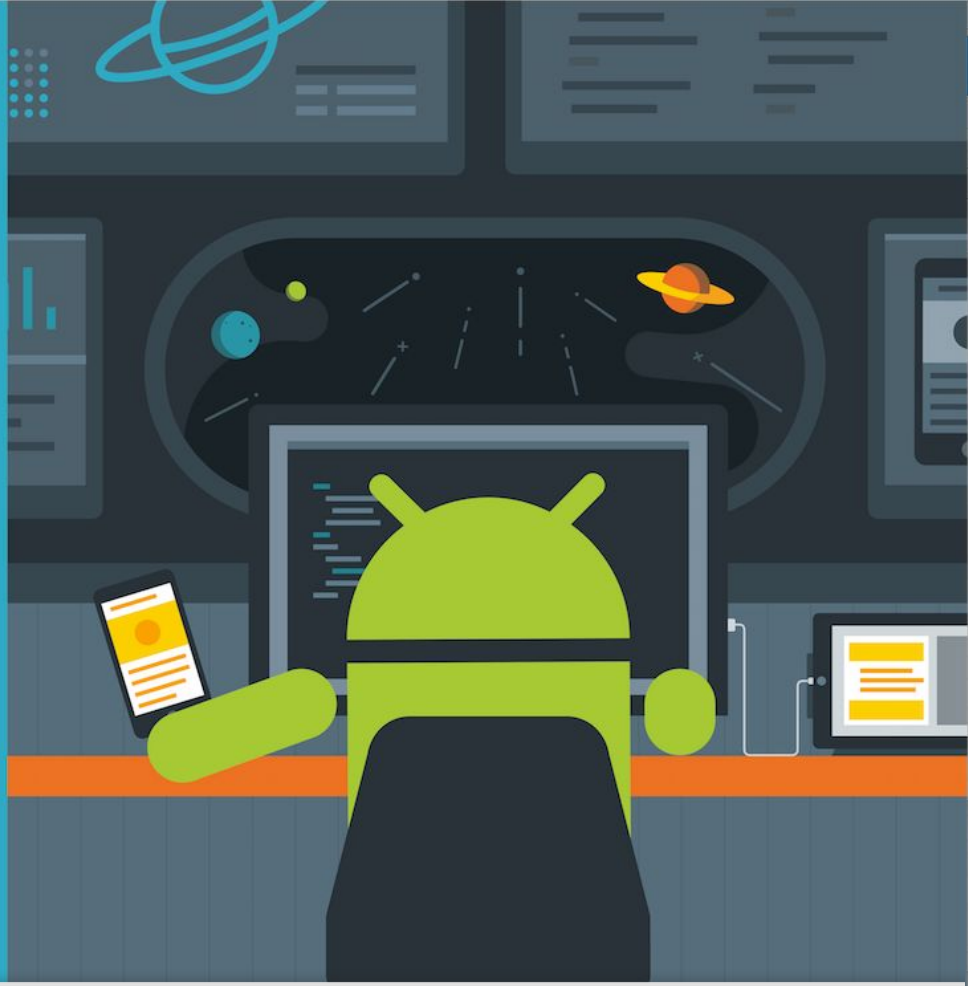
Développement Android avancé

# Fragments

Jordan Hiertz

Contact :
hiertzjordan@gmail.com
jordan.hiertz@al-enterprise.com



# 6.1 Fragments

Un composant UI réutilisable avec son propre cycle de vie

# Contenu

- Comprendre la classe Fragment
- Créer un `Fragment`
- Utiliser un layout pour un `Fragment`
- Ajouter un `Fragment` à une `Activity`

