Smart Home Control
Navigationstemplates für einen LP Controller

Prof.Dr.Ing.Birgit Wendholt

Inhalt

- Master-Detail Flow
- Aufbau der Android Projekte (Activities, Fragments, Klassen)
- laterale Navigation mit Tabulatoren (→ Android Projekt ***LPControlTabSample***)
- laterale Navigation über ViewPager (→ Android Projekt ***LPControlPagerSample***)
- laterale Navigation mit ViewPager und Tabulatoren (→ Android Projekt ***LPControlTabPagerSample***)
- Up-Navigation in der ***ControlActivity*** zur Liste der Räume (Version für Smartphones im Designmuster Master-Detail Flow)
- ActionBar Kompatibilität

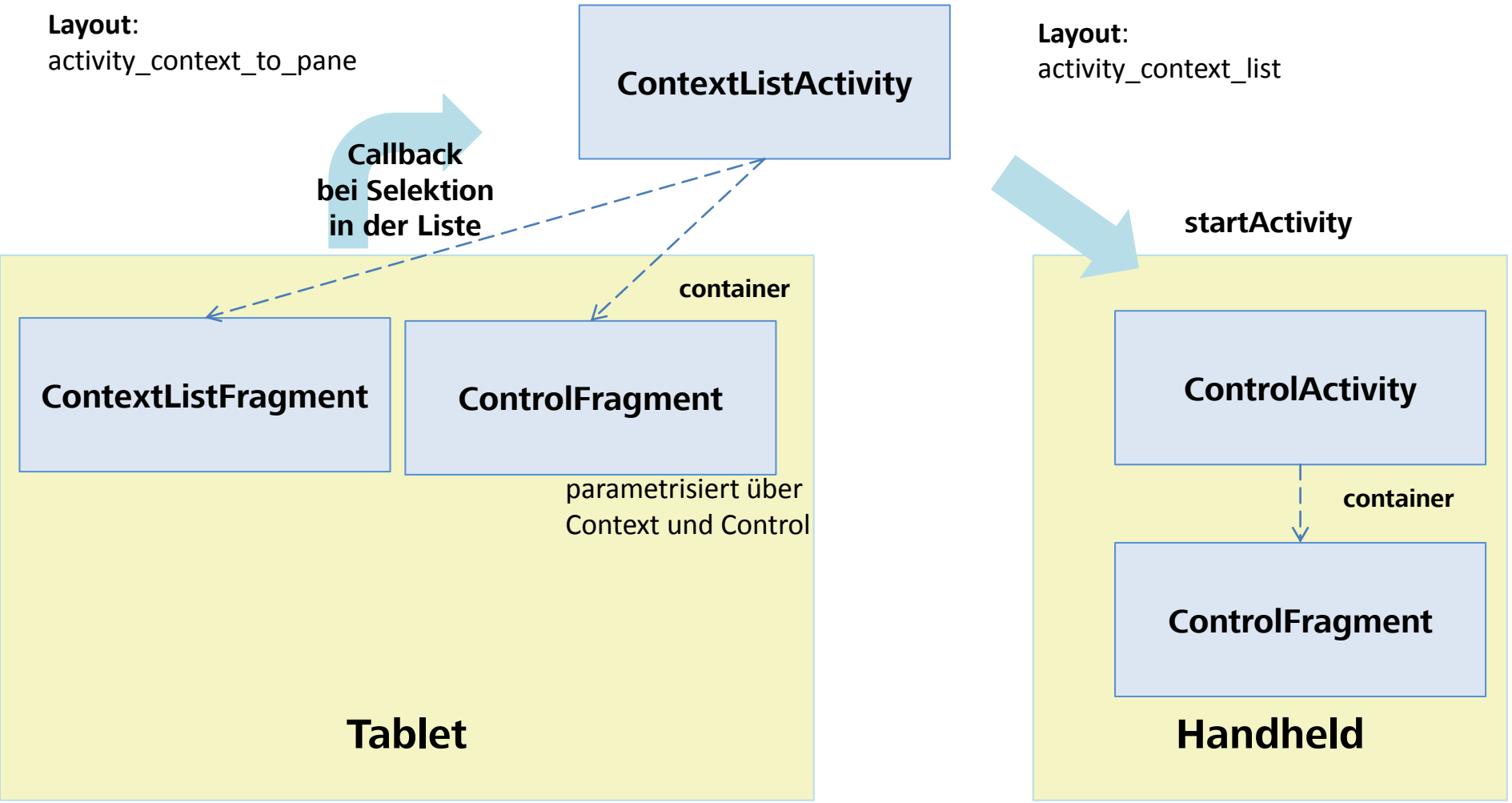
Master Detail Flow

- **Aufbau**:
  - Master Bildschirm → eine ListView zur Selektion von Elementen
  - Detail Bildschirm → ein Editor für die Elemente

