



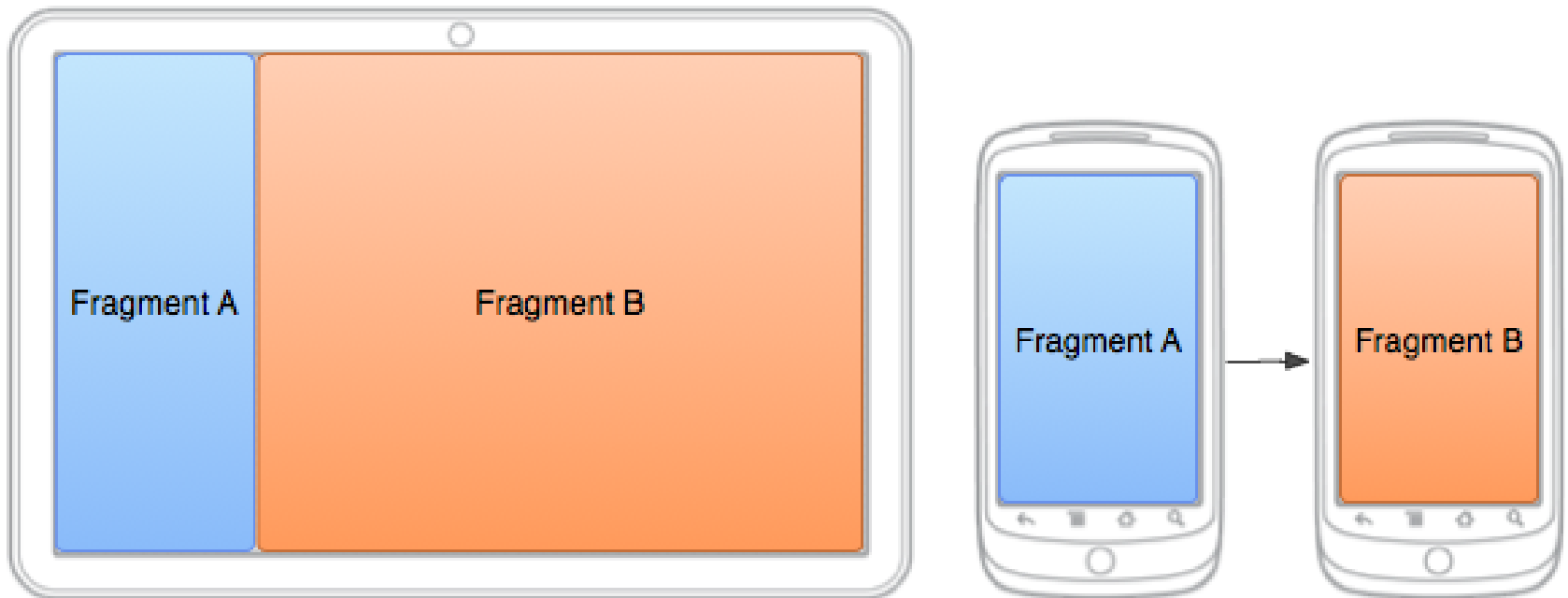
ANDROID

An Open Handset Alliance Project

Les fragments

G. Seront

Motivation



Les Fragments

- Un fragment est un morceau d'interface avec
 - Son layout
 - Son comportement (classe et cycle de vie)
- Un fragment est toujours inclus dans une activité
- Une activité contient 0 ou plusieurs fragments
- Une activité peut changer dynamiquement ses fragments



