Smart Home Control
Arbeiten mit Benutzerpräferenzen

Prof.Dr.Ing.Birgit Wendholt

Einleitung

- **Settings**: Einstellungen
  - des Benutzer über PreferenceActivities / Fragments
  - über Property Files
- **SharedPreferences:**
  - Präferenzen, die sich alle Komponenten einer Android Applikation teilen → konkurrierender Zugriff möglich
- Einstellungen des Benutzers → Eintrag in die SharedPreferences
- PreferenceActivity / Fragment
  - PreferenceFragment nicht Bestandteil der Kompatibilitätsbibliothek
  - **Empfehlung**: Verwenden von PreferenceActivitiy

Überblick (→ Projekt LPPreferences)

Erstellen von Präferenzdialogen

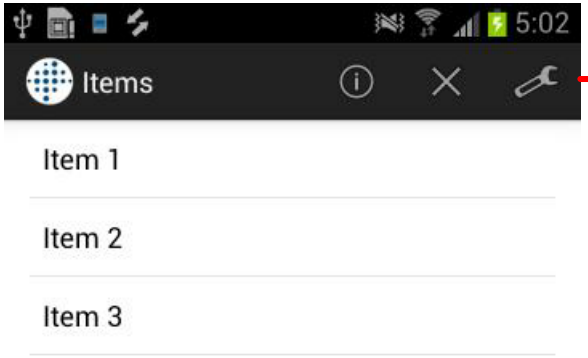
Verwenden von Präferenzen

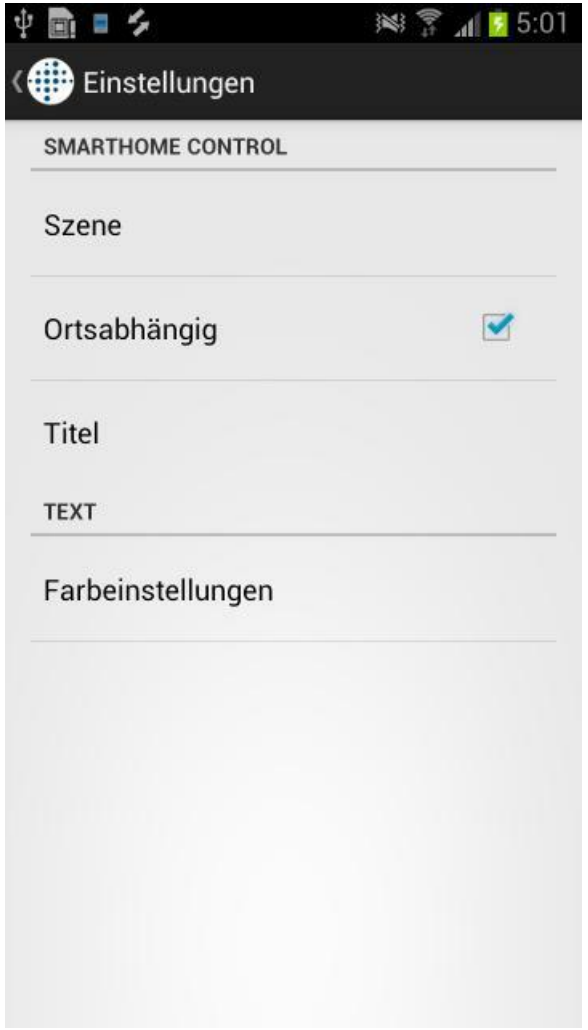
Eigene Preferences definieren

Listener für das Ändern von Präferenzen

Programmatisches Ändern von Präferenzen

Erstellen von Präferenzdialogen


5:02
Items
Item 1
Item 2
Item 3

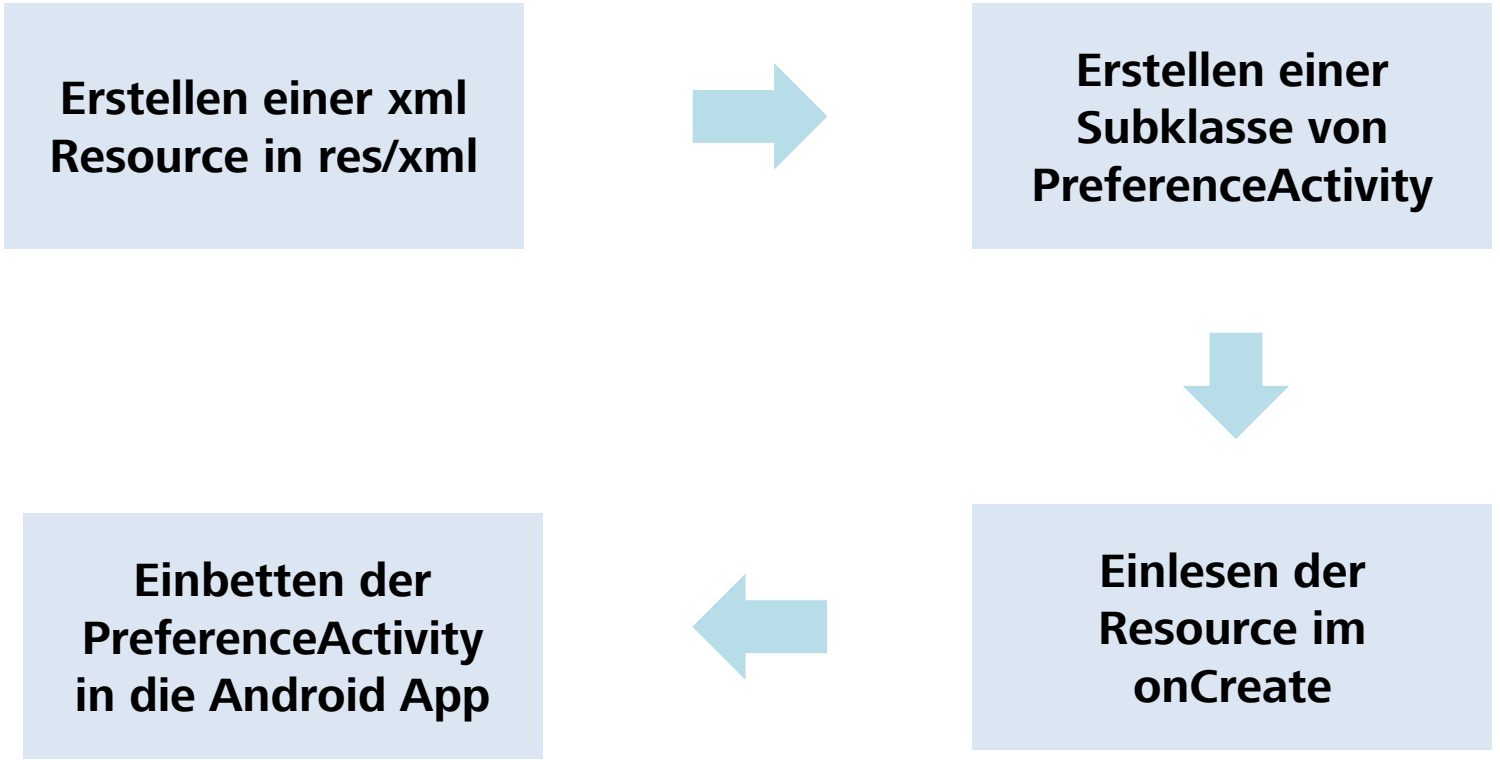

5:01
Einstellungen
SMARTHOME CONTROL
Szene
Ortsabhängig
Titel
TEXT
Farbeinstellungen

