Mon application : Lecteur RSS

Explications

Le principe de cette application est d'affiché le contenu d'un flux RSS (ici celui du monde.fr).

L'affichage est basé sur une « ListView » (comme pour le TP précédent) ainsi qu'un Adapter. Ce dernier devra s'occuper de récupérer le flux depuis Internet à l'aide d'une « AsyncTask[1] », les requêtes http étant interdites sur le thread s'occupant de l'UI (cela peut entraîner un blocage de l'application).

Ajout des permissions

L'application nécessite de pouvoir accéder à internet, il faut donc penser à ajouter la permission « INTERNET ».

Adapter

Au niveau de l'adapter, la base est quasiment la même que le précédent TP. Cependant, nous veillerons à utiliser le layout «android.R.layout.two_line_list_item » permettant d'obtenir facilement deux lignes par item.

Suppression des balises HTML

Le contenu (« description » dans le flux RSS) était, dans mon cas, en html. Voici une astuce permettant de supprimer facilement les balises html ainsi que les caractères « [obj][2] » résultant de la transformation par « Html.fromHtml »

```
Html.fromHtml(n.getContent()).toString().replace((char) 65532, (char) 32).trim()
```

HttpURLConnection

Il s'agit de l'objet Java standard permettant d'effectuer une requête http.

Pour aller plus loin : http://developer.android.com/reference/java/net/HttpURLConnection.html

AsyncTask

Cet objet permet d'exécuter de manière asynchrone un morceau de code (tel un thread). Tout en ayant une méthode « onPostExecute » invoqué par le thread UI permettant ainsi le rafraîchissement de l'interface sans « plantage » (il n'est pas possible/fiable demander le rafraîchissement de l'interface en dehors du thread principal).

Vous devriez vous approcher de cet exemple :

[1] https://developer.android.com/reference/android/os/AsyncTask.html
[2] http://stackoverflow.com/questions/8560045/android-getting-obj-using-textview-settextcharactersequence

CC BY

```
AsyncTask<Void, Void, List> a = new AsyncTask<Void, Void, List>() {
	@Override
	protected List doInBackground(Void... params) {
		ArrayList<News> res = new ArrayList<News>();
		try {
			URL url = new URL("http://feeds.lefigaro.fr/c/32266/f/438191/in-
dex.rss");
			   HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
			   RssParser parser = new RssParser();
			try {
				return parser.parse(urlConnection.getInputStream());
			} catch (XmlPullParserException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		} catch (IOException e) {
			e.printStackTrace();
		}
		return res;
	}

	@Override
	protected void onPostExecute(List result) {
		adapter.update(result);
	}
};
a.execute();
```

Parser XML

Le but de ce TP n'étant pas de parser le contenu d'un flux XML, vous trouverez ci-dessous la classe que j'ai écrite pour faire ce TP (et devrait théoriquement suffire).

CC BY

```
public class RssParser {
      private String readText(XmlPullParser parser) throws IOException, XmlPull-
ParserException {
          String result = "";
          if (parser.next() == XmlPullParser.TEXT) {
              result = parser.getText();
              parser.nextTag();
          }
          return result;
      }
      private static final String ns = null;

      private void skip(XmlPullParser parser) throws XmlPullParserException, IOEx-
ception {
          if (parser.getEventType() != XmlPullParser.START_TAG) {
              throw new IllegalStateException();
          }
          int depth = 1;
          while (depth != 0) {
              switch (parser.next()) {
              case XmlPullParser.END_TAG:
                  depth--;
                  break;
              case XmlPullParser.START_TAG:
                  depth++;
                  break;
              }
          }
       }

      public ArrayList<News> parse(InputStream in) throws XmlPullParserException,
IOException {
              XmlPullParser parser = Xml.newPullParser();
        parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
        parser.setInput(in, null);
        parser.nextTag();
        ArrayList<News> entries = new ArrayList<News>();

        parser.require(XmlPullParser.START_TAG, ns, "rss");
        parser.nextTag();
        parser.require(XmlPullParser.START_TAG, ns, "channel");
        while (parser.next() != XmlPullParser.END_TAG) {
            if (parser.getEventType() != XmlPullParser.START_TAG) {
                continue;
            }
            String name = parser.getName();
            if (name.equals("item")) {
             String title = "";
                String content = "";
             while (parser.next() != XmlPullParser.END_TAG) {
                    if (parser.getEventType() != XmlPullParser.START_TAG) {
                        continue;
                    }
                    name = parser.getName();
                    if (name.equals("title")) {
                        title = readText(parser);
                    } else if (name.equals("description")) {
                        content = readText(parser);
                    } else
                          skip(parser);
             }
                entries.add(new News(title, content));
            }  else
             skip(parser);
        }
        return entries;
      }
}
```


CC BY

Résultat

L'ensemble du projet « NewsReader » est disponible sur le dépôt github : https://github.com/steven-martins/tp-android

CC BY