

Sistemas Distribuidos

REST FAQ

Rodrigo Santamaría

+ Proyecto descargado de Google Drive no funciona

Eclipse

- Google Drive elimina los archivos ocultos `.settings`, `.classpath` y `.project`, entre otros, de Eclipse
 - Si importamos un proyecto con la estructura de Eclipse pero sin estos archivos, no puede interpretar la configuración y Eclipse da errores, generalmente de rutas no encontradas
- Solución:
 - Crea un proyecto Eclipse nuevo y arrastra las clases y bibliotecas del antiguo
 - O mejor, usa un pendrive o Z para tus copias de seguridad: mantienes tu privacidad, reduces gasto energético y sabes que lo que subes es lo que bajas.
 - Si usas Z, no trabajes directamente con Z como workspace, es mucho más lento que X. Úsalo sólo como copia de seguridad.

+ No hay salida en el navegador

REST

- El código no funciona o no aparece nada al visitar la URL del servicio (o Estado HTTP 404).
- En `web.xml`, `display-name` debe contener el nombre del proyecto, y el valor `param-value` del `param-name` `'com.sun.jersey.config.property.packages'` debe contener el nombre del paquete donde están las clases.
- Observa si, en la pestaña de Consola de Eclipse, Tomcat ha reportado algún error
 - Si ha arrancado correctamente la última línea será `"Server startup in xxx ms"` o similar



No hay salida en el navegador

REST

- Si tienes un error 404, recuerda que la URL incluye secciones que dependen del web.xml y de las anotaciones `@Path` (ver diap. 19 del seminario de REST)
- La anotación `@Path` debe declararse siempre después de la anotación `@GET`



+ Puerto en uso

REST

- Sobre todo en vuestros ordenadores personales, puede que tengáis alguna aplicación usando el puerto 8080
 - Buscadlas y cerradlas
 - En GNU/Linux se pueden ver los puertos en activo con `netstat` o `lsof`
 - O bien cambiad el puerto en que arranca Tomcat
 - En Eclipse, en la pestaña Servers, haciendo doble clic en el servidor se puede cambiar su configuración.





No puedo seleccionar Tomcat

Tomcat



- Al ejecutar por primera vez con `Run as > Run on Server` no se habilita el botón de finalizar tras elegir la versión de Apache Tomcat
- Parece que a veces es necesario indicárselo así:
 - 1) `Window > Preferences > Server > Runtime Environments`
 - 2) `Add`
 - 3) `Apache > Apache Tomcat X.X`

+ ClassNotFoundException

Jersey

- `java.lang.ClassNotFoundException:`
`com.sun.jersey.spi.container.servlet.ServletContainer`
- Los jar de Jersey no pueden meterse en cualquier carpeta, deben ir directamente en `WebContent/WEB-INF/lib`
 - Sin subcarpetas!



+ ContainerException – no root

Jersey

- `com.sun.jersey.api.container.ContainerException`: The `ResourceConfig` instance does not contain any root resource classes.
- El parámetro `com.sun.jersey.config.property.packages` en `web.xml` debe tener como valor el paquete que contiene los servicios en el proyecto

+ ContainerException-no method

Jersey

- `org.glassfish.jersey.server.ContainerException:`
`java.lang.NoSuchMethodError: 'void`
`jakarta.servlet.http.HttpServletResponse.setStatus(int,`
`java.lang.String)'`
- Se trata de un problema de versionado entre Tomcat y jersey, generalmente. Qué versiones exactamente dependerá del contexto, pero parece que el problema generalmente es tener una versión de Tomcat más moderna que la versión de jersey. En este caso de Stack Overflow habland de Tomcat 10 con versiones antiguas (<3) de jersey

+ No injection source

REST

- `[[FATAL] No injection source found for a parameter of type XXX at index 0.;`
 - Este problema suele darse por tener como parámetro de la interfaz un tipo no básico (p. ej. un List, HashMap, etc.)



Unsupported major.minor version

REST

- Caused by: `java.lang.UnsupportedClassVersionError: org/glassfish/jersey/servlet/ServletContainer : Unsupported major.minor version 51.0 (no puedo cargar clase org.glassfish.jersey.servlet.ServletContainer)`
 - Suele indicar que estamos usando una versión de jersey compilada en una versión de Java más moderna que la versión de Java en la que ejecutamos nuestro proyecto.
 - Solución: recompilar con la misma versión java con la que se ejecuta.
 - Posiblemente se deba a el desarrollo del proyecto en tu propio ordenador, al migrarlo al laboratorio
 - Recuerda que en el laboratorio tenemos java 1.8, cambia la configuración de la ruta a la JRE y la versión de compilación si usas un proyecto importado de Eclipse.





