UNIVERSITY OF PLYMOUTH

COMP2000: Software engineering 2

Android Development

Outline

Troubleshoots

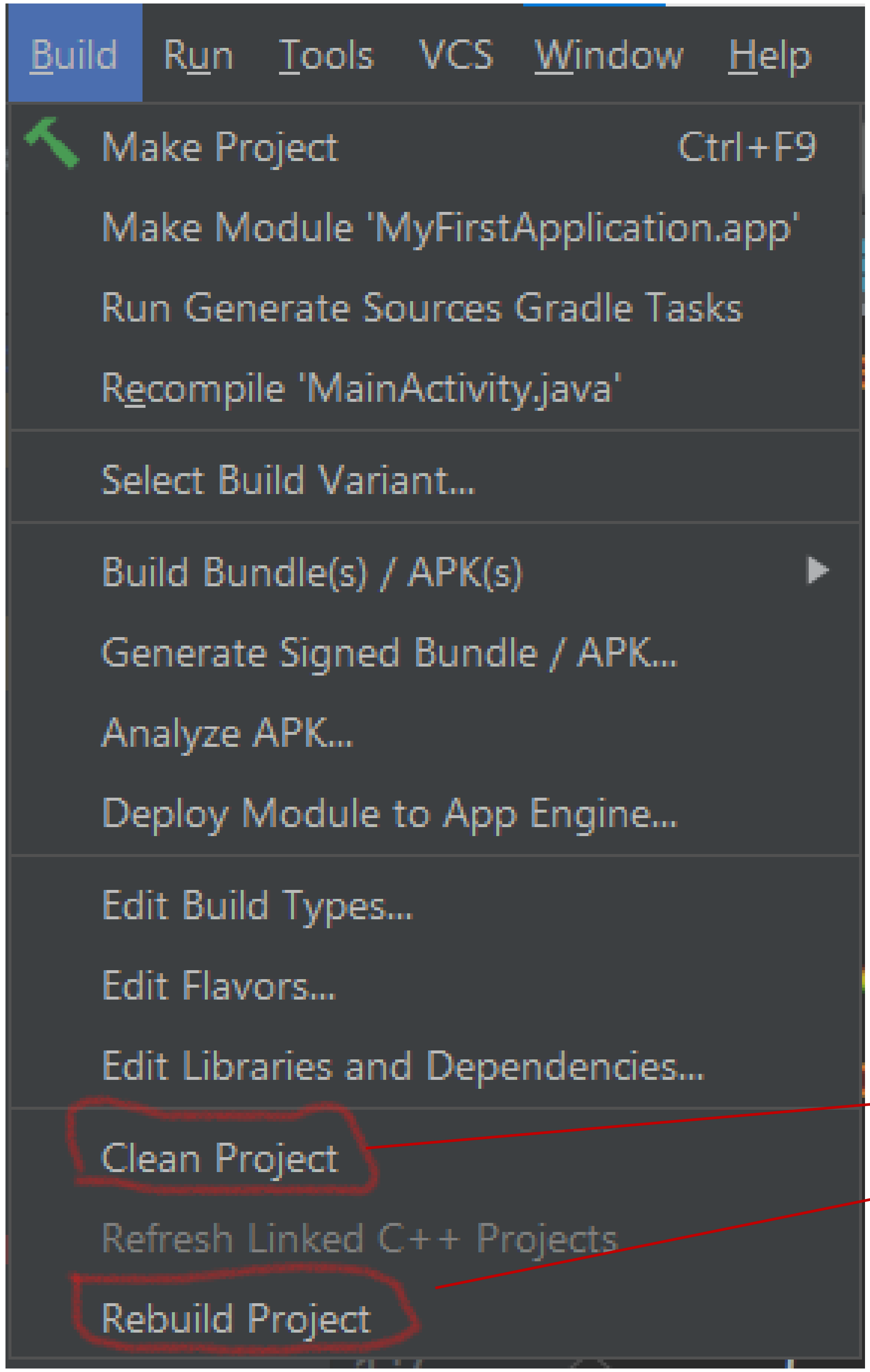
Intent and Intent-filter

Sending data between Activities

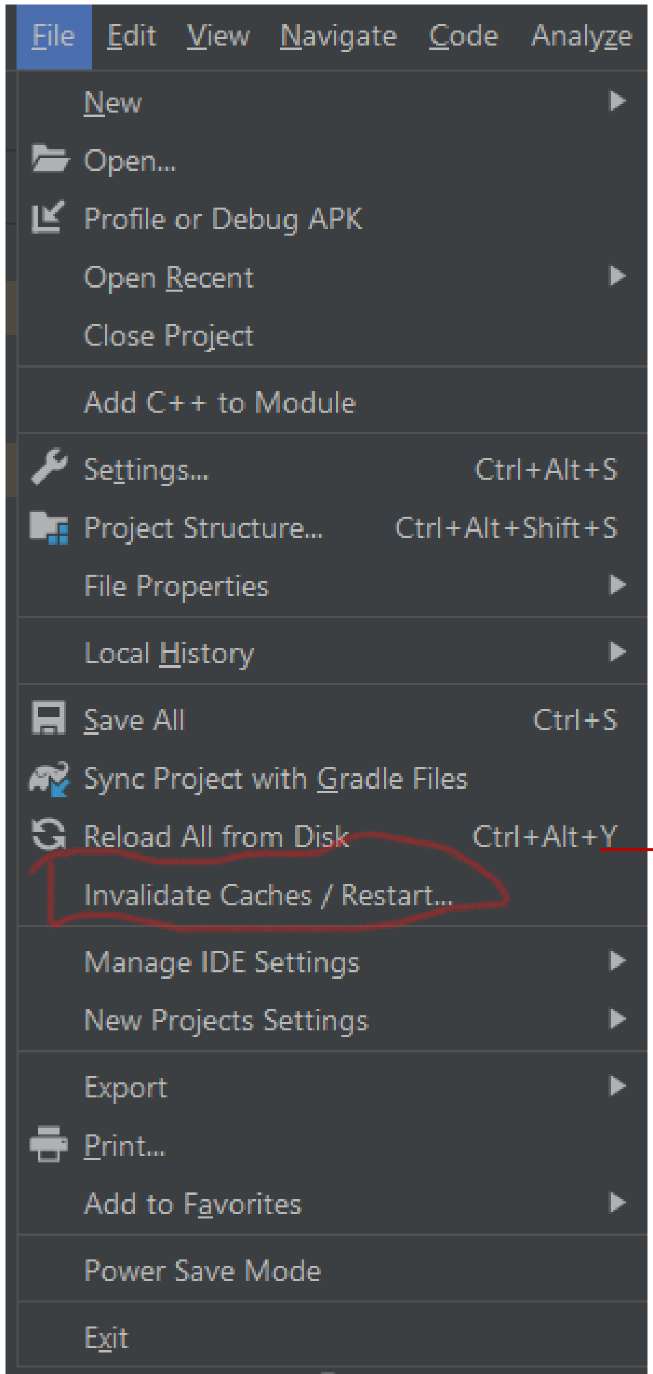
Responding to click events

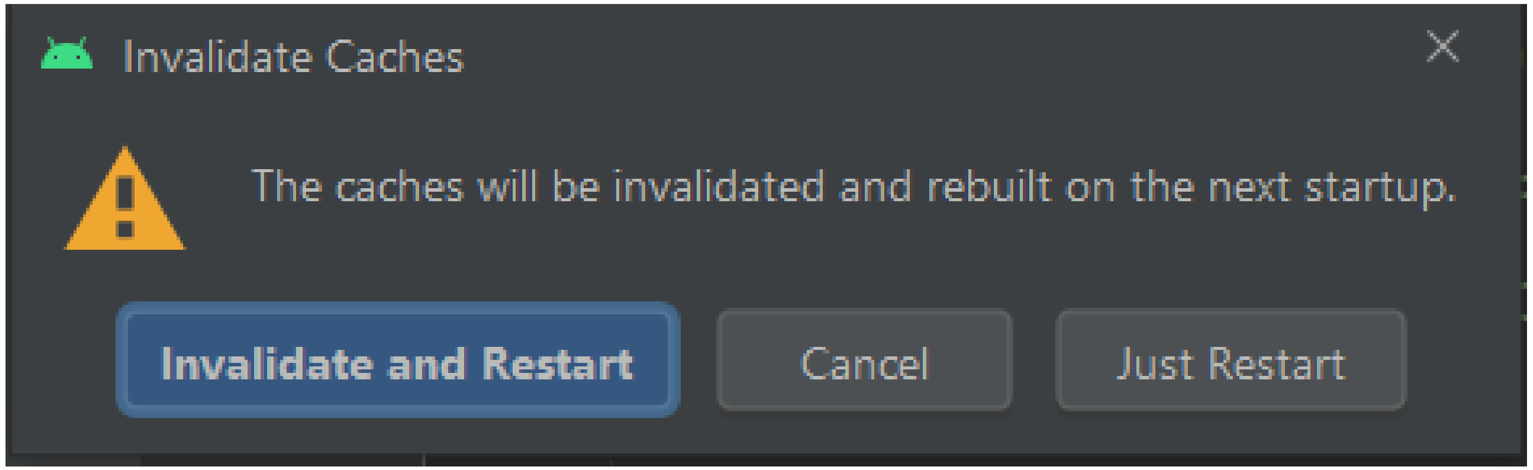
Troubleshoots

- Errors in XML files
- SDK version
- Firewall


Build
Run
Tools
VCS
Window
Help
Make Project Ctrl+F9
Make Module 'MyFirstApplication.app'
Run Generate Sources Gradle Tasks
Recompile 'MainActivity.java'
Select Build Variant...
Build Bundle(s) / APK(s)
Generate Signed Bundle / APK...
Analyze APK...
Deploy Module to App Engine...
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Clean Project
Refresh Linked C++ Projects
Rebuild Project

Use these tools to clean your project and re-build it again





The caches will be invalidated and rebuilt on the next startup.

Helpful resources

- https://developer.android.com/studio/troubleshoot
- https://developer.android.com/studio/known-issues
- StackOverflow

Using your Android device

- Enable USB debugging on your device.
- In Android 4.0 and newer, it's in Settings > Developer options.
- On Android 4.2 and newer, Developer options is hidden by default.
- To make it available, go to Settings > About phone and tap Build number seven times.
- Return to the previous screen to find Developer options.

For more information visit the Android Developer website:

https://developer.android.com/studio/run/device

Intents and Intent Filters

An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use cases:

Starting an activity:
An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to startActivity() . The Intent describes the activity to start and carries any necessary data.

Starting a service:
A Service is a component that performs operations in the background without a user interface. You can start a service to perform a one-time operation (such as downloading a file) by passing an Intent to startService(). The Intent describes the service to start and carries any necessary data.

Delivering a broadcast
A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast() or sendOrderedBroadcast().

- Intent filters are a very powerful feature of the Android platform. They provide the ability to launch an activity based not only on an *explicit* request, but also an *implicit* one.
- For example, an explicit request might tell the system to “Start the Send Email activity in the Gmail app". By contrast, an implicit request tells the system to “Start a Send Email screen in any activity that can do the job."
- When the system UI asks a user which app to use in performing a task, that’s an intent filter at work.

- You can take advantage of this feature by declaring an<intent-filter>attribute in the<activity>element in the manifest file.
- The definition of this element includes an<action>element and, optionally, a<category>element and/or a<data>element.
- These elements combine to specify the type of intent to which your activity can respond.
- For example, the following code snippet shows how to configure an activity that sends text data, and receives requests from other activities to do so:

Manifest file

```
<activity android:name=".ExampleActivity”
android:icon="@drawable/app_icon">
     <intent-filter>
         <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
  </intent-filter>
</activity
```

- In this example, the<action>element specifies that this activity sends data.
- Declaring the<category>element as DEFAULT enables the activity to receive launch requests.
- The<data>element specifies the type of data that this activity can send.
-
- The following code snippet shows how to call the activity described above:

- // Create the text message with a string
- Intent sendIntent=new Intent();
- sendIntent.setAction(Intent.ACTION_SEND);
- sendIntent.setType("text/plain");
- sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
- // Start the activity
- startActivity(sendIntent);

Another example:

```
<activity
              android:name=".AnotherActivity"
              android:label="@string/AnotherActivityName“>
  <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
                  android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
</activity>
```

Sending data between activities

When an app creates an Intent object to use in startActivity(android.content.Intent) in starting a new Activity, the app can pass in parameters using the putExtra(java.lang.String, java.lang.String) method.
The following code snippet shows an example of how to perform this operation.

```
Intent intent = new Intent(this, MyActivity.class);
intent.putExtra("media_id", "a1b2c3");
// ...
startActivity(intent);
```

Example:

```
Intent intent= new Intent(this, MenuActivity.class);
 startActivity(intent);
```

Buttons and Image Buttons

- A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.

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

```xml
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher_background"
    android:contentDescription="@string/app_name"/>
```

```xml
<ImageView
    android:id="@+id/imageView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    android:drawableStart="@drawable/ic_launcher_foreground"
     />
```

How to Respond to Click Events

When the user clicks a button, the Button object receives an on-click event.

To define the click event handler for a button, we should add the android:onClick attribute to the <Button> element in the XML layout.

The value for this attribute must be the name of the method you want to call in response to a click event.

```
<Button
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="button_send"
    android:onClick="sendMessage" />
```

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

Using an OnClickListener

To declare the event handler programmatically, create an View.OnClickListener object and assign it to the button by calling setOnClickListener(View.OnClickListener). For example:

```
Button button = (Button) findViewById(R.id.button_send);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

Thank you