TABLA DE CONTENIDO

[TABLA DE CONTENIDO 1](#_Toc494576405)

[PRE REQUISITOS 2](#_Toc494576406)

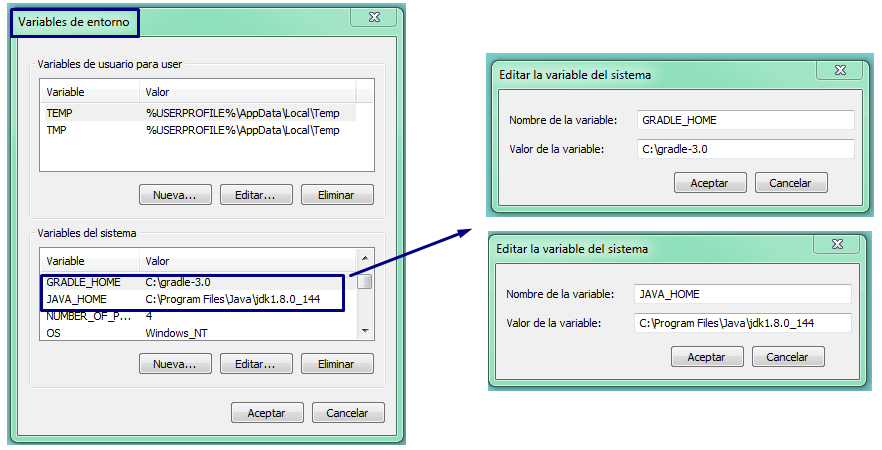
[Leyenda 7](#_Toc494576407)

[Objeto $http 7](#_Toc494576408)

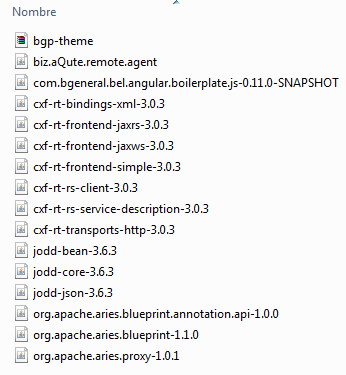
[Protocolo http 7](#_Toc494576409)

PRE REQUISITOS

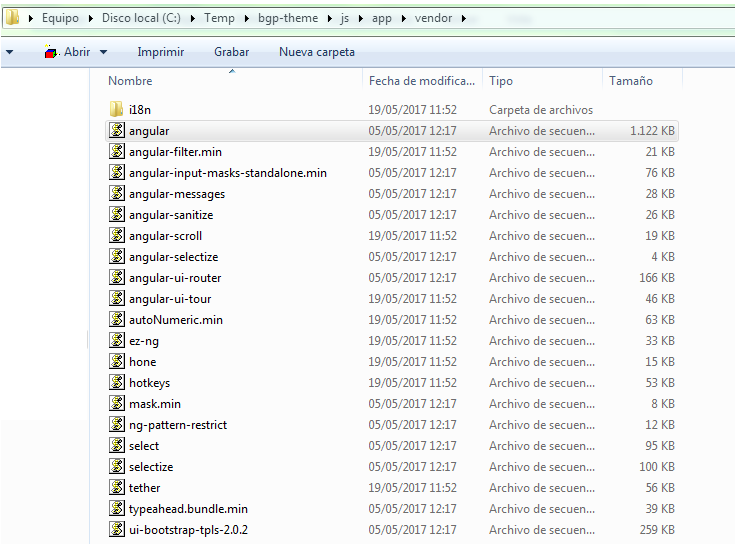
* JDK 1.8: (Java Development Kit). Kit para desarrolladores.
* Gradle 3.0: Herramienta de automatización para la construcción de proyectos
* Liferay Developer Studio: Entorno de desarrollo integrado para la creación de aplicaciones y módulos para Liferay Digital Experience Platform (DXP).
* Liferay Portal CE (liferay-ce-portal-7.0-ga4): Portal Liferay DXP
* GIT: Control de versiones diseñado
* Variables de Entorno