ServletContainer cannot be cast REST



- Caused by: class `org.glassfish.jersey.servlet.ServletContainer` cannot be cast to class `jakarta.servlet.Servlet`
 - Hemos detectado que este fallo ocurre cuando la versión de Tomcat es más moderna que la versión de jersey.
 - En particular, sabemos que ocurre si tenemos Tomcat 10 y jersey 2.3 (Eclipse EE 2020)
 - Solución: Usar Tomcat 9.

+ IllegalArgumentException

REST

- Caused by: `IllegalArgumentException`
 - Hemos detectado que este fallo en el arranque del Servidor ocurre cuando la versión de Java es más moderna que la versión de Jersey.
 - En particular, sabemos que ocurre si tenemos Java 11 y Jersey 2.3
 - Solución: Usar Java 1.8 (o bien usar Jersey 3 y Tomcat 10)
 - En particular, esto ocurre en algunos ordenadores del aula que, aunque tienen instalada la versión 1.8, utilizan por defecto Java 11
 - En estos casos, en Eclipse, tenemos que modificar para el proyecto la biblioteca en Properties/Java Build Path, y Properties/Facets

+ startup.sh no responde en ssh

Despliegue: tomcat

- Cuando ejecuto startup desde ssh con una orden como:
 - `ssh 127.0.0.1 "/ruta/a/tomcat/bin/startup.sh"`
- Me indica que tomcat se ha arrancado, pero luego si accedo a <http://localhost:8080> veo que tomcat no está arrancado
- Solución:
 - Si te fijas en la salida de tomcat, es distinta a cuando la ejecutas sin ssh

```
Using CATALINA_BASE: /ruta/a/tomcat
Using CATALINA_HOME: /ruta/a/tomcat
Using CATALINA_TMPDIR: /ruta/a/tomcat/tmp
Using JRE_HOME: /opt/jdk1.6.0
```

en local

```
Using CATALINA_BASE: /ruta/a/tomcat
Using CATALINA_HOME: /ruta/a/tomcat
Using CATALINA_TMPDIR: /ruta/a/tomcat/tmp
Using JRE_HOME: /usr
```

con ssh 127.0.0.1



startup.sh no responde en ssh (ii)

Despliegue: tomcat

- ¿Por qué, si las dos son llamadas 'locales', aunque sea a través de ssh?
 - ssh no ejecuta archivos .profile o .bashrc que puedan definir variables de entorno
 - En concreto, no definiremos la versión de java a ejecutar
- En el laboratorio suele haber varias versiones de Java
 - Basta ejecutar `whereis java` para comprobarlo
- Solución: exportar la versión correcta de java en el ssh:
 - ```
ssh 127.0.0.1 'export PATH=$PATH:/opt/jdk1.6.0/bin; /ruta/a/tomcat/bin/startup.sh'
```





# startup.sh no responde en ssh (iii)

Despliegue: tomcat

- Otra solución: crear variables de entorno en Tomcat:
  - Crear un archivo `setenv.sh` dentro de la carpeta `bin` de Tomcat, con las siguientes líneas\*:

```
export JAVA_HOME=/opt/jdk1.8.0_60/
```

```
export JRE_HOME=/opt/jdk1.8.0_60/jre
```

\* La ruta exacta puede cambiar, hay que asegurarse primero de cuál es la ruta en la que está instalado java, por ejemplo con `whereis java`

# + No consigo compartir claves

Despliegue: ssh

- Consigo ejecutar el script `shareKeys.sh` disponible en <http://vis.usal.es/rodrigo/documentos/sisdis/scripts>
  - Y comparte bien las claves
  - Pero al ejecutar otros script me las vuelve a pedir
- Solución: las sentencias al final del script anterior (`ssh-agent` y `ssh-add`) deben añadirse al inicio de cualquier otro script que quiera utilizar el sistema de claves exportadas (por ejemplo, si queremos ejecutar un `ssh` o `scp` desde una orden `ssh`)

# + No consigo acceder a variables

Despliegue: ssh

- Cuando ejecuto una orden ssh que contiene alguna variable del servidor destino, esta aparece vacía:
  - `ssh "echo $variable"`
- Las comillas dobles (") resuelven los caracteres de escape (\) y los nombres de variables (\$) *antes* de interpretar el texto. Es decir, `$variable` se resuelve como una variable local antes de ejecutar ssh
- Solución: utilizar comillas simples ('), que toman el texto literalmente:
  - `ssh 'echo $variable'`