Création fragment

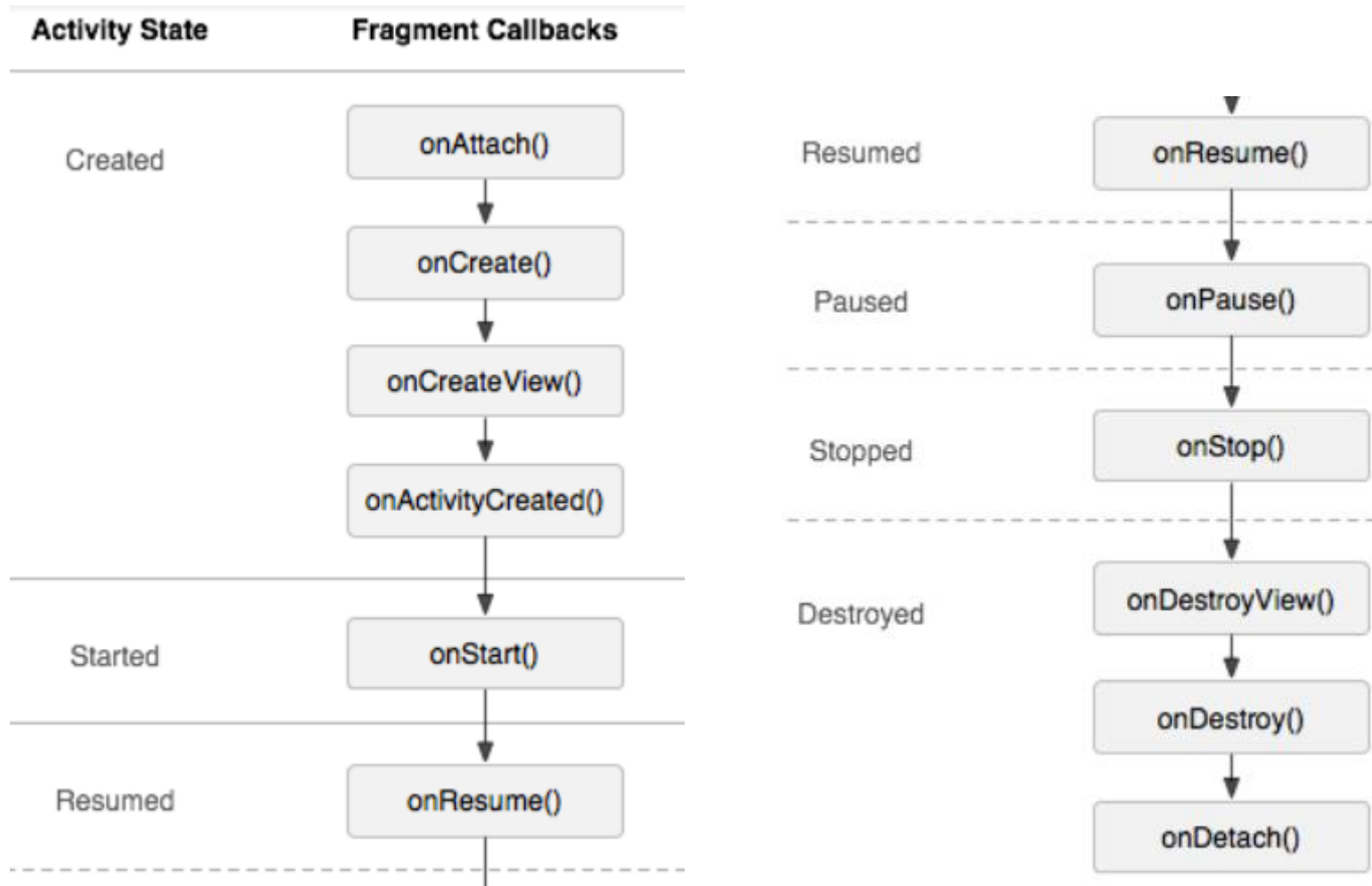
- Définir XML (comme pour une activité)
- Créer une classe pour le fragment

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.ViewGroup;

public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```



Cycle de vie fragment



Lien statique

Activité -> Fragment

- On l'inclus simplement dans le XML de l'activité


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment android:name="com.example.android.fragments.HeadlinesFragment"
        android:id="@+id/headlines_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

    <fragment android:name="com.example.android.fragments.ArticleFragment"
        android:id="@+id/article_fragment"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

</LinearLayout>
```

Classe du
fragment




Lien statique

Activité -> Fragment

- On ne pourra pas changer dynamiquement de fragment



Lien dynamique

Activité -> Fragment

- On met un conteneur pour le fragment dans le layout de l'activité:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```



Lien dynamique

Activité -> Fragment

- On charge le fragment dans le onCreate de l'activité hôte:

```
// Create a new Fragment to be placed in the activity layout
HeadlinesFragment firstFragment = new HeadlinesFragment();

// In case this activity was started with special instructions from an
// Intent, pass the Intent's extras to the fragment as arguments
firstFragment.setArguments(getIntent().getExtras());