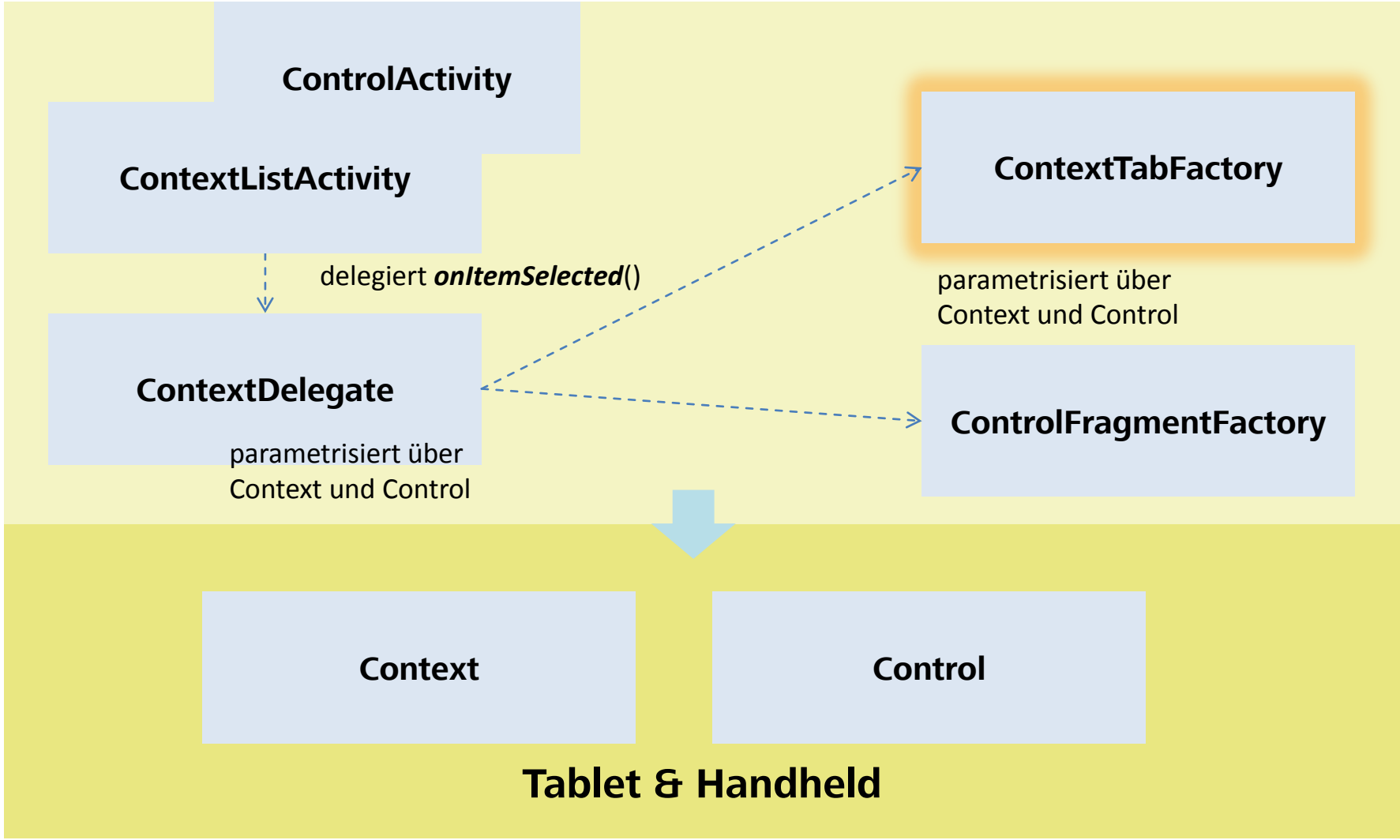
- **Handheld**:
  - Implementierung als Abfolge zweier Aktivitäten („descendant mavigation“)

- **Tablets**:
  - Darstellung von Master und Detail-View auf einem Bildschirm
  - descendant Navigation wird durch das Austauschen von Fragments in der Detail-View implementiert.
- Eclipse:
  - Muster zum Master-Detail Flow erzeugt Aktivitäten und Fragments für Handheld und Tablet
  - Ziel: Implementierung der GUI’s in Fragments zur Wiederverwendung für Handheld Tablet Lösung

