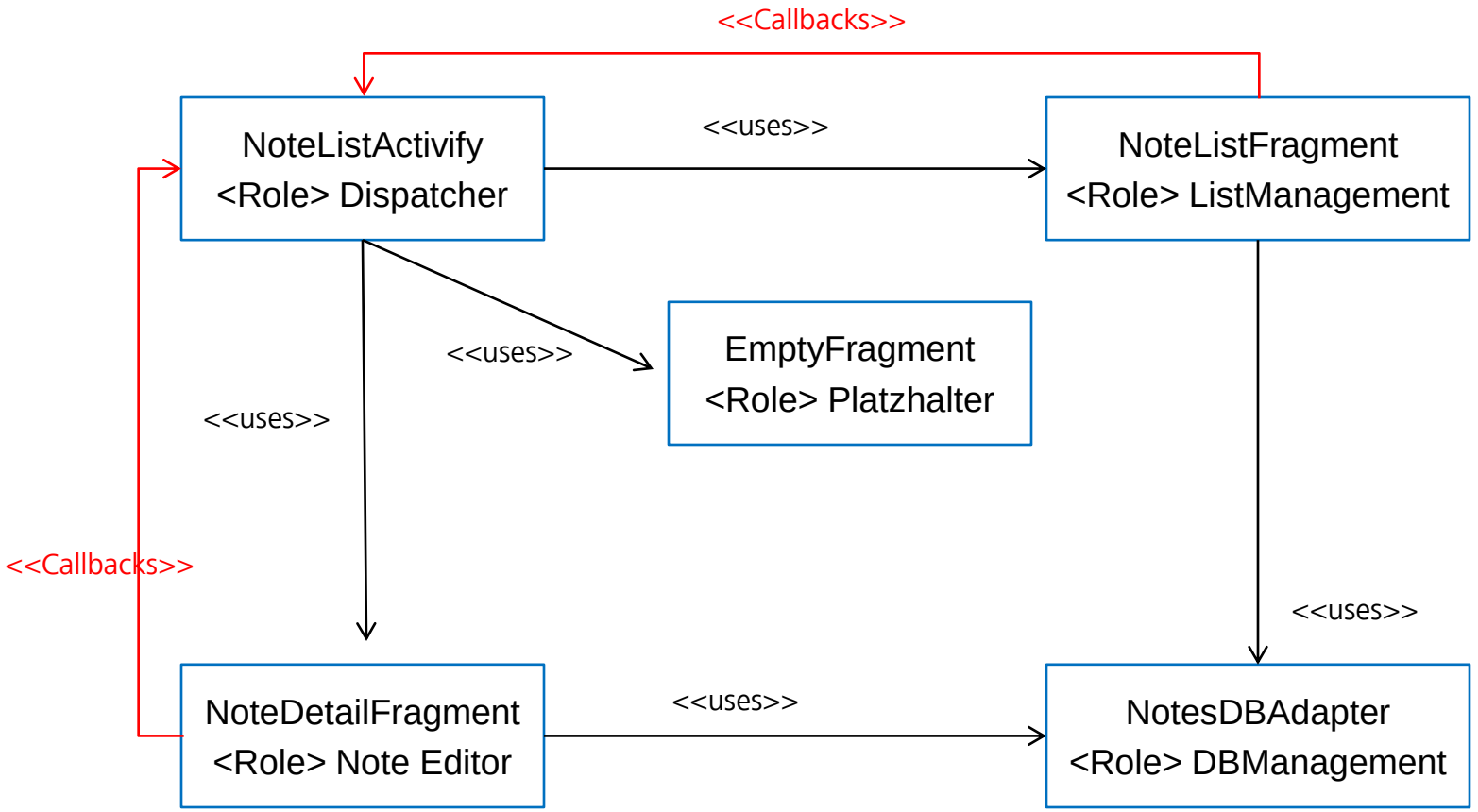
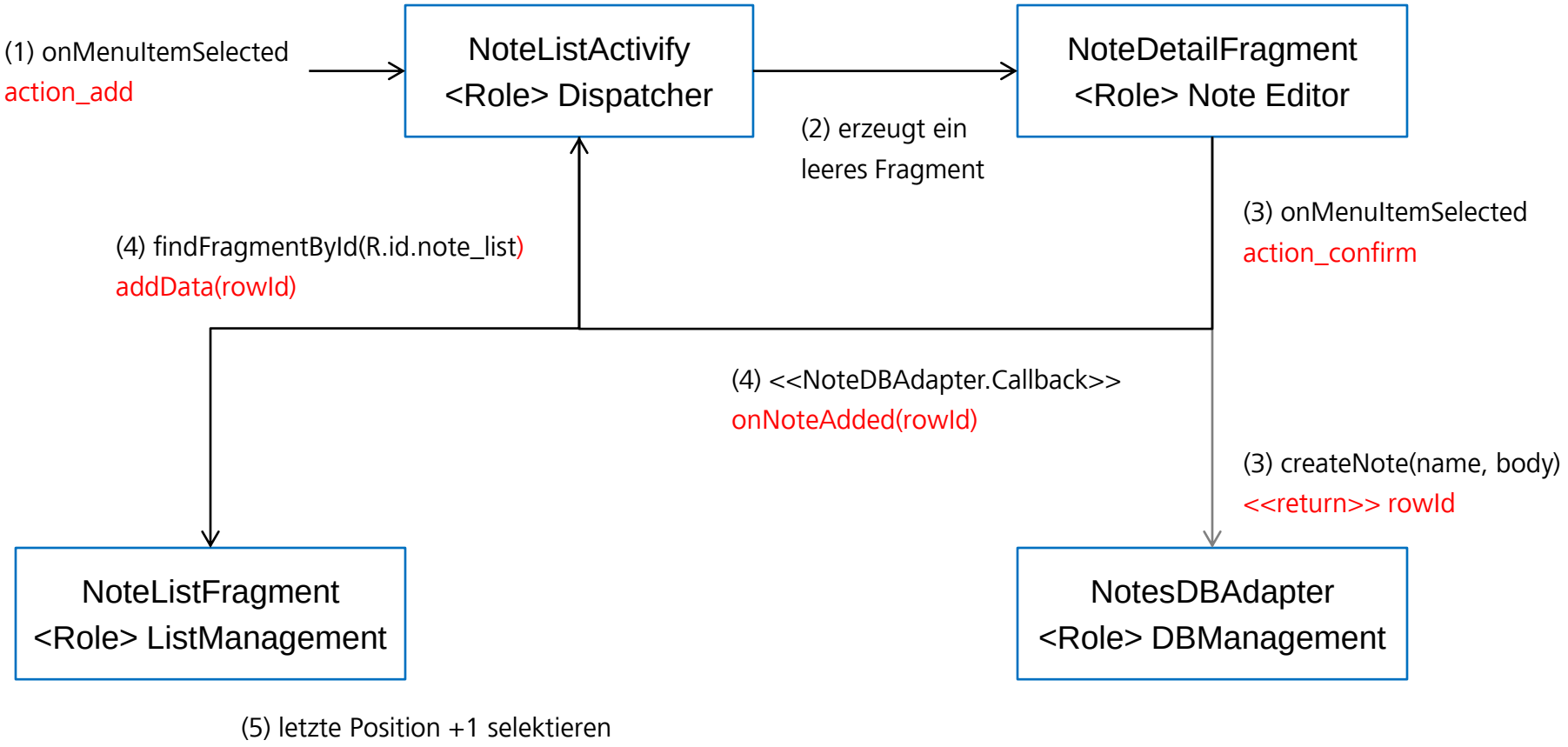
WP SmartHome
Notepad Example mit Fragments

