Licence Professionnelle

Android - 2014-2015

Passer à la vitesse supérieure

Android TD5

Dans ce TD, nous allons manipuler plusieurs aspects (fournisseurs de contenus, interactions entre activités, bases de données, fragments). Contrairement aux TDs précédents, qui étaient très guidés, nous vous donnons ici une succession d'objectifs à réaliser de manière autonome. La seule assistance que nous fournissons (outre l'accompagnement en salle) est une liste de liens vers les pages où vous pouvez trouver les informations utiles. À vous de les extraires et les utiliser au mieux.

Objectif global : créer un gestionnaire (hyper basique) d'arbre généalogique, capable d'importer des personnes depuis les contacts du téléphone et leur affecter des liens de parenté (au minimum : conjoint, parent, enfant).

Travail à rendre. Une archive contenant votre projet, prêt à être exécuté après importation. Le tout est à rendre pour le mercredi 3 décembre, par courriel à votre enseignant Android (Arnaud Casteigts ou Stéphane Fossé, selon votre groupe). Ce mini-projet constituera la majorité de votre notre de TP (pas l'intégralité). Outre les fonctionalités demandées, votre travail sera évalué en fonction de la personalisation de votre application (voir point 8 ci-dessous).

1. **Création du squelette.** Pour gagner du temps, nous allons partir de la version corrigée du tutoriel de la semaine dernière (version 1 : `Notepadv1Solution`).
   (a) Adaptez les différents codes source afin de remplacer la table "`notes`" par une table "`persons`" qui comportera les deux colonnes `_id`, et `name`. Ainsi l'ancienne classe `NotesDbAdapter` ne contient plus que `KEY_ROWID` et `KEY_NAME` au lieu de `KEY_ROWID`, `KEY_TITLE` et `KEY_BODY`. Pensez à changer le numéro de version de votre base de donnée pour forcer l'invocation de la méthode `onUpgrade()` lors de la prochaine exécution.
   (b) Conservez, dans un premier temps, l'item de menu et la méthode permettant d'ajouter une personne automatique et numérotée (utile pour tester le reste).
   (c) Rajoutez aussi le code permettant de supprimer une entrée via le menu contextuel (`public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo)`), en vous aidant des étapes 1 à 3 de l'exercice 2 du gestionnaire de notes (`http://developer.android.com/training/notepad/notepad-ex2.html`). Vérifiez que tout fonctionne bien avant de passer à la suite.
2. **Importer un contact.** Plutôt que de saisir les personnes unes à unes, on aimerait pouvoir importer des contacts déjà présents dans le téléphone. (Ceux qui utilisent l'émulateur pourront aller saisir quelques contacts fictifs dans l'application dédiée.) Dans un premier temps, on veut pouvoir importer un contact donné en lançant le gestionnaire de contacts sur le mode `startActivityForResult` (vu la semaine dernière). Vous pouvez vous aider de la page `http://developer.android.com/training/basics/intents/result.html` en l'adaptant pour récupérer le nom du contact plutôt que son numéro de téléphone. Lorsque vous aurez récupéré ces infos, mettez à jour votre base de donnée pour y ajouter cette personne, et si besoin, réactualisez l'affichage de votre liste.
   (a) Ajouter au menu, un item d'importation de contact
   (b) Lors d'un clic sur cet item, invoquez `startActivityForResult` avec un intent construit de la façon suivante `Intent intent = new Intent(Intent.ACTION_PICK, ContactsContract.Contacts.CONTENT_URI)`
   (c) Redéfinissez la méthode `void onActivityResult(int requestCode, int resultCode, Intent data)` pour récupérer le contact sélectionné
3. **Importer tous les contacts.** On aimerait disposer d'une autre fonctionnalité afin d'importer d'un coup tous les contacts du téléphone. Dans ce cas, plutôt que de lancer une autre activité, il faut utiliser le fournisseur de contenu (*content provider*) que l'application de gestion des contacts met à votre disposition. Cette opération peut durer un certain temps, donc à terme nous la ferons exécuter par un autre *thread* afin de ne pas bloquer l'interface graphique Vous pouvez vous référer à la page `http://developer.android.com/guide/topics/providers/content-provider-basics.html#SimpleQuery` pour l'utilisation d'un fournisseur de contenu en général, et la page `http://developer.android.com/guide/topics/providers/contacts-provider.html` pour le cas particulier des données de contact. Si cela vous semble plus simple, vous pouvez commencer par faire tout ça depuis le thread de l'interface graphique, puis adapter votre solution pour utiliser une tâche asynchrone (classe `AsyncTask`).
   (a) Ajouter au menu un item pour tout importer

