IUT Lyon 1
l'excellence technologique

Institut Universitaire de Technologie
Département Informatique
Site de Bourg-en-Bresse

Utilisation des Web Services RestFull avec Android

LP Metinet - Janvier 2016
lionel.buathier@univ-lyon1.fr

Planning prévisionnel

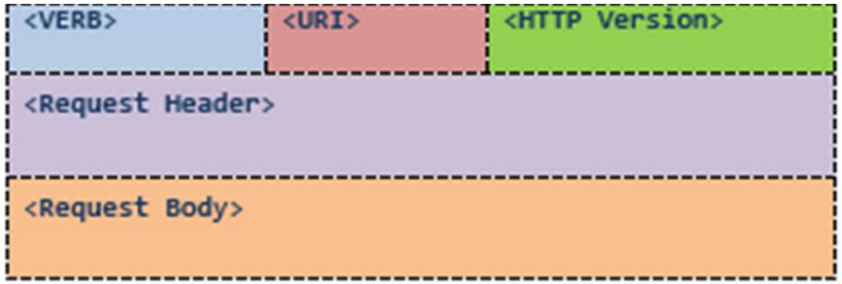
- Introduction aux WebServices
- Connexion distante - AsyncTask : 4h
- Parsage Json et affichage personnalisé : 4h
- Amélioration de l'API : 4h
- Évaluation sur le projet

Introduction aux WebServices RestFul

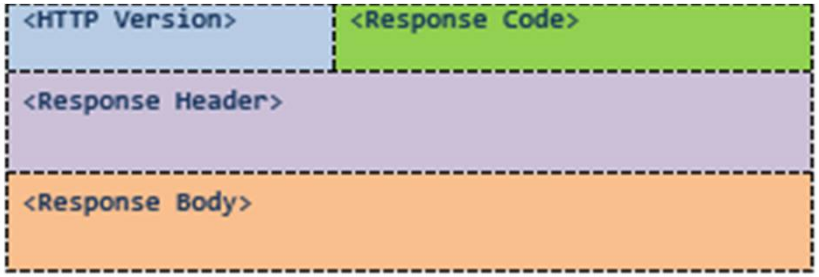
- REST est une architecture de services Web, à la manière de SOAP et de XML-RPC. C'est l'acronyme de *REpresentational State Transfer*.
- Elaboré en l'an 2000 par Roy Fielding, un des créateurs du prot. HTTP
- Principe : Internet est composé de ressources accessibles à partir d'une URL. Par exemple, http://www.meteo.fr/paris/ où Paris est une ressource définie par Météo France.
- A la requête de cet URL serait renvoyée une représentation de la ressource demandée (paris.php, par exemple). Cette représentation place l'application cliente dans un état (*state*) donné.
- Si l'application cliente lance un appel sur un des liens de la représentation en cours, une autre ressource est appelée. Ainsi, l'application cliente change d'état (*state transfer*) pour chaque représentation de ressource.

Rest s’appuie sur le protocole http

Requête :



Réponse :



Un des objectifs fondamentaux d’une API REST est d’utiliser HTTP comme protocole applicatif, pour homogénéiser et faciliter les interactions entre SI et pour ne pas avoir à façonner un protocole « maison » de type *SOAP/RPC* ou *EJB*, qui a l’inconvénient de “réinventer la roue” à chaque fois.

Il convient donc d’utiliser systématiquement les verbes HTTP pour décrire les actions réalisées sur les ressources (CRUD).

Correspondance Verbes / CRUD

Verbe HTTP	Correspondance CRUD	Collection : /orders	Instance : /orders/{id}
GET	READ	Read a list orders. 200 OK.	Read the detail of a single order. 200 OK.
POST	CREATE	Create a new order. 201 Created.	–
PUT	UPDATE/CREATE	–	Full Update. 200 OK. Create a specific order. 201 Created.
PATCH	UPDATE	–	Partial Update. 200 OK.
DELETE	DELETE	–	Delete order. 200 OK.

http://blog.octo.com/designer-une-api-rest/#crud

Le bon usage

- Un service REST devrait respecter les "conventions" suivantes :
  - Bien identifier les ressources devant être exposées au travers du service et de manière unique.
  - Chaque ressource devra se voir assigner une URL, de la forme :
    formehttp://www.site.com/contenus/1789
    plutôt que http://www.site.com/contenus.php?id=1789.
  - catégoriser les ressources selon les possibilités offertes à l'application cliente : lecture seule (GET) ou possibilité de modifier/créer une ressource (POST, PUT, DELETE) ?
  - chaque ressource devrait faire un lien vers les ressources liées.
  - documenter l'API.