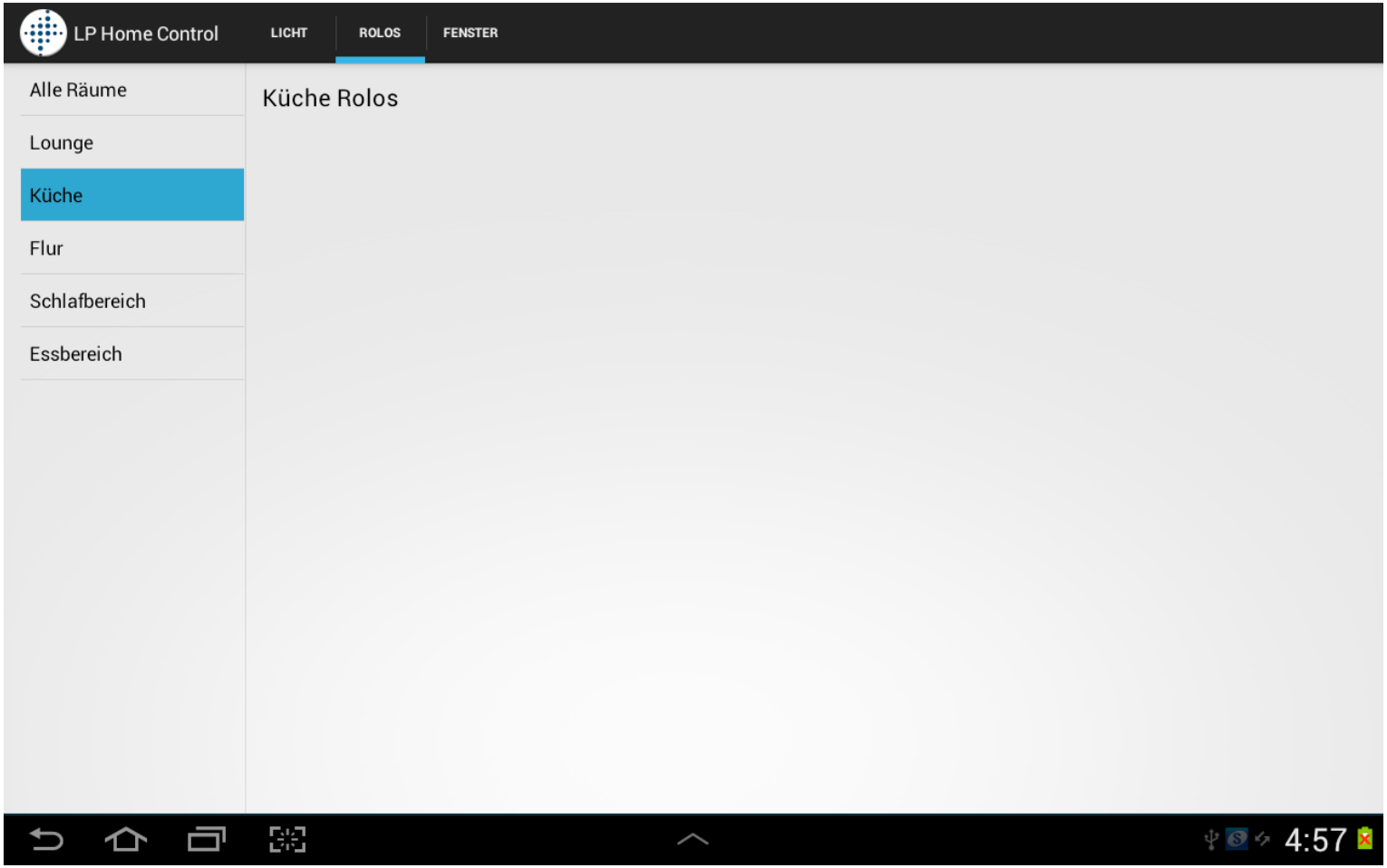
Aufbau der Android Projekte



Aufbau der Android Projekte



Laterale Navigation mit Tabulatoren



Laterale Navigation mit Tabulatoren

- ***ContextDelegate***:
  - konfiguriert die ActionBar für Tabulatoren
  - erzeugt einen ***TabListener*** für die einzelnen Tabs (Tabs entsprechen den kontextabhängigen Kontroll-Elementen)
  - stösst das Erzeugen der Tabulatoren in der ***ActionBar*** an

- ***ContextTabFactory***
  - erzeugt spezifisch für jeden räumlichen Kontext Tabulatoren in der ***ActionBar***
  - registriert für jeden Tab einen ***TabListener*** aus ***ContextDelegate***

- anonymer ***TabListener***:
  - erzeugt mit Hilfe der ***ControlFragmentFactory*** Fragments für spezifische Controls der einzelnen Kontexte
  - tauscht die Fragments im ***control_detail_container*** mittels einer Fragment-Transaktion aus

- Anzahl und Art der Tabulatoren wird generisch über die Enums ***Context*** und ***Control*** gesteuert

ContextDelegate für laterale Navigation mit Tabulatoren

- Konfiguration der ***ActionBar*** für Tabulatoren
- ***getSupportActionBar***() liefert eine ***SherlockActionBar***, die die Verwendung von Action Bars für Android < Version 4 unterstützt

```
final ActionBar actionBar = activity.getSupportActionBar();
// Specify that tabs should be displayed in the action bar.
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
```

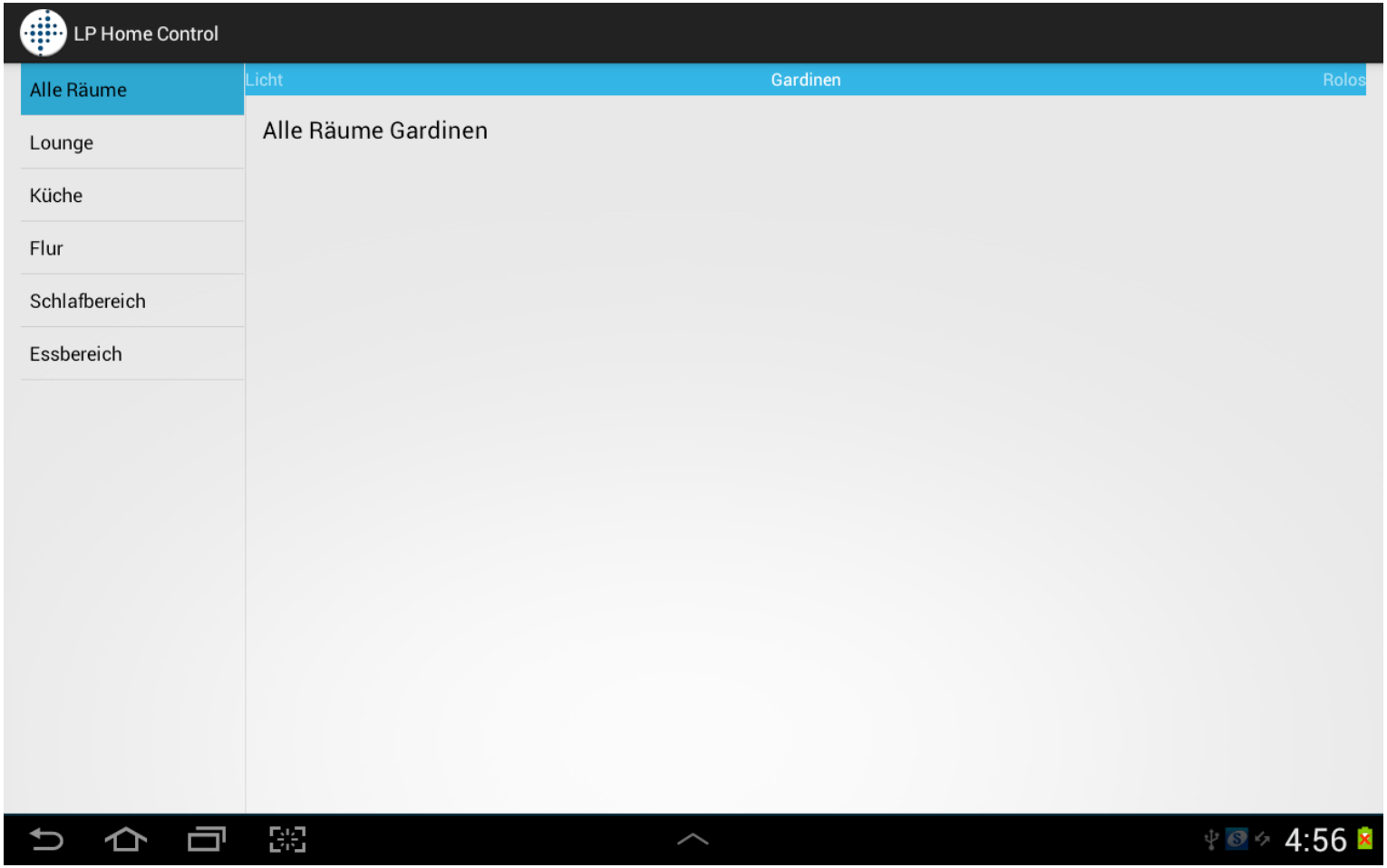
ContextDelegate TabListener

```
ActionBar.TabListener tabListener = new ActionBar.TabListener() {

@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {
   mContext = id;
   mControl = (Control) tab.getTag();

   activity.getSupportFragmentManager()
   .beginTransaction()
   .replace(
        R.id.control_container,
        ControlFragmentFactory.getInstance(id,
        (Control) tab.getTag())).commit();
}
...
};
```

