Лекция 3

Мобильная разработка

Пирская Любовь Владимировна,
к.т.н., доцент кафедры МОП ЭВМ
lpirskaya@sfedu.ru

Ссылки на сайты дисциплины

- Разработка приложений для мобильных устройств (бакалавриат 09.03.04)

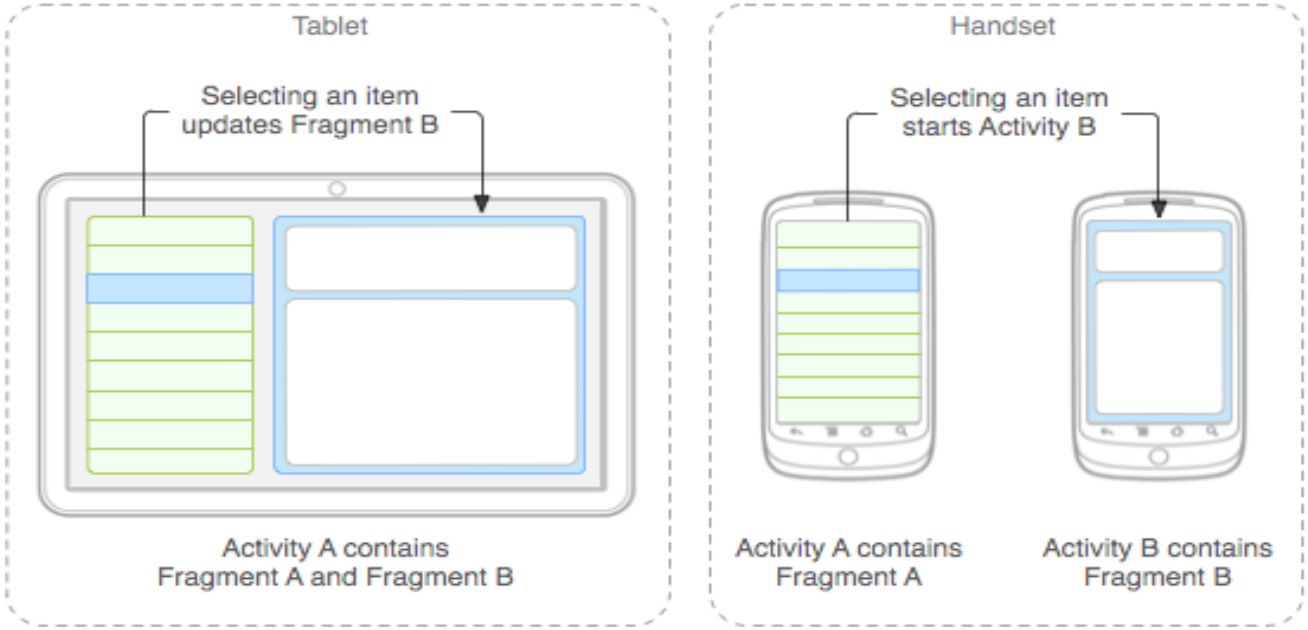
https://sites.google.com/site/progformob

- Программное обеспечение для мобильных платформ и системы цифровой обработки сигналов (магистратура 09.04.04)

https://sites.google.com/view/prog-for-mob-mag

Fragments

Fragments



- Представлены в API 11 (Android 3.0) – планшеты – большой экран – разумное распределение места
- Живет в activity
- Представляет собой «кусок» интерфейса
- Переиспользуемые
- Свой Life Cycle

Создание Fragment

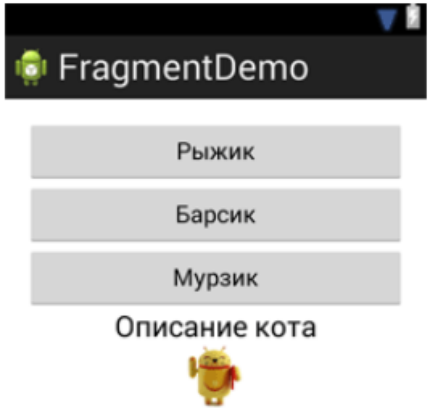
- Подкласс **Fragment**
- Пустой конструктор
- Такой же жизненный цикл как у activity
- onCreateView()
- Собственный layout
- Могут быть объявлены в разметке

