



Deploy QueryManager on a Develop Environment (Eclipse Luna)

Deploy QueryManager on a Develop Environment (Eclipse Luna).....	1
Prerequisites	1
Repositories	1
Deploy	2
Apidocs	2
Web Sample Client	4
Query Manager	4
Errors after deploy	6
DataBase	8
First Run	8
Auth Operations, Bypass and SPARQL Queries API	10
Google Access Token & 3Cixty Access Token	10
Web Sample Client & Bypass	11
SPARQL Queries APIs	12

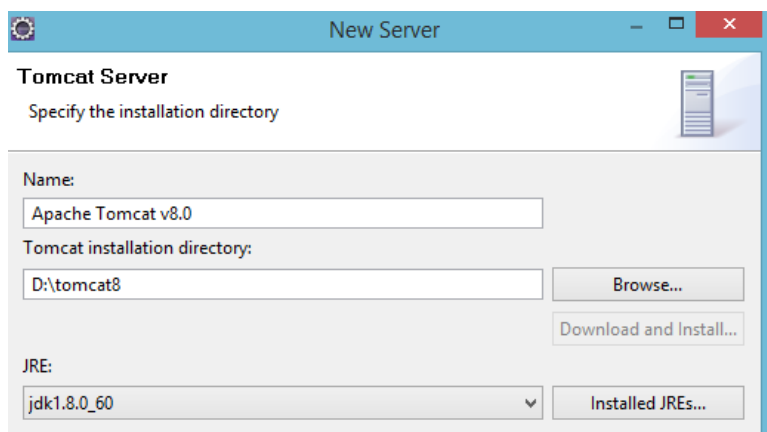
Prerequisites

- Eclipse Luna
- Java 8
- Tomcat 8 running in port 8080
- MySQL

Note: we need to use a JDK not JRE

Valid			Non Valid		
Name	Location	Type	Name	Location	Type
<input checked="" type="checkbox"/>  jdk1.8.0_6...	C:\Java\jdk1.8.0...	Standard ...	<input checked="" type="checkbox"/>  jre1.8.0_60...	C:\Java\jre1.8.0...	Standard ...

Note: tomcat configuration on eclipse



Repositories

We need three repositories:

- <https://github.com/3cixty/web-client-sample>
- <https://github.com/3cixty/apidocs>
- <https://github.com/3cixty/querymanager>

We need to download the content of these "repos" in our machine.

Deploy

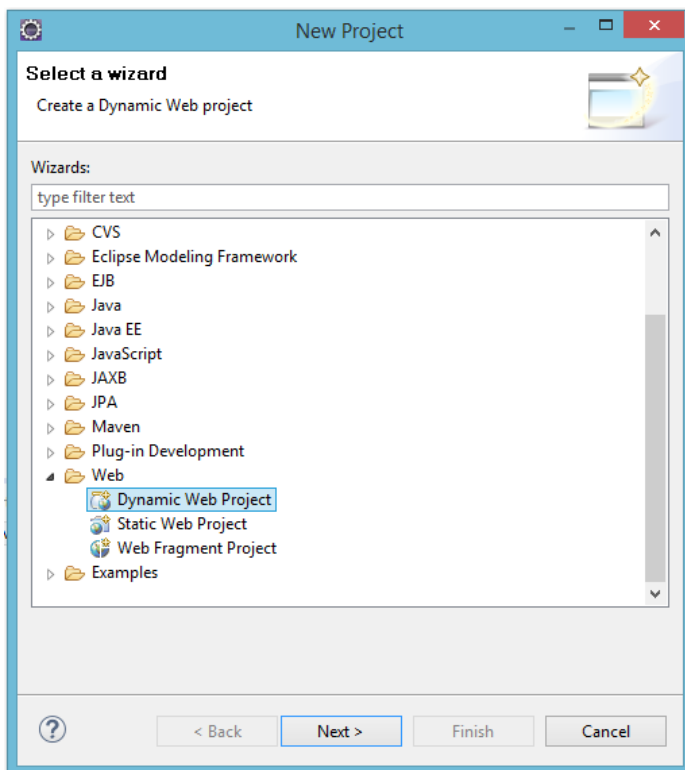
"Web client sample" and "Apidocs" are two html5 apps. We copy the content of this apps in our environment.

About query manager, we must deploy two applications that are inside on our tomcat.

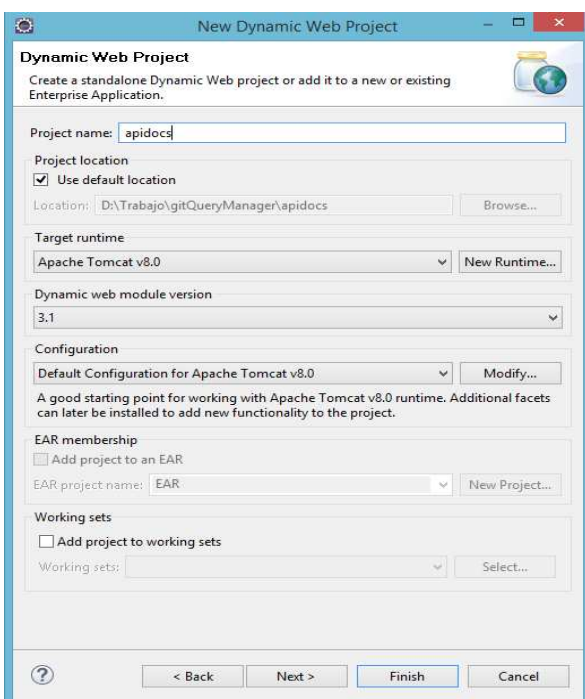
We can deploy all on Tomcat follow the next steps.

