WP SmartHome
Aktivitäten Fragments

Prof. Dr. Ing. Birgit Wendholt

Fragment

- Teil des UIF einer Aktivität / Fragments ohne UI: unsichtbare “Worker” eine Aktivität
- eingebettet in eine Aktivität
- Kombination mehrerer Fragments in einer einzelnen Aktivität möglich

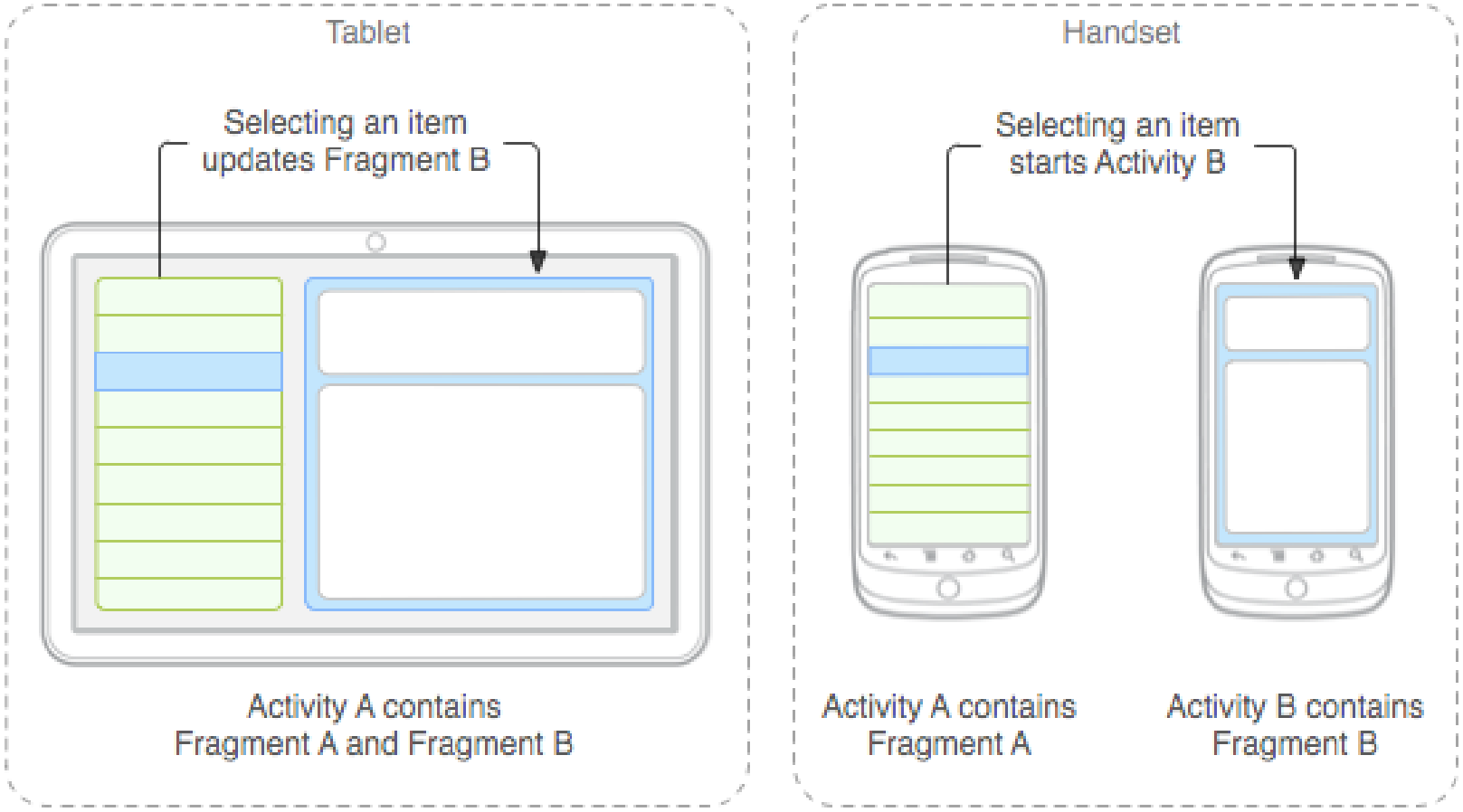
- Module mit
  - eigenem Lebenszyklus,
  - eigenem Event Handler
  - Transaktionen für das Hinzufügen und Entfernen zu / von einer Aktivität zur Laufzeit