Управление Fragment

- FragmentManager
- Из activity всегда можно получить fragment(id или tag)
- Во фрагменте всегда есть ссылка на activity(getActivity)
- Фрагменты можно добавлять, удалять в реальном времени
- Транзакции сохраняются в backStack

Пример. Стандартная разметка

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Рыжик" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Барсик" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Мурзик" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Описание кота"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="fitCenter"
        android:src="@drawable/cat_yellow" />

</LinearLayout>
```



Пример. Делим на Fragment

res/layout/fragment1.xml

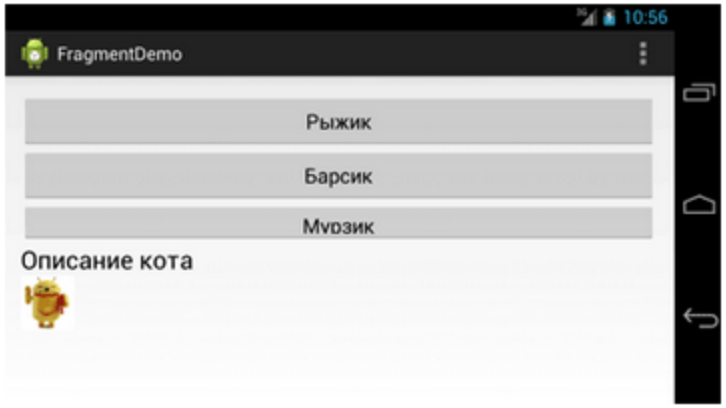
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Рыжик" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Барсик" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Мурзик" />

</LinearLayout>
```

res/layout/fragment2.xml

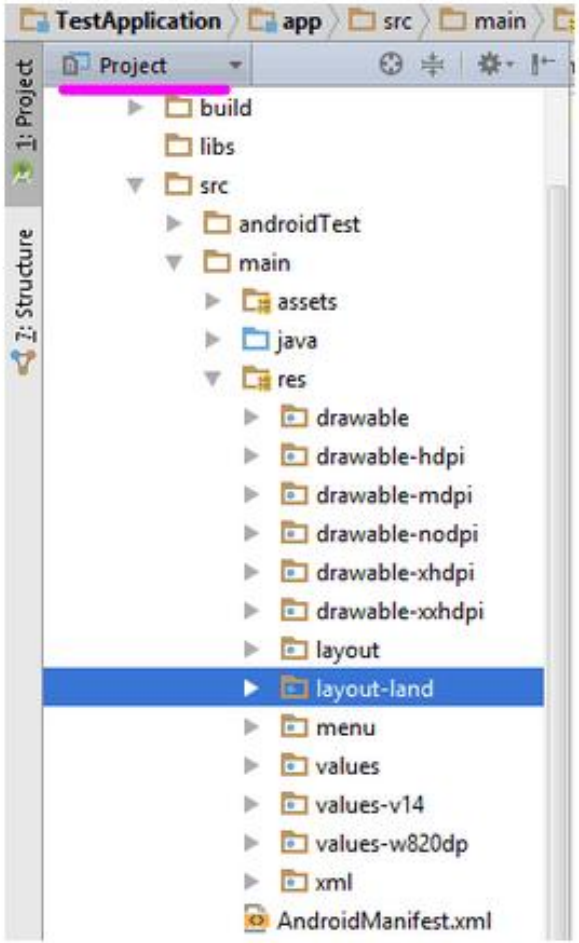
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Описание кота"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="fitCenter"
        android:src="@drawable/cat_yellow" />

