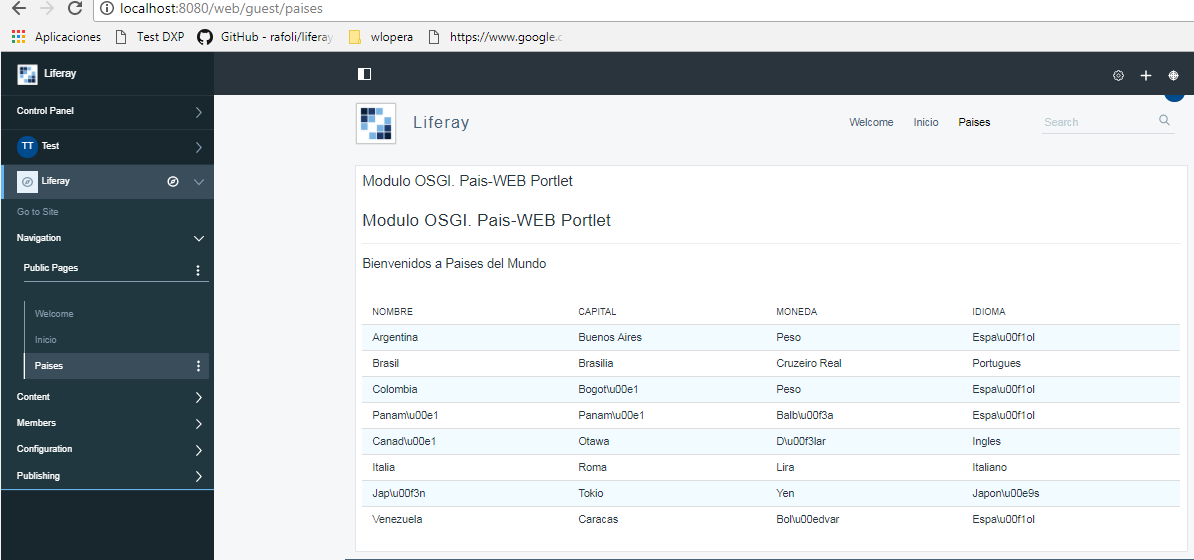
**Crear página y publicar el Portlet.**

* Abrir el servidor <http://localhost:8080/>
* Crear una página publica (Paises)

****



* Publicar el portlet .(Arrastrar el Portlet a la vista)

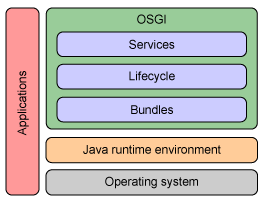
Se agrega el Portlet. Refrescar la vista.

Leyenda

OSGI Architecture

OSGI (Open Services Gateway initiative) es la solución que aporta Java a la programación modular. Un módulo es un componente auto-contenido con una funcionalidad perfectamente delimitada dentro de un sistema mayor.

Los módulos OSGI pueden ser arrancados y parados en caliente (sin tener que para y reiniciar un servidor), pueden comunicarse entre ellos y además pueden recibir avisos cuando otro módulo es arrancado o parado.



### Protocolo http

El **Protocolo** de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo **protocolo** cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores **HTTP**.

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

* Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo Location del cliente Web.
* El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
* Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.   
  Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,…
* El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
* Se cierra la conexión TCP.

### bnd.bnd

Es un archivo de plantilla de configuración MANIFEST que se utiliza para crear automáticamente MANIFEST.MF.

### build.gradle

Se utiliza para la gestión de dependencias y compilaciones de los módulos Liferay 7. Liferay 7 proporciona el panel de tareas de generación de Gradle para ejecutar tareas de creación, implementación, limpieza y compilación.

    Los artefactos de generación generados se pueden copiar en la carpeta deploy / bundles / osgi / modules para la implementación manual.

    Para el despliegue automático, asegúrese de que tomcat se crea en la carpeta liferay-workspace / bundle.