Erstellen von Präferenzdialogen

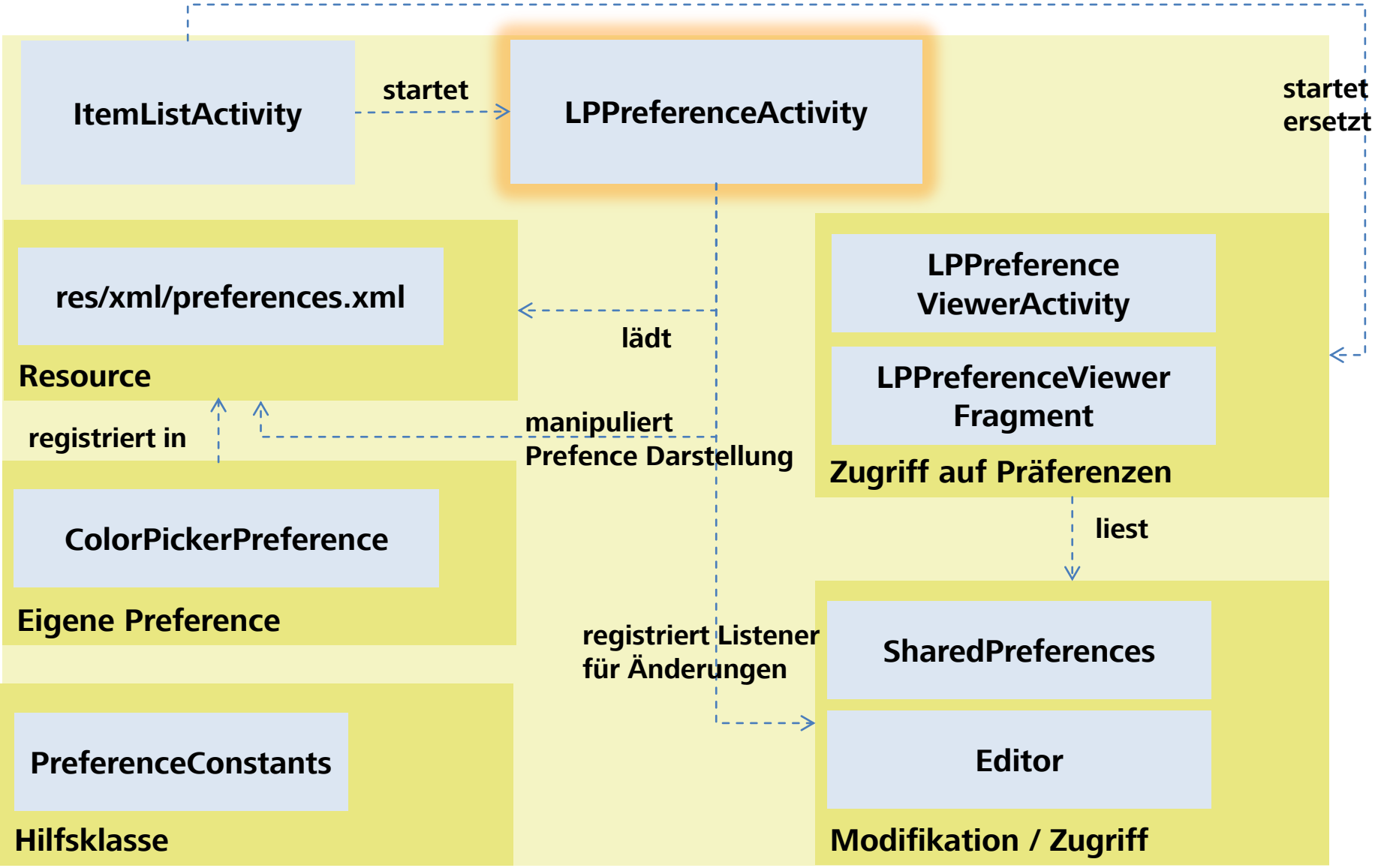
- **Settings**: Einstellungen
  - des Benutzer über PreferenceActivities / Fragments
  - über Property Files
- **SharedPreferences:**
  - Präferenzen, die sich alle Komponenten einer Android Applikation teilen → konkurrierender Zugriff möglich
- Einstellungen des Benutzers → Eintrag in die SharedPreferences

- PreferenceActivity / Fragment
  - PreferenceFragment nicht Bestandteil der Kompatibilitätsbibliothek
  - **Empfehlung**: Verwenden von PreferenceActivitiy
- PreferenceScreen
- Preference
  - ListPreference
  - CheckboxPreference
  - etc.
- Definition in xml Dateien
  - ListPreference
  - etc…
- Laden von Präferenzen
- Bezug zu den SharedPreferences