</LinearLayout>
```

Пример. Добавление Fragment в layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <fragment
        android:id="@+id/fragment1"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

    <fragment
        android:id="@+id/fragment2"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment2"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

</LinearLayout>
```

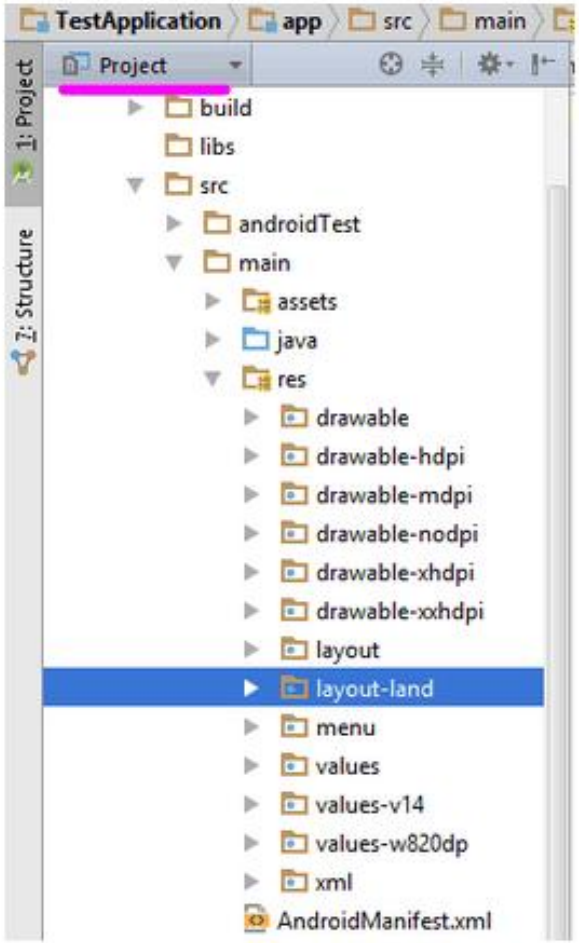


Пример. Добавление Fragment в layout




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:baselineAligned="false"
    tools:context=".MainActivity" >

    <fragment
        android:id="@+id/fragment1"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment1"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        tools:layout="@layout/fragment1" />

    <fragment
        android:id="@+id/fragment2"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment2"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        tools:layout="@layout/fragment2" />

</LinearLayout>
```

Пример. Добавление Fragment в layout



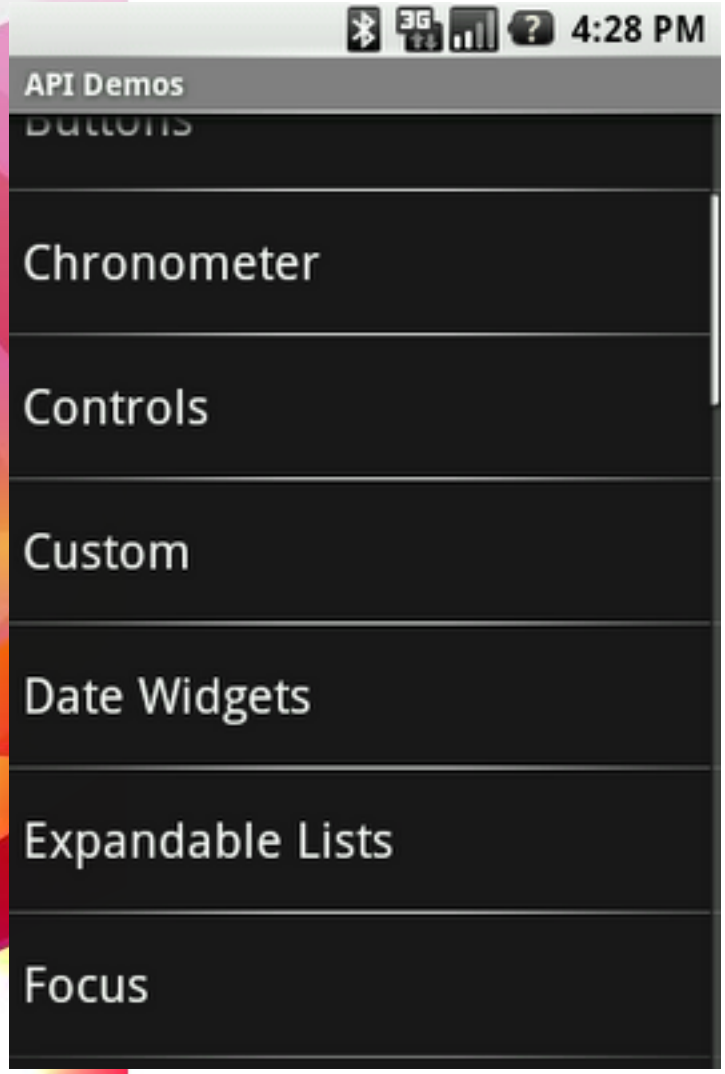

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:baselineAligned="false"
    tools:context=".MainActivity" >

    <fragment
        android:id="@+id/fragment1"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment1"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        tools:layout="@layout/fragment1" />

    <fragment
        android:id="@+id/fragment2"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment2"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        tools:layout="@layout/fragment2" />

</LinearLayout>
```