// Add the fragment to the 'fragment_container' FrameLayout
getSupportFragmentManager().beginTransaction()
    .add(R.id.fragment_container, firstFragment).commit();
```



Lien dynamique

Activité -> Fragment

- Avec quelques précautions avant:

```
public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);

        // Check that the activity is using the layout version with
        // the fragment_container FrameLayout
        if (findViewById(R.id.fragment_container) != null) {

            // However, if we're being restored from a previous state,
            // then we don't need to do anything and should return or else
            // we could end up with overlapping fragments.
            if (savedInstanceState != null) {
                return;
            }
        }
    }
}
```



Remplacement fragment

- Code dans l'activité:

```
FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

- Notez l'utilisation du BackStack
- Permet la navigation par la touche back



Gestion des évènements

- Rappel deux méthodes:
 - Par listener
 - Dans le XML du fragment

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="Click me" />
```

- Appelle la méthode *onClick* dans **l'activité**; pas dans le fragment!
- **KO !**



Gestion des évènements

- On doit définir un listener dans le code pour appeler une méthode du fragment
- Code dans le *onCreateView* du fragment :

```
Button button = (Button) findViewById(R.id.myButton);

button.setOnClickListener(
    new Button.OnClickListener() {
        public void onClick(View v) {
            // Code to be performed when
            // the button is clicked
        }
    }
);
```



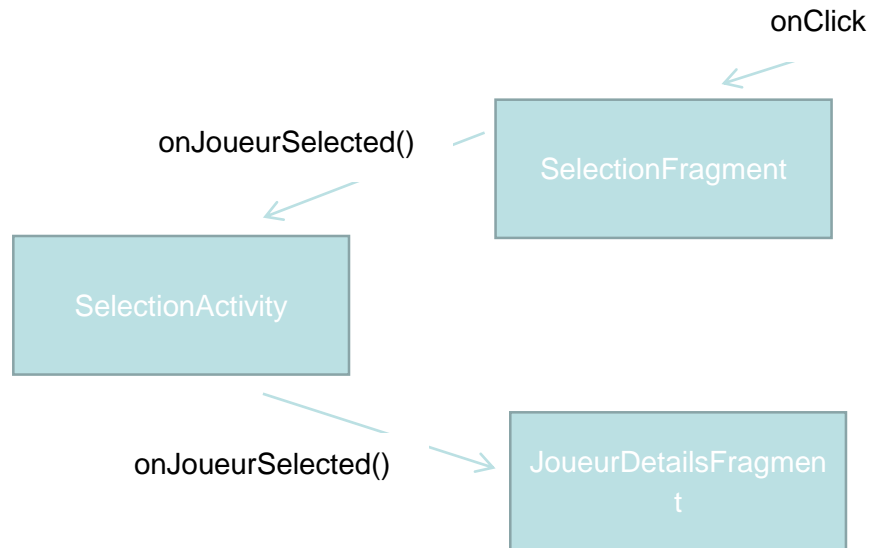
Communication entre fragments

- Un fragment doit être
 - Réutilisable
 - Complet (XML + comportement)
 - Indépendant
- Deux fragment ne peuvent donc jamais communiquer directement
- Ils doivent passer par l'activité hôte
- Et cela de façon découplée



Communication entre fragments

- Exemple:



Interface de communication

- Le fragment publie une interface que l'activité hôte implémentera
- On est donc indépendant de l'activité (découplé)

```
public class HeadlinesFragment extends ListFragment {
    OnHeadlineSelectedListener mCallback;

    // Container Activity must implement this interface
    public interface OnHeadlineSelectedListener {
        public void onArticleSelected(int position);
    }
}
```



Interface de communication

- Le fragment enregistre l'activité comme observer de cette interface

```
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);

    // This makes sure that the container activity has implemented
    // the callback interface. If not, it throws an exception
    try {
        mCallback = (OnHeadlineSelectedListener) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()
            + " must implement OnHeadlineSelectedListener");
    }
}
```



Interface de communication

- Lors d'un évènement, le fragment notifie l'activité hôte:

