TP Android : Consultation de la météo sur Android (2nde partie)

Objectifs

Cette seconde partie vise à améliorer l'application météo en modifiant par étape la persistance des données et la gestion asynchrone des données collectées à partir du service Web.

1 Création d'une base de données

Jusqu'à présent, les ajouts ou suppressions de villes ne sont pas conservés entre deux lancements de l'application. Pour pallier ce problème, il est demandé de remplacer le stockage des données fait dans un `ArrayList` par une base de données SQLite.

Écrivez une classe héritant de `SQLiteOpenHelper` pour gérer les données météo dans une base de données. En sus des columnes nécessaires pour conserver le nom de la ville, le pays, la date du relevé météo, les informations sur la température, le vent et la pression atmosphérique, ajoutez un identifant utilisé comme clé primaire. Pour tester cette nouvelle classe, vous pourrez dans un premier temps conserver l'`ArrayAdapter` déjà défini pour gérer votre `ListView`. Une méthode `List<City> getAllcities()` pourra ainsi retourner l'ensemble des villes stockées actuellement dans la base de données.

Afin d'empêcher plusieurs ajouts d'une même ville, il est demandé d'introduire la contrainte `UNIQUE (CITY_NAME, COUNTRY)` lors de la création de la base de données.

2 Utilisation d'un fournisseur de contenu

Les fournisseurs de contenu (*content providers*) sont principalement employés pour partager des données entre applications ; ils sont parfois également nécessaires pour utiliser certains types de classes comme les `Loaders`.

Définissez un `ContentProvider` permettant d'accéder à la base de données par des URI ayant l'un des deux types suivants :

- `content://<content-authority>/weather` retournant un lien vers l'ensemble de la base de données (type `ContentResolver.CURSOR_DIR_BASE_TYPE`),
- `content://<content-authority>/weather/<country>/<city>` retournant les informations sur la ville *city* située dans le pays *country* (type `ContentResolver.CURSOR_ITEM_BASE_TYPE`).

Il est conseillé de définir des tests dans une classe héritant de `ApplicationTestCase<Application>` [1] pour tester la création, la consultation, la mise à jour, l'ajout ou la suppression d'entités du fournisseur de contenu.

1. La classe `ApplicationTestCase` étant obsolète depuis la version 7.0 d'Android, il est préférable d'employer `ProviderTestCase2`.

3 Utilisation d'un chargeur

Les classes de type `Loader` sont très pratiques pour charger de manière asynchrone des données dans une activité. Elles nécessitent de définir, comme cela vient d'être fait, un fournisseur de contenu, pour accéder aux données.

Dans cette section, il est demandé de modifier l'accès aux données au niveau des activités de l'application, en recourant à un *loader* pour interroger le fournisseur de contenu. Il faudra pour ce faire que vos activités implémentent `LoaderManager.LoaderCallbacks<Cursor>`. Pour gérer la `ListView`, remplacez l'`ArrayAdapter` par un `SimpleCursorAdapter`. La création de ce `SimpleCursorAdapter` pourra être faite en utilisant comme paramètre `android.R.layout.simple_list_item_2` pour afficher d'une part le nom de la ville et d'autre part le pays (voir figure 1).

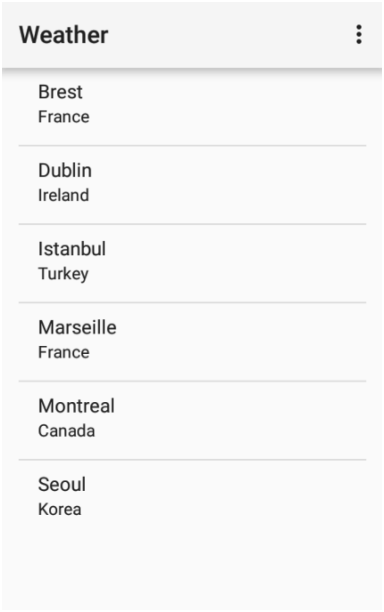

FIGURE 1 – Affichage de la liste des villes en utilisant `android.R.layout.simple_list_item_2`.

4 Emploi d'un adaptateur de synchronisation

Les adaptateurs de synchronisation (*sync adapters*)[2] sont des composants facilitant les tâches de transfert de données entre un appareil mobile et un serveur. Remplacez l'accès au service Web réalisé auparavant par un `AsyncTask` par un *sync adapter*.

Ce type d'adaptateur nécessite de définir quatre classes dont deux sont chargées de gérer l'authentification au serveur. Dans la mesure où le service Web employé ne nécessite pas d'authentification, vous pourrez créer deux classes minimalistes pour cette opération.

Le *sync adapter* devra être appelé dans deux circonstances :

— lors d'une sélection de l'item « rafraîchir »,

— de manière automatique toutes les heures.

2. `https://developer.android.com/training/sync-adapters/index.html`

5 Rendu

Votre projet Android créé sous Android Studio est à déposer sur le site du cours de la plateforme e-uapv.

Il vous est demandé d'accompagner votre projet par un fichier « README » expliquant votre avancement dans cette partie du TP.