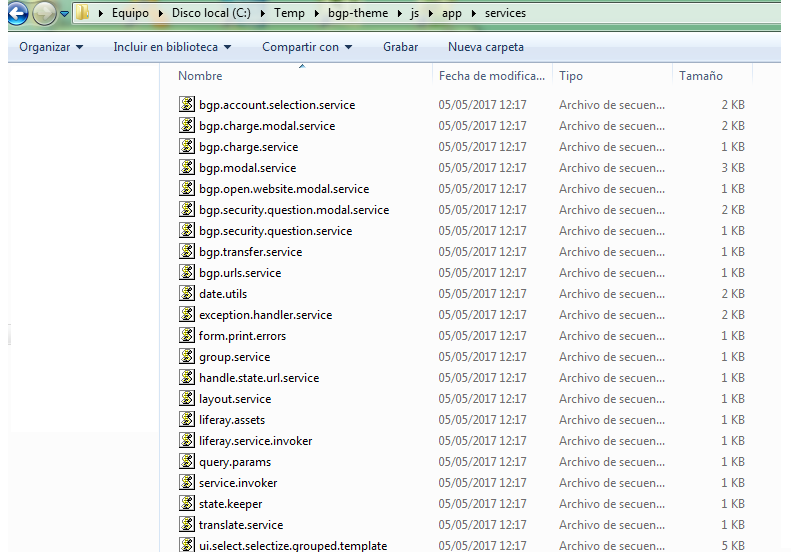


* Librerías y Theme:



Contenidos del tema: Herramientas y dependencias requeridas como angular , manejos de estilos , archivos de propiedades, controladores, filtros, directivas, servicios (ejemplo: $http), entre otras.





**Nota:** Agregar todas las dependencias y war al servidor. Para ello con el servidor arriba agregarlos los .jar y .war al directorio deploy del servidor.

Ejemplo de Servicio: service.invoker

Permite realizar peticiones HTTP ([Ver Leyenda Objeto $Http](#_Objeto_$http))

..\bgp-theme\js\app\services\service.invoker

(function(win) {

"use strict";

var serviceInvoker = function($log, $http) {

$log.debug("[Liferay/Angular/ServiceInvoker] Initializing...");

function invoke(method, path, data) {

return $http({

method: method,

headers: {

'Content-type': 'application/json'

},

url: path,

data: data

});

}

return {

invoke: invoke

};

};

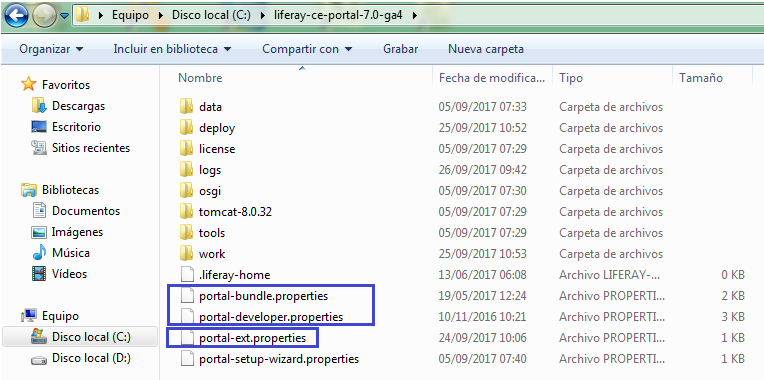
serviceInvoker.$inject = ["$log", "$http"];

win.MainApp.Services.service("serviceInvoker",

serviceInvoker);

}(window));

* Archivos de propiedades requeridos para configurar el servidor y configuraciones para el entorno de desarrollo, ejemplo subir de versión al servidor al publicar un proyecto, entre otras.

****

Ejemplo portal-ext.properties: Configurar ambiente de desarrollo.

Cuando estás cambiando estilos de Liferay, o trabajando un tema, verás que los cambios no aparecen conforme tú los vas haciendo. Para solucionar esto y que los cambios de CSS se vayan viendo conforme los vas haciendo, hay que poner el Liferay en modo desarrollo. Para ello utilizaremos el fichero portal-ext.properties.

Este archivo de configuraciones se utiliza para sobrescribir las propiedades del portal.properties. Por defecto no viene creado, por lo que lo tenemos que crear en LIFERAY\_HOME.