# + Tomcat en distintos puertos



- El puerto en el que escucha Tomcat es configurable en `server.xml`, en el tag `Connector`:

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000" RedirectPort="8443" />
```

- También es necesario cambiar los otros dos puertos que aparecen en el `server.xml`:

```
<Server port="8005" shutdown="SHUTDOWN"> [...]
```

```
<Connector port="8009" protocol="AJP/1.3"
redirectPort="8443" />
```

- En resumen, tenemos que duplicar la carpeta Tomcat y en una de ellas cambiarle a otros puertos, y luego ejecutar ambos Tomcats

*NOTA: esta solución es más compleja que utilizar un despachador de procesos en un solo Tomcat, pero puede resultar más elegante*

# + Tomcat en distintos puertos

## Instancias de Jersey

- ¿Por qué no un sólo Tomcat escuchando en dos puertos distintos, mediante la creación de otro `Connector`?
  - Porque Jersey, si tiene clases `Singleton`, **no** va a crear dos instancias de ellas, una por puerto
  - Si queremos dos instancias, tendríamos que usar dos Tomcats, cada uno escuchando en su puerto

# + Tomcat en distintos puertos

## Distinguiendo el puerto en Jersey

- Cualquier servicio puede tener acceso a la cabecera de la petición HTTP, que contiene el puerto al que se invoca
- En Jersey 1.x, mediante la etiqueta `@Context`  
`HttpHeaders` o `@HeaderParam("Host")`
  - <https://dzone.com/articles/using-context-jax-rs>
- En Jersey 2, la cosa se complica, es necesario implementar la interfaz `ClientResponseFilter`
  - <http://www.hascode.com/2013/12/jax-rs-2-0-rest-client-features-by-example/>

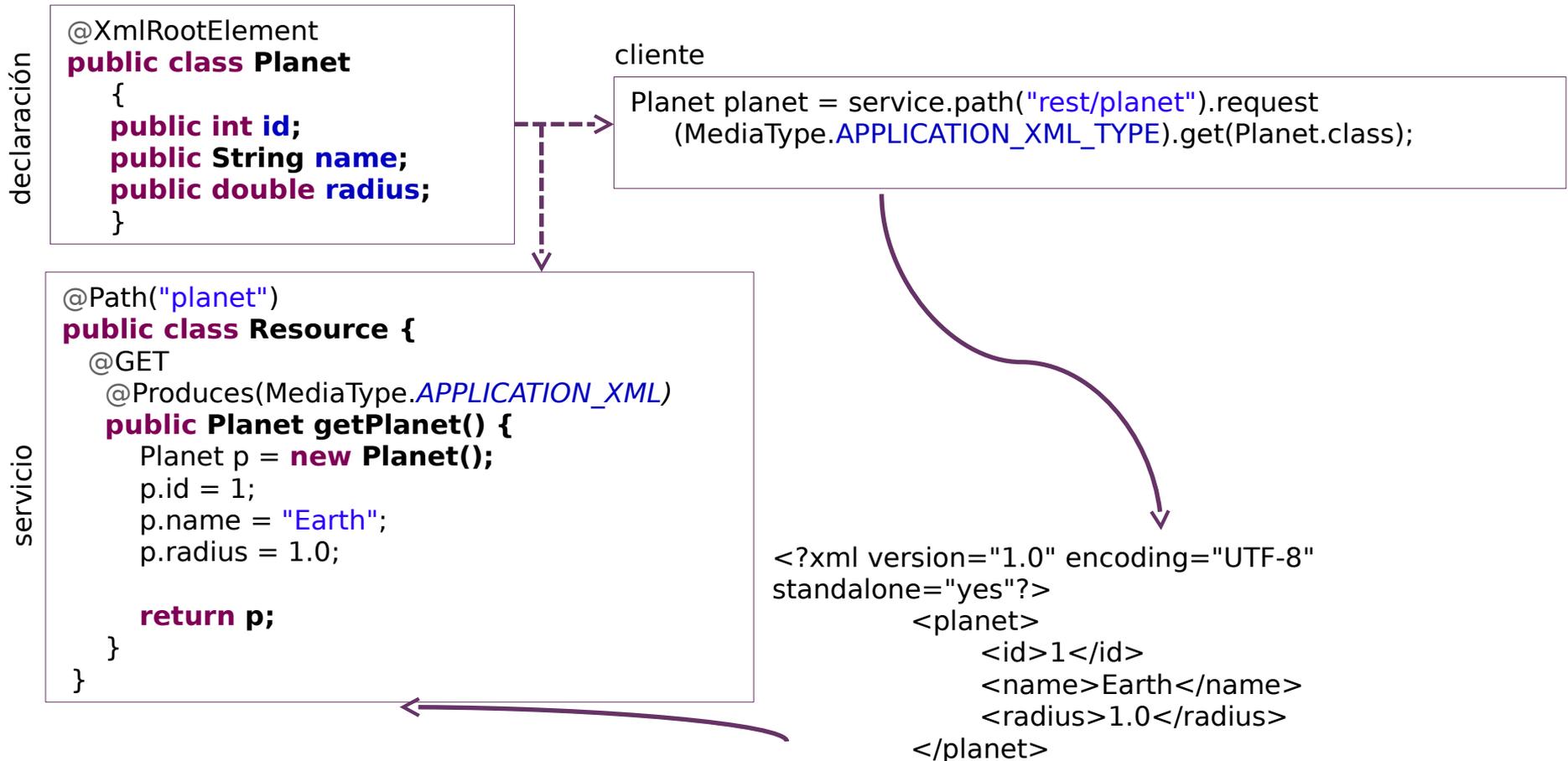
# + Retorno de objetos en Jersey

- No va a ser necesario para las prácticas, pero por si a alguien le interesa se explica en las siguientes diapositivas.
- En principio, Jersey retorna tipos MIME (es decir, texto, en distintos formatos)
  - Jersey no soporta la serialización de tipos primitivos
  - Debemos usar String + documentación de la API
  - Si intentamos, por ejemplo, retornar un long:
    - [com.sun.jersey.api.MessageException: A message body writer for Java class java.lang.Long, and Java type long, and MIME media type XXX was not found](#)
- Solución: convertir objetos Java en texto (p. ej. XML)
  - Jersey da soporte para ello a **JAXB**, una arquitectura para asociar clases Java a representaciones XML

# + Retorno de objetos en Jersey (ii)

## Retorno de objetos 'nuevos'

- Usamos @XmlRootElement + APPLICATION\_XML



# + Retorno de objetos en Jersey (iii)

## Retorno de POJOs

- Usamos la clase JAXBElement + APPLICATION\_XML

servicio