Erstellen von Präferenzdialogen



Überblick über die Lösung

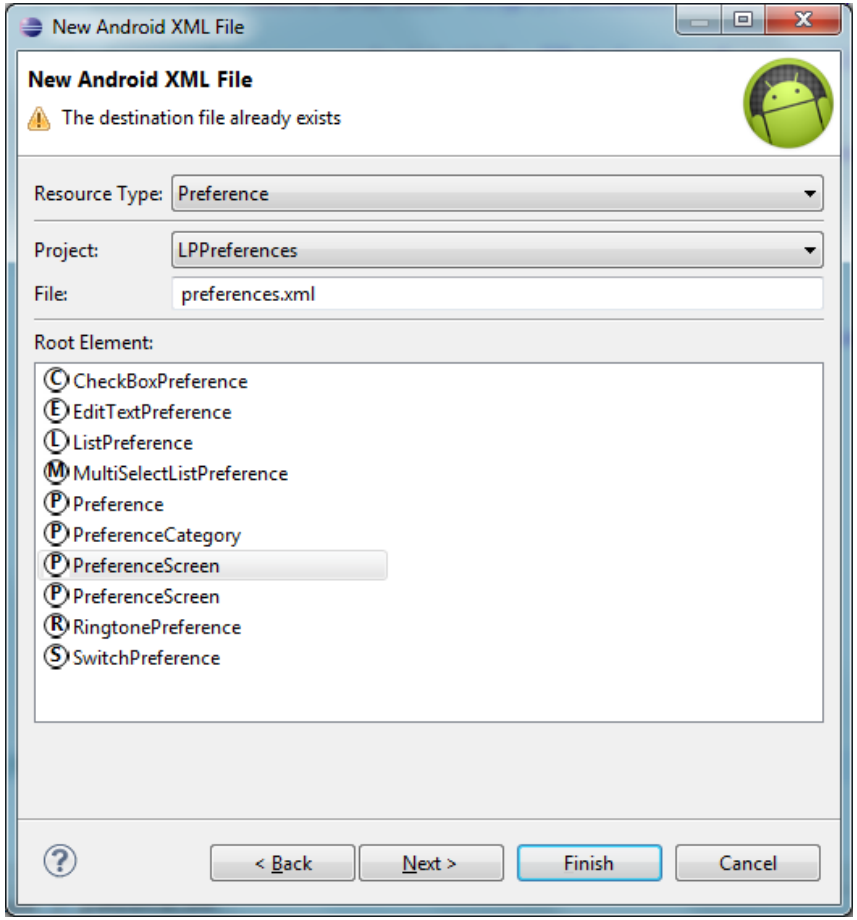


Konfiguration für Präferenzdialoge

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <PreferenceCategory android:title="@string/pref_controller_list" >
        <ListPreference
            android:entries="@array/pref_scenes_entries"
            android:entryValues="@array/pref_scenes_entry_values"
            android:key="pref.scenes.key"
            android:title="@string/pref_scenes_title" />

        <CheckBoxPreference
            android:key="pref.location.based.key"
            android:title="@string/pref_location_based_title" />

        <EditTextPreference
            android:key="pref.title.key"
            android:title="@string/pref_title" />
    </PreferenceCategory>

    <PreferenceCategory android:title="@string/pref_text">
    <de.wpsmarthome.ColorPickerPreference
            android:key="pref.color.1"
            android:showDefault="true"
            android:title="@string/pref_color" />
    </PreferenceCategory>