### Clase:

* Una clase es un molde del que luego se pueden crear múltiples objetos, con similares características.
* Una clase es una plantilla (molde), que define atributos (variables) y métodos (funciones)
* La clase define los atributos y métodos comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas funciones.

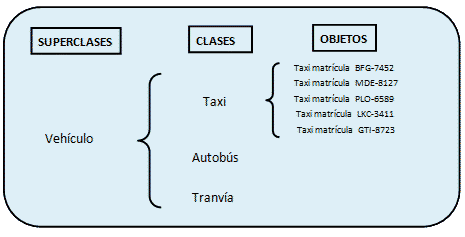
### Herencia - Clases:

Clase**:** abstracción que define un tipo de objeto especificando qué propiedades (atributos) y operaciones (métodos) disponibles va a tener.

**Objeto**: entidad existente en la memoria del ordenador que tiene unas propiedades (atributos o datos sobre sí mismo almacenados por el objeto) y unas operaciones disponibles específicas (métodos).

Herencia

* Existen dos clases, a las que llamaremos (superclase o clase base) y (subclase o clase derivada)
* La subclase hereda todo lo que tiene la clase superclase
  + Atributos
  + Métodos
* Un objeto de la clase subclase es también un objeto de la clase superclase



* Se indica usando la palabra reservada **extends**
  + class Autobus extends Vehiculo
* Visibilidad
  + Los miembros privados de la superclase no son visibles desde la subclase.
  + Los miembros públicos de la superclase son visibles y siguen siendo públicos en la subclase.
* Se puede acceder a los miembros de la superclase usando la palabra reservada super.
* Si una clase se declara como final no se puede heredar de ella.
* En java, todas las clases heredan implícitamente de la clase Object.

### Clases Abstractas

* En ciertos casos, una clase se diseña directamente para ser extendida por un conjunto de subclases.
* En estos casos suele interesar no implementar alguno de sus métodos, pues no tiene ningún significado en la clase base.
* Es necesario declarar tanto la clase como los métodos no implementados como abstractos a través de la palabra reservada **abstract**.
* Una clase abstracta es, por tanto, aquella que tiene alguno de sus métodos definido pero no implementado.

### Interfaces

* A veces nos interesa que todos los métodos de una clase abstracta sean abstractos.
  + Lo usamos para obligar a que todas las subclases implementen esos métodos
* Para estos casos, Java proporciona unas clases especiales llamadas **interfaces**
* Se declaran como **interface**, no como **class**.
* Representan el concepto de **clase abstracta pura**
* Una clase declarada como interface no puede tener ningún método implementado

**Importante:**

* **Para heredar de una interface se usa la palabra reservada implements**

**Ejemplo: public class MiClase implements Serializable**

**Reglas:**

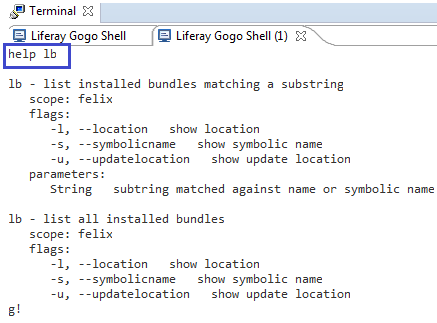
* Una interface puede heredar de otra interface
* Una clase (abstracta o no) puede heredar de una interface
* Una interface **NO** puede heredar de una clase