ContextTabFactory

```
public class ContextTabFactory {

public static void createTabs(ActionBar actionBar, Context
   context, TabListener tabListener) {
   for (Control control : context.getControls()) {
        actionBar.addTab(actionBar.newTab()
               .setText(control.toString())
               .setTabListener(tabListener).setTag(control));
   }
}
}
```

Laterale Navigation mit ViewPagern



Laterale Navigation mit ViewPagern

- Registrieren eines ***ViewPager*** Layouts in den XML Dateien ***activity_context_two_pane*** und ***activity_control***

- ***ContextDelegate***:
  - registriert einen anonymen PagerAdapter (***FragmentStatePagerAdapter***) beim ViewPager Layout

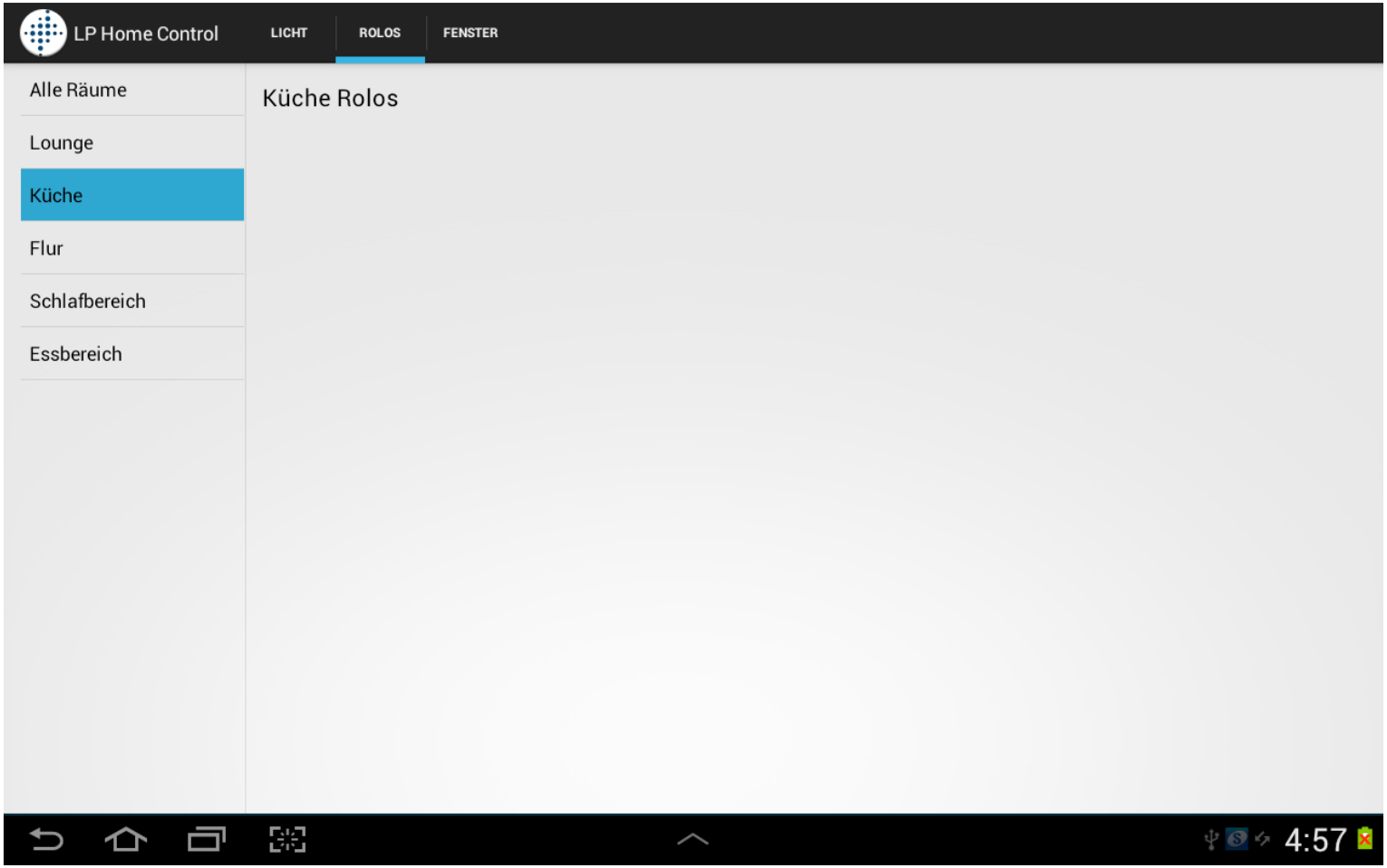
- anonymer ***FragmentStatePagerAdapter***:
  - kontext-abhängige Implementierung der Inhalte der einzelnen Seiten in der Methode ***getItem***
  - ***getItem*** liefert ein ***ControlFragment*** mit Hilfe der ***ControlFragmentFactory***

- Anzahl und Art der Pages wird generisch über die Enums ***Context*** und ***Control*** gesteuert

