AMIO : Applications mobiles et Internet des Objets
Troisième année

T.P. 2: Android et Web Services REST

Ressources à consulter durant les séances de TP :
Documentation API Android:
http://developer.android.com/guide/index.html
Cours avec exemples de code:
http://www.univ-orleans.fr/lifo/Members/Jean-Francois.Lalande/teaching.html
Documentation API Telecom IoT lab:
http://iotlab.telecomnancy.eu/api

Exercice 1 : Consultation d'un web service

L'objectif de cet exercice est de montrer comment un smartphone Android peut interagir avec un Web Service de type REST. Nous utiliserons pour cela le Web Service « IoT lab » fourni par Telecom Nancy. Il est recommandé d'étendre l'application réalisée durant le TP 1 afin d'enrichir progressivement ses fonctionnalités en vue du projet final.

- Question 1 – Ajouter un bouton qui, une fois pressé, déclenchera l'envoi d'une simple requête au Web Service.
  - avant toute chose, il est nécessaire d'ajouter la permission INTERNET dans le AndroidManifest.xml de votre application.

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

  - la requête en question est une requête GET qui permettra d'obtenir la plus récente valeur de luminosité mesurée par chaque capteur:
    GET http://iotlab.telecomnancy.eu:8080/iotlab/rest/data/1/light1/last
  - La classe **HttpURLConnection** sera employée pour envoyer la requête.
  - il est recommandé de réaliser la communication avec le serveur en tant que tâche asynchrone (classe **AsyncTask**) afin de ne pas bloquer le thread principal de l'application (i.e. l'Activité au premier plan). Réaliser la connexion HTTP dans une tâche asynchrone en suivant : http://developer.android.com/training/basics/network-ops/connecting.html#AsyncTask
  - astuce: si vous utilisez un émulateur, il est possible de vérifier que la requête est bien transmise au serveur en utilisant Wireshark sur la même machine que l'émulateur.

- Question 2 – Traiter la réponse du Web Service à la requête précédemment transmise.
  - dans la même tâche asynchrone que pour la question 1 (car même communication), ajouter le code nécessaire pour récupérer la réponse du serveur, et en extraire le code HTTP que vous afficherez dans les logs.
  - si le code HTTP correspond à une erreur (différent de 200), utiliser la classe **Toast** pour afficher une pop-up sur l'écran afin de prévenir l'utilisateur qu'un problème est survenu.
  - remarque : doInBackground() ne peut pas interagir avec l'UI, il faut utiliser onProgressUpdate() ou onPostExecute() pour afficher la pop-up.
  - en cas de succès, afficher dans les logs le corps de la réponse retournée par le serveur.
    - on observe que le corps de la réponse est formaté en JSON.

- Question 3 – Parser la réponse JSON renvoyée par le Web Service.
  - utiliser la classe **JsonReader** qui est dédiée à cet effet. Le Json reçu contient un objet « data » à

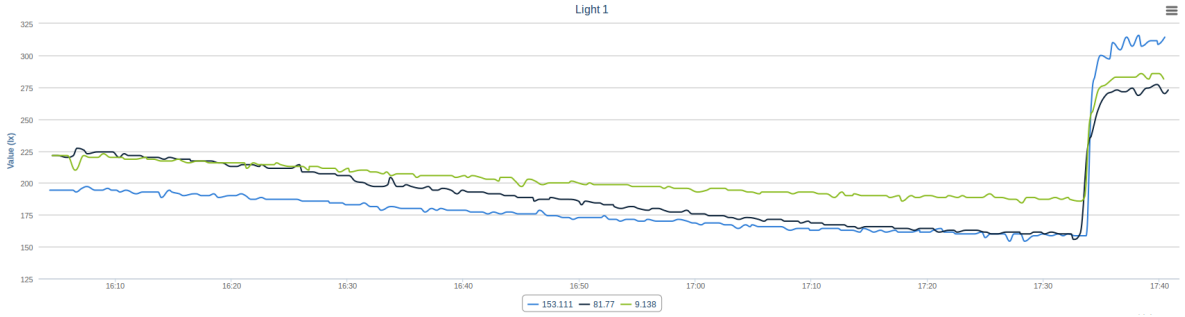
lire avec *reader.beginObject()* qui lui-même contient un tableau *reader.beginArray()* d'objets.

- dans le layout de l'activité, utiliser une Textview qui sera mise à jour après chaque échange avec le Web Service. Typiquement, cette Textview pourra afficher par exemple la dernière valeur relevée et la date de mesure.
- optionnellement, ajouter dans le layout de l'activité une autre Textview (plus grande cette fois) avec une Scrollbar; parser, pour chaque mote, l'ID et les mesures associées (température, luminosité, etc.) et afficher la liste des motes et leurs informations dans la grande Textview nouvellement créée.

Exercice 2 : Traitement des données

Les données issues du réseau de capteurs peuvent désormais être récupérées au sein de l'application. Nous allons maintenant leur appliquer le traitement spécifique à notre application.

- Question 1 – Calibrage
  - observer le graphique ci-dessous et identifier les valeurs de luminosité traduisant une lumière de bureau allumée ou éteinte.



- Question 2 – Détection des lumières allumées
  - créer une structure de données permettant de conserver les derniers relevés extraits du JSON.
  - lors de la réception de nouvelles données, recenser les motes dont la luminosité mesurée indique une lumière allumée et les afficher au niveau de l'activité

Exercice 3 : Notification

Nous souhaitons maintenant tirer profit du Service créé au TP1 pour collecter périodiquement les données relevées par les motes, et notifier l'utilisateur lors d'un changement d'état (allumé → éteint ou éteint → allumé).

- Question 1 – Utiliser un service dont le but est de vérifier périodiquement (et en background) les valeurs relevées
  - reprendre le code de l'exercice 2 du TP 1 (Q1 et Q2) pour implanter un service avec un timer périodique et de l'exercice 1 de ce TP pour l'échange avec le Web Service

- Question 2 – Ajouter le code nécessaire afin d'afficher une notification Android lorsqu'un changement brusque de la luminosité d'un mote a été détecté après la vérification périodique.
  - la documentation officielle d'Android explique la création de notifications:
    - https://developer.android.com/guide/topics/ui/notifiers/notifications.html
  - restreindre l'envoi d'une notification suite à un changement d'état à l'intervalle horaire [18h-23h]