- Lebenszyklus abhängig vom Lebenszyklus der Host Aktivität

- Verwaltung über den Backstack ➔ Rückwärts-Navigation über Fragments mittels Back Button
- Teil einer ViewGroup in der View Hierarchie der Aktivität

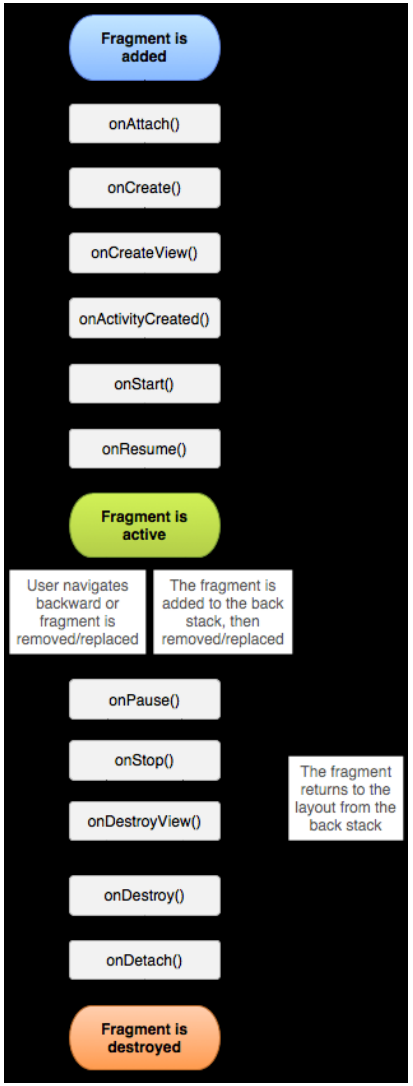
- Einfügen in die ViewHierarchie:
  - in der Layout Datei mittels des <fragment> Element
  - programmatisch

Konfigurieren von Fragments für unterschiedliche Geräte



Fragment Lebenszyklus

- **Erzeugen**: Subklasse von Fragment erstellen
- Callback Methoden ähnlich einer Aktivität
- **Migration**: Callback-Methoden der Aktivität → Callback-Methoden des Fragments

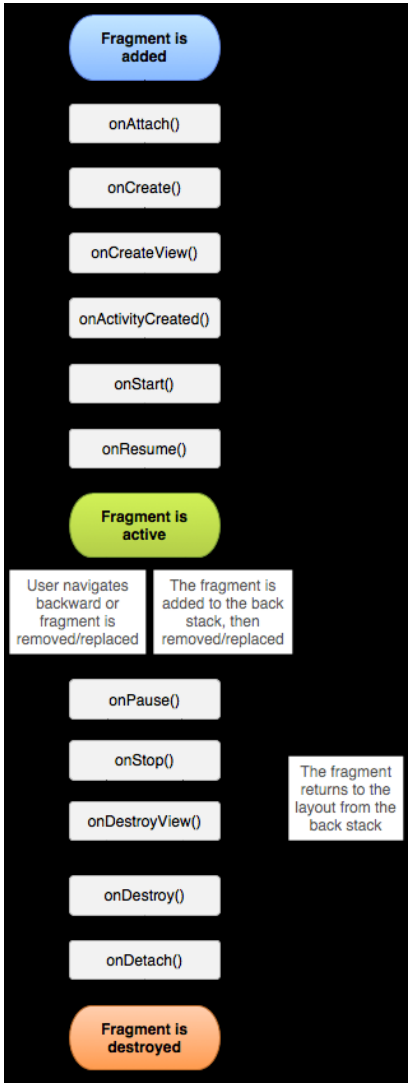


Fragment Lebenszyklus

- Minimal: Implementieren der Callbacks

- **onCreate**:
  - Initialisierung der wesentlichen Komponenten, die im Zustand stopped oder paused erhalten bleiben sollen

- **onCreateView**:
  - Anzeige des Fragment UIF's
  - Ergebnis der Methode ein View Objekt, die Root View des Fragments
  - null, für Fragments ohne UI
- **onPause**:
  - Aufruf, wenn der Benutzer das Fragment verlässt
  - Zeitpunkt, zu dem Änderungen persistiert werden sollten



Vordefinierte Fragments

- **DialogFragment**:
  - für Dialoge
  - Alternative zu Dialog Helfer Methoden einer Aktivität.

