

Android Resources

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

Resources

- Things that are embedded (bundled) into the app
- Resources in res/ directory
- Several resource categories
- Accessible through code: `R.<category>.<resourceName>`
- Do NOT hard-code values inside codes

Contents

- Values
- Layouts
- Drawables
- Raw
- Styles, Design Guidelines

Layouts

- **Remind**
- **Definition**
- **Layout in XML**
- Popular Layout classes

Remind

- Remind: HTML
- Old-school: `<div>` and `<table>`
- HTML5: `<header>`, `<nav>`, `<section>`, `<aside>`, `<footer>`
- CSS float and padding

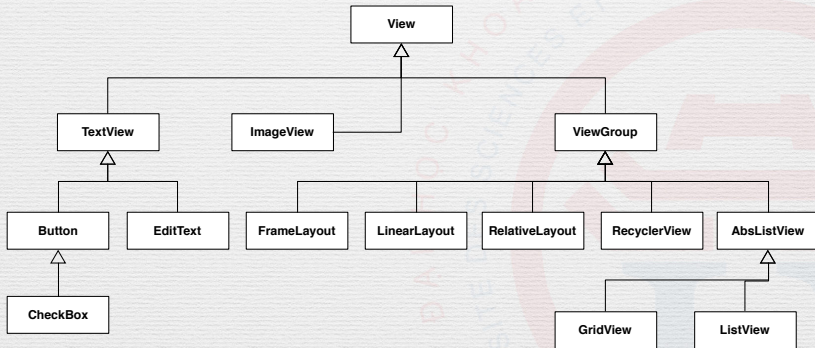
Remind

- Remind: what's the base View class for layouting in android?

View class for layouting in
t, RelativeLayout

- Remind: what's the base View class for layouting in android?
- ViewGroup
- FrameLayout, LinearLayout, RelativeLayout

Remind



Layout

- A way to organize Views inside an UI
- Can be created by code (see Practical Work #4)
- XML files in **res/layout**
- Hierarchical “structure” of one UI
- Can be nested
- WYSIWYG or manual editor

Layout XML

- Containers (ViewGroups) contain Views (TextView, ImageView, EditText, Button, ImageButton...)
- Required: `layout_width`, `layout_height`
- Optional: `id` (for later `findViewById()`)

Layout 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"
    android:padding="16dp">

    <TextView
        android:id="@+id/question"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Is Android a Linux Distribution?"
        android:textAppearance=
            "?android:attr/textAppearanceMedium" />

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="True"
            android:id="@+id/btnTrue" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="False"
            android:id="@+id/btnFalse" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Next"
            android:id="@+id/btnNext" />

    </LinearLayout>
</LinearLayout>
```

«Adaptive» Layout on Android

- Use different layout XMLs in different **directories**
- Tablet: layout-large, layout-xlarge
- Phone: layout-normal
- Small: layout-small
- Orientation: -land, -port
- Examples

Normal

Layout XML

- How to load XML layout?
 - Activity: in `onCreate()`, with `setContentView()`
 - Fragment: in `onCreateView()`

// Activity

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_layout);  
}
```

// Fragment

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,...) {  
    // Inflate the layout for this fragment  
    return inflater.inflate(R.layout.fragment_layout, container, false);  
}
```


Layouts

- Remind
- Definition
- Layout in XML
- **Popular Layout classes**

Popular Layout Classes

- `FrameLayout`
- `LinearLayout`
- `RelativeLayout`
- `ViewPager`

FrameLayout

- Can contain multiple children (Views)
- Multiple layers, Z-based order: like a Photoshop design
- First child will be at the bottom
- Support child margins
- Support gravity

FrameLayout

```

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:background="#20FF0000"
        android:src="@drawable/usth" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#FF00FF00"
        android:text="Is USTH awesome?" />
</FrameLayout>

```



LinearLayout

- One direction
- Horizontal or Vertical

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Logout" />

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

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

</LinearLayout>

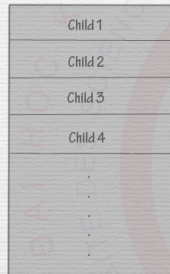
```

LinearLayout

```

<LinearLayout
    android:orientation="vertical">

```



```

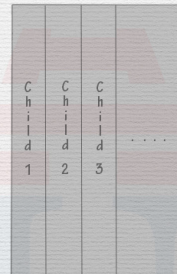
</LinearLayout>

```

```

<LinearLayout
    android:orientation="horizontal">

```



```

</LinearLayout>

```

horizontal orientation

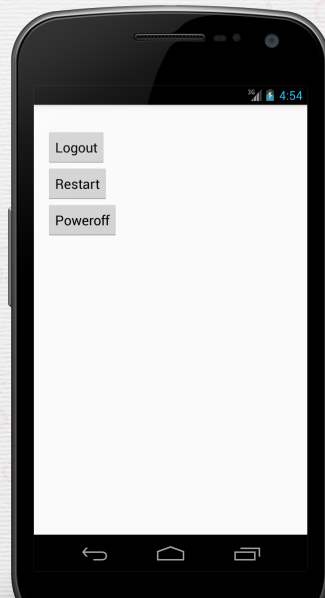
LinearLayout

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

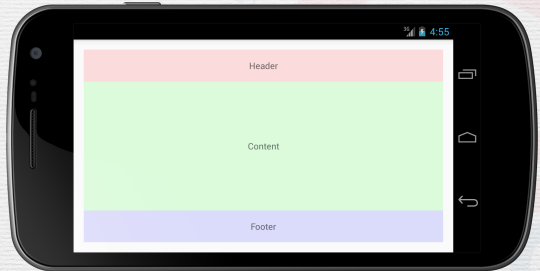
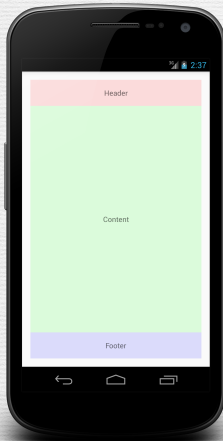
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Logout" />

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

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Poweroff" />
</LinearLayout>
```



LinearLayout



LinearLayout Stretching

- Use layout_weight
- Based on orientation
 - horizontal: stretch width
 - vertical: stretch height
- no layout_weight: no stretch
- width/height ω_i is calculated based on weight γ_i of child i as

$$\omega_i = \frac{\gamma_i}{\sum_{j=0}^{n-1} \gamma_j} \times (\omega_{parent} - \sum_{k=0}^{n-1} \omega_k | \gamma_k = 0)$$

LinearLayout: Exercise

```

<LinearLayout
    android:layout_width="720px"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/button"
        android:layout_width="0px"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Logout" />

    <Button
        android:id="@+id/button2"
        android:layout_width="100px"
        android:layout_height="wrap_content"
        android:text="Restart" />

    <Button
        android:id="@+id/button3"
        android:layout_width="0px"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Poweroff" />
</LinearLayout>

```

What's the width value of each child in this layout?

$$\omega_i = \frac{\gamma_i}{\sum_{j=0}^{n-1} \gamma_j} \times (\omega_{parent} - \sum_{k=0}^{n-1} \omega_k | \gamma_k = 0)$$

LinearLayout: Exercise

```
<LinearLayout
```

```
    android:id="@+id/container"
    android:layout_width="720px"
    android:layout_height="48px"
    android:orientation="horizontal"
    android:padding="4px">
```

```
<View
```

```
    android:layout_width="0px"
    android:layout_height="1px"
    android:layout_weight="1" />
```

```
<TextView
```

```
    android:id="@+id/item1"
    android:layout_width="100px"
    android:layout_height="match_parent"
    android:paddingLeft="8px"
    android:paddingRight="8px" />
```

```
<View
```

```
    android:layout_width="1px"
    android:layout_height="match_parent"
    android:layout_marginLeft="8px"
    android:layout_marginRight="8px"
    android:background="@drawable/divider" />
```

```
<TextView
```

```
    android:id="@+id/item2"
    android:layout_width="0px"
    android:layout_height="match_parent"
    android:layout_weight="2"
    android:paddingLeft="16px"
    android:paddingRight="16px" />
```

```
<View
```

```
    android:layout_width="1px"
    android:layout_height="match_parent"
    android:layout_marginLeft="8px"
    android:layout_marginRight="8px"
    android:background="@drawable/divider" />
```

```
<TextView
```

```
    android:id="@+id/item3"
    android:layout_width="120px"
    android:layout_height="match_parent"
    android:paddingLeft="8px"
    android:paddingRight="8px" />
```

```
<View
```

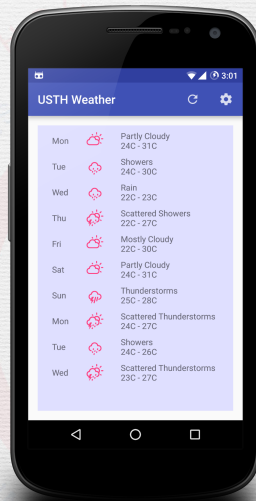
```
    android:layout_width="0px"
    android:layout_height="1px"
    android:layout_weight="1" />
```

```
</LinearLayout>
```

$$\omega_i = \frac{\gamma_i}{\sum_{j=0}^{n-1} \gamma_j} \times (\omega_{parent} - \sum_{k=0}^{n-1} \omega_k | \gamma_k = 0)$$

Practical Work 5

- Modify your ForecastFragment layout
- Use LinearLayout to have the blue forecast area



RelativeLayout

- Multiple layers, Z-order based: similar to Photoshop layers
- Relativity of children's position and size
 - to parent
 - to each other

RelativeLayout

- Children are «relative» to parent



`android:layout_alignParentLeft`



`android:layout_alignParentTop`



`android:layout_alignParentRight`



`android:layout_alignParentBottom`



`android:layout_centerHorizontal`



`android:layout_centerVertical`



`android:layout_centerInParent`

RelativeLayout

- Children are «relative» to each other



android:layout_toLeftOf



android:layout_above



android:layout_toRightOf



android:layout_below



android:layout_alignTop



android:layout_alignBottom



android:layout_alignLeft



android:layout_alignRight

RelativeLayout



android:layout_alignParentLeft



android:layout_alignParentTop



android:layout_alignParentRight



android:layout_alignParentBottom



android:layout_centerHorizontal



android:layout_centerVertical



android:layout_centerInParent



android:layout_toLeftOf



android:layout_above



android:layout_toRightOf



android:layout_below



android:layout_alignTop



android:layout_alignBottom



android:layout_alignLeft



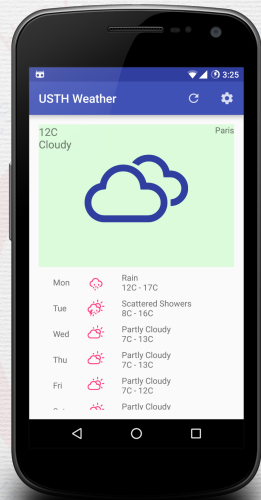
android:layout_alignRight

RelativeLayout

- Questions:
 - How to simulate vertical LinearLayout using RelativeLayout?
 - Is it possible to simulate LinearLayout's stretching feature (layout_weight) using RelativeLayout?

Practical Work 6

- Create a new WeatherFragment
- Modify your WeatherActivity layout
 - Add the WeatherFragment above ForecastFragment
 - Use LinearLayout
 - Use `<fragment>` tag
- Produce the green area (WeatherFragment) with RelativeLayout



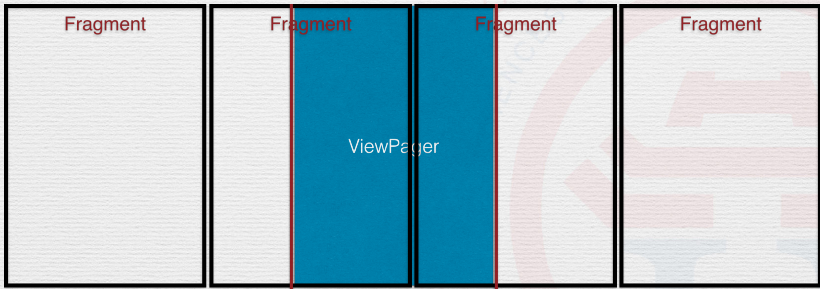
Layout

- Many other ViewGroups
- ScrollView
- GridView
- CardView
- ListView
- **ViewPager**
- DrawerLayout
- CoordinatorLayout
- RecyclerView

ViewPager



ViewPager

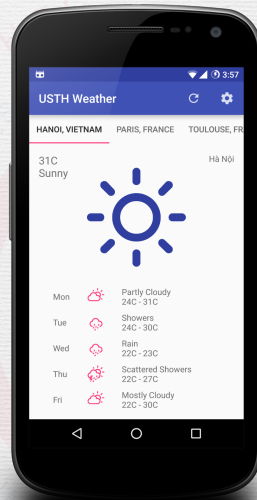


ViewPager

- Tab-like container
 - Widely used, user-friendly
 - Horizontal swipe gesture
 - «Page-by-page» scrolling

ViewPager

- No header. Use a separate View for that
 - Previously, use PagerSlidingTabStrip on GitHub
 - Now: Android design library's TabLayout



ViewPager

- Each tab content is a fragment
 - UI inflated / created by fragment
 - Controlled by fragment
 - Can be nested
 - «Off-screen» limit: fragments outside this limit are destroyed and recreated when necessary.
 - Why?

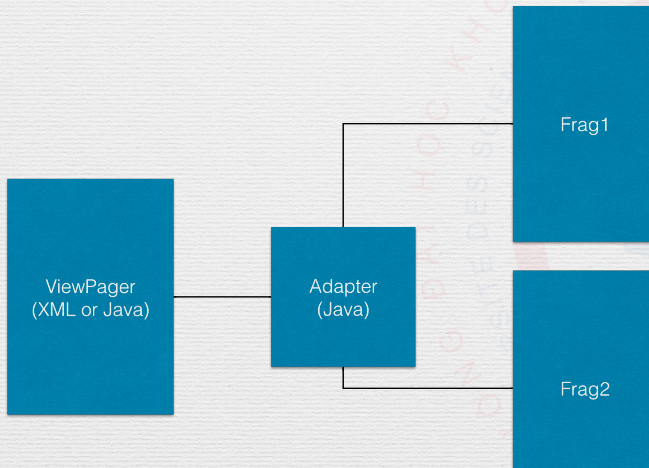
ViewPager

- ViewPager in XML layout

```
<android.support.v4.view.ViewPager  
    android:id="@+id/pager"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

ViewPager

- «Adapter» in Java class (in parent Activity or Fragment)
- Specify what fragment is in what page



Adapter

```
public class HomeFragmentPagerAdapter extends FragmentPagerAdapter {
    private final int PAGE_COUNT = 3;
    private String titles[] = new String[] { "Hanoi", "Paris", "Toulouse" };

    public HomeFragmentPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public int getCount() {
        return PAGE_COUNT;        // number of pages for a ViewPager
    }

    @Override
    public Fragment getItem(int page) {
        // returns an instance of Fragment corresponding to the specified page
        switch (page) {
            case 0: return Fragment1.newInstance();
            case 1: return Fragment2.newInstance();
            case 2: return Fragment3.newInstance();
        }
        return new EmptyFragment(); // failsafe
    }

    @Override
    public CharSequence getPageTitle(int page) {
        // returns a tab title corresponding to the specified page
        return titles[position];
    }
}
```

How to use ViewPager and Adapter

- Setup ViewPager in Activity's onCreate() or Fragment's onCreateView()

```
PagerAdapter adapter = new HomeFragmentPagerAdapter(  
    getSupportFragmentManager());
```

```
ViewPager pager = (ViewPager) findViewById(R.id.pager);  
pager.setOffscreenPageLimit(3);  
pager.setAdapter(adapter);
```

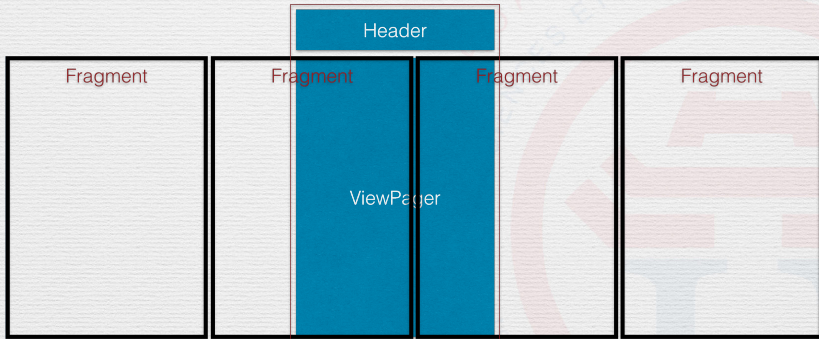
Practical Work 7

- Create a new WeatherAndForecastFragment
- Put your two fragments (WeatherFragment and ForecastFragment) into it
- Remove WeatherFragment and ForecastFragment from WeatherActivity
- Add a ViewPager into WeatherActivity
- Put 3 WeatherAndForecastFragments into the ViewPager
- Swipe!

Header for ViewPager

- A specialized ViewGroup
- TabLayout
- PagerSlidingTabStrip

Header for ViewPager



Header for ViewPager

- build.gradle (of your app, not your project)

```
compile "com.android.support:design:23.1.0"
```

- layout XML: **above** your ViewPager (LinearLayout or RelativeLayout may work)

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tab"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" />
```

- In Activity or Fragment: setup a link between them

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tab);  
tabLayout.setupWithViewPager(pager);
```

Practical Work 8

- Add a header to your ViewPager
 - TabLayout