Prof. Dr. Ing. Birgit Wendholt

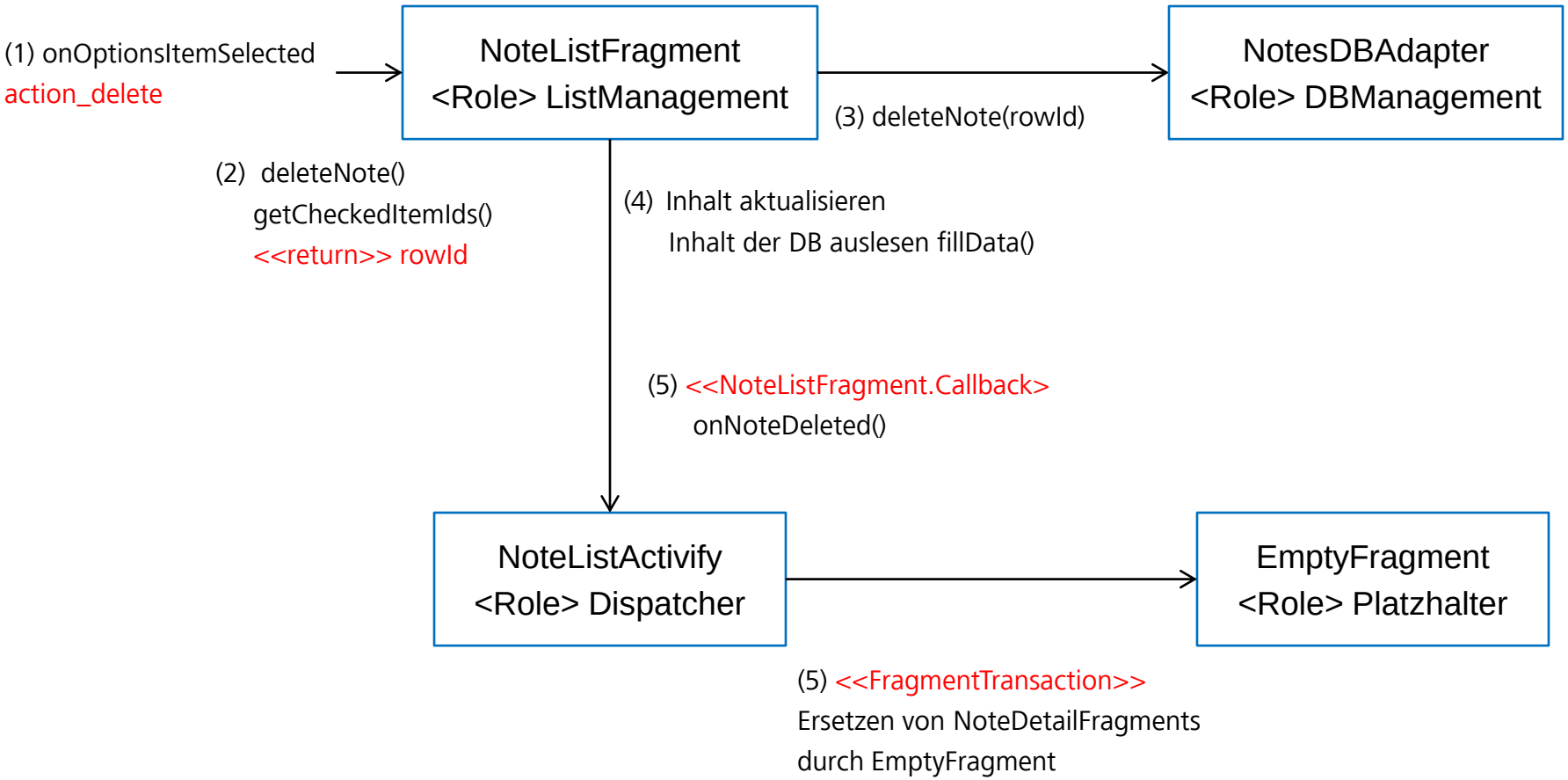
Übersicht über die Klassen



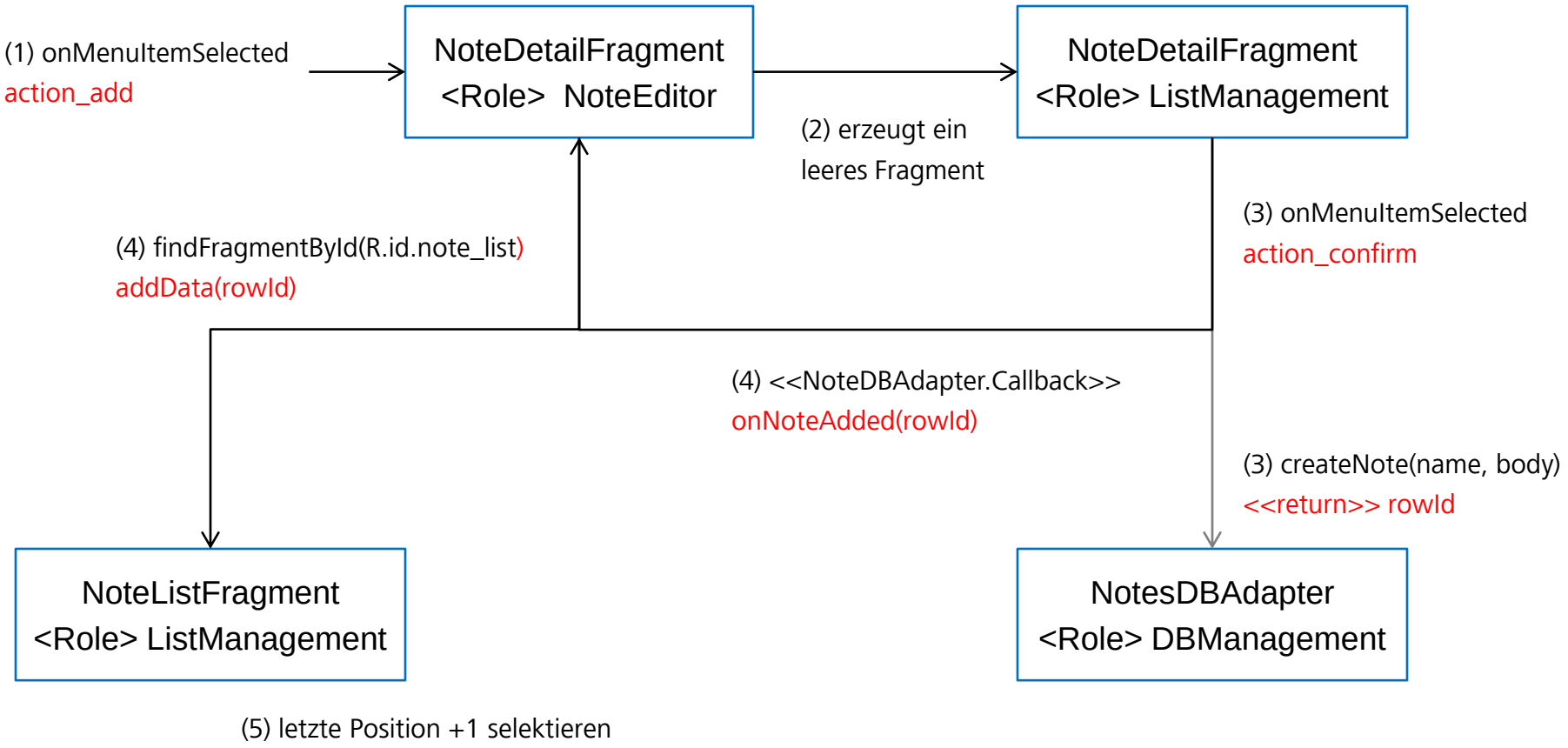
Szenario – Notes Erzeugen



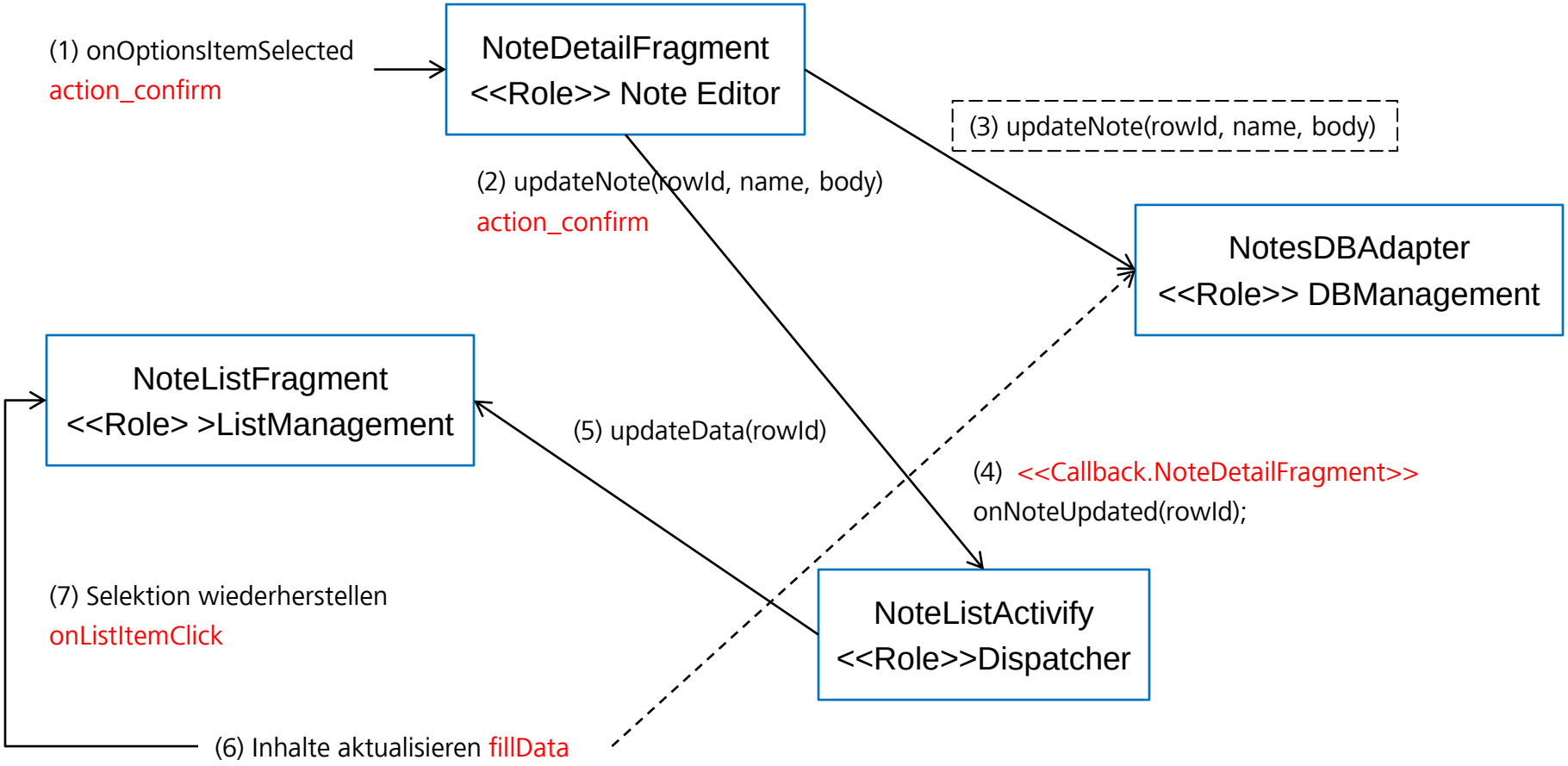
Szenario – Notes Löschen



Szenario – Notes Editieren



Szenario – Notes Editieren



Dynamisches Tauschen von Fragments

Situationen

1. Selektion einer Note in der Liste
2. Erzeugen eíner neuen Note
3. Löschen einer Note

Aktionen

1. NoteListFragment onItemClickEvent → informiert NoteListActivity → erzeugt ein NoteDetailFragment für das selektierte Element
2. NoteListActivity onMenuItem Selected Event (action_added) → erzeugt ein leeres NoteDetailFragment
3. NoteListFragment onOptionsItemSelected Event (action_delete) → NoteListActivity ersetzt NoteDetailFragment durch EmptyFragment