- **ListFragment**:
  - Liste von Elementen, die über einen Adapter gemanaged werden (→ SimpleCursorAdapter)
  - ähnlich der ListActivity
  - Methoden für das Verwalten von List Views, z.B. Eventhandler der onListItemClick() Callback

- **PreferenceFragment**:
  - Hierarchie von Preference Objekten, die als Liste angeboten werden
  - für eine "settings" Aktivität einer Applikation

Fragment Layout implementieren

- im onCreateView():
  - **Defaultimplementierung für** ListFragment liefert eine ListView
  - Erzeugen aus einer Layout Ressource mittels eines LayoutInflater Objektes
- inflate:
  - resource ID des Layouts
  - ViewGroup des Parents (Container) vererbt Layout Parameter
  - boolean: steuert, ob das Layout während der Expansion an die ViewGroup angehängt wird (hier **false**, da das System das Layout bereits dem Container hinzufügt)

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

Fragments einer Aktivität hinzufügen deklarativ


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

Fragments einer Aktivität hinzufügen programmatisch

- Fragment Transaktionen: add, remove, replace mittels des API's der Klasse FragmentTransaction

```
FragmentManager fragmentManager = getFragmentManager()
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Fragment Management

- **FragmentManager**:
  - Fragments einer Aktivität auffinden: findFragmentById() (Fragments mit UI) findFragmentByTag() (Fragments ohne UI)
  - Fragments vom Backstack löschen (popBackStack()) → simuliert das ***Back*** durch den Benutzer
  - Registrieren eines Listeners auf Änderungen im Backstack addOnBackStackChangedListener()
  - Öffnen einer Fragmenttransaktion

Fragment Transaktionen

- Abfolge von Änderungen auf Fragments einer Aktivität
  - mit **commit** abgeschlossen
  - Methoden: **add**, **remove**, **replace, addToBackStack**
- API der Klasse FragmentTransaction
- **addToBackStack** → Back-Navigation über Fragment Änderungen

```
// Create new fragment and transaction
Fragment newFragment = new ExampleFragment();
FragmentTransaction transaction = getFragmentManager().beginTransaction();

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

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

Kommunikation mit der Aktivität

- Instanzen von Fragments sind direkt mit der enthaltenen Aktivität verbunden

- Zugriff eines Fragments auf die Aktivität:

```
View listView = getActivity().findViewById(R.id.list);
```

- Zugriff einer Aktivität auf ein Fragment:

```
nt = (ExampleFragment) getFragmentManager().findFragmentById(R.id.example_fragment);
```

Event Callbacks zur Aktivität

- Definition eines Callback Interfaces im Fragment, das die Host Aktivität implementiert

- Beispiel: eine "News" Applikation mit 2 Fragments
  - Fragment A: Liste von Artikeln
  - Fragment B: Anzeige der Artikel
  - Fragment A muss der Aktivität mitteilen, wenn ein Listelement selektiert wurde, damit das Element in Fragment B angezeigt werden kann.
  - **dazu**: Definition eines **OnArticleSelectedListener** Interfaces in Fragment A:

```
public static class FragmentA extends ListFragment {
    ...
    // Container Activity must implement this interface
    public interface OnArticleSelectedListener {
        public void onArticleSelected(Uri articleUri);
    }
    ...
}
```

Event Callbacks zur Aktivität

- Host Aktivität:
  - implementiert das OnArticleSelectedListener Interface
  - informiert in der Methode onArticleSelected() Fragment B über das Event von Fragment A.
- Sicherstellen, dass die Host Aktivität das Interface implementiert:
  - in der Methode onAttach() von Fragment A wird ein OnArticleSelectedListener instantiiert, durch Casten der in onAttach übergebenen Aktivität auf das Interface

```
public static class FragmentA extends ListFragment {
    OnArticleSelectedListener mListener;
    ...
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnArticleSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString() + " must implement On
        }
    }
    ...
}
```

Nutzen des Aktivität Event Callbacks

- Element der Liste wird gewählt → Fragment A ruft den EventHandler der Aktivität auf (mListener)

```
public static class FragmentA extends ListFragment {
    OnArticleSelectedListener mListener;
    ...
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        // Append the clicked item's row ID with the content provider Uri
        Uri noteUri = ContentUris.withAppendedId(ArticleColumns.CONTENT_URI, id);
        // Send the event and Uri to the host activity
        mListener.onArticleSelected(noteUri);
    }
    ...
}
```

