Introducción a AndroidAnnotations

Vimos como instalar AndroidAnnotations, para lo cual había que poner en el `app/build.gradle` las siguientes líneas:

- annotationProcessor "org.androidannotations