Université Paris Diderot - Paris 7

Master 1 Informatique, Programmation de composants mobiles.

14 janvier 2016

Durée : 2h Tous les documents papier autorisés. Documentation android autorisée. Les transparents du cours autorisés.

Toutes les autres ressources web (forums de développeurs etc.) interdites

Le sujet comporte 4 pages.

Consignes

La documentation se trouve localement dans

`/opt/android-sdk-linux_x86-r8/docs`

Si vous utilisez votre ordinateur portable alors c'était à vous d'installer la documentation localement sur votre machine avec Android Studio.

Après chaque exercice vous devez appeler un enseignant pour valider votre travail. Ensuite compressez le répertoire `VotreAppli/app/src/main` qui contient les sources de votre application :

```
tar cvf VotreNom_VotrePrenom_Gr1Exoi.tar VotreAppli/app/src/main
gzip VotreNom_VotrePrenom_Gr1Exoi.tar
```

où *i* sera remplacé par le numéro de l'exercice et `VotreNom` c'est bien sûr votre nom.

Si vous travaillez sur une machine de l'ufr copiez le fichier `VotreNom_VotrePrenom_Gr1Exoi.tar.gz` dans le répertoire `/ens/zielonka/Enseignement/Android/G1`

Si vous utilisez votre propre ordinateur portable copier le fichier `tar.gz` qui contient la solution de l'exercice dans `didel` (cours Android)

Travaux → ExamGr1

Avant de commencer les exercices récupérez depuis `didel` (Documents et liens) ou à partir de répertoire `/ens/zielonka/Enseignement/Android/LAYOUTS_EXAM_GR1` les fichiers `layout`.

Il y a deux groupes indépendants d'exercices.

le premier : les exercices 1 et 2 doivent être faits dans l'ordre,

et le deuxième : les exercices 3,4,5,6 sont aussi à faire dans cet ordre.

Vos programmes seront testés avec un logiciel de détection de plagiat.

Le dépôt sur `didel` est protégé, sauf vous et moi-même personne n'aura accès à vos programmes.

Si vous travaillez sur les machines de l'ufr c'est à vous de protéger votre répertoire de travail contre les visites non-autorisées.

Exercices

Exercice 1 Écrire une application composée de deux activités. L'activité principale (layout `content_exo5.xml`) contient deux boutons : pair et impair et un `TextView` initialement vide.

Quand l'utilisateur appuie sur un de deux boutons, l'activité principale mémorise dans une variable l'information sur la parité du boutons (pair ou impair) et lance une nouvelle activité dans le but d'obtenir un résultat.

Dans la nouvelle activité (layout `activity_exemple.xml`) utilisateur met un nombre entier dans un `ViewEdit` et appuie sur le bouton `ENVOYER`. Ceci envoie ce nombre comme le résultat dans la première activité.

Activité principale compare le nombre envoyé par la deuxième activité et si la parité correspond au bouton choisi initialement (nombre pair et le bouton pair ou le nombre impair et le bouton impair) alors on affiche `bravo` dans le `TextView`, sinon on affiche `oops`.

Exercice 2 Si on tourne l'appareil alors le mot `bravo` ou `oops` affiché précédemment dans l'activité principale disparaît.

Modifier l'application pour que la réponse ne disparaisse plus quand l'utilisateur tourne l'appareil.

(Un problème similaire peut avoir lieu dans l'activité auxiliaire mais on ne demande pas de s'en occuper).

Exercice 3 Créer l'activité principale (layout `content_main.xml`) qui contient deux boutons : « afficher liste » et « initialiser liste ».

Copier dans le répertoire `res/values` de votre projet le fichier ressources `arrays.xml` qui contient la définition de deux tables : `cles` et `urls` de `Strings`.

Quand l'utilisateur appuie sur le boutons « initialiser liste » l'activité doit prendre un objet `SharedPreferences` dont le nom est `WEBPAGES` et remplir cet objet avec les couples (clé, valeur) en utilisant les éléments de la table `cles` comme les clés et les éléments de la table `urls` comme les valeurs. On suppose que *i*-ème clé correspond à *i*-ème valeur (les deux tables dans le fichier ressources `arrays.xml` sont de même longueur).

(Rappel : pour obtenir les ressources `getResources()` et ensuite la méthode qui convient pour récupérer les tables.)