```
@Override
public void onItemClick(ListView l, View v, int position, long id) {
    // Send the event to the host activity
    mCallback.onArticleSelected(position);
}
```



Interface de communication

- L'activité hôte implémente l'interface:

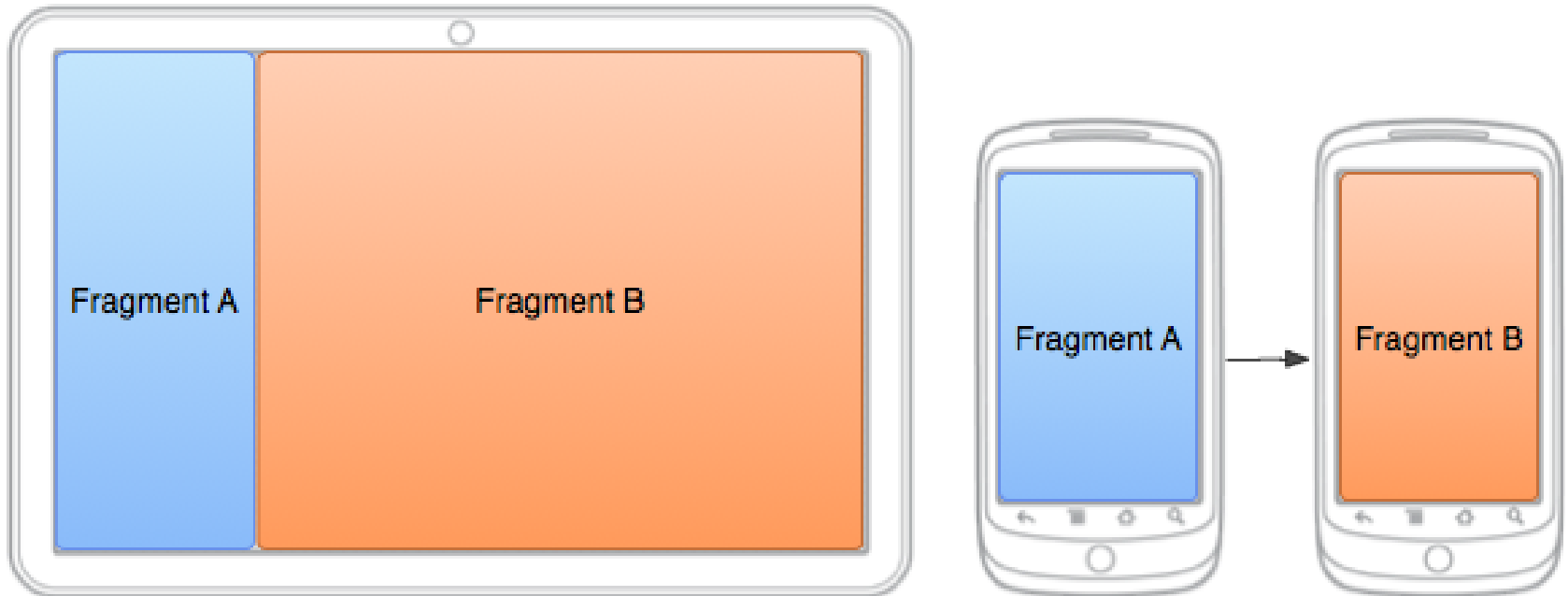
```
public static class MainActivity extends Activity
    implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article
    }
}
```



Communication

Activity => Fragment



Communication

Activity => Fragment

- L'activité hôte notifie le deuxième fragment:

```
public static class MainActivity extends Activity
    implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article

        ArticleFragment articleFrag = (ArticleFragment)
            getSupportFragmentManager().findFragmentById(R.id.article_fragment);

        if (articleFrag != null) {
            // If article frag is available, we're in two-pane layout...

            // Call a method in the ArticleFragment to update its content
            articleFrag.updateArticleView(position);
        }
    }
}
```



Communication

Activity => Fragment

- Si l'on doit changer de fragment:

```

} else {
    // Otherwise, we're in the one-pane layout and must swap frags...

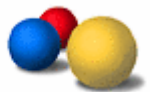
    // Create fragment and give it an argument for the selected article
    ArticleFragment newFragment = new ArticleFragment();
    Bundle args = new Bundle();
    args.putInt(ArticleFragment.ARG_POSITION, position);
    newFragment.setArguments(args);

    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

    // Replace whatever is in the fragment_container view with this fragment,
    // and add the transaction to the back stack so the user can navigate back
    transaction.replace(R.id.fragment_container, newFragment);
    transaction.addToBackStack(null);

    // Commit the transaction
    transaction.commit();
}

```



Et MVC dans tout ça?

- Prenons le temps de réfléchir, ...