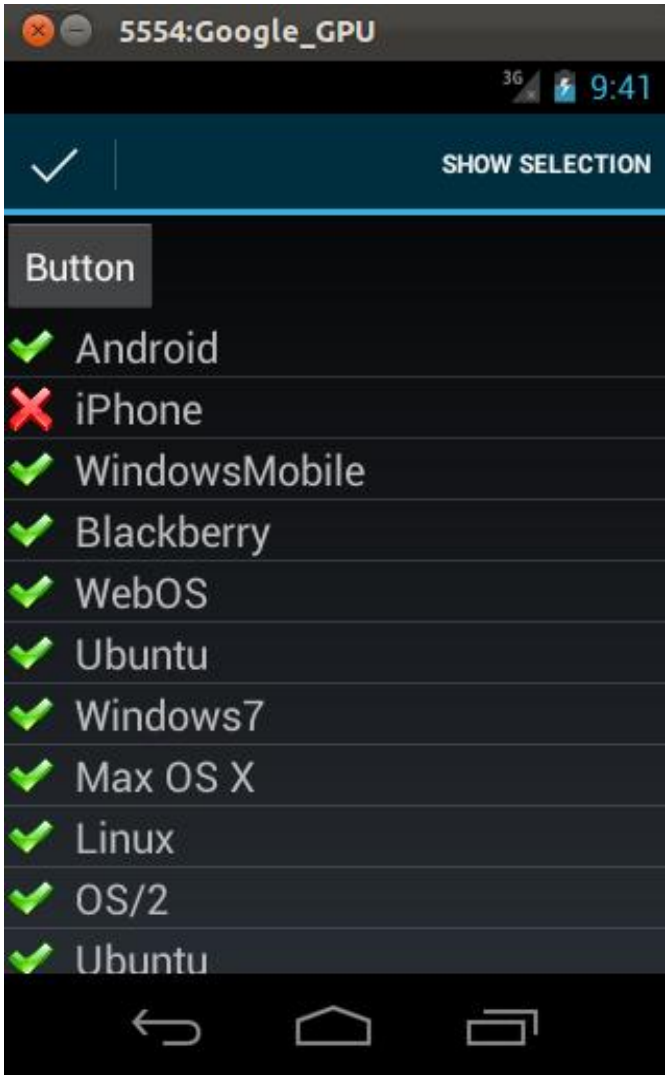
AdapterViews

AdapterViews

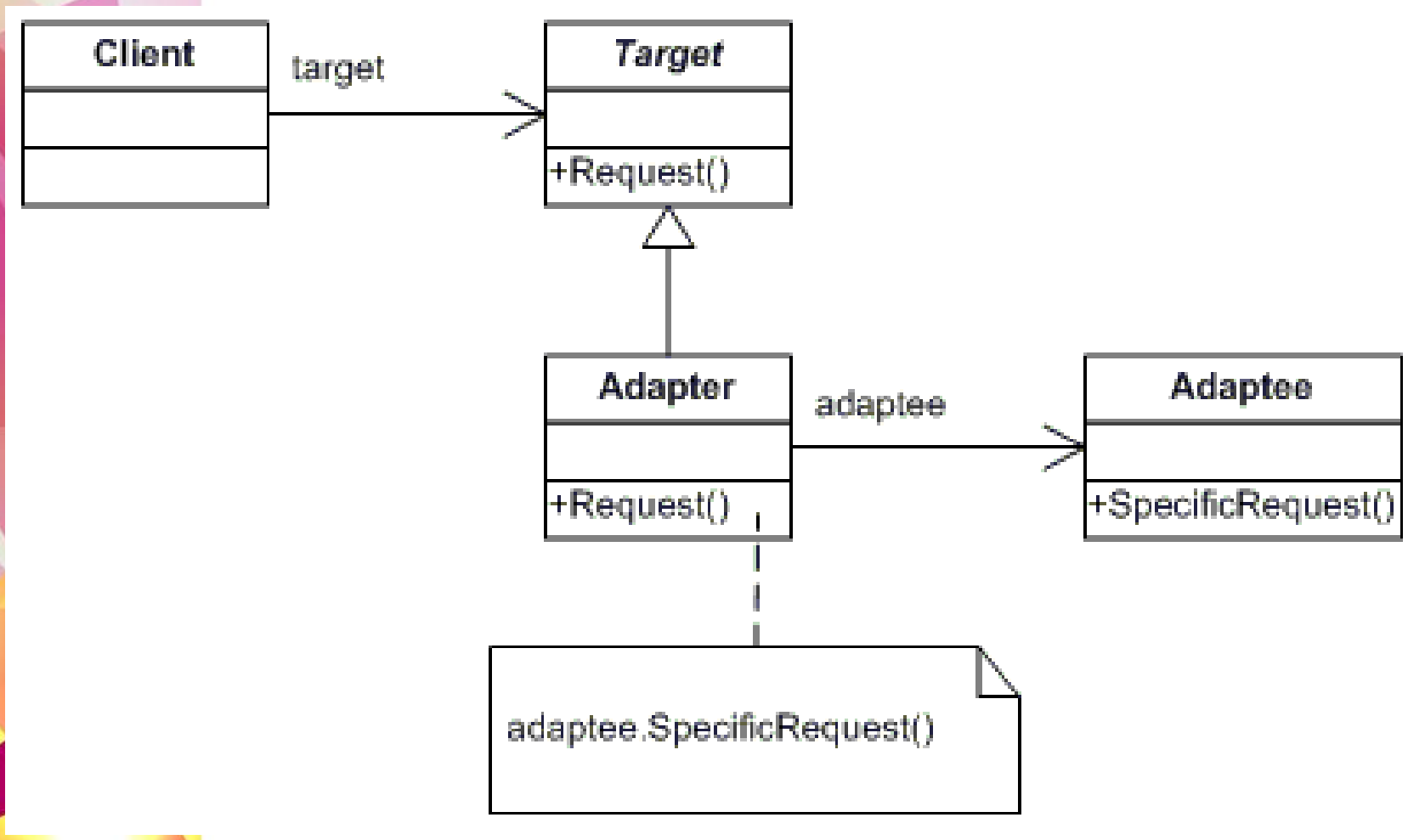
- Для однородных динамических данных, например списки
- ListView, GridView, Spinner, etc
- Для связи данные-отображение используется паттерн adapter
- RecyclerView (с 5.0)
- Много реализаций адаптеров