# Comprendre la classe Fragment

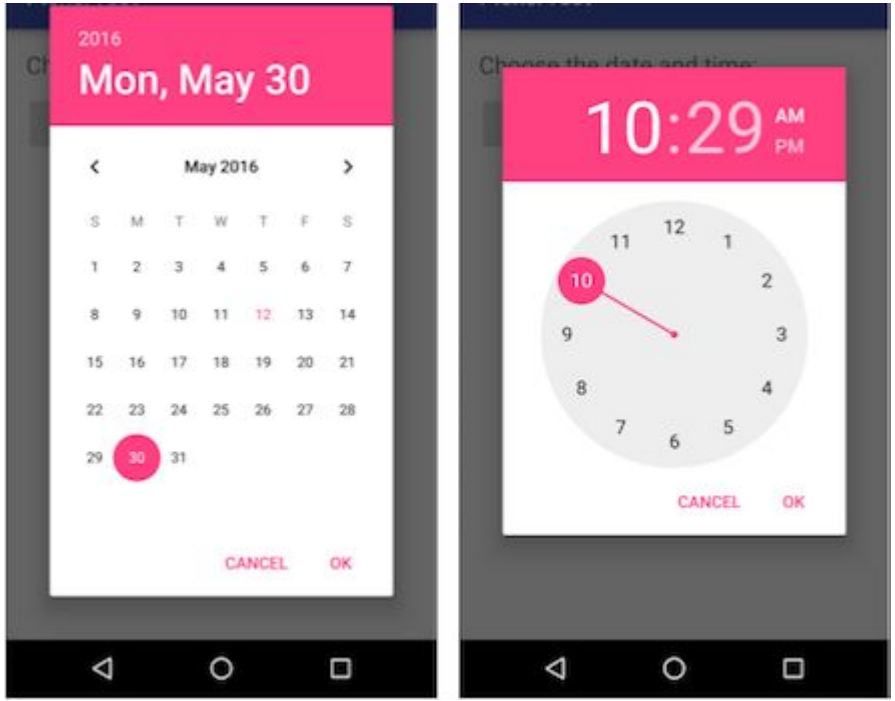
# Fragment class

- Contient une partie de l'UI et de son comportement
- Possède ses propres états de cycle de vie (comme une `Activity`)
- Réutilisable—à partager entre plusieurs activités
  - Chaque instance de `Fragment` est exclusivement liée à l'activité hôte
  - Le code du `Fragment` définit le layout et le comportement
- Représente des sections d'une interface utilisateur

# Exemple : Les sélecteurs

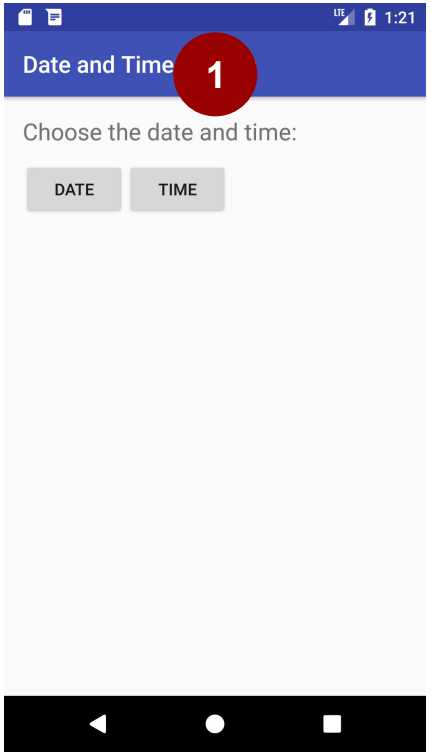
Les sélecteurs de date et d'heure :

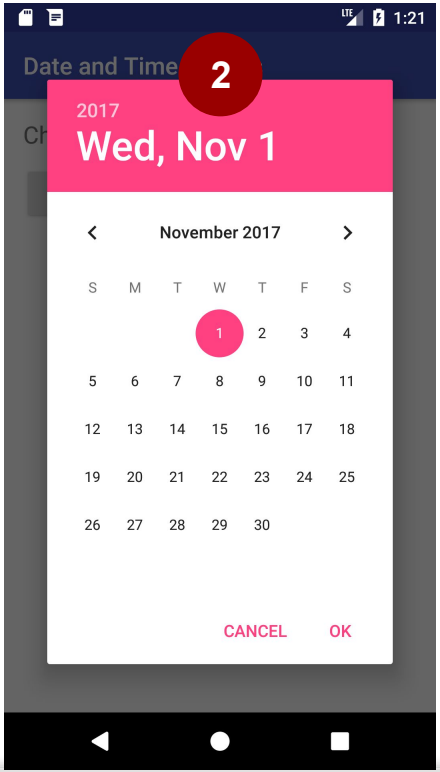
→ Hérite de `DialogFragment` (sous classe de `Fragment`)