Apidocs

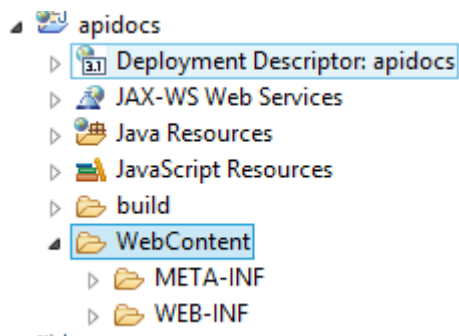
New Project -> Dynamic Web Project



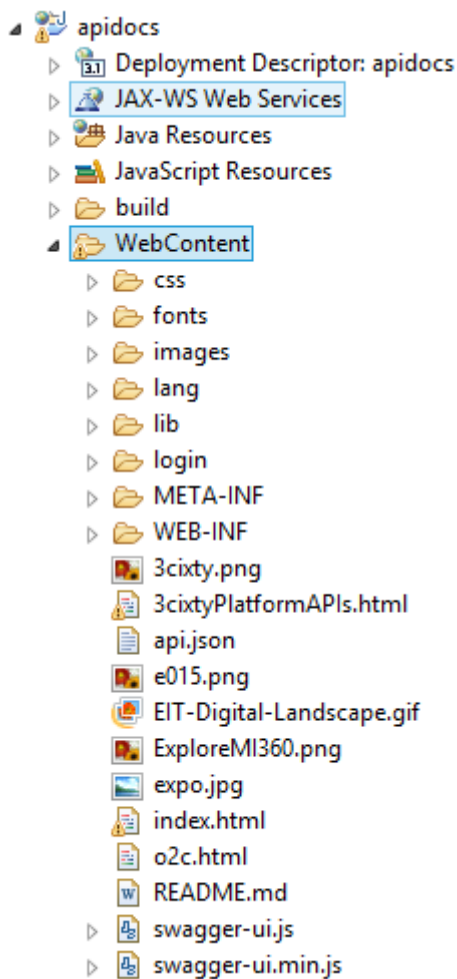
Write "apidocs" in the Project name and push the button "Finish"



This is the project structure:



Paste inside WebContent all the files:



After these you need to follow these steps:

In the file api.json cambiar "host" y "schemes":

```
"host": "localhost:8080",  
"basePath": "/v2",  
"schemes": [  
    "http"  
]
```

In index.html at linea 36:

```
url = "http://localhost:8080/apidocs/api.json";
```

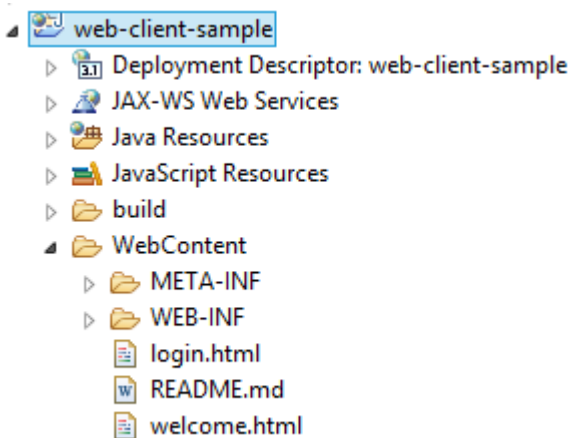
Then in `swagger-ui.js` we must change the url of this function:

```
showPetStore: function(){
    this.trigger('update-swagger-ui', {
        url: 'http://localhost:8080/apidocs/api.json'
    });
},
```

Web Sample Client

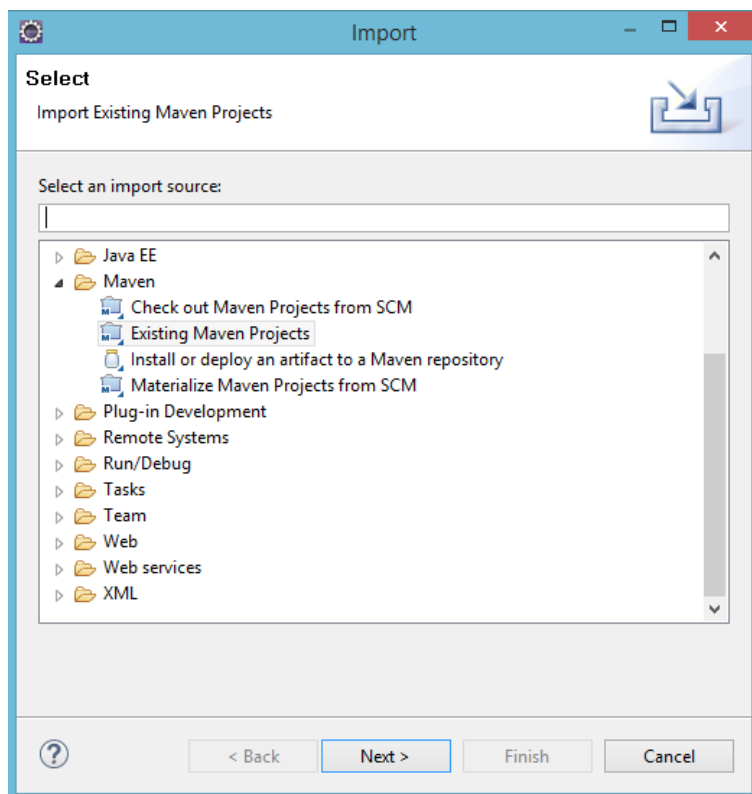
With this project you need to do these steps (similar of apidocs):

- New Project -> Dynamic Web Project
- Write " web-client-sample" in the Project name and push the button "Finish"
- Paste inside WebContent all the files.