Indication. Rappelons que `SharedPreferences` est une sorte de fichier qui permet de stocker les couples (clé,valeur). Pour obtenir `SharedPreferences` on utilise la méthode `getSharedPreferences()` de `Context`.

Pour remplir `SharedPreferences` on utilise l'objet `SharedPreferences.Editor`, et n'oubliez pas de faire l'opération `commit` à la fin, sinon rien ne sera sauvegardé dans le fichier de préférences (voir la documentation).

Exercice 4 Ajouter une nouvelle activité `ListeSimple`, layout `activity_liste_simple.xml` Cette nouvelle activité démarre quand l'utilisateur appuie sur « afficher liste » dans l'activité principale.

L'activité contient un `LinearLayout` dont le id est `buttons` et deux `CheckBox`. Le `LinearLayout` `buttons` n'a initialement aucun élément.

(L'autre `LinearLayout` qui contient les deux `CheckBox` n'a pas d'intérêt ici et n'a pas d'id).

Au démarrage l'activité `ListeSimple` lira le `SharedPreferences WEBPAGES` et mettra dans le `LinearLayout buttons` les objets `Button`, chaque des ces `Buttons` affichera le texte qui est une clé récupérée dans `SharedPreferences WEBPAGES`. Donc le nombre de `Buttons` à mettre dans le `LinearLayout buttons` correspond au nombre de couple (clé,valeur) dans `SharedPreferences`.

Indications. Pour récupérer l'ensemble de clés on peut utiliser le code suivant :

```
Map<String, ?> all = preferences.getAll();  
Set<String> keys = all.keySet();
```

où `preferences` se sont nos `SharedPreferences`.

Pour ajouter `Buttons` dans `LinearLayout`, comme d'habitude il faut récupérer `LinearLayout` depuis le fichier `layout` en utilisant son `id`, et ensuite utiliser la méthode `addView(View)` pour ajouter les boutons un par un. `addView()` est une méthode de la classe `GroupView` héritée par chaque `layout`, en particulier par `LinearLayout`.

Exercice 5 On complète l'activité `ListeSimple` de l'exercice précédent.

Quand l'utilisateur appuie sur un des boutons que vous avez contruit dans l'exercice précédent alors

(1) si le premier `CheckBox` est marqué on lancera une nouvelle activité `AffichePageWeb` en lui envoyant dans `Intent` la valeur (un `String` contenant `url`) correspondante à la clé qui s'affiche dans le bouton.

(Pour trouver la valeur à partir d'une clé il faudra bien sûr utiliser `SharedPreferences WEBPAGES`).

L'activité `AfficherWeb` sera vide pour l'instant.

(2) si le deuxième `CheckBox` est marqué alors

— il faut supprimer ce bouton dans `LinearView` (la méthode `removeView(View)` de `GroupView` hérité par `LinearLayout`) et

— supprimer la clé correspondante (avec sa valeur) de `SharedPreferences`.

Indication. Équiper chaque `Button` à sa création d'un `View.OnClickListener` qui effectuera les actions demandées : `Button.setOnClickListener( new View.OnClickListener() ) ...`

Exercice 6 Et la dernière activité `AfficherPageWeb`, `layout activity_web_page.xml`.

L'`intent` qui lance cette activité contient comme un extra un `String` avec l'`url` de la page web (voir l'exo précédent).

`Layout` de `AfficherWebPage` contient un `WebView` et deux boutons.

L'activité démarre en affichant la page web qui correspond à l'`url` fourni dans l'`Intent`.

Pour cela on récupère `WebView` à partir de fichier `layout`, ensuite en exécute :

```
webView.setWebViewClient(new WebViewClient());  
webView.loadUrl(url);
```

La première instruction est nécessaire si le `WebView` doit s'afficher à l'intérieur de l'activité courante au lieu de lancer une nouvelle activité. La deuxième charge la page `url` dans `WebView`. (`url` est un `String` qui contient l'adresse de la page à charger).

Si l'utilisateur appuie sur le boutons SAVE alors l'activité ajoutera dans SharedPreferences WEBPAGES l'url de la page courante (comme valeur) et le titre de la page courante comme la clé.

Indication. Pour récupérer le titre et l'adresse web deux méthodes de WebView : `getTitle()` et `getUrl()`.