# DialogFragment hébergé par une Activité

1. Activité avant l'ajout du fragment de sélecteur de date
2. Fragment sélecteur de Date apparaît en haut de l'activité



# Ajouter un fragment à une activité

- Partie statique de l'UI (dans le layout de l'activité):

  à l'écran pendant tout le cycle de vie de l'activité

- Partie dynamique de l'interface utilisateur :

  ajoutée et retirée pendant le déroulement de l'activité

# Avantages de l'utilisation de fragments

- Réutiliser un `Fragment` dans plus d'une activité
- Ajoutez ou supprimez dynamiquement selon les besoins
- Intégrer une mini-UI dans une activité
- Conserver les instances de données après un changement de configuration
- Représenter des sections d'un layout pour différentes tailles d'écran

# Étapes de l'utilisation d'un fragment

1. Créer une sous-classe de `Fragment`
2. Créer un layout pour le `Fragment`
3. Ajouter un fragment à une activité hôte
    - Statiquement dans le layout
    - Dynamiquement en utilisant les transactions du fragment

# Créer un Fragment

# Dans Android Studio…

- Développez **app > java** dans le projet
- Clique droit sur un package > New > Fragment > Fragment (Blank)
- Créer un fichier, hériter de la classe Fragment ou d'une sous classe.
  - **Inclure une méthode statique** `newInstance()` qui instancie le Fragment.
  -

# Nouveau Fragment

```
public class SimpleFragment extends Fragment {
    public SimpleFragment() {
        // Required empty public constructor
    }
    ...
}
```

# Nouveau Fragment

```
public static SimpleFragment newInstance(String param1) {
        SimpleFragment fragment = new SimpleFragment ();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        fragment.setArguments(args);
        return fragment;
}
```

# Hériter de la classe Fragment class

- Hériter de la classe `Fragment`
  - `public class SimpleFragment extends Fragment`
- Hériter d'une sous-classe spécifique de `Fragment` :
  - `DialogFragment`: Dialogue flottant (exemples : sélecteurs de date et d'heure)
  - `ListFragment`: Liste d'éléments gérés par un adapter
  - `PreferenceFragment`: Hiérarchie d'objets de préférence (utile pour les paramètres)

# Utiliser un layout pour un Fragment

# Créer un layout pour un Fragment

- **Créer le fichier layout XML**
- `La` callback `onCreateView() du Fragment` créé la vue
- Override la méthode pour “inflate” le layout du Fragment
  - Renvoyer l’élément root du layout

# Inflate le layout du Fragment (1)

```
@Override
public View onCreateView(LayoutInflater inflater,
            ViewGroup container, Bundle savedInstanceState) {
    // Inflate the fragment layout and return it as root view.
    return inflater.inflate(R.layout.fragment_simple,
                                        container, false);
}
```

# Inflate le layout du Fragment (2)

- Le layout du Fragment est inséré dans le conteneur ViewGroup de l'Activité
- `Le LayoutInflater` inflate le layout et renvoie la root View à l'Activité
- `Le Bundle savedInstanceState` sauvegarde l'instance précédente du Fragment

# Inflate le layout du Fragment (3)

```
return inflater.inflate(R.layout.fragment_simple, container, false);
```

- ID de la ressource du layout (`R.layout.fragment_simple`)
- `Le ViewGroup` qui sera le parent du layout (`container`)
- `Boolean`: Si le layout doit être attachée au parent
  - Devrait être à `false`
  - Si l'on ajoute le Fragment dans le code, ne pas mettre à `true` pour éviter un ViewGroup redondant.