(b) Pensez à ajouter le droit de lecture de la liste des contacts
`<uses-permission android:name="android.permission.READ_CONTACTS" />`

(c) Créez une classe héritant de `AsyncTask`. Redéfinissez le type de paramètre pour récupérer une référence à l'activité parente plutôt que des URIs. Cette référence vous servira à deux endroits : dans la méthode `doInBackground()`, elle vous permettra de récupérer une référence vers le résolveur de contenu (`getContentResolver()`) ; puis dans la méthode `onPostExecute()`, elle vous permettra d'invoquer la méthode `fillData()` pour actualiser l'affichage de la liste. Gardez en tête que la méthode `onPostExecute()`, contrairement à `doInBackground()`, s'exécute depuis le thread de votre activité parente (et donc, de l'interface graphique), et ce, justement pour permettre une mise à jour de l'affichage une fois la tâche terminée.

(d) Utilisez la syntaxe suivante pour lancer l'import de manière aynchrone : `maTacheAsynchrone.execute();`

4. **Activité d'édition de liens.** Une seconde entrée dans le menu contextuel est ajouté avec pour texte "Add a relation". Le choix de cette entrée par l'utilisateur doit lancer une nouvelle activité, dont nous décomposerons le développement comme suit :

(a) Contentez vous d'abord de ré-afficher le nom de la personne, ainsi qu'un menu déroulant (`Spinner`) qui contient les trois choix "parent de", "enfant de", et "conjoint de".

(b) Ajouter un second menu déroulant contenant toutes les personnes de votre table "persons". Cette étape n'est pas triviale. Le mieux est d'utiliser un `SimpleCursorAdapter`, qui permet de charger automatiquement dans une `ListView` (dont hérite `Spinner`) le contenu d'une colonne de donnée souhaitée. Vous pourrez vous aider de la documentation officielle, ainsi que de la réponse `http://stackoverflow.com/a/11216746` (à première vue correcte).

(c) Modifiez le code source de gestion de la base de donnée (`xxxHelper.java`) afin que la prochaine exécution fasse créer une seconde table "`relations`" qui comportera les trois champs suivants : "`id1`","`id2`", et "`type`". Le champs `type` est un entier qui correspond au type de lien (par exemple 0 pour enfant de, 1 pour parent de, 2 pour conjoint de). Notez que les types 0 et 1 sont redondants car on peut toujours coder `(id1, id2, 0)` comme `(id2, id1, 1)`. Vous pouvez adaptez ce schéma si vous n'aimez pas la redondance, du moment que votre logique s'y adapte ensuite.

(d) Toujours dans le même fichier, ajoutez deux méthodes pour encapsuler l'accès aux données : `addRelation(id, id, type)` permettant d'ajouter une nouvelle relation, et `getRelations(id)` permettant de récupérer une `HashMap` contenant toutes les relations d'une personne donnée.

(e) De retour dans la nouvelle activité, rajoutez un bouton de validation dont le traitement appelera addRelation.

5. **Affichage des relations.** Intégrez à votre application une troisième activité permettant de lister les relations existantes pour une personne donnée. Vous pouvez les récupérez avec votre nouvelle méthode `getRelations()` et remplir votre liste à l'aide d'un `ArrayAdapter<String>` qui utilisera le layout `android.R.layout.simple_list_item_1` (layout de base du SDK pour remplir des liste avec un `TextView` par ligne). Chaque ligne sera d'abord composée de la concaténation de la relation et de l'autre personne. Par exemple `<0, Madame Y>` deviendra "Enfant de Madame Y".

6. **Optimisez la navigation.** Adaptez vos menu contextuel ou autres événements de clics pour que la navigation entre les fonctionalités soit intuitive et cohérente. Par exemple, on pourrait imaginer qu'un clic (normal, plutôt que "long") sur une personne envoie vers la page où l'on voit ses relations, page depuis laquelle un item de menu (de type normal, pas contextuel, à l'image du "Add note" initial) permet de saisir une nouvelle relation. Vous pouvez aussi intégrer de nouvelles commandes pour supprimer des personnes ou des relations. Enfin, empêchez les relations impossibles (par exemple être l'enfant d'un de ses parent).

7. **Fragments.** Sauvegardez votre projet pour éviter les dégâts. Puis utilisez des `Fragments` plutôt que des activités. Votre application sera alors composée de trois "panneaux". Testez son exécution sur un téléphone puis sur une tablette. Comparez le comportement de votre application sur les deux plateformes.

8. **Extension.** Personnalisez votre application de la manière que vous voulez. Vous pouvez par exemple ajouter des fonctionnalités et/ou parfaire l'expérience utilisateur. Vous êtes libre d'aller jusqu'où vous voudrez. Nous tiendrons compte du travail effectué dans l'évaluation.