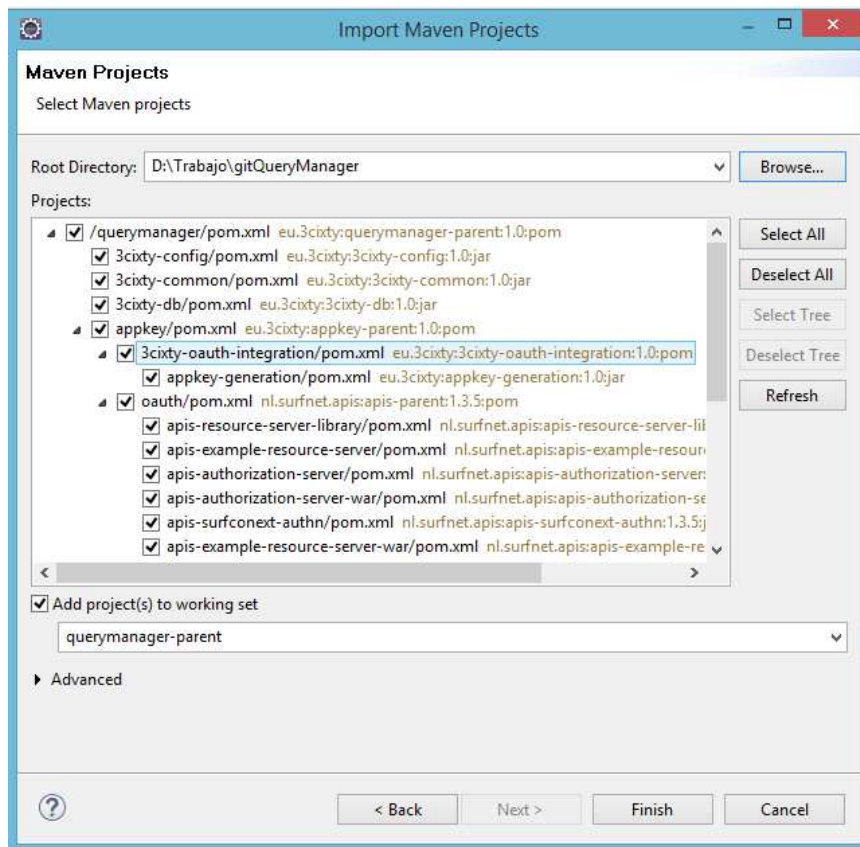


Query Manager

File -> Import -> Existing Maven Projects

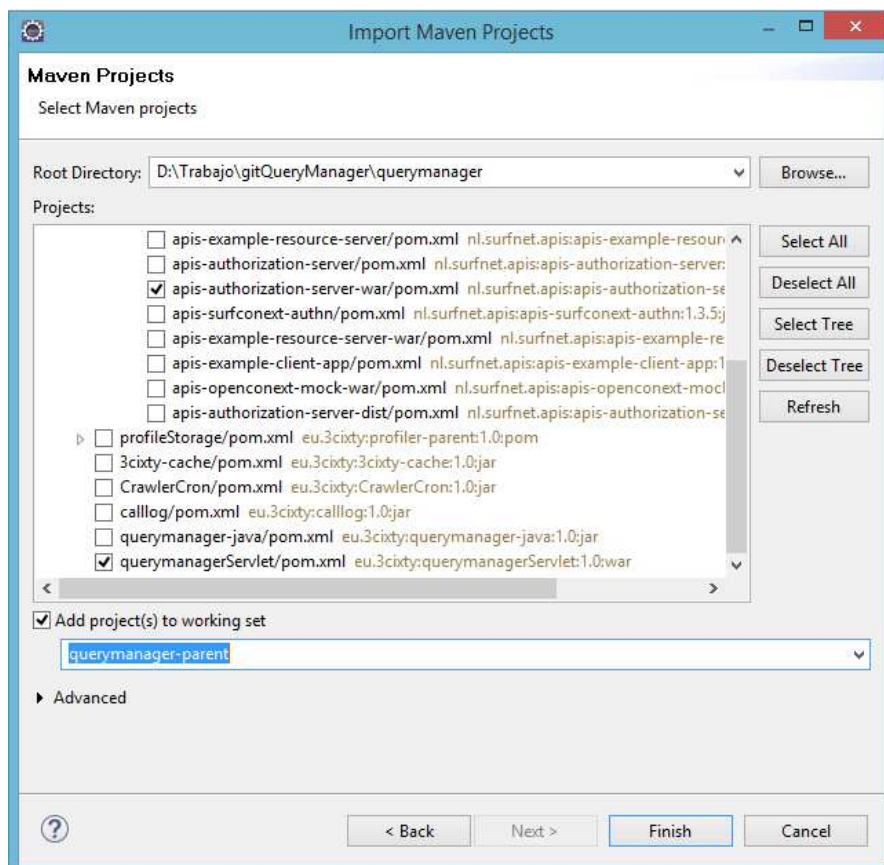


Press button "Next >" and in the next screen select in "Root Directory" the QueryManager repositories source:



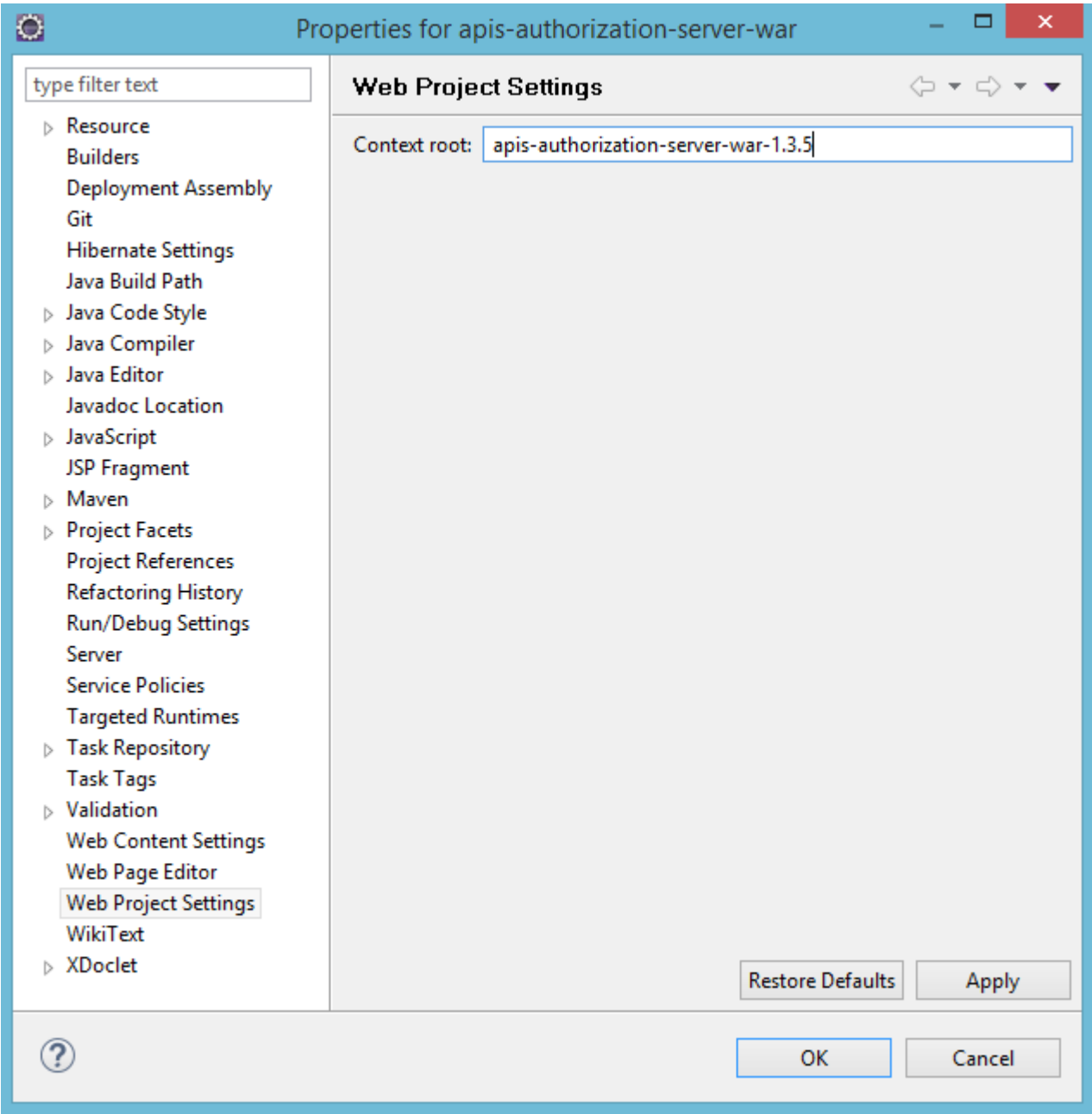
Then "Deselect All" and choose manually the projects:

- apis-authorization-server-war
- querymanager-parent



Note: you must go to the properties of "apis-authorization-server-war" and change the context to:

apis-authorization-server-war-1.3.5



Errors after deploy

In Query Manager we have some error that I fixed follow this steps:

Maven Java EE Configuration Problem (2 items)		
Dynamic Web Module 3.1 requires Java 1.7 or newer.	querymanagerServlet	
One or more constraints have not been satisfied.	querymanagerServlet	
Maven Problems (2 items)		
XML Problem (5 items)		
cvc-complex-type.2.4.d: Invalid content was found starting with element 'listener-class	web.xml	/querymanagerServlet/WebContent/WE...
No grammar constraints (DTD or XML Schema) referenced in the document.	apis-logback.xml	/apis-authorization-server-war/src/main...
No grammar constraints (DTD or XML Schema) referenced in the document.	apis-logback.xml	/apis-authorization-server-war/src/test/r...
No grammar constraints (DTD or XML Schema) referenced in the document.	logback.xml	/apis-authorization-server-war/src/main...
No grammar constraints (DTD or XML Schema) referenced in the document.	logback.xml	/apis-authorization-server-war/src/test/r...

