Libraries & Publishing Application

Android - Week12

Animations, Picasso, ButterKnife

Libraries

- Many Android developers have produced useful **libraries**.
  - There is a **Maven repository** to store various libraries.
  - This makes it easy to add them to your Android Studio projects.
  - Most libraries use permissive licenses so that you can use them for free and can include them in the code of commercial apps/products.

(Some libraries must be downloaded as .JARs and added manually to your project as we do with the Stanford Android library.)

maven

Adding Library to Project

- Edit the **build.gradle** file for your 'app' module and add lines to the following section at the bottom:
  - You can usually find out what file name to write below by going to various libraries' home pages / GitHub pages.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'

    compile 'your library file here'
    compile 'your library file here'
    ...
    compile 'your library file here'
}
```

Example Lib: ButterKnife

- **ButterKnife** is a popular library intended to simplify usage of Android widgets and events in Java code.
  - written by Jake Wharton
  - http://jakewharton.github.io/butterknife/

- To add ButterKnife to your Android Studio project:

```
dependencies {
    ...
    compile 'com.jakewharton:butterknife:7.0.1'
}
```

ButterKnife Field Bindings

- Using the @Bind annotation, you can declare a field that will always be set to the value of a widget with a certain ID.
  - equivalent to setting it equal to findViewById(R.id.***id***);
  - but retains its state if the activity is closed / reopened

```
public class MyActivity extends Activity {
    @Bind(R.id.mytext) TextView myText;
    @Bind(R.id.myedit) EditText myEdit;

    public void onCreate(Bundle bundle) {
        setContentView(R.layout.activity_my);
        ButterKnife.bind(this);
        myEdit.setText("Wow, cool!");
    }
}
```

ButterKnife Adding Event

- Using @On***Event*** annotations, you can easily attach methods to be event handlers for various widget events.
  - equivalent to calling setOn***Event***Listener on a given view

```
    @OnClick(R.id.mybutton)
    public void handleClick(View view) {
        Log.v("example", "Clicked the button!");
    }

    @OnLongClick(R.id.mytextview)
    public void handleLongClick(View view) {
        Log.v("example", "Long-clicked text view!");
    }
}
```

Example Lib: Android Animations

- An ambitious Android user named *daimajia* has created several libraries, including one to do **animation effects** on Views.
  - https://github.com/daimajia/AndroidViewAnimations
  - To use this library, add the following dependencies:

```
dependencies {
    ...
    compile 'com.nineoldandroids:library:2.4.0'
    compile 'com.daimajia.easing:library:1.0.1@aar'
    compile 'com.daimajia.androidanimations:library:1.1.3@aar'
}
```

Using Animation

- Anywhere in your app's Java code, write:

```
YoYo.with(Techniques.AnimationName)
    . set various properties of the animation
    .playOn(View);
```

- Example:

```
// play a "tada" animation for 700 ms
// that will affect the "edit_area" view
YoYo.with(Techniques.Tada)
    .duration(700)
    .playOn(findViewById(R.id.edit_area));
```

Animations Provided

- Attention
  - Flash, Pulse, RubberBand, Shake, Swing, Wobble, Bounce, Tada, StandUp, Wave
- Special
  - Hinge, RollIn, RollOut, Landing, TakingOff, DropOut
- Bounce
  - BounceIn, BounceInDown, BounceInLeft, BounceInRight, BounceInUp
- Fade
  - FadeIn, FadeInUp, FadeInDown, FadeInLeft, FadeInRight
  - FadeOut, FadeOutDown, FadeOutLeft, FadeOutRight, FadeOutUp
- Flip
  - FlipInX, FlipOutX, FlipOutY
- Rotate
  - RotateIn, RotateInDownLeft, RotateInDownRight, RotateInUpLeft, RotateInUpRight, RotateOut, RotateOutDownLeft, RotateOutDownRight, RotateOutUpLeft, RotateOutUpRight
- Slide
  - SlideInLeft, SlideInRight, SlideInUp, SlideInDown, SlideOutLeft, SlideOutRight, SlideOutUp, SlideOutDown
- Zoom
  - ZoomIn, ZoomInDown, ZoomInLeft, ZoomInRight, ZoomInUp, ZoomOut, ZoomOutDown, ZoomOutLeft, ZoomOutRight, ZoomOutUp

Animation Events

- To hear animation events, pass a class that implements interface AnimatorListener (or extends AnimatorListenerAdapter) that implements some/all of the following methods:

Method	Description
onAnimationStart	called when animation begins
onAnimationEnd	called when animation ends
onAnimationCancel	called if animation is canceled
onAnimationRepeat	called if a looping animation repeats

```
// example
YoYo.with(Techniques.Wobble)
    .withListener(new AnimatorListenerAdapter() {
        public void onAnimationEnd(Animator anim) {
            Log.v("demo", "Animation has ended!");
        }
    }).playOn(findViewById(R.id.myview));
```

Picasso Library

- **Picasso** is a powerful library for manipulating images.
  - written by Square, inc.
  - http://square.github.io/picasso/

- To add Picasso to project:

```
// in build.gradle
dependencies {
    ...
    compile 'com.squareup.picasso:picasso:2.5.2'
}

<!-- in AndroidManifest.xml -->
<uses-permission android:name="android.permission.INTERNET" />
```

Displaying a Photo

- In your app's Java code, write:

```
Picasso.with(this)
    .load("url")
    .into(ImageView);
```

- Example:

```
// show a cute puppy photo
ImageView img = (ImageView) findViewById(R.id.photo);
Picasso.with(this)
  .load('imageurl/imagename                    .jpg")
  .into(img);
```

Picasso Image Methods

Method	Description
`centerCrop()`	center and crop image inside view
`centerInside()`	center image inside view
`error(`***`id`***`)`	show given drawable as error
`fetch()`	download image in the background
`fit()`	resize image to fit view bounds
`get()`	return image as a Bitmap
`into(`***`view`***`)`	puts image into given view
`placeholder(`***`id`***`)`	show given drawable while loading
`resize(`***`width, height`***`)`	change image size in pixels
`rotate(`***`degrees`***`)`	rotate clockwise
`tag("`***`tag`***`")`	attaches a "tag" to a loading image *(useful for bulk operations shown later)*
`transform(`***`trans`***`)`	apply complex transformations

Picasso Methods

Method	Description
`cancelRequest(`***view***`)`	abort any image loading in that view
`cancelTag("`***tag***`")`	cancel all images with given tag
`invalidate("`***url***`")` `invalidate(`***File***`)`	flush out cache of given image, so that it will be re-downloaded the next time

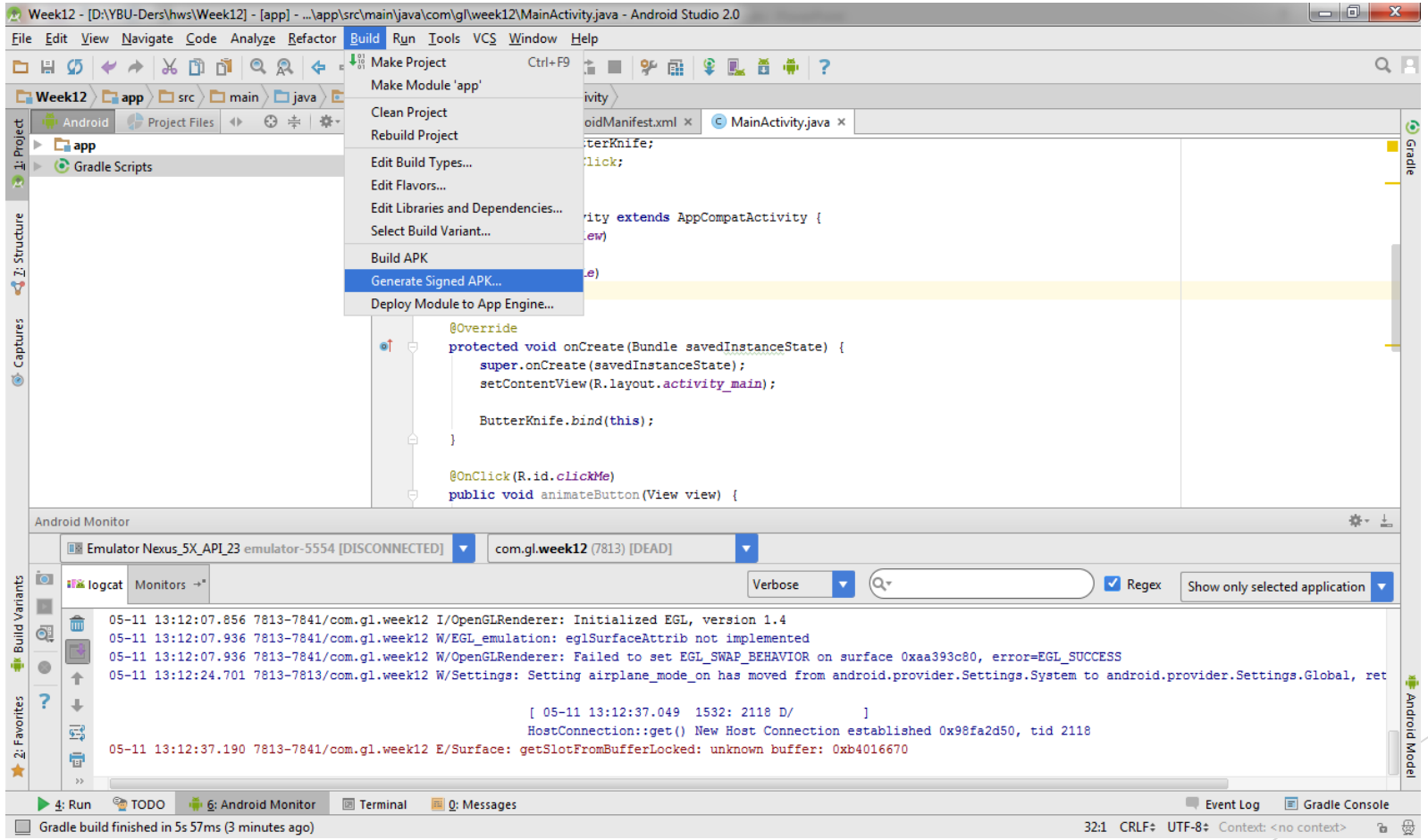
`load("`***url***`")`
`load(`***id***`)`
`load(`***File***`)`
	load an image from various sources

`pauseTag("`***tag***`")`	pause all image loads for given tag
`resumeTag("`***tag***`")`	unpause all image loads for given tag
`shutdown()`	stop entire Picasso system
`with(`***context***`)`	use given activity/fragment as context

More Libraries

- There are of course more than this.
  - Literally thousands of Android Libraries out there
- How to find them
  - The search according to your needs

How to publish



More with Examples

- Class Examples

References

- CS193A Standford Android Course