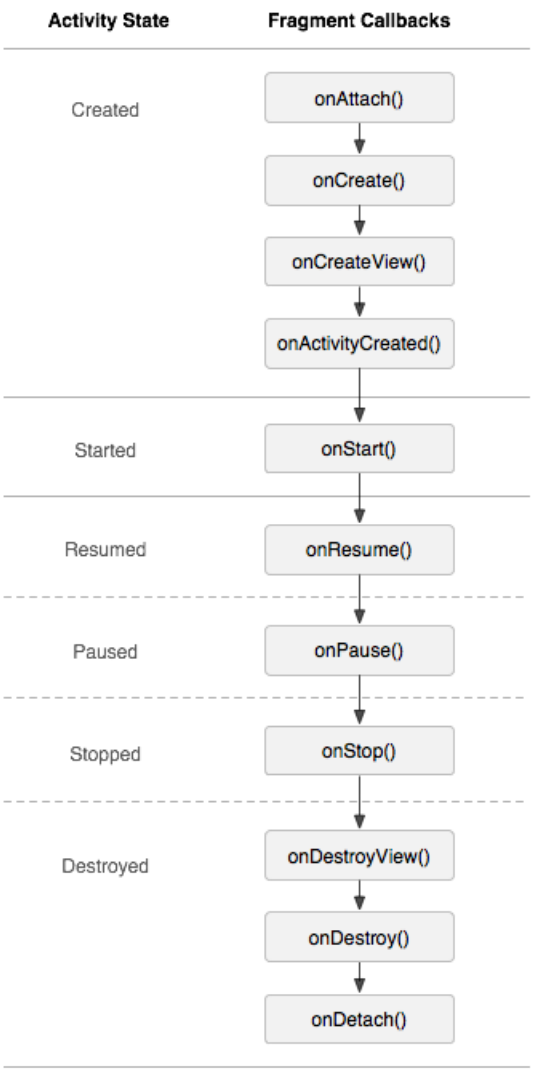
Implementierung des Event Handler Callbacks

- Element der Liste wird gewählt → Fragment A ruft den EventHandler der Aktivität auf (mListener)
- id Parameter ist die Zeilen ID des selektierten Items: Aktivität benutzt diese um den zugehörigen Artikel zu ermitteln

```
public static class FragmentA extends ListFragment {
    OnArticleSelectedListener mListener;
    ...
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        // Append the clicked item's row ID with the content provider Uri
        Uri noteUri = ContentUris.withAppendedId(ArticleColumns.CONTENT_URI, id);
        // Send the event and Uri to the host activity
        mListener.onArticleSelected(noteUri);
    }
    ...
}
```