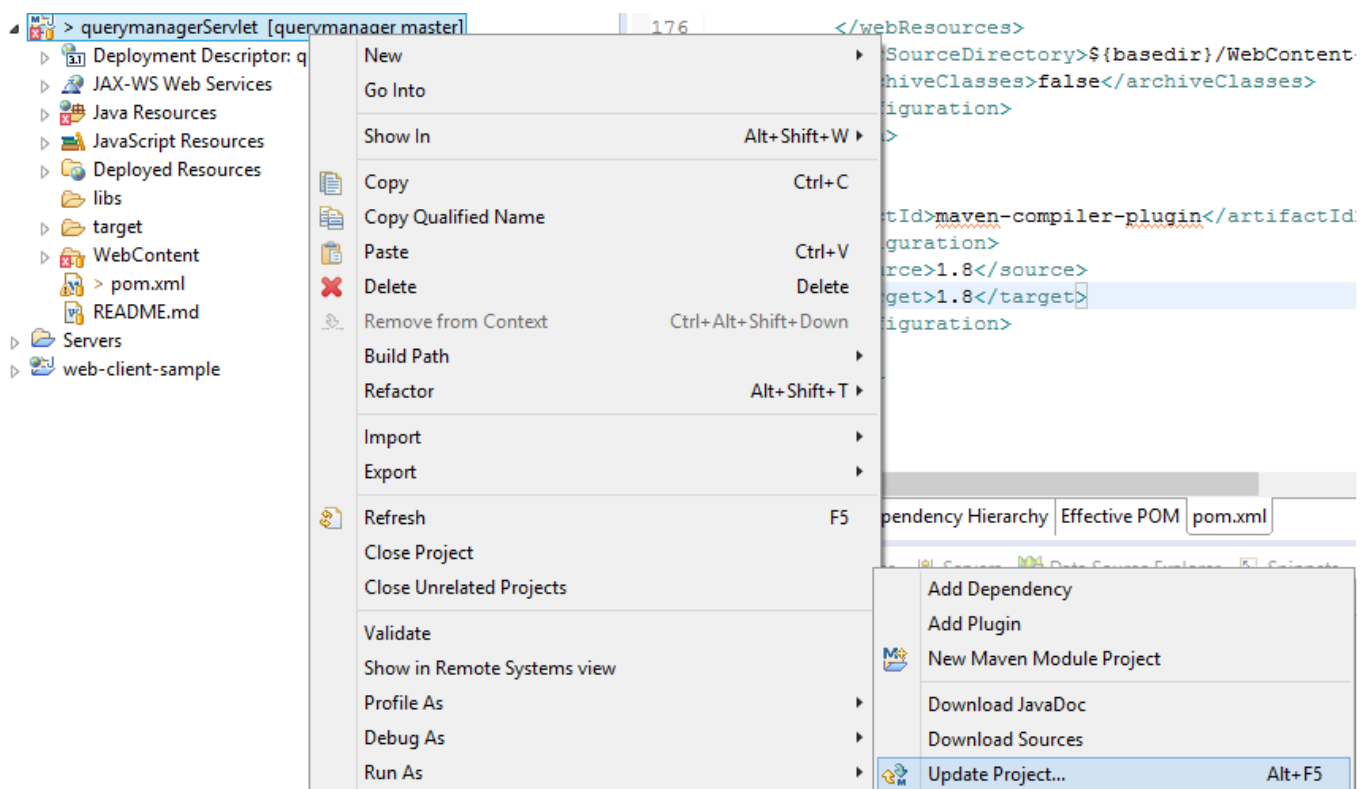
1. Go to the Pom File and change the version 1.6 to 1.8 on this lines:

```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <source>1.6</source>
    <target>1.6</target>
  </configuration>
</plugin>
```

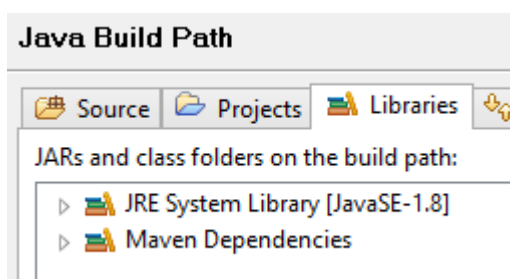
To

```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
  </configuration>
</plugin>
```

Then update the Maven Project:



2. Check if your Java Library is correct in project properties



3. Comment the line 13 in web.xml

```
10
11  <listener>
12    <listener-class>eu.threecixty.querymanager.ThreeCixtyContextListener</listener-class>
13    <listener-class>net.bull.javamelody.SessionListener</listener-class>
14  </listener>
15
```

To

```
<listener>
  <listener-class>eu.threecixty.querymanager.ThreeCixtyContextListener</listener-class>
  <!--
  <listener-class>net.bull.javamelody.SessionListener</listener-class>
  -->
</listener>
```

DataBase

We need to init a MySQL Schema before run the APPS.

