Laboratoire 5

REST

Plan

Labo 5 – But :

Vous familiariser avec le style architectural **REST**.

Ce que vous devez réaliser :

Réaliser une application ‘pense-bête’ utilisant une API de type REST.

➔L’API sera fournie, mais vous devrez implémenter le côté client !

➔L’API est codée avec Flask (en Python)
voir : http://flask.pocoo.org/

Laboratoire

Procédure d'installation :

1. Installer Python 2.7.5 (et non pas 3!)

   Si vous êtes sur Windows, ajustez vos variables d'environnement.

2. Installer pip et Flask (*pip install flask*)
   -> http://stackoverflow.com/questions/4750806/how-to-install-pip-on-windows

3. Roulez l'application fournie en ligne de commande (app.py)

4. Elle sera disponible à l'adresse suivante : http://localhost:5000/

Laboratoire

Description de l'API fournie :

1 – à **/tasks**

➔ Retourne toutes les tâches actuelles. (GET)

2 – à **/tasks**

➔ Permet de créer une nouvelle tâche. (POST)

Laboratoire

Description de l'API fournie :

3 – à **/tasks /<id>**

➔ Permet de modifier la tâche associée à l'id. (PUT)
(Remplace les valeurs de la tâche par la nouvelle passée dans la requête)

4 – à **/tasks /<id>**

➔ Permet de supprimer la tâche associée à l'id. (DELETE)

Laboratoire

Consignes :

Concevoir une application web dynamique utilisant toutes les commandes de l'API.

➔ Utilisez jQuery abondamment !

➔ Pensez à concevoir un UI facile à utiliser.

➔ REST Client (plugin Chrome/Firefox) permet de tester facilement les requêtes de type autre que GET.

https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo

Laboratoire

Remarques sur les requêtes

Prenez un certain temps pour regarder les requêtes qui passent dans votre console... vous pourrez admirer la magie du HTTP!

```
127.0.0.1 - - [08/Oct/2013 18:54:57] "POST /tasks HTTP/1.1" 201 -
127.0.0.1 - - [08/Oct/2013 18:55:07] "GET /tasks/0?__debugger__=yes&cmd=resource
&f=style.css HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2013 18:55:07] "GET /tasks/0?__debugger__=yes&cmd=resource
&f=jquery.js HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2013 18:55:07] "GET /tasks/0?__debugger__=yes&cmd=resource
&f=debugger.js HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2013 18:55:08] "GET /tasks/0?__debugger__=yes&cmd=resource
&f=ubuntu.ttf HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2013 18:55:08] "GET /tasks/0?__debugger__=yes&cmd=resource
&f=console.png HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2013 18:55:08] "GET /tasks/0?__debugger__=yes&cmd=resource
&f=source.png HTTP/1.1" 200 -
```

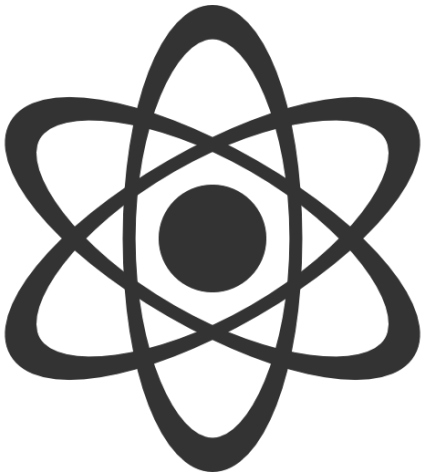
À REMETTRE

Zip contenant :

- Fichiers noms.txt avec vos noms/matricules
- Votre code (seulement front-end)
- Instructions si nécessaires

POINTS ALLOUÉS

- Ajout, Édition, Suppression de tâches
- Interfaçe dynamique (pas de rechargement de page!)
- Implémentation des **toutes** les méthodes
  GET tasks GET tasks/:id, POST, PUT, DELETE
- Utilisation des méthodes ajax de jquery ($.get, $.post et $.ajax)
- Vous avez droit d'utiliser Bootstrap ou autres librairies CSS



N’hésitez pas à poser

vos questions ☺

*P.-S. Si vous avez des problèmes d’**encodage** avec Flask, assurez-vous de rouler l’application d’un endroit dons le **path** ne comprend pas d’ **accents**!*