Dynamisches Tauschen von Fragments

1. Selektion einer Note in der Liste
   - `NoteListFragment onListItemClick:` ruft über den Callback die Methode `onItemSelected` der `NoteListActivity`
   - `NoteListActivity` ersetzt Inhalt des Containers (`R.id.note_detail_container)` durch ein `NoteDetailFragment` für die selektiere `id`

```
@Override
public void onItemSelected(long id) {
    if (id < 0) return;
    Bundle arguments = new Bundle();
    arguments.putLong(NoteDetailFragment.ARG_ITEM_ID, id);
    NoteDetailFragment fragment = new NoteDetailFragment();
    fragment.setArguments(arguments);
         getSupportFragmentManager().beginTransaction()
         .replace(R.id.note_detail_container, fragment).commit();
}
```

Fragment Container

- Definiert im Layout der `NoteListActivity`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    ...
    tools:context=".NoteListActivity">
    <fragment android:name="com.example.notepad.NoteListFragment"
        android:id="@+id/note_list"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <FrameLayout android:id="@+id/note_detail_container"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3" />

</LinearLayout>
```

Dynamisches Tauschen von Fragments

2. Erzeugen einer neuen Note
   - `NoteListActivity` `noteCreateFragment:` Container (`R.id.`*`note_detail_container`*`)` wird mit einem leeren `NoteDetailFragment` ersetzt

```
private void noteCreateFragment() {

    NoteDetailFragment fragment = new NoteDetailFragment();
    fragment.setArguments(new Bundle());
    getSupportFragmentManager().beginTransaction()
          .replace(R.id.note_detail_container, fragment).commit();
}
```

Dynamisches Tauschen von Fragments

3. Löschen einer Note
   - `NoteListFragment` `onOptionsItemSelected` Event (action_delete) →
   - `NoteListFragment` ruft über den Callback die Methode `deleteNote` der `NoteListActivity` auf
   - `NoteListActivity` setzt im Container ein leeres Fragment ein

```
@Override
public void onNoteDeleted() {
    EmptyFragment ef = new EmptyFragment();
    getSupportFragmentManager().beginTransaction()
         .replace(R.id.note_detail_container, ef).commit();
};
```

Shared ActionBars

- bietet Benutzer Aktionen und Navigation und enthält die wichtigsten Aktionen einer Aktivität / Applikation

- seit API Level 11 für Android Theme.Holo und abgeleitete

- **Aufgaben**:
  - Identifikation der Applikation (z.B. Logo)
  - Konsistente Navigation: Tab Navigation für Fragmente
  - enthält die wesentlichen Aktionen einer Akktivität bzw. einer Applikation – Einträge im Optionsmenu können in die ActionBar übertragen werden.
  - Menu Einträge, die nicht als Action Item deklariert sind, werden in das Overflow Menu eingetragen

- Fragments
  - können Action Items in die Action Bar eintragen
  - können auf Actions, die von der Aktivität eingetragen wurden im `onOptionsItemSelected` Callback reagieren

Shared ActionBars

Menu-Layout Datei der Aktivität `NoteListActivity`

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_add"
        android:icon="@drawable/ic_action_add"
        android:showAsAction="always"
        android:title="@string/add">
    </item>
    <item android:id="@+id/action_confirm" android:showAsAction="always"
   android:icon="@drawable/ic_action_confirm" android:title="ok">
    </item><item
        android:id="@+id/action_delete"
        android:icon="@drawable/ic_action_delete"
        android:showAsAction="always"
        android:title="@string/delete">
    </item>
</menu>
```

Action Items hinzufügen

- Attribut `android.showAsAction`: wandelt einen Menu-Eintrag in ein Action Item
- `ifRoom|withText` zeigt den Text in der ActionBar an, wenn der Platz ausreichen ist
- `always`: zeigt einen Menüeintrag immer als Action Item an