Aquí dentro creamos un fichero llamado portal-ext.properties. Para ponerlo en modo desarrollo hay que poner estas líneas:

theme.css.fast.load=false

theme.images.fast.load=false

javascript.fast.load=true

javascript.log.enabled=true

layout.template.cache.enabled=false

browser.launcher.url=

combo.check.timestamp=true

freemarker.engine.cache.storage=soft:1

freemarker.engine.modification.check.interval=0

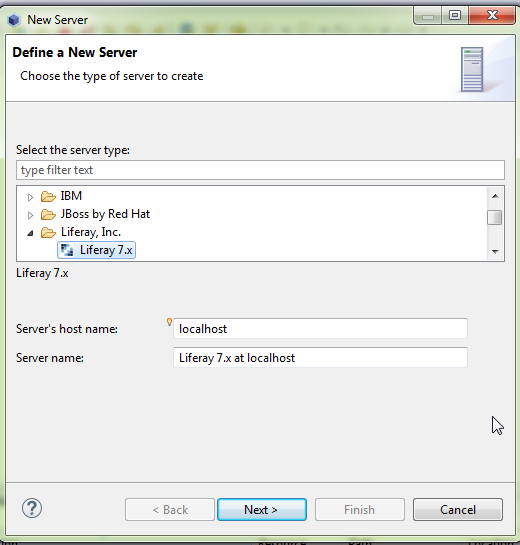
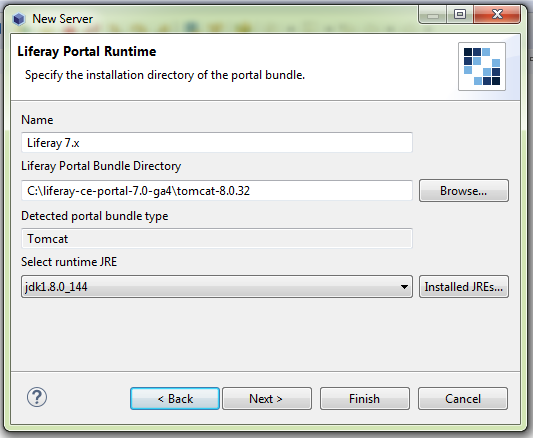
openoffice.cache.enabled=false

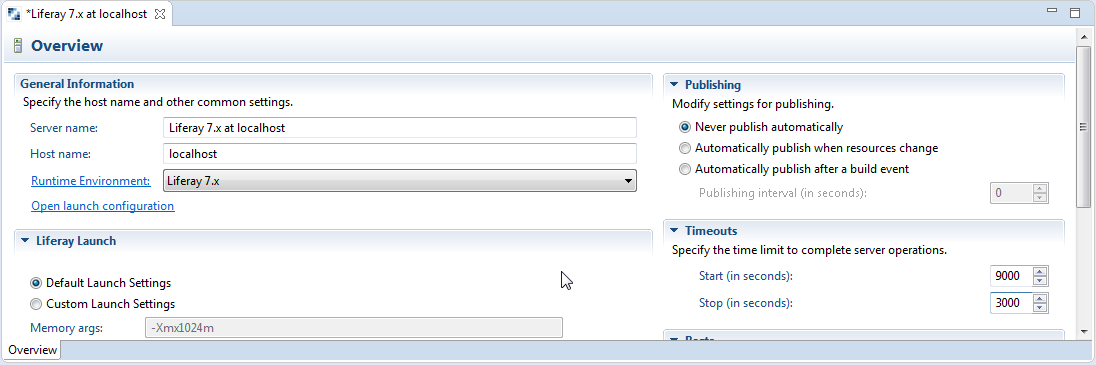
velocity.engine.resource.manager.cache.enabled=false

com.liferay.portal.servlet.filters.cache.CacheFilter=false

com.liferay.portal.servlet.filters.themepreview.ThemePreviewFilter=true

* Configurar y levantar el servidor en Liferay Developer Studio (IDE)

** **

****

Leyenda

### Objeto $http

Es un servicio que viene incorporado con Angular, y que permite comunicarnos con servidores HTTP remotos, a través de pedidos asincrónicos que transmiten datos ([ver Protocolo http](#_Protocolo_http))

* utilizando el objeto [XMLHttpRequest](https://developer.mozilla.org/en/xmlhttprequest) del browser
* o vía [JSONP](http://en.wikipedia.org/wiki/JSONP)

Se inyecta en el constructor de cada service que necesite hacer una llamada remota.

### 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.