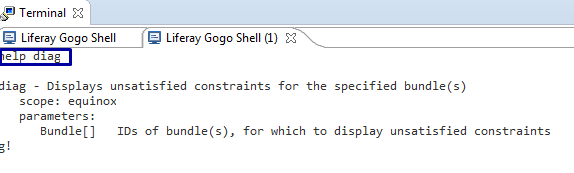
Notas

### Mostrar todas las tareas de Gradle

### Gogo Shell

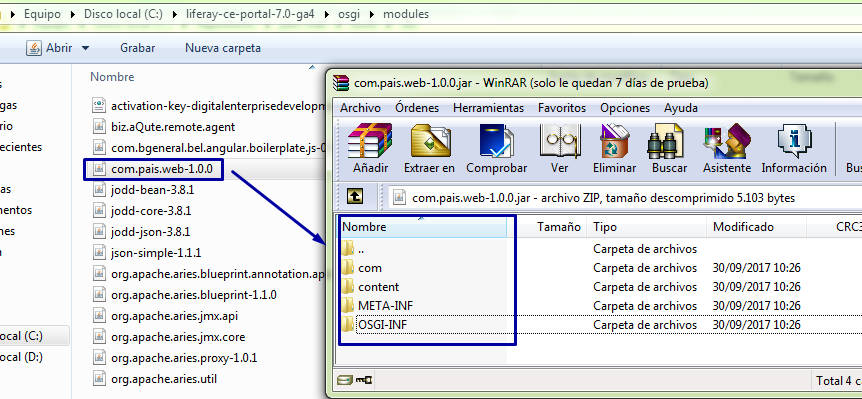
Ayuda de comandos (Help \_COMANDO\_)





### Revisar Modulo Generado

Abrir el proyecto com.paises.web-1.0.0 creado con Wirar.



Se genera las carpetas:

* + COM: Clases compiladas (Carpeta com)
  + CONTENT: Recursos del proyecto (Carpeta content)
  + META-INF: Archivos JSP y archivo MANIFEST.MF

Manifest-Version: 1.0

Bnd-LastModified: 1506828388550

Bundle-ManifestVersion: 2

Bundle-Name: Modulo OSGI -- Pais-WEB

Bundle-SymbolicName: com.pais.web

Bundle-Version: 1.0.0

Created-By: 1.8.0\_144 (Oracle Corporation)

Import-Package: com.liferay.portal.kernel.portlet.bridges.mvc;version=

"[1.0,2)",javax.portlet;version="[2.0,3)",javax.servlet,javax.servlet

.http

Javac-Debug: on

Javac-Deprecation: off

Javac-Encoding: Cp1252

Private-Package: com.pais.portlet,content

Provide-Capability: osgi.service;objectClass:List<String>="javax.portl

et.Portlet",liferay.resource.bundle;bundle.symbolic.name="com.pais.we

b";resource.bundle.base.name="content.Language"

Require-Capability: osgi.extender;filter:="(&(osgi.extender=jsp.taglib

)(uri=http://java.sun.com/portlet\_2\_0))",osgi.extender;filter:="(&(os

gi.extender=jsp.taglib)(uri=http://liferay.com/tld/aui))",osgi.extend

er;filter:="(&(osgi.extender=jsp.taglib)(uri=http://liferay.com/tld/p

ortlet))",osgi.extender;filter:="(&(osgi.extender=jsp.taglib)(uri=htt

p://liferay.com/tld/theme))",osgi.extender;filter:="(&(osgi.extender=

jsp.taglib)(uri=http://liferay.com/tld/ui))",osgi.ee;filter:="(&(osgi

.ee=JavaSE)(version=1.8))"

Service-Component: OSGI-INF/com.pais.portlet.PaisPortlet.xml

Tool: Bnd-3.2.0.201605172007

* + OSGI-INF: Archivo con la información del Portlet. (com.pais.portlet.PaisPortlet.xml).

<?xml version="1.0" encoding="UTF-8"?>

<component name="com.pais.portlet.PaisPortlet" immediate="true">

<implementation class="com.pais.portlet.PaisPortlet"/>

<service>

<provide interface="javax.portlet.Portlet"/>

</service>

<property name="javax.portlet.init-param.view-template" type="String" value="/view.jsp"/>

<property name="javax.portlet.security-role-ref" type="String" value="power-user,user"/>

<property name="javax.portlet.resource-bundle" type="String" value="content.Language"/>

<property name="javax.portlet.init-param.template-path" type="String" value="/"/>

<property name="com.liferay.portlet.instanceable" type="String" value="true"/>

<property name="javax.portlet.display-name" type="String" value="Modulo OSGI. Pais-WEB Portlet"/>

<property name="com.liferay.portlet.display-category" type="String" value="categoria.pais"/>

</component>