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_save"
          android:icon="@drawable/ic_menu_save"
          android:title="@string/menu_save"
          android:showAsAction="ifRoom|withText" />
</menu>
```

Action Items hinzufügen

- direkter Zugriff auf Elemente des Optionsmenüs → Deklaration als Action Item
- Action Items = Icon oder Text
- Menu Items ohne Action Items erscheinen im Overflow Menu
- Action Bar und Overflow Menu werden in `onCreateOptionsMenu()` erzeugt

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.note_list_menu, menu);
    return true;
}
```

Shared ActionBars

- Beispiel: `NoteDetailFragment`
  - reagiert im `onOptionsItemSelected` Callback auf die Aktion `action_confirm`

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.action_confirm:
    if (mItem == null)
         createNote(mNoteName.getText().toString(), mNoteBody.getText()
                 .toString());
    else
         updateNote(mItem, mNoteName.getText().toString(), mNoteBody
                 .getText().toString());
    break;
    default:break; }
    return super.onOptionsItemSelected(item);
}
```

Arbeiten mit ListFragments

- Selektieren des letzten Eintrages nach dem Erzeugen einer neuen Note
  - Anzahl der Elemente des Adapter → Position des letzten eingetragenen Elementes
  - `setActivatedPosition`: stellt sicher, dass die Postion gültig ist und selektiert das Element an der Position
  - `getCheckItemListPosition`: gibt die Position des selektieren Elementes
  - `getCheckedItemIds`: liefert die Id des selektierten Elementes

```
public void addData(Long rowId) {
    int lastPos = getListAdapter().getCount();
    fillData();
    setActivatedPosition(lastPos);
    long[] ids = getListView().getCheckedItemIds();
    if (ids.length > 0 && ids[0] == rowId ){
         onListItemClick(getListView(), getView(),
              getListView().getCheckedItemPosition(), ids[0]);
    }
}
```

Arbeiten mit ListFragments

- Selektieren des letzten Eintrages nach dem Update einer Note
  - `getCheckItemListPosition`: gibt die Position des selektieren Elementes
  - `setActivatedPosition`: stellt sicher, dass die Postion gültig ist und selektiert das Element an der Position
  - `getCheckedItemIds`: liefert die Id des selektierten Elementes

```
public void updateData(Long rowId) {
    int lastPos = getListView().getCheckedItemPosition();
    fil  lData();
    setActivatedPosition(lastPos);
    long[] ids = getListView().getCheckedItemIds();
    if (ids.length > 0 && ids[0] == rowId ){
    onListItemClick(getListView(), getView(),
         getListView().getCheckedItemPosition(), ids[0]);
    }
}
```

Styles

- Style:
  - Sammlung von Eigenschaften, die das Aussehen und Format einer View beeinflussen
  - spezifiziert Eigenschaften wie Höhe, Fontfarbe, Fontgröße, Hintergrundfarbe etc.
  - definiert in einer XML Datei getrennt vom Layout
  - für einzelne Views über das android:style Attribut
  - für die Applikation/Aktivität über das **`android:theme`** Attribut im Manifest

```
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
</application>
```

Styles

- Style:
  - Sammlung von Eigenschaften, die das Aussehen und Format einer View beeinflussen
  - spezifiziert Eigenschaften wie Höhe, Fontfarbe, Fontgröße, Hintergrundfarbe etc.
  - definiert in einer XML Datei getrennt vom Layout
  - für einzelne Views über das android:style Attribut
  - für die Applikation/Aktivität über das **`android:theme`** Attribut im Manifest

```
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
</application>
```

Styles

- Zuordnung der Namens AppTheme zum verfügbaren Android Theme: in `styles.xml` Dateien für unterschiedliche API Level in `values` Verzeichnissen

- Bsp: Api Level 11 → `values-11/styles.xml`

```
<resources>
    <style name="AppTheme" parent="android:Theme.Holo.Light" />
</resources>
```

IconSets für Theme Holo.Light

- Unter http://developer.android.com/design/downloads/index.html können die zum Theme passenden Icons heruntergeladen werden

- Hinzufügen zum Projekt: New → Other → Android → Android Icon Set

- **Ergebnis**: Erzeugt für die verschiedenen Auflösungen ein Icon für das gewählte Theme