ContextDelegate plus *FragmentStatePagerAdapter*

```
ViewPager pager = (ViewPager) activity.findViewById(R.id.pager);

pager.setAdapter(new FragmentStatePagerAdapter(activity
        .getSupportFragmentManager()) {
    @Override
    public int getCount() {
       return context.getControls().length;
    }
    @Override
    public Fragment getItem(int arg0) {
       return ControlFragmentFactory.getInstance(context,
         context.getControls()[arg0]);
    }
    @Override
    public CharSequence getPageTitle(int position) {
       return context.getControls()[position].toString();
    }
});
```

Laterale Navigation mit ViewPager und Tabulatoren



Laterale Navigation mit ViewPager und Tabulatoren

- Layouts wie beim ViewPager Beispiel ohne die Titelzeile
- ***ContextDelegate*** und ***FragmentStatePagerAdapter:***
  - implementiert wie im ViewPager Beispiel

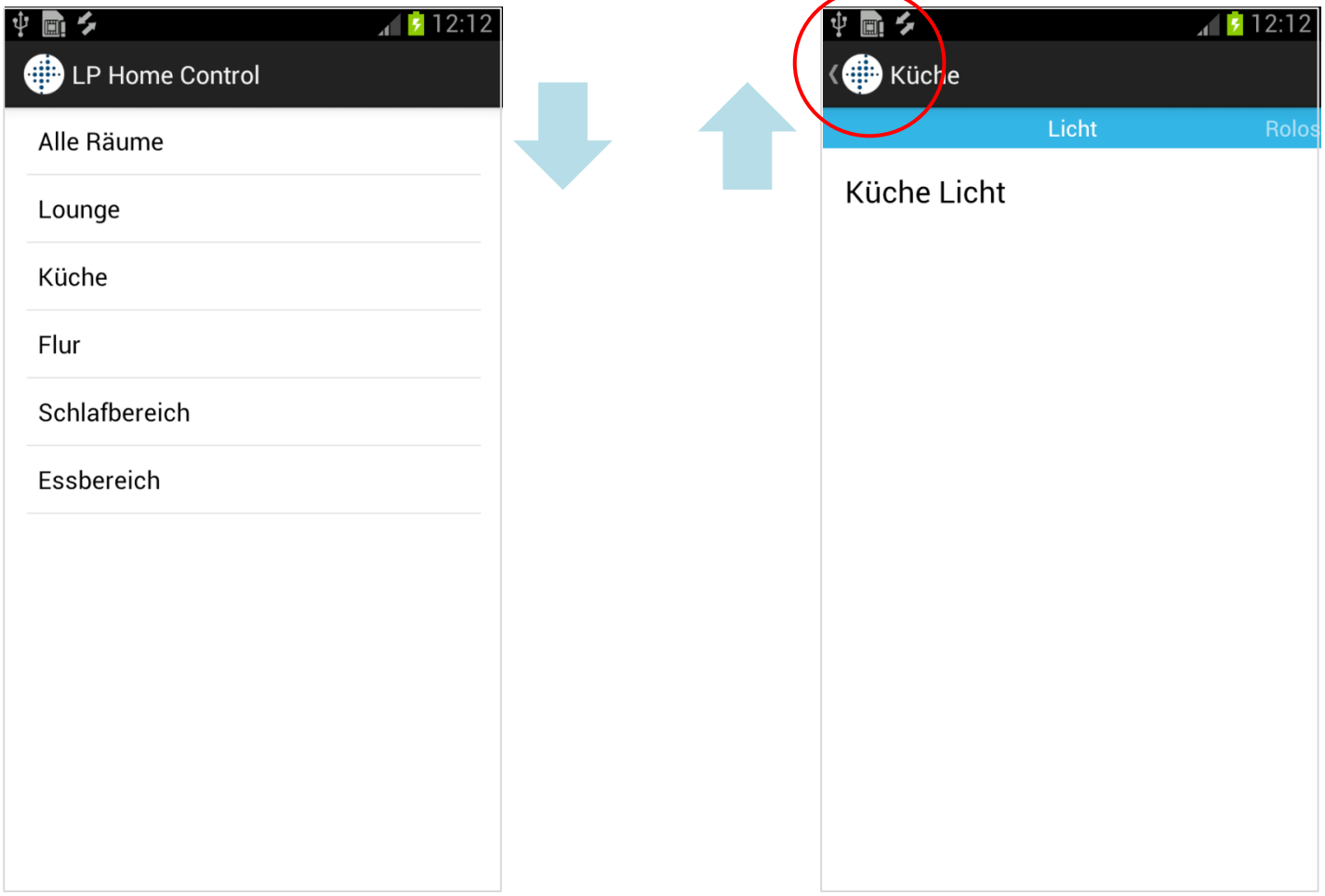
- ***ContextDelegate*** Erweiterungen:
  - Registrierung eines PageChangeListener beim View-Pager
  - Erzeugen von Tabs in der ActionBar
  - Implementierung des TabListener Interfaces

- ***SimpleOnPageChangeListener***:
  - setzt die Position des Navigationselementes in der ActionBar → Selektion des entsprechenden Tabs
- ***TabListener***:
  - setzt aktuelles Item des View Pagers

ContextDelegate Erweiterungen

```
mPager.setOnPageChangeListener(new ViewPager.SimpleOnPageChangeListener(){

      @Override
      public void onPageSelected(int position) {
          actionBar.setSelectedNavigationItem(position);
          super.onPageSelected(position);
      }

});


@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {
    mPager.setCurrentItem(tab.getPosition(), true);
}
@Override
public void onTabUnselected(Tab tab, FragmentTransaction ft) {
//Nothing to do
}
@Override
public void onTabReselected(Tab tab, FragmentTransaction ft) {
// nothing to do
}
```

