Examen Web Service

Année 2014/2015

Durée : 1h30 (hors préparation)
Documents autorisés (cours : **`http://www.shipstone.org/siad/m2.pdf`**)

Préparation

1. Lancer votre machine virtuelle
2. récupérer le code source de l'examen : https://github.com/ptitbob/siad-m2-2014-rattrapage
   code à executer : `git clone https://github.com/ptitbob/siad-m2-2014-rattrapage.git`
3. placer vous dans le répertoire `siad-m2-2014-rattrapage` et executer la commande suivante : `mvn clean install` ne vous inquiétez pas, il y a des erreurs, c'est voulu ;)
4. ouvrez le projet avec netbeans
5. *Pour cette partie, je pourrai vous aider.*

Partie 1 - Serialisation XML

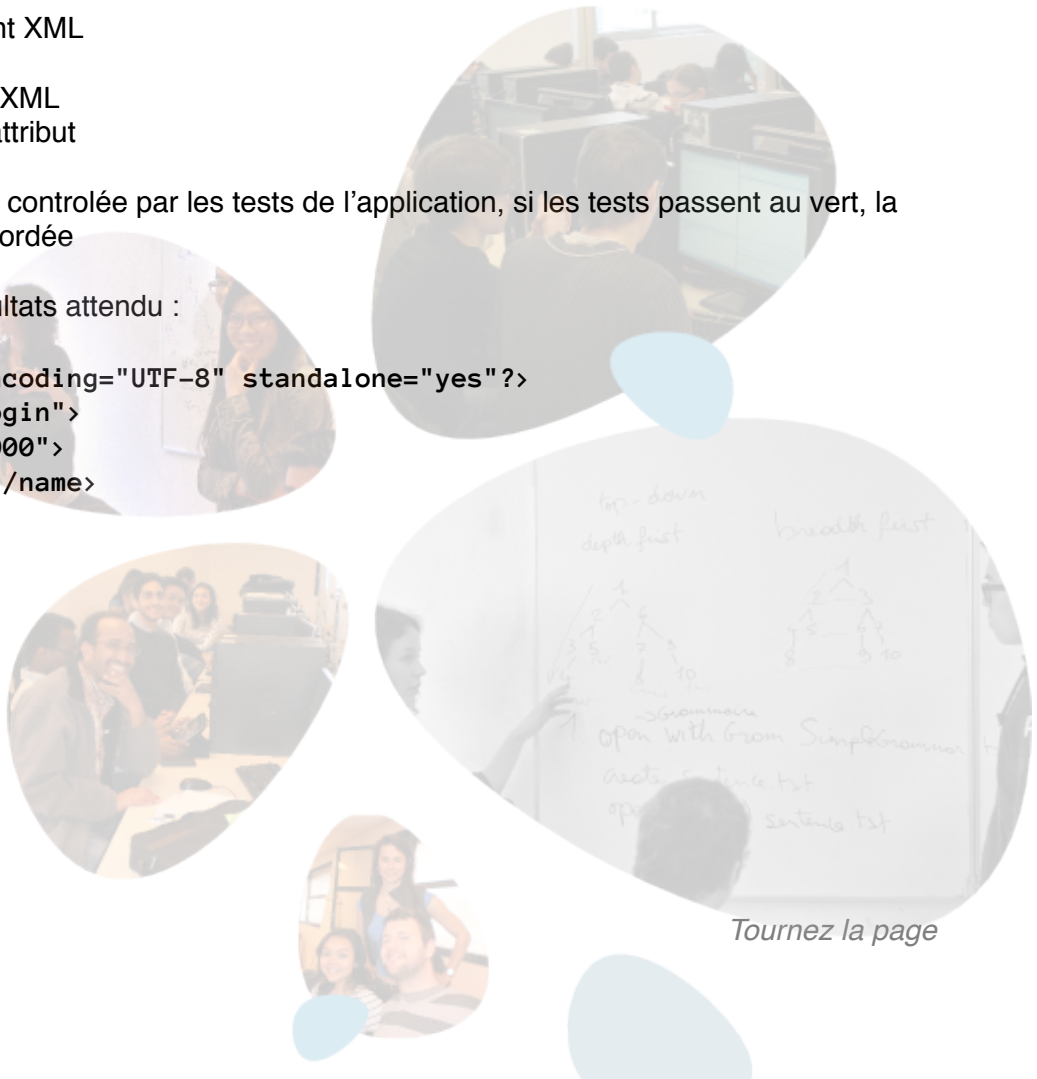
Modifiez les classes **`fr.univ.blois.siad.m2.todolist.model.User`** et **`fr.univ.blois.siad.m2.todolist.model.Town`** afin qu'elles répondent aux contraintes suivantes :