```
@Path("calendario")
public class Calendario {
 @GET
 @Produces(MediaType.APPLICATION_XML)
 public JAXBElement<Date> getDate()
 {
 Date p = new Date(System.currentTimeMillis());
 return new JAXBElement<Date>(new QName("date"), Date.class, p);
 }
}
```

cliente

```
GenericType<JAXBElement<Date>> dateType = new GenericType<JAXBElement<Date>>() {};
Date fecha = (Date) service.path("rest/calendario").request
 (MediaType.APPLICATION_XML_TYPE).get(dateType).getValue();
System.out.println("### " + fecha.getTime());
```

# + Obtener mi IP desde Java



- Podemos usar la clase `InetAddress`, con un socket:

```
DatagramSocket datagramSocket = new DatagramSocket();
datagramSocket.connect(InetAddress.getByName("8.8.8.8"), 12345);
```

```
String ip = datagramSocket.getLocalAddress().getHostAddress().toString();
```

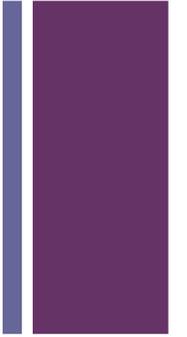
- Otra opción es ejecutar `ifconfig` del terminal desde Java con el método estático `Runtime.exec()`, aunque requiere parsear la salida para tomar la ip



# + Aportaciones



- Según se vayan identificando errores y su causa, cread un hilo en el foro de Studium con un título indicativo del error y en el cuerpo:
  - Stacktrace (si procede)
  - Causa
  - Solución



NEVER HAVE I FELT SO  
CLOSE TO ANOTHER SOUL  
AND YET SO HELPLESSLY ALONE  
AS WHEN I GOOGLE AN ERROR  
AND THERE'S ONE RESULT  
A THREAD BY SOMEONE  
WITH THE SAME PROBLEM  
AND NO ANSWER  
LAST POSTED TO IN 2003



<https://xkcd.com/979/>