Docs: semantic web project

Code, scripts..

All the code and scripts are in my github, in this repository. Please Clone the project in your computer to find all the sources code's components.

Vocabularies

The vocabularies we require for the project are as following:

code_postale	code postale
commune	Example: nantes
name	nom de la station
address	adresse indicative de la station
position	[-] Object, 2 properties **lat** latittude **lng** longitude
banking	false ou true
bonus	indique s'il s'agit d'une station bonus
bike_stands	nombre total des quais
available_bike_stands	quais disponibles
available_bikes	velos disponibles
status	indique l'état de la station, peut être CLOSED ou OPEN
last_update	timestamp indiquant le moment de la dernière mise à jour

In addition to that, we generate another vocabulary item `(ID)` to identify each station. this ID should be unique, so we take the name of the station (clean from special characters) concatenated with its ZIP code.

Turle file example

Here is an example (few N-triples) of the turtle file that we generate from data coming from the different data sources (APIs):

```
@prefix ex: <http://example.org/#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
```

```
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rel: <http://relations.example.com/>.

ex:Chateaucreux42000 a geo:SpatialThing;
rdfs:label "Chateaucreux"@fr;
geo:lat "45.442972"^^xsd:decimal;
geo:lon "4.398855"^^xsd:decimal;
rel:STATUS "OPEN";
rel:ADDRESS "101 Châteaucreux "^^xsd:langString;
rel:FREE_BIKES "23"^^xsd:integer;
rel:EMPTY_SLOTS "9"^^xsd:integer;
rel:TOTAL_SLOTS "32"^^xsd:integer;
rel:LAST_UPDATE "1572010093"^^xsd:integer;
rel:CB_PAYMENT "0"^^xsd:integer;
rel:ZIP_CODE "42000"^^xsd:integer;
rel:COMMUNE "Saint-Etienne"@fr.
```

> Note: you can find an extended turtle file called `Project_data2.ttl` inside the `server` folder.

RDFa and Json-ld

As said in the presentation, the data is encoded and embedded as RDFa and Json-ld in all the pages of each specific station. In order to find it in the code, and since the pages are client rendered (in the front end):

1- please tap `F12` to inspect the page element 2- copy the `html` tag (**the first element of the page**) 3- paste it in your RDFa or Json-ld provider. (I use RDFa or Json-ld)

Install Instruction

Requirements:

1- Jena fuzeki server **running** on default port (3030). 2- Nodejs Installed (how to install Nodejs)

The project is a full stack application, so there is two component:

Back end (server)

The back end has been created using `Nodejs`. This part's code is in the `server` folder, this latter contains 2 important files:

- **index.js**: this is the file that contains the Nodejs code (main) of the API
- **package.json**: this file contains *the dependencies* as well as some additional information. (description, test script,keywords...)

In order to get the setup of the project:

1. Make sure you are in the directory `Bicycle-Sharing-system-SemanticWeb/server/`.
2. Run the following command: `npm install` and Done ! In order to run the application, run this command: `node index.js`

There is 4 .js files in the same folder:

- **lyon-data.js, saint-etienne-data.js** : code for creating lyon and saint-etienne cities.
- **newCity-data.js**: code for creating any city available in the Jcdeceaux API.
- **createCity-data.js**: code for creating any *new* city manually.

Front end (client)

The Front end has been created in`Vue.js`. This part's code is in the `client/myclient` folder, this folder contains the views, the components created, and all the files related to the front end of the project.

In order to get the setup of the project:

1. Make sure you are in the directory `Bicycle-Sharing-system-SemanticWeb/client/myclient`.
2. Run the following command: `npm install` and Done ! In order to run the application, run this command: `npm run serve`.

Finally, go to the web page using this URL: http://localhost:8080/