Sitographie

http://blog.nicolashachet.com/niveaux/confirme/larchitecture-rest-expliquee-en-5-regles/

http://blog.octo.com/designer-une-api-rest/

http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

http://www.journaldunet.com/developpeur/tutoriel/xml/030707xml_rest1b.shtml

IUT Lyon 1
l'excellence technologique

Institut Universitaire de Technologie
Département Informatique
Site de Bourg-en-Bresse

ANDROID

Connexion à un serveur distant - AsyncTask Affichage du flux Json

LP Metinet - Janvier 2016
lionel.buathier@univ-lyon1.fr

1. Mise en place du serveur RestFull

- Importer la BDD (spiral) dans Wamp.
- Mettre en place le serveur
- Tester son fonctionnement en localhost

2. Connexion à un serveur distant

- Sous Android, l'interface utilisateur ne peut pas être bloquée plus de 10s
- Le temps de réponse d'une connexion distante est toujours incertain
- Par protection, sous Android, on ne plus ouvrir une socket directement dans l'interface utilisateur, sauf dans une webView (avec la méthode loadUrl)

- Il faut donc mettre en place un mécanisme asynchrone pour la connexion :
  - soit une AsyncTask,
  - soit un Thread

3. Utilisation de la classe AsyncTask

- La classe **AsyncTask<U,V,W>** basée sur 3 types génériques :
  - ⇨ U: le type du paramètre envoyé à l'exécution (reçu par l'asyncTask)
  - ⇨ V: le type de l'objet permettant de notifier de la progression de l'exécution
  - ⇨ W: le type du résultat de l'exécution retourné à onPostExecute

- AsyncTask définie 4 méthodes, dont 2 nous intéressent :
  - ⇨ **void onPreExecute() :** permet de créer une barre de progression
  - ⇨ **W doInBackground(U...):** travail dans le thread qui s'exécute en tâche de fond
  - ⇨ **void onProgressUpdate(V...)** qui permet d'actualiser une ProgressBar
  - ⇨ **void onPostExecute(W):** permet de mettre à jour l'UI lorsque que le travail est terminé
- Ressource : http://developer.android.com/reference/android/os/AsyncTask.html

3. Utilisation de la classe AsyncTask

Exécution et passage de paramètres

- Dans l'activity :
  - new MyAsyncTAsk().execute("toto", textview);

- Dans l'asyncTask<Object, Void, String> :
  - String doInBackGround(Object… params){
    String chaine = (String)params[0];
    tv = (Textview)params[1];
    Traitement
    Return chaine;

  - onPostExecute(String result)){
    tv.setText(result); }

4. Exercice : connexions http diverses

- Nous allons tester différentes manières de traiter des informations provenant de serveurs distants.

- Créer une activité avec :
  - 3 boutons
  - un TextView
  - une WebView

- Donner au préalable la permission à l'application de se connecter à l'internet dans le manifest.xml :

```
<uses-permission android:name="android.permission.INTERNET" />
```

4.1 Chargement direct dans une WebView

- **Bouton LoadURL:**
  - La méthode loadUrl de la WebView permet d'afficher le contenu d'une page web directement dans l'activité :

```
webview.loadUrl("http://10.0.2.2:80/rest/villes.php");
```

- **Bouton Text HTML :**
  - On peut également afficher directement une chaine HTML avec LoadData (si chaine stockée en local, donc chargement rapide) :

```
String strHtml = "<html><body><b> Ceci est un texte au format HTML
   </b></br>qui s'affiche très simplement</body></html>";

webview.loadData(strHtml , "text/html; charset=utf-8", "UTF-8");
```

4.3 Connexion à au serveur avec 'AsyncTask'

- **Bouton AsyncTask :**

  On veut maintenant afficher dans un TextView le flux Json retourné par le serveur.

  ⇨ Lorsqu'on appuie sur un bouton, **exécuter une AsyncTask** qui :

  - ouvre une connexion http de type HttpUrlConnection,

    ```
    HttpURLConnection  urlConnection = (HttpURLConnection) monUrl.openConnection();
    if (urlConnection.getResponseCode() == HttpURLConnection.HTTP_OK){ ....}
    ```

  - ouvre un flux en entrée et récupérer la chaine envoyée par le serveur

    ```
    BufferedReader in = new BufferedReader(newInputStreamReader(urlConnection.getInputStream()));
    ```

  - lit avec in.readLine(), tant qu'il y a des données dans le flux d'entrée
  - ferme la connexion (et donc le flux)
  - affiche la chaine retournée par le serveur dans un TextView

- Attention à bien afficher sur le TextView dans la méthode 'onPostExecute ' afin de laisser Android gérer l'affichage dans l'UI Thread

http://developer.android.com/reference/java/net/HttpURLConnection.html

http://developer.android.com/training/basics/network-ops/connecting.html