Up-Navigation



Up-Navigation

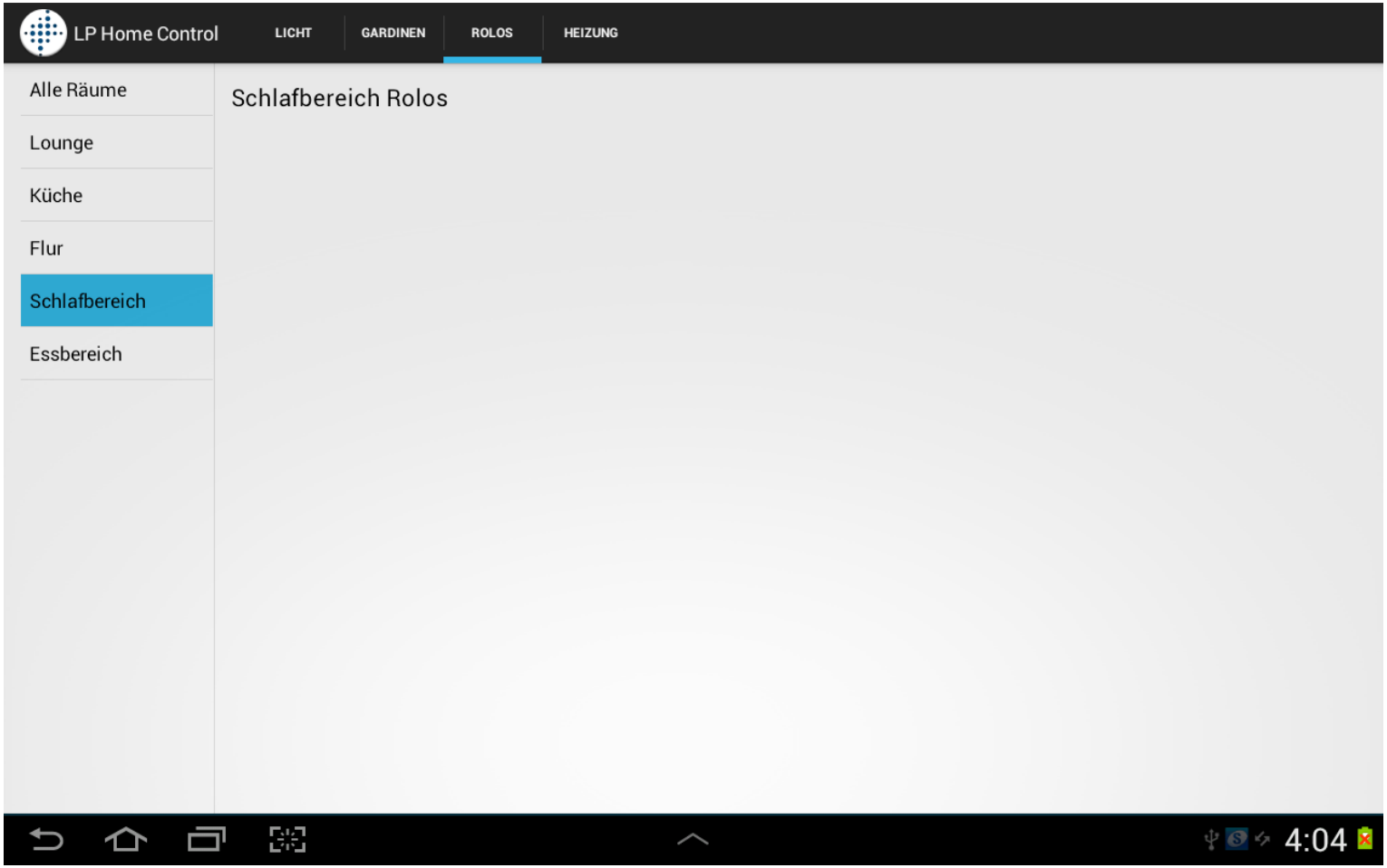
Konfiguration

```
<activity
  android:name=".ControlActivity"
  android:label="@string/title_control"
    >
  <meta-data
    android:name="android.support.PAREN
    T_ACTIVITY"
    android:value=".ContextListActivity
    " />
</activity>
```

Navigation zum Parent

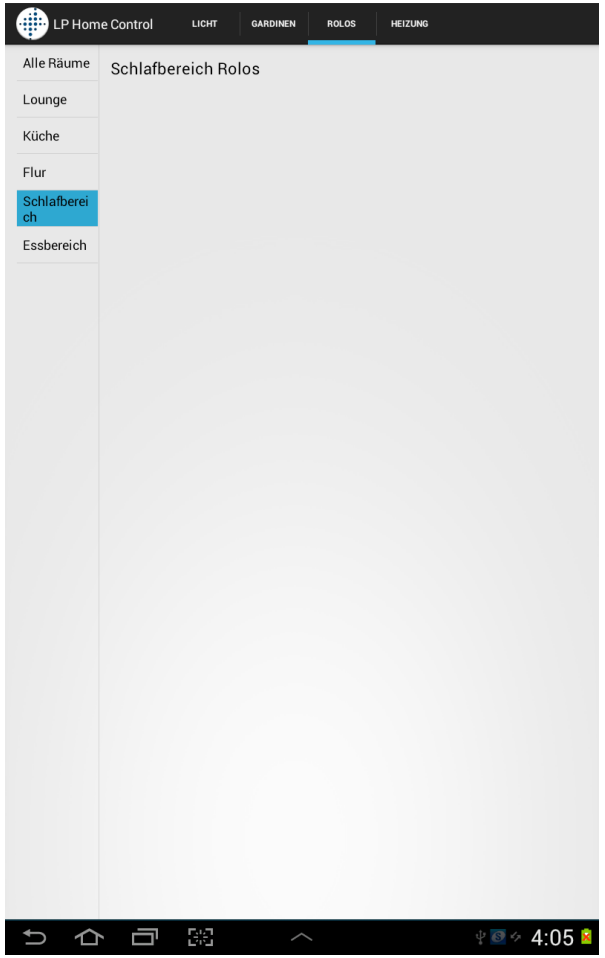
```
@Override
public boolean onOptionsItemSelected(
MenuItem item) {
 if (item.getItemId() ==
    android.R.id.home) {
 NavUtils.navigateUpTo(this, new
    Intent(this,
 ContextListActivity.class));
 return true;
}
return
    super.onOptionsItemSelected(item);
}
```