- Classe User
  - Login est un attribut XML
  - Id est un attribut XML
    - La ville est un élément XML
- Classe Town
  - La ville est un élément XML
  - Le code postal est un attribut

Cette serialisation XML est controlée par les tests de l'application, si les tests passent au vert, la totalité des points sera accordée

Pour exemple, voici le résultats attendu :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user id="1" login="login">
    <town zipCode="45000">
        <name>Orléans</name>
    </town>
</user>
```



Tournez la page

Partie 2 - Exposition d'API REST

Il est important de faire les points dans l'ordre pour vous assurer le maximum de points ;)

Le but de cette partie de l'exercice est de mettre en place une API REST exposant une API CRUD-List concernant les utilisateurs.

Pour cela, il vous faudra amender la classe **`fr.univ.blois.siad.m2.todolist.ws.rs.UserAPI`**, vous trouverez des indications qui rappèleront les points suivant dans le code source:

1. Annotez la classe pour faire en sorte que l'URL d'accès de base soir : **`http://localhost:8080/todo/api/user`**
   Prenez en note que l'application REST (décrite dans la classe **`fr.univ.blois.siad.m2.todolist.ToDoListeApplication`**, laquelle est annoté pour que l'URL de base des API soit : **`http://localhost:8080/todo/api`**)
   De plus, toutes les méthode REST qui seront exposées par la classe UserAPI doivent renvoyer des donnée serials sous forme JSON et XML selon le type de contenu demandé lors de négociation de contenu lancé par le client.
2. Faire en sorte que la méthode **`getUserList`** réponde à l'URL **`http://localhost:8080/todo/api/user`** et renvoi la liste des utilisateurs.
3. Créer une méthode (annotée pour exposer une API REST) qui renvoi un utilisateur en fonction de son id. ID qui sera partie intégrante de l'IRL (exemple : **`http://localhost:8080/todo/api/user/1`** renverra l'utilisateur avec un identifiant 1)
4. Créer une méthode permettant d'effacer un identifiant dont l'Id fera partie intégrante de l'URL, comme au point 3.
5. Créer une méthode qui permettra de renvoyer tous les utilisateurs d'une ville en fournissant le code postal comme paramètre d'une requête URL (http://localhost:8080/todo/api/user/zip?zipcode=41000)
6. Ecrivez une méthode permettant de créer un **`User`** (contenu dans le body de la requête
   Pour, par exemple répondre à une commande de ce type : **`curl -H "Content-Type: application/json" -X POST -d '{"login":"fifi","town":{"name":"Blois","zipCode":"41000"}}' http://localhost:8080/todo/api/user`**
   Rappelez vous bien des norme HTTP ;)
7. Ecrire une méthode permettant de modifier un User avec ce qui sera contenu dans le body de la requête (pour trouvez l'inspiration regarder la classe de service)
8. Point bonus, faire en sorte que l'exception **`NotFoundEntityException`** renvoi un code retour 499

Attention, vous vous devez d'utiliser tous les verbes HTTP que vous jugerez utile pour exposer l'API REST.