You can init with this command in a Terminal:

```
mysql -u root -p
CREATE USER '3cixty'@'localhost' IDENTIFIED BY '3cixtydatabase001';
create database 3cixty;
GRANT ALL ON 3cixty.* TO '3cixty'@'localhost';
create database 3cixtyTest;
GRANT ALL ON 3cixtyTest.* TO '3cixty'@'localhost';
exit;
```

Note: be carefull, if you are in another machine you can't connect using this user. In this case you must use these commands:

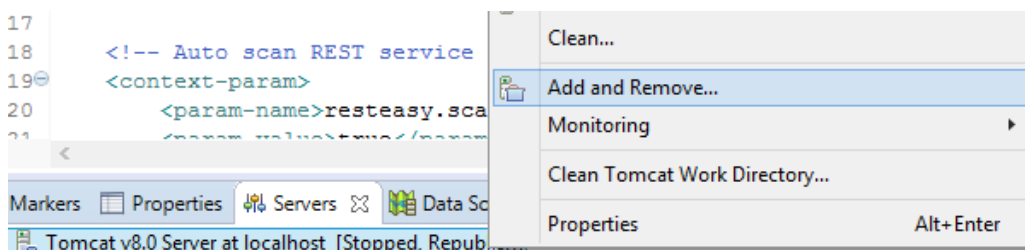
```
CREATE USER '3cixty'@" IDENTIFIED BY '3cixtydatabase001';
CREATE DATABASE 3cixty;
GRANT ALL ON 3cixty.* TO '3cixty'@";
CREATE DATABASE 3cixtyTest;
GRANT ALL ON 3cixtyTest.* TO '3cixty'@";
```

Note: if you need delete all, you can use:

```
DROP USER '3cixty'@";
DROP DATABASE 3cixty;
DROP DATABASE 3cixtyTest;
```


First Run

In Eclipse over Tomcat -> Add and Remove...






Choose only "apis-authorization-server-war".

In Tomcat Web Modules you MUST Verify that the path is "apis-authorization-server-war-1.3.5"

 **Web Modules**

Web Modules
Configure the Web Modules on this server.

Path	Document Base
 /apidocs	apidocs
 /web-client-sample	web-client-sample
 /apis-authorization-server-war-1.3.5	apis-authorization-server-war

Overview **Modules**

If doesn't change try to restart Eclipse.

And Run Tomcat.


In the Console you must see these messages:


```
13:37:02.736 [localhost-startStop-1] INFO    openjpa.Runtime - Starting OpenJPA 2.2.0
13:37:03.548 [localhost-startStop-1] INFO    c.g.f.c.m.MetadataTableImpl - Creating
Metadata table: `3cixty`.`schema_version`
13:37:04.007 [localhost-startStop-1] INFO    c.g.flyway.core.command.DbMigrate -
Current version of schema `3cixty`: << Empty Schema >>
13:37:04.007 [localhost-startStop-1] INFO    c.g.flyway.core.command.DbMigrate -
Migrating schema `3cixty` to version 0
13:37:04.179 [localhost-startStop-1] INFO    c.g.flyway.core.command.DbMigrate -
Successfully applied 1 migration to schema `3cixty` (execution time 00:00.635s).
13:37:04.236 [pool-3-thread-1] DEBUG org.surfnet.oaaas.support.Cleaner - Cleaning up
expired access tokens
13:37:04.243 [localhost-startStop-1] INFO    o.s.web.context.ContextLoader - Root
WebApplicationContext: initialization completed in 12457 ms
```



And now, if you check your Database now you must see the tables.



Again got to Eclipse over Tomcat -> Add and Remove...


And choose Add All.

 **Tomcat v8.0 Server at localhost [Stopped, Synchronized]**

 apidocs [Synchronized]

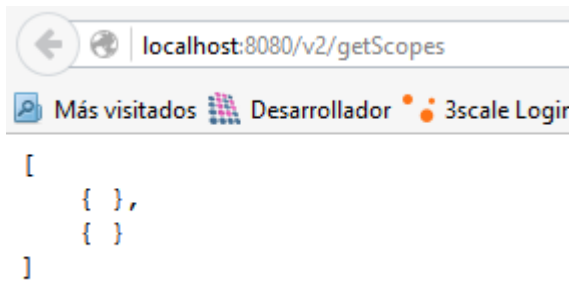
  apis-authorization-server-war [Synchronized]

  querymanagerServlet [Synchronized]

 web-client-sample [Synchronized]

Then Run it again.