ListView


4:28 PM
API Demos
Chronometer
Controls
Custom
Date Widgets
Expandable Lists
Focus


5554:Google_GPU
9:41
SHOW SELECTION
Button
Android
iPhone
WindowsMobile
Blackberry
WebOS
Ubuntu
Windows7
Max OS X
Linux
OS/2
Ubuntu

Adapter pattern



Терминология

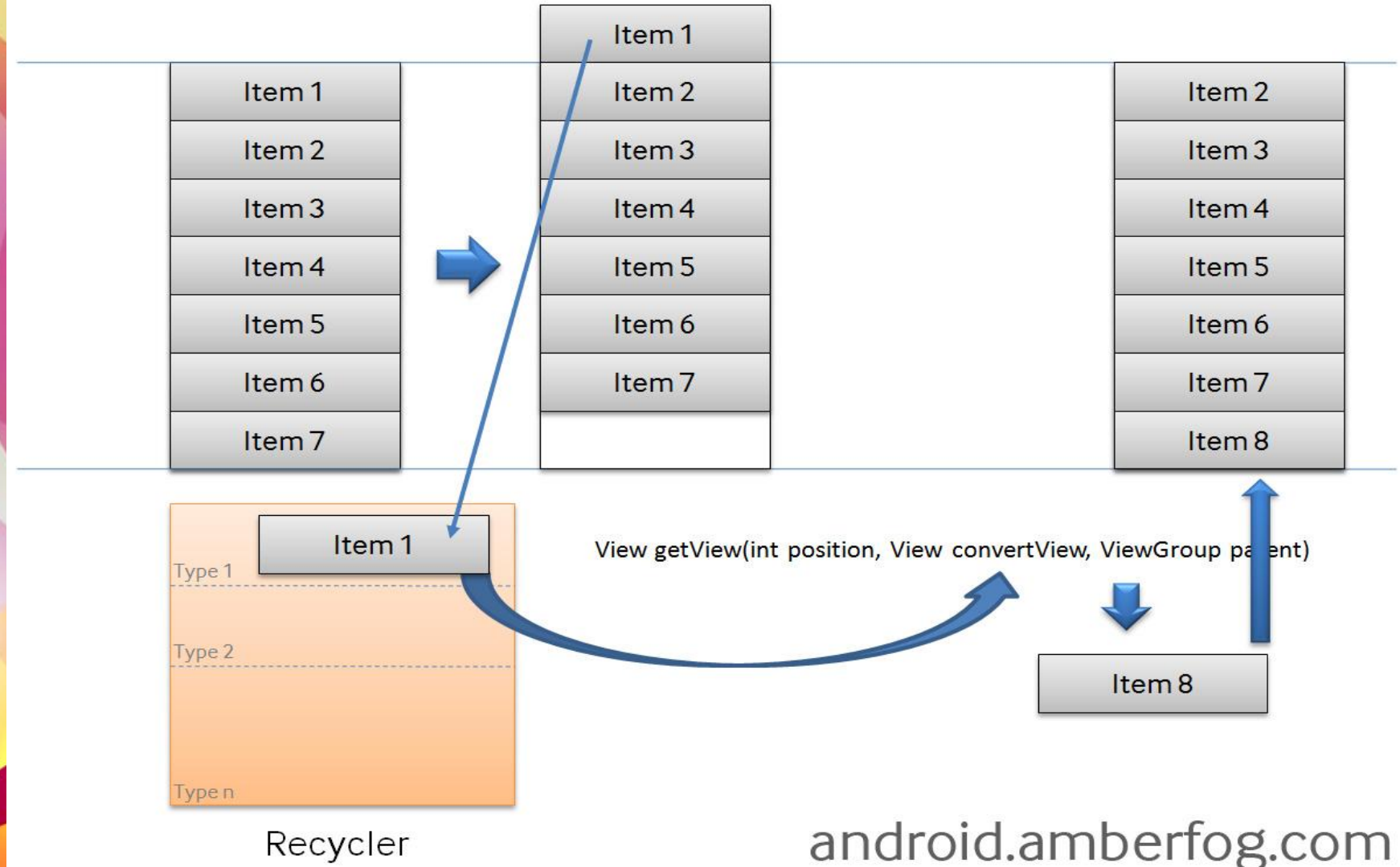
- Childs
- Position
- Id

- getView() – получение данных

Adapter interface

- Класс от BaseAdapter
- getCount()
- getItem(int position)
- getItemId(int position)
- getView (int position, View convertView, ViewGroup parent)

Recycle Bin



android.amberfog.com

Как использовать?

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.item, parent, false);
    }

    ((TextView) convertView.findViewById(R.id.text)).setText(DATA[position]);
    ((ImageView) convertView.findViewById(R.id.icon)).setImageBitmap(
            (position & 1) == 1 ? mIcon1 : mIcon2);

    return convertView;
}
```

LayoutInflater. Использование

Создание view фрагментов

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fmt_task_list, null);
}
```

Avoid passing null as the view root (needed to resolve layout parameters on the inflated layout's root element) more... (⌘F1)

Создание элементов списков в адаптере

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = LayoutInflater.from(mContext).inflate(R.layout.li_task, null);
    }
    ((TextView) convertView).setText(getItem(position));
    return convertView;
}
```

Варианты метода *inflate*

- inflate(int resource, ViewGroup root)
- inflate(int resource, ViewGroup root, boolean attachToRoot)
- inflate(XmlPullParser parser, ViewGroup root)
- inflate(XmlPullParser parser, ViewGroup root, boolean attachToRoot)

root - корневой элемент, к которому будут присоединены “надутые” элементы.

attachToRoot - флаг, отвечающий за непосредственное присоединение элементов к корневому элементу. По умолчанию равен (**root != null**).

root=null для ячеек списка

```xml
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="?attr/listPreferredItemHeightSmall"