Zustandserhaltung bei Wechsel der Orientierung

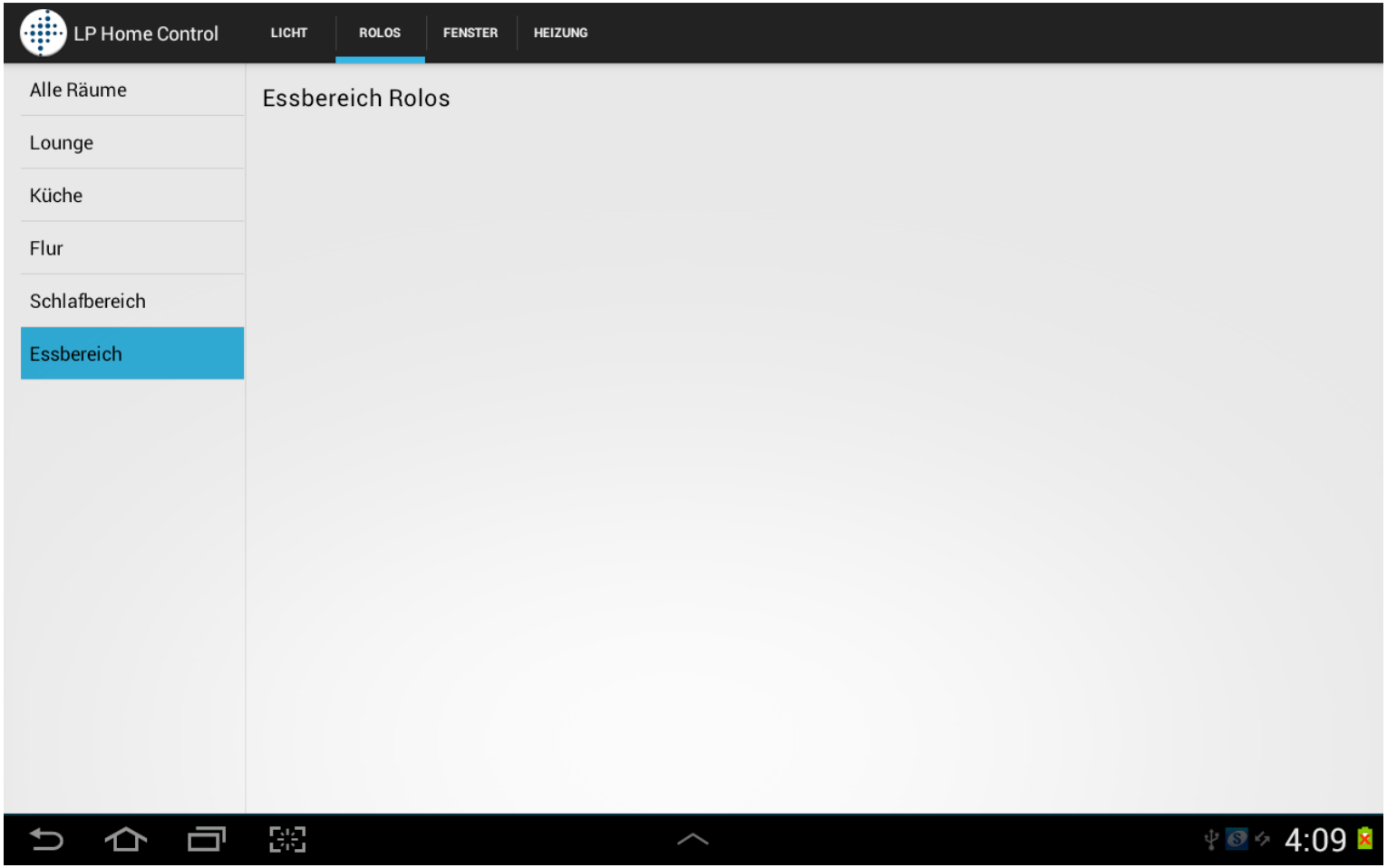


Zustandserhaltung bei Wechsel der Orientierung

- Projekt: ***LPControlTabSample***
- Drehen des Tabs / Handhelds → letzte TabSelektion bleibt erhalten



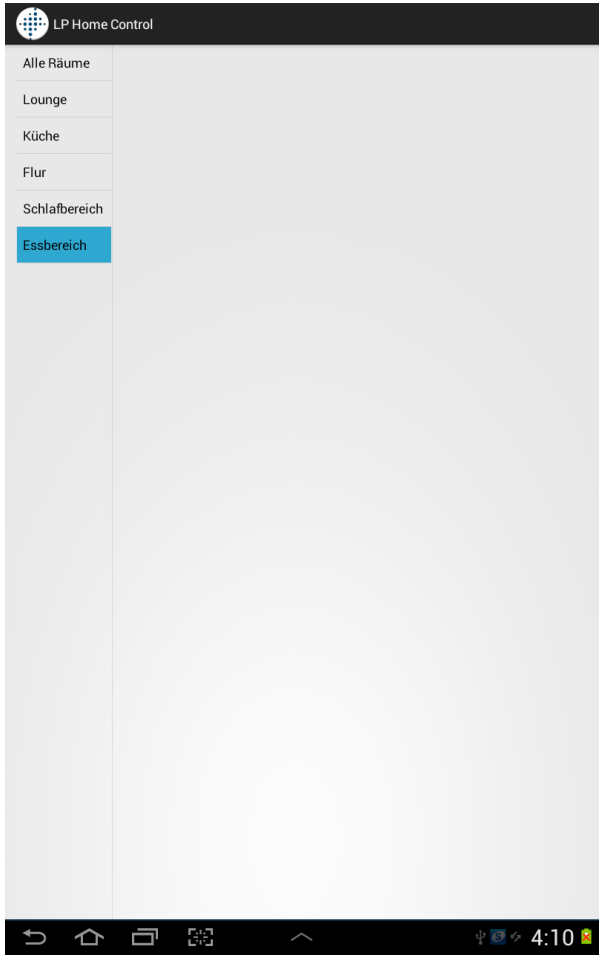
Zustandserhaltung bei Wechsel der Orientierung