And to test you can check this URI: <http://localhost:8080/v2/getScopes>

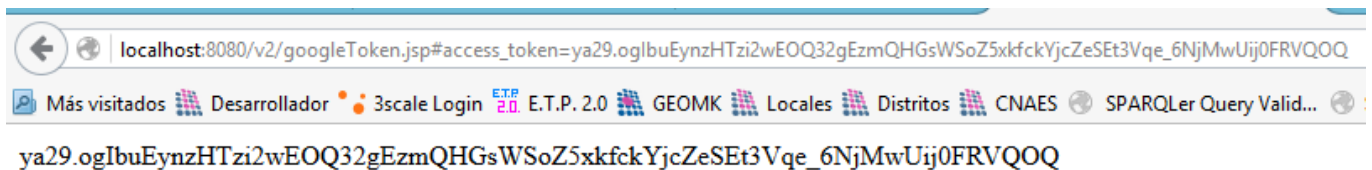
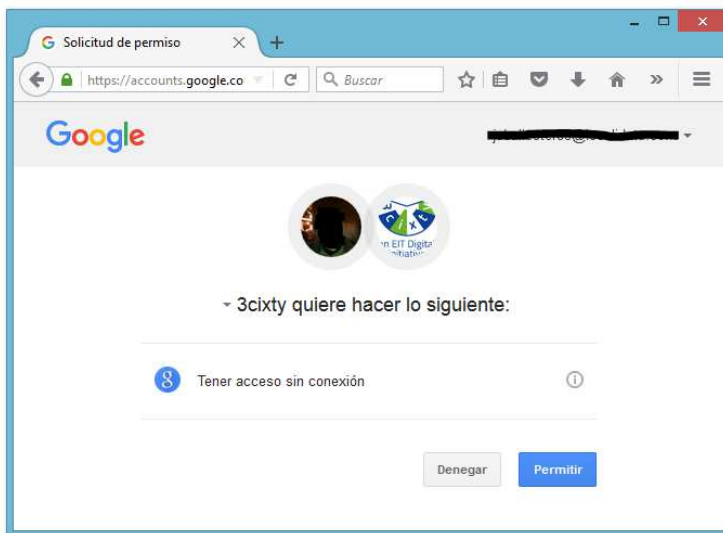
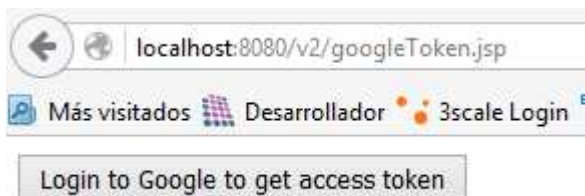


Auth Operations, Bypass and SPARQL Queries API

Google Access Token & 3Cixty Access Token

We need to generate our Google Access Token. We use this URI to do that:

<http://localhost:8080/v2/googleToken.jsp>



Now go to the "apidoc" app: <http://localhost:8080/apidocs/>

Go to the api: /getAppKey in Oauth API Section

GET /getAppKey

Get/generate 3cixty App key

And test it with your Google Access Token.

Fill the mandatory fields.

In Scopes put: Profile, Wishlist

And then check the API with the button "Try it out".

```
{
  "key": "3fda9615-1475-4ba6-8e88-159f55a38d8b"
}
```

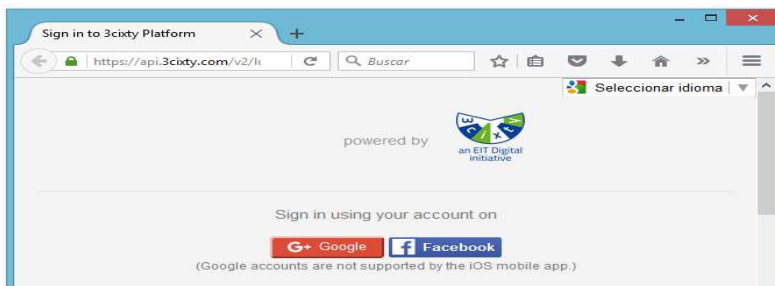
Then you must have your 3Cixty Access Token.

Web Sample Client & Bypass

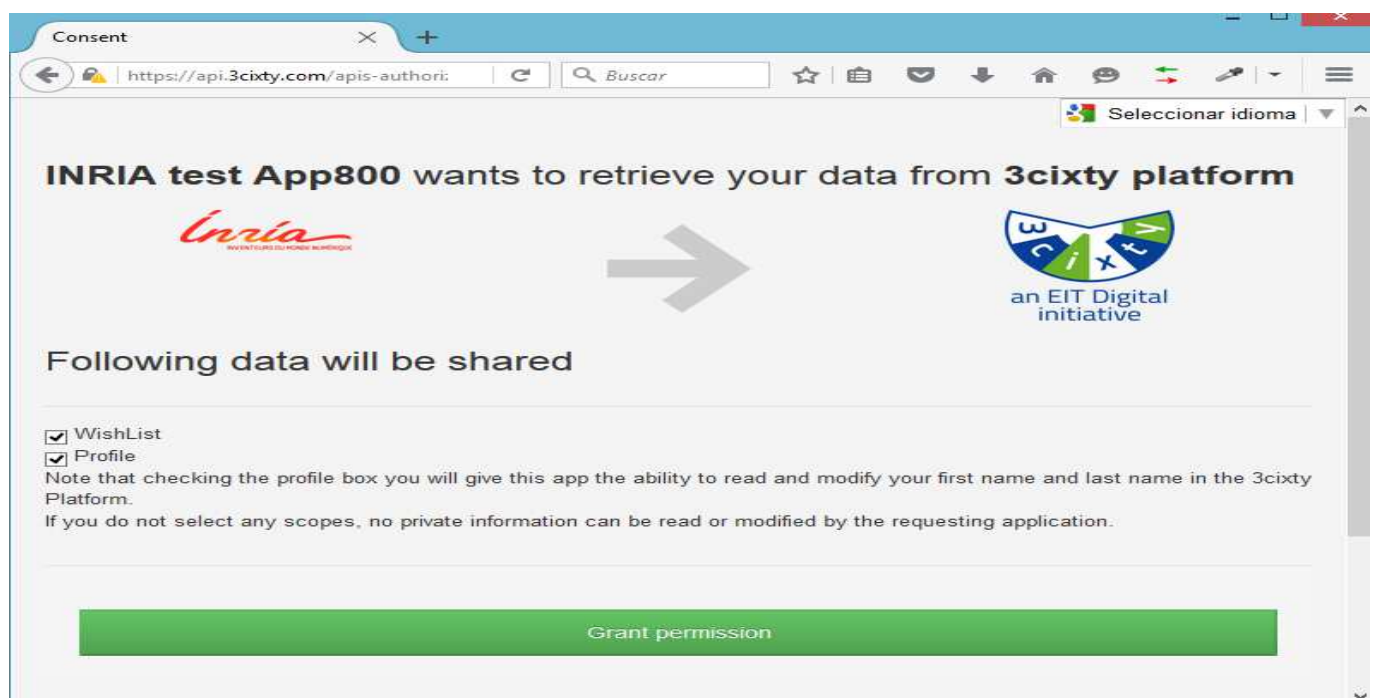
Access to this URI: <http://localhost:8080/web-client-sample/login.html> and push the button.

