Persistance des données sur une courte durée

Chantal Keller

16 octobre 2015

Plan

Jours de “révision”

Date de “contrôle continu” :

- jeudi 3 décembre ?
- (samedi 28 novembre ?)

Rendu du TD6 (1/2)

Ne pas se reposer sur l'interface :

- **les valeurs connues (qui ne sont pas des entrées) ne doivent pas dépendre de l'interface** : ex. ne pas aller lire un `TextView`
- **préférer des variables globales du programme**
- ex. du TD6 : le score et le niveau doivent être contenus dans des variables globales de type `int`

Principe d'une application avec interface graphique :

- la programmation Java reste la même
- en complément, on affiche certaines variables dans l'interface (mais pas le contraire)
- on ne lit des variables de l'interface que pour des entrées de l'utilisateur

Rendu du TD6 (2/2)

Disposition des boîtes de l'interface :

- **ne pas utiliser de dimensions fixes** : les appareils Android ont des tailles très différentes !
- **préférer des dispositions et dimensions relatives** : utiliser les poids

Cas des fragments :

- on réserve un espace avec un `FrameLayout` : pas de `wrap_content`
- pas de dimensions fixes non plus : il reste donc les poids

Rappel sur les poids (cf. TD2)

Les poids :

- s'utilisent dans un `LinearLayout` (vertical ou horizontal)
- pour les boîtes qui ont besoin de `wrap_content`, inchangé
- pour les autres : on met la dimension correspondante à `0dp`, et on donne un poids relatif

Exemple

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="bouton"/>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="texte"/>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"/>

</LinearLayout>
```

Plan

Observation

Lorsqu'on tourne la tablette :

- on perd le score et le niveau
- le fragment est attaché plusieurs fois

Explications :

- l'activité est détruite et recréée à partir de rien
- mais les fragments sont en partie sauvegardés
- il ne faut donc pas les attacher à chaque passage dans `onCreate` et il faut s'en servir pour conserver des données

Destruction pour une "courte durée"

Le système détruit une activité mais dans le but de la recréer ensuite :

- on tourne la tablette : pour redessiner l'interface
- le système a besoin de mémoire

Buts des fragments :

- factoriser le code d'interfaces
- créer des interfaces dynamiques
- **sauvegarder des données lors des destructions pour une courte durée**

Sauvegarder des données lors des destructions pour une courte durée

Méthode :

- utiliser les fragments déjà présents dans l'activité
- s'il n'y en a pas, en ajouter un sans interface

Sans interface :

- pas lié à un `FrameLayout`
  (rappel : `transaction.add(R.id.frameLayoutFragment, frag);`)
- lié à un *tag* : `transaction.add(frag, "tagDuFragment");`

Récupération d'un fragment d'une activité

Lorsqu'une activité est détruite pour une courte durée :

- ses fragments **attachés** sont gardés en mémoire
- on peut les retrouver grâce au gestionnaire de fragments :

```
frag = (MyFragment) fragmentManager.findFragmentById(R.id.frameLayoutFragment);
```

ou

```
frag = (MyFragment) fragmentManager.findFragmentByTag("tagDuFragment");
```

- `frag` vaut `null` lors de la première création de l'activité, et contient le fragment lors d'une recréation

Exemple

Dans la classe **de l'activité** :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity);

    // On récupère le fragment, ou null s'il n'existe pas encore
    MyFragment frag = (MyFragment) getFragmentManager().findFragmentById(R.id.frameLayoutFragment);

    // S'il n'existe pas, on le crée et on l'attache
    if (frag == null) {
        frag = new MyFragment();
        FragmentTransaction transaction = fragmentManager.beginTransaction();
        transaction.add(R.id.frameLayoutFragment, frag);
        transaction.commit();
    }
}
```

Sauvegarde des données d'un fragment

Par défaut :

- les fragments **attachés** sont gardés en mémoire, **mais pas leurs données**

Pour que leurs données soient sauvées aussi :

- dans la classe **du fragment** :

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // On veut que les données du fragment soient mémorisées
    setRetainInstance(true);
}
```

Plan

TD

Suite du TD6 :

1. Corriger les erreurs décrites au début de ces transparents (si besoin)
2. "À vous de jouer"
3. À rendre pour le mercredi 21 octobre à 23h59 (jusqu'à un numéro de question communiqué en fin de TD)