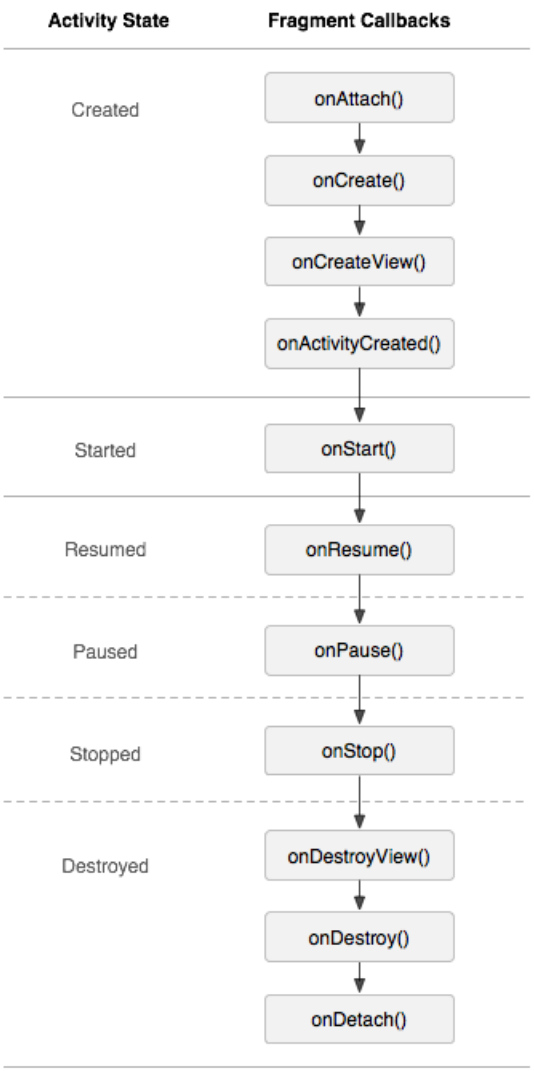
Fragment Lebenszyklus

- **Resumed**: sichtbar in der laufenden Aktivität
- **Paused**: eine andere Aktivität im Vordergrund besitzt den Fokus
  - Host-Aktivität des Fragments noch sichtbar
  - Aktivität im Vordergrund transparent oder nutzt nur eine Teil des Bildschirms
- **Stopped**: Fragment nicht mehr sichtbar, aber noch im Speicher
  - Host Aktivität wurde gestoppt oder Fragment aus der Aktivität entfernt und auf den BackStack gelegt
  - wird gelöscht, wenn Host Aktivität gelöscht wird



Zustandssicherung für Fragments

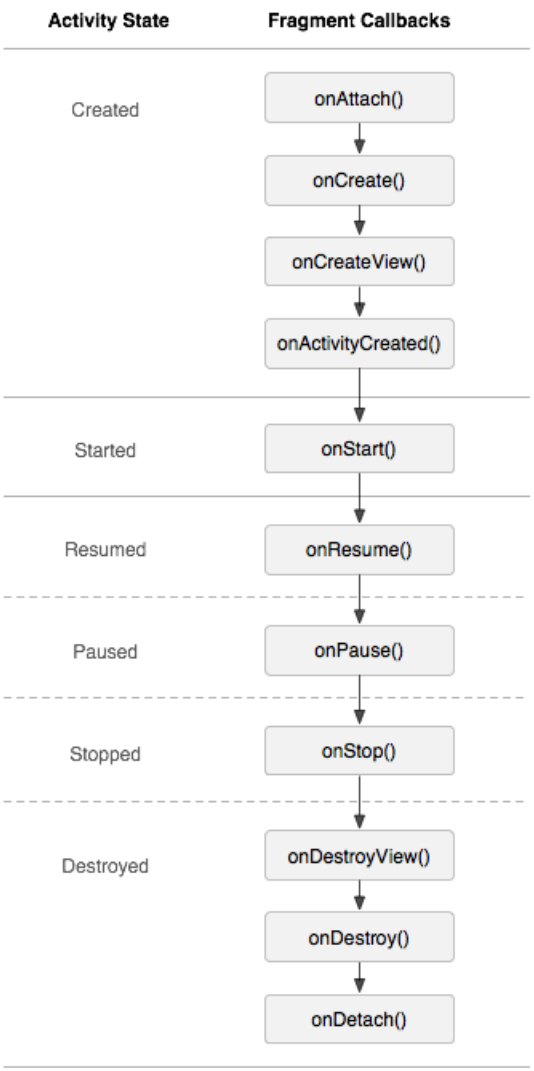
- wenn der Prozess der Host-Aktivität zerstört wurde und die Aktivität neu erzeugt wird
- wie für Aktivitäten über ein **Bundle** Objekt
- **Speichern**: ***onSaveInstanceState***
- **Wiederherstellen*: onCreate(), onCreateView(), onActivityCreated()***
- Unterschied im Lebenzyklus der Aktivität zu Fragment:
  - Aktivität: System managed den BackStack
  - Fragment: Aktivität managed den BackStack für Fragmensts → **addToBackStack**



Koordination mit dem Lifecyle der Aktivität

- Lifecycle Callbacks einer Aktivität → entsprechender Callback des Fragments (onResume)

- Zusätzliche Callbacks
  - **onAttach**() Fragment wird mit der Aktivität verbunden (add)
  - **onCreateView**() UI wird aufgebaut
  - **onActivityCreated**() aufgerufen in der onCreate Methode der Aktivität
  - **onDestroyView**() aufgerufen, wenn die ViewHierarchie, zu dem das Fragments gehört gelöscht wird.
  - **onDetach**() Verbindung zur Aktivität wird aufgehoben



Fragments und Action Bar

- Menüeinträge im Options Menü / Action Bar einer Aktivität → Fragment implementiert die onCreateOptionsMenu()
- **vorher**: setHasOptionsMenu() im onCreate() des Fragments
- Fragment Items werden an vorhandene Menu Items angehängt
- Erzeugen von Kontextmenüs → registerForContextMenu()
- Methode onCreateContextMenu des Fragments wird aufgerufen, wenn der Benutzer das Menu öffnet.
- Methode onContextItemSelected wird aufgerufen, wenn der Benutzer ein Item selektiert.