Log in to Pizza4ever!!

Login using 3cixty



Press the G+ Button



You can see the Scopes that you write in the previous steps.

Press the big button.

Now we configure the app to avoid the Permission step:

Edit the file `authBypass.list` in `queryManagerServlet WEB-INF`

Add our 3cixty key: `3fda9615-1475-4ba6-8e88-159f55a38d8b`

```
1 6f0b60ec-2ef5-4ae5-9d90-3e94be8227ea
2 fb8a79c0-b2b3-4147-82b7-14099cd6f4c9
3 bb669013-a57f-4e77-8e53-fab1aefe9756
4 39d9908f-eb91-43b1-a56c-6f7c0499788d
5 a1c85c6b-a42a-40bc-929d-f71bafaaa482
6 ef15ff3c-a8d4-4991-9e56-57f1ce683adc
7 4bb5323e-8c09-4a5d-b147-64b7dc69fcd2
8 01f99cd9-5c90-4d09-9812-9c8a1a5ffd67
9 f128dd83-1ecc-4875-91da-3fe6650b53fa
10 6f0b60ec-2ef5-4ae5-9d90-3e94be8227eb
11 3fda9615-1475-4ba6-8e88-159f55a38d8b
12
```

Restart Tomcat and try Again: `http://localhost:8080/web-client-sample/login.html`

Now we jump the permission page.

SPARQL Queries APIs

To run these APIs:

QueryManager/Augmentation API - REST API

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

QueryManager/Augmentation API - Rich REST API

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

We need a 3cixty App Access Token (is not the same as 3Cixty App Key). To obtain it you need to run this API:

GET `/getAccessToken`

[Get 3cixty access_token from Google access_token](#)

Implementation Notes

This API is used to get 3cixty access_token from Google access_token. **This API is only used by trusted 3cixty apps.** Let us now describe the parameters required

And you need your Google access token and your 3Cixty App key (`3fda9615-1475-4ba6-8e88-159f55a38d8b`)

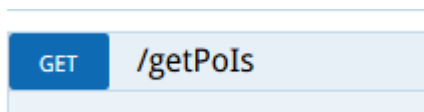
Parameters

Parameter	Value	Description	Parameter Type	Data Type
google_access_token	<code>osffdqFRvZ5H-fHY_CVGEQjznüfHmgJV9ecDApQ</code>	A Google access_token.	header	string
key	<code>3fda9615-1475-4ba6-8e88-159f55a38d8b</code>	A 3cixty App key.	header	string
scope	<input type="text"/>	list of 3cixty scopes in a comma separated syntax.	header	string

```
{
  "access_token": "b4782725-68ac-409a-b675-d69a45fa471f",
  "refresh_token": "2729eef2-2412-4d65-bf9c-6349e9d46392",
  "scopeNames": [],
  "used": false,
  "token_type": "Bearer",
  "expires_in": 86400
}
```

Now we have our access Token!!!

Now we can try a SPARQL API:



I write my access token and select Milan city and try...

And I have an Connection refused: connect

That is because the SPARQL Endpoint is well not configured. You can change in the file 3Cixty.properties in the WEB-INF folder.

By default we have: http://localhost:8890, if we don't have a Virtuoso Server in our local environment we need to target to another. For example: http://91.250.81.138:8890/sparql

```
24 # Note that no need to use https for Virtuoso as Virtuoso should be a private IP address
25 VIRTUOSO_SERVER_LOCAL = http://91.250.81.138:8890
26 VIRTUOSO_SERVER_FOR_OUTSIDES_LOCAL = http://91.250.81.138:8890
27 VIRTUOSO_JDBC_LOCAL = jdbc:virtuoso://91.250.81.138:1111/
```

Now restart Tomcat and Try Again:



If you want to debug the query you can go to the class (QuerymanagerServlet):

```
eu.threecixty.querymanager.rest.QueryManagerServices
```

```
function getPoIs
```

You can see where is building that query:

```
680     AccessToken userAccessToken = OAuthWrappers.findAccessTokenFromDB(access_token);
681     if (userAccessToken != null && OAuthWrappers.validateUserAccessToken(access_token)) {
682         String user_id = userAccessToken.getUid();
683         String key = userAccessToken.getAppkey();
684
685         String query = createSelectSparqlQueryForPoI(offset, limit, category, minRating, maxRating, key, city);
686     }
```