    android:gravity="center_vertical"
    android:textColor="?android:attr/textColorPrimary" />
```

```java
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = LayoutInflater.from(mContext).inflate(R.layout.li_task, null);
    }
    ((TextView) convertView).setText(getItem(position));
    return convertView;
}
```

Попытка “надуть” элемент списка с фиксированной высотой (**android:layout_height="?attr/listPreferredItemHeightSmall"**) без указания родительского элемента приводит к неожиданному результату

root=parent для ячеек списка


```java
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = LayoutInflater.from(mContext).inflate(R.layout.li_task, parent, false);
    }
    ((TextView) convertView).setText(getItem(position));
    return convertView;
}
```

Ячейки списка пришли в норму.

Стоит обратить внимание на то что **attachToRoot = false**. **ListView** (строго говоря, это параметр **parent**) сам компонует ячейки, получая их из ***Adapter**. В противном случае, приложение упадет с фатальной ошибкой.

Когда root=null оправдан

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    return new AlertDialog.Builder(getActivity())
            .setView(LayoutInflater.from(getActivity()).inflate(R.layout.fmt_input_alert, null))
            .setPositiveButton(android.R.string.ok, null)
            .setNegativeButton(android.R.string.cancel, null)
            .create();
}
```

Самый распространенный случай - в диалогах. Диалог не нуждается в родительском элементе для построения своего окна.

Когда root=null оправдан

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    return new AlertDialog.Builder(getActivity())
            .setView(LayoutInflater.from(getActivity()).inflate(R.layout.fmt_input_alert, null))
            .setPositiveButton(android.R.string.ok, null)
            .setNegativeButton(android.R.string.cancel, null)
            .create();
}
```

Самый распространенный случай - в диалогах. Диалог не нуждается в родительском элементе для построения своего окна.

Палитра UI элементов