# Ajouter un fragment à une activité





# Ajouter un fragment à une activité

- Ajout statique dans le layout de l'activité, visible pendant tout le cycle de vie de l'activité
- Ajouté (ou supprimé) dynamiquement, selon les besoins, au cours du cycle de vie de l'activité en utilisant des transactions de fragments.

# Ajouter un fragment de manière statique

1. Déclarer le fragment dans le layout de l'activité (`activity_main.xml`) en utilisant la balise `<fragment>`.
2. Spécifiez les propriétés du layout du fragment comme s'il s'agissait d'une vue. (*width, height, id*…)

# Exemple de fragment statique

Ajout de SimpleFragment au layout de l'activité :

```
<fragment android:name="com.example.appname.SimpleFragment"
            android:id="@+id/simple_fragment"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
```

# Ajouter un fragment de façon dynamique

1. Spécifier le ViewGroup pour le Fragment dans le layout
2. Instanciation du fragment dans l'activité
3. Instancier le FragmentManager
    - Utiliser `getSupportFragmentManager()` pour la compatibilité
4. Utiliser les transactions de fragments

# Spécifier le ViewGroup pour le Fragment

```
<FrameLayout
    android:id="@+id/fragment_container"
    android:name="SimpleFragment"
    tools:layout="@layout/fragment_simple"
    ... />
```

# Instancier le Fragment

1. Créez la méthode newInstance() dans le Fragment

```
public static SimpleFragment newInstance() {
      return new SimpleFragment();
}
```

2. Dans l'activité, instancier le Fragment en appelant newInstance() :

```
SimpleFragment fragment = SimpleFragment.newInstance();
```

# Instancier le FragmentManager

Dans une activité, récupérer l'instance du `FragmentManager` avec `getSupportFragmentManager()`:

```
FragmentManager fragmentManager = getSupportFragmentManager();
```

Préférer la Support Library —`getSupportFragmentManager()` plutot que `getFragmentManager()`—pour la compatibilité avec les versions antérieures d'Android.

# Utiliser les transactions de Fragment

Les opérations de Fragment sont regroupées dans une transaction:

- Commencer la transaction avec `beginTransaction()`
- Effectuer toutes les opérations sur le fragment (ajout, suppression, etc.)
- Terminer la transaction avec `commit()`

# Les opérations de transaction

- Ajouter un fragment à l'aide de `add()`
- Suppression d'un fragment avec `remove()`
- Remplacer un fragment par un autre en utilisant `replace()`
- Masquer et afficher un fragment avec `hide()` et `show()`
- Ajouter la transaction de fragment dans la pile arrière en utilisant `addToBackStack(null)`

# Exemple de transaction de fragment (1)

```
FragmentTransaction fragmentTransaction =
            fragmentManager.beginTransaction();
fragmentTransaction
            .add(R.id.fragment_container, fragment)
            .addToBackStack(null)
            .commit();
```

# Exemple de transaction de fragment (2)

Dans le code de la diapositive précédente :

- Les arguments de la méthode `add()` :
  - Le `ViewGroup` (`fragment_container`)
  - Le `fragment` à ajouter
- `addToBackStack(null)`:
  - Ajouter une transaction à la Back stack des transactions de fragments
  - La Back stack est gérée par l'activité
  - L'utilisateur peut appuyer sur le bouton "Back" pour revenir à l'état précédent.

# Exemple de transaction de fragment (3)

Suppression d'un fragment :

```
SimpleFragment fragment = (SimpleFragment) fragmentManager
                    .findFragmentById(R.id.fragment_container);
if (fragment != null) {
     FragmentTransaction fragmentTransaction =
                    fragmentManager.beginTransaction();
     fragmentTransaction.remove(fragment).commit();
}
```

# END