Zustandserhaltung bei Wechsel der Orientierung

- Projekt:
  LPControlTabPagerSample & LPControlPagerSample

- Drehen des Tabs / Handhelds → letzte TabSelektion geht verloren



Zustandserhaltung bei Wechsel der Orientierung

- Wechsel der Orientierung
  - Erneutes „Erzeugen“ einer Aktivität und aller enthaltenen Fragmente
  - Aufruf von ***onCreate*** mit dem Bundle ***savedInstanceState*** != null

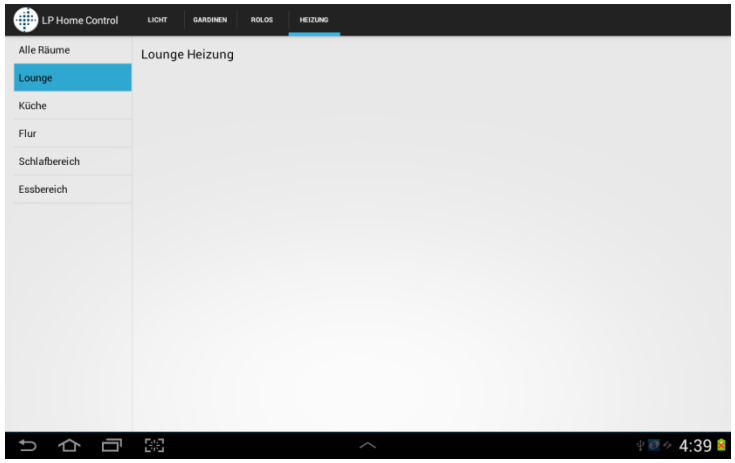
```
@Override
public void onCreate(Bundle
    savedInstanceState) {
 ...
if (savedInstanceState != null) {
Log.d(getClass().getSimpleName(),"onCreate
    mit savedInstanceState");
clfDelegate.restoreFrom(savedInstanceState,
    this);
} ...}
```

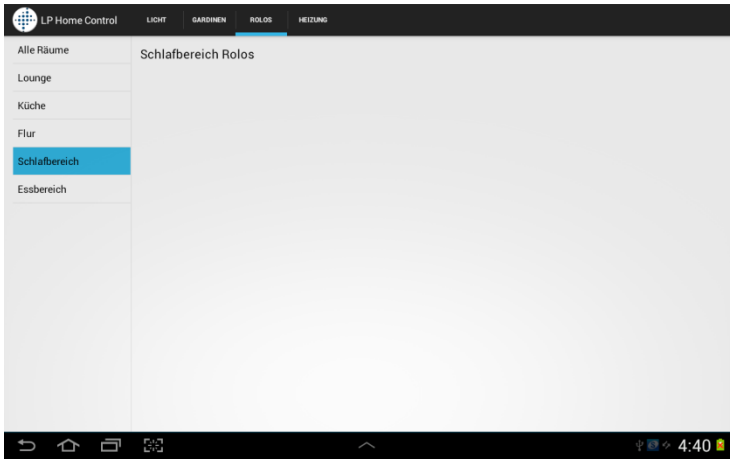
  - **Vorher**: Speichern des Zustands der Applikation in ***onSaveInstanceState***

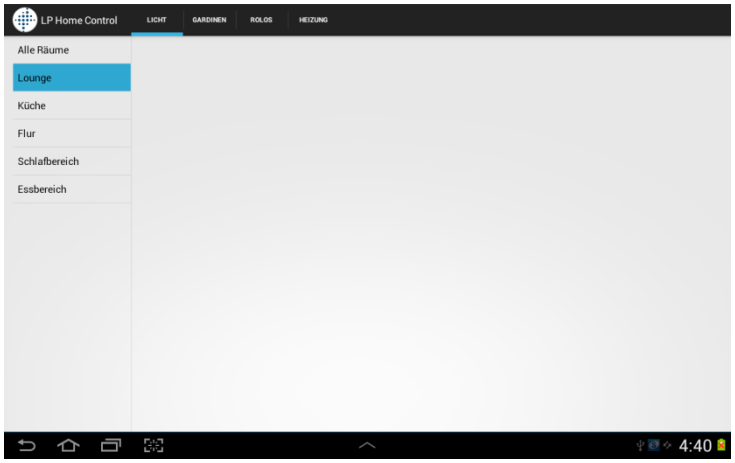
```
@Override
protected void onSaveInstanceState(Bundle
    outState) {
Log.d(getClass().getSimpleName(),"onSavedInst
    anceState");
super.onSaveInstanceState(outState);
clfDelegate.onSaveInstanceState(outState);
}
```

Zustandserhaltung beim Wechsel zwischen Detail Fragments

Keine Zustandserhaltung beim Wechsel zwischen Detail Fragments







Zustandserhaltung beim Wechsel zwischen Detail Fragments

- Klasse ***ContextDelegate***
  - kennt den ***Context*** und das letzte ***Control*** pro ***ControlFragment***
  - merkt sich die für jeden Context das letzte Control Einstellungen in einer Map (***ctxCtrlMap)***
  - liest bei der Initialisierung die Map aus und reselektiert den entsprechenden Tab
  - TODO: retten der Info im ***savedInstanceState***

```
if (ctxCtrlMap.containsKey(id)) {
 int pos =
    Arrays.asList(id.getControls()).indexOf(
    ctxCtrlMap.get(id));
    actionBar.setSelectedNavigationItem(pos);
} else {
    actionBar.setSelectedNavigationItem(0);
}
```

ActionBar Kompatibilität

- ActionBar erst seit Android 4
- Für Versionen < 4: Verwenden der Sherlock Kompatibilitäts Bibliothek
- Projekt ActionBar Sherlock als Library hinzufügen
- anstelle von ***FragmentActivity*** und ***Fragment*** ***SherlockFragmentActivity*** und ***SherlockFragment*** verwenden
- Referenz auf die ***ActionBar*** mit ***getSupportActionBar***()