</PreferenceScreen>
```

Konfiguration für Präferenzdialoge (1)

1. Anlegen eines Verzeichnisses res/xml
2. Erzeugen einer neuen Präferenzdatei File→New→Other→Android→Android XML File
3. als Ressource Type **Preference** angeben

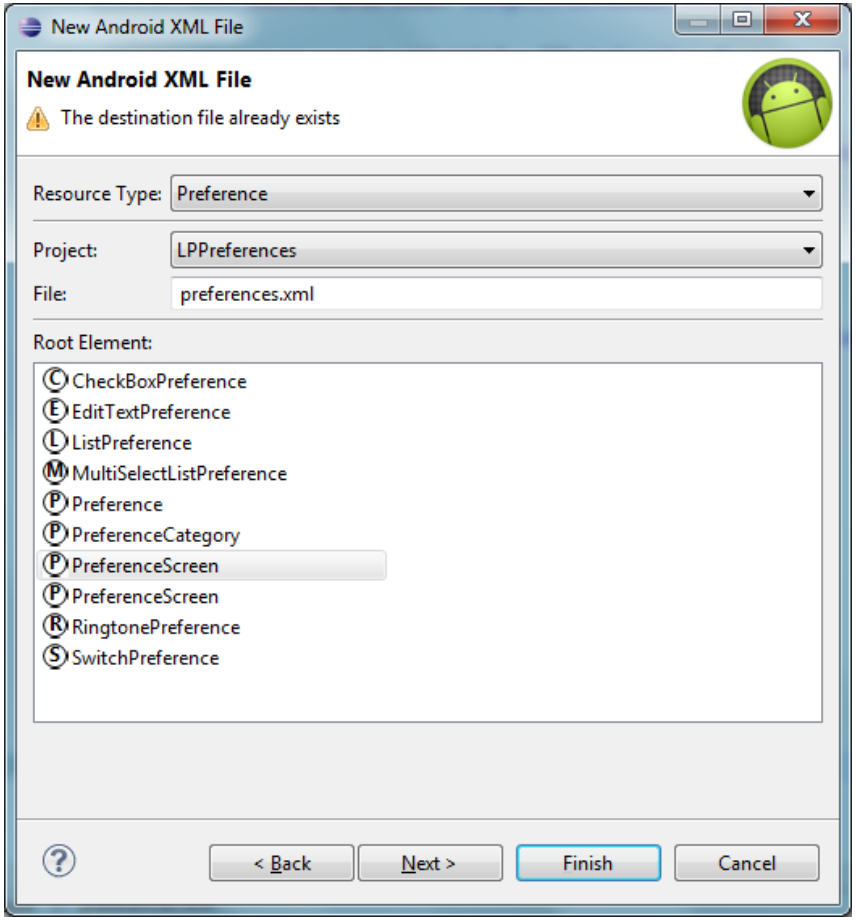


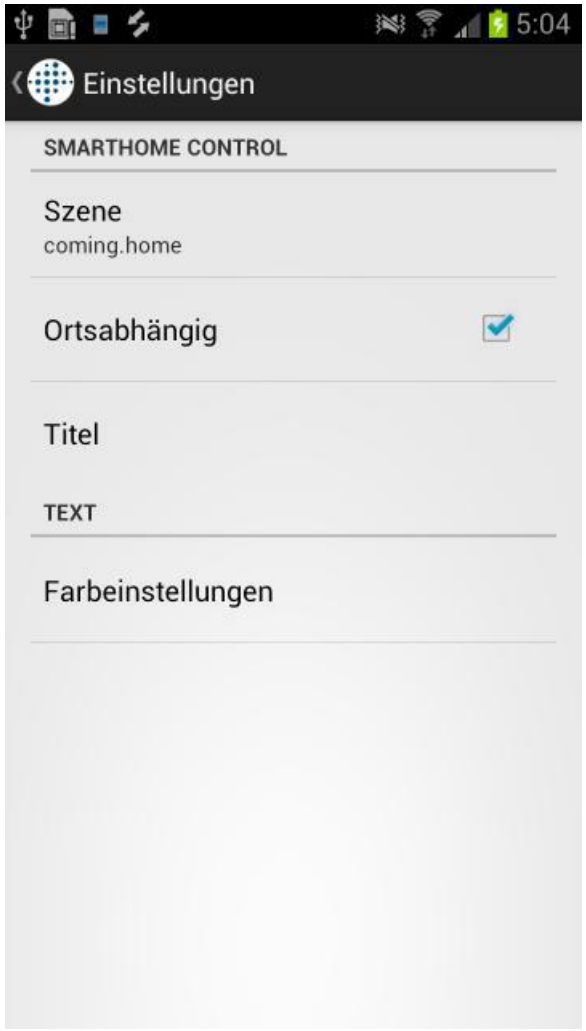
Konfiguration für Präferenzdialoge (2)

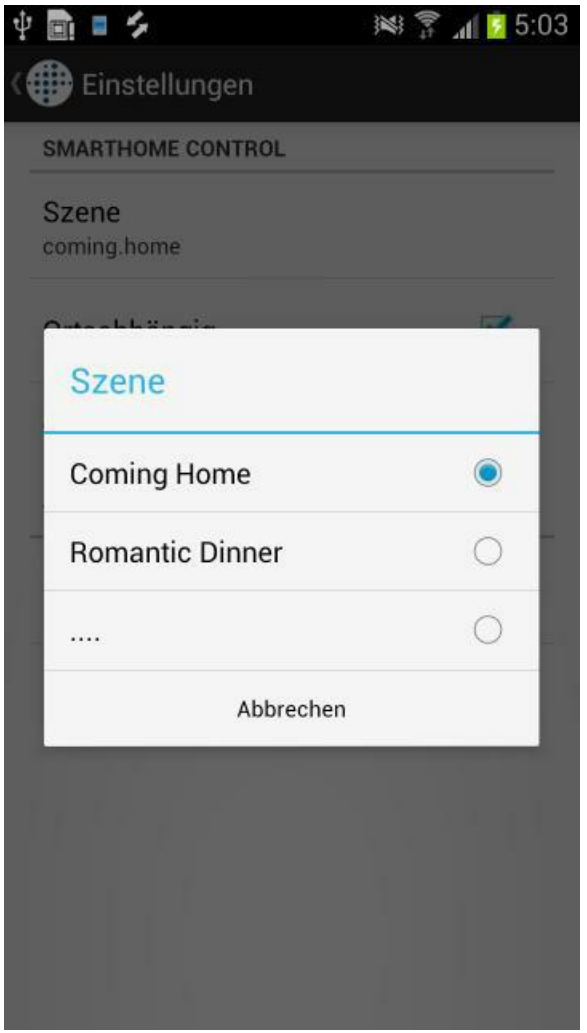
4. Gruppieren über Kategorien

5. Einfügen einzelner Preferences
   - List / EditText / SwitchPreference etc

6. im Beispiel verwendet
   - EditTextPreference
   - ListPreference
   - CheckboxPreference
   - selbst definierte Preference → ColorPickerPreference




5:04
Einstellungen
SMARTHOME CONTROL
Szene
coming.home
Ortsabhängig
Titel
TEXT
Farbeinstellungen


5:03
Einstellungen
SMARTHOME CONTROL
Szene
coming.home
Szene
Coming Home
Romantic Dinner
....
Abbrechen

Konfiguration für ListPreference

- entries und entryValues in res/values/arrays.xml definieren:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="pref_scenes_entries">
        <item>Coming Home</item>
        <item>Romantic Dinner</item>
        <item>....</item>
    </string-array>
    <string-array
    name="pref_scenes_entry_values">
        <item>coming.home</item>
        <item>romantic.dinner</item>
        <item>misc</item>
    </string-array>
</resources>
```

```
<ListPreference
    android:entries="@array/pref_scenes_entri
    es"

    android:entryValues="@array/pref_scenes_e
    ntry_values"
            android:key="pref.scenes.key"

    android:title="@string/pref_scenes_title"
    />
```

Erzeugen einer PreferenceActivity
Lesen der Resource

```
public class LPPreferenceActivity extends PreferenceActivity implements
 {

   public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         getActionBar().setDisplayHomeAsUpEnabled(true);
         addPreferencesFromResource(R.xml.preferences);
   }

}
```

Verwenden von Präferenzen Listener auf Änderungen Anzeige

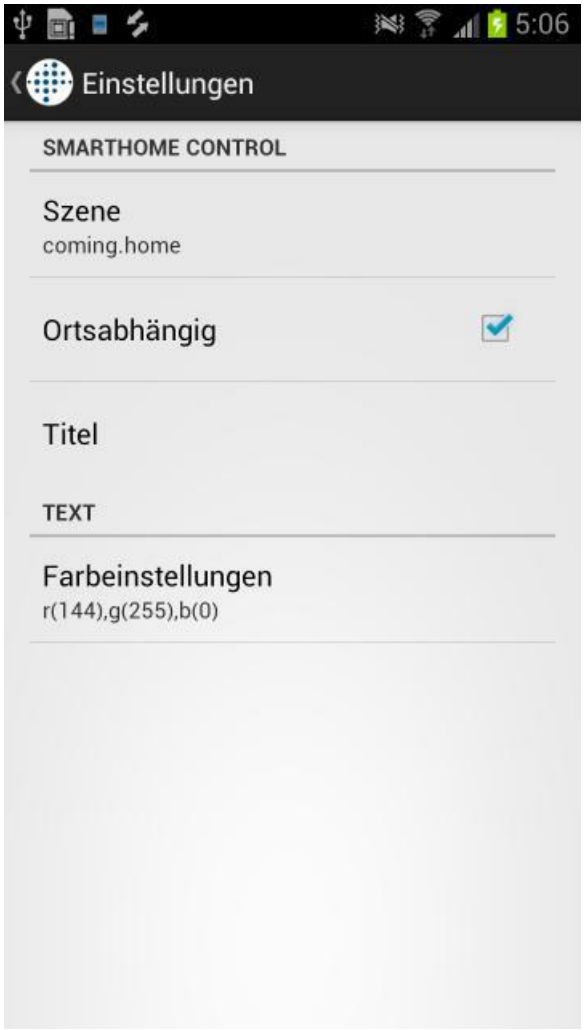
Erzeugen einer Preference Summary


```
public class LPPreferenceActivity extends PreferenceActivity implements
    OnSharedPreferenceChangeListener {

protected void onResume() {
 PreferenceManager.getDefaultSharedPreferences(this)
        .registerOnSharedPreferenceChangeListener(this);
 super.onResume();
}

protected void onPause() {
 PreferenceManager.getDefaultSharedPreferences(this)
         .unregisterOnSharedPreferenceChangeListener(this);
 super.onPause();
}

public void onSharedPreferenceChanged(
         SharedPreferences sharedPreferences, String key) {
  if (key.equals(PreferenceConstants.SCENES_KEY)
     || key.equals(PreferenceConstants.TITLE_KEY)) {
         Preference pref = findPreference(key);
         pref.setSummary(sharedPreferences.getString(key, ""));
   }
...
}
```

Auslesen von Präferenzen

```
public class LPPreferenceViewerFragment extends Fragment {

 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
      Bundle savedInstanceState) {
   View rootView = inflater.inflate(R.layout.fragment_preference_viewer, null);
    return rootView;
}

 @Override
 public void onViewCreated(View view, Bundle savedInstanceState) {
   EditText text = (EditText)view.findViewById(R.id.pref_viewer);
   Log.d(getClass().getSimpleName(),"in onCreateView");
   SharedPreferences sharedPrefs =
      PreferenceManager.getDefaultSharedPreferences(getActivity());
   for(Map.Entry<String, ?> singlePref :sharedPrefs.getAll().entrySet()) {
      text.append(singlePref.getKey() + ":" + singlePref.getValue() );
      text.append("\n");
 }
 super.onViewCreated(view, savedInstanceState);
}

}
```

Eigene Präferences definieren


5:06
Einstellungen
SMARTHOME CONTROL
Szene
coming.home
Ortsabhängig
Titel
TEXT
Farbeinstellungen
r(144),g(255),b(0)


5:06
Farbeinstellungen
Abbrechen
OK

Eigene Preferences (1)

1. Layout für den Präferenzdialog definieren
2. Subklasse von DialogPreference erstellen

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.
    com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

  <com.oliver.ColourWheel

    android:id="@+id/color_wheel1"

    android:layout_width="match_parent"

    android:layout_height="wrap_content" >

  </com.oliver.ColourWheel>

</LinearLayout>
```

Eigene Preferences (2)

1. Layout für den Präferenzdialog definieren

2. Subklasse von DialogPreference erstellen
   - im Konstruktior das Layout laden
   - Zugriff auf die View-Komponenten im ***onBindDialogView***

```
public class ColorPickerPreference extends
    DialogPreference implements
   OnColourWheelChangeListener {

public ColorPickerPreference(Context
    context, AttributeSet attrs) {
   super(context, attrs);
  setDialogLayoutResource(
    R.layout.color_picker_dialog);
   ...
}
protected void onBindDialogView(View view)
    {
    ColourWheel colorWheel = (ColourWheel)
     view.findViewById(R.id.color_wheel1);
   colorWheel.
     setOnColourWheelChangeListener(this);
   super.onBindDialogView(view);
}
```

Programmatisches Ändern von Präferenzen

Programmatisches Ändern von Präferenzen

- Laden von Preference Ressourcen → Eintrag in die SharedPreferences
- Konkurrierender Zugriff auf SharedPreferences durch Komponenten einer App möglich
- Editor → „transaktionales“ Ändern der SharedPreference
- Ohne Verwendung des Editors → Änderungen nicht dauerhaft

```
public class ColorPickerPreference extends
    DialogPreference implements
   OnColourWheelChangeListener {
...
@Override
protected void onDialogClosed(boolean
    positiveResult) {

    if (!positiveResult)
          return;

    SharedPreferences.Editor editor =
          getEditor();
    int colint = Color.rgb(red, green,
    blue);
    editor.putInt(PreferenceConstants.
                    COLOR_1, colint);
    editor.commit();
    super.onDialogClosed(positiveResult);
}
}
```

Zusammenfassung

Erstellen von Präferenzdialogen

Verwenden von Präferenzen

Eigene Preferences definieren

Listener für das Ändern von Präferenzen

Programmatisches Ändern von Präferenzen