KCL Tech Build X: Android

Lecture One: Environment Setup and First App

- Course and project introduction
- Setting up your development environment
- Getting started
- Components of a basic app
- Exercise: adding a view and an event listener

Course & Project Intro

The best way to learn Android is to **get stuck in** and start playing with it. That's exactly what we're going to do. Over the 8-9 lectures in this course, we will start at the very beginning of Android development and build our way through many basic topics, and some more advanced ones. Throughout the course, everything you need can be found at [https://android.kcl.tech], including lecture notes, code samples, etc.

Throughout this course, **we will work together on a project**: a todo app. Everyone will build their own app, and there will be chances for you to customise and experiment. We will build step by step, exploring the basic features of Android before moving on to some of the more complex topics.

Development Environment Setup

Android apps can be developed from a wide range of IDEs: IntelliJ, Eclipse, Android Studio, etc. All of them have their benefits and drawbacks. For this course, **we are going to be using Android Studio**. It has a range of Android-specific tools, it's easy to set up, and it's simple to install on Linux, Windows and Mac.

You can **download Android Studio** from here [http://tiny.cc/android-studio], where you can also find a guide on how to install it. We will be installing it together, but just in case the WiFi lets us down, **please try to download the file before the lecture.**

The easiest way to install Android Studio is to follow the guide offered on the link before (you will be forwarded to the guide after the download). **The KCL Tech team are here to help with any problems.**

Getting Started

To start things quickly, we have prepared an "empty shell" Android app for you to get started with. You can download this from [https://android.kcl.tech] - just click on `blank-project.zip` under *Lecture One* . **We will explain everything** in the initial project during the first session, but by starting here we can get you building your own apps even quicker!

Download the `.zip` file, extract the contents, and then open up the project in Android Studio. We'll show you exactly how to do this, so don't worry if you're not sure where to start.

Components of an App

Android apps are built from many components, but the ones you'll use most commonly can be grouped into a few categories. Everything we're going to explain now is also **explained in the blank app project files**, so feel free to follow along with us.

Resources

Resource files are written in a markup language called XML. Here's an example that describes a note:

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder to buy pasta</heading>
  <message>Don't forget, buy pasta!</message>
</note>
```

As you can see, it's fairly **self-descriptive**: you can see all the details of the note, you can see it's to "Tove", it's from "Jani", and it's heading is "Reminder to buy pasta". You can describe *anything* in this fashion. For instance:

```
<person>
  <name>Josh</name>
  <age>22</age>
  <gender>Male</gender>
  <favouritefood>Pasta</favouritefood>
</person>
```

When we are writing Android resources, we use XML in the same manner. For instance, we have an XML resource to hold all of our strings (all of our chunks of text). Here's an example from the todo app:

```
<resources>
  <string name="app_name">KCL Tech Todo</string>
  <string name="hello_world">Hello world!</string>
</resources>
```

Notice we have the same format, but this time we have some extra detail in our tags - the `name` part of our `<string>` tags is called an **attribute**. Attributes contain data relating to a specific object in our XML - in this case, we use it to give a `name` to our `<string>` object. So now we know the `<string>` tag containing `Hello World!` has the name `hello_world`.

Resources can be used to define, strings, numbers and booleans, as well as sizes for items, styles and appearances, layouts and views (which we'll cover later) and a whole host of other things. If something for Android *isn't* **Java**, then it's **probably XML**.

The AndroidManifest.xml File

The `AndroidManifest.xml` file cataloges a lot of **information *about* your app**: what version of Android it needs, what permissions it is requesting, what components it includes, etc. As you can tell by the extension, this file also uses XML as a markup language. It includes many Android-specific tags, such as `<application>`, `<activity>`, `<intent-filter>`, `<service>`, etc.

We'll explain each one as we use them in the project.

Layouts & Views

In Android, a layout tells the device **how things should look** - an activity, a fragment, a custom view, etc.

Layouts are made up of **views**, some of which can contain other views (this type of view is called a **view group**). For example, a `TextView` (which is a lot like a `JLabel`, for those who have a Java Swing background) is a simple view so it *cannot* contain other views, but a `LinearLayout` is a view group so it *can* contain other "child" views. (FYI, a `LinearLayout` will stack the views inside it in a line, vertically or horizontally).

Views can be given **custom IDs**, which you can use to access them from your code.

We will work with different layouts and views to create parts of the app, which we will explain along the way.

Activities

An activity is one of the basic building blocks of an Android app. An activity can be thought of as a **distinct "page"** in your app. They have layouts made up of many views (taken from an XML resource), they can handle user interactions, and your user can navigate between them.

Activities have a fairly complex **lifecycle** that they travel through as they are created and destroyed. We will cover the four main events that an Activity can go through in the lecture, but here's a summary:

`onCreate`	the activity has been **created** for the first time, but may not be visible to the user yet
`onResume`	the activity is now "on top" and **can be seen** by the user
`onPause`	the activity has been **obscured or hidden** by something, such as another application opening, the user pressing "Home", a phone call, etc.
`onStop`	the activity has been **stopped** and will terminate completely

The most important thing to know about the lifecycle is that an activity **can exist in the background** without being fully closed. For example, consider that a user is using your app and then receives a phone call - the call covers your activity, but it isn't fully closed; it's "paused". It will be "resumed" when their call ends.

Exercise

Your challenge, should you choose to accept it:

1. Verify that you can run the app on your phone or an emulator.

2. Change the text in some way, via the XML resources. You could change the size, the colour (see [http://www.colorpicker.com] for helping getting the colour codes), the wording, or all three!

3. Add a `<Button>` below your text view, and set the label on it using a string resource. Feel free to look online for help or ask anyone from the KCL Tech Team.

4. **Challenge:** give the button an ID, "find" the button in your activity Java code, and then make it do *something* when you press it. (Need an idea for something to do? Search "Android toast" online.)

The KCL Tech team are **here to help** with any of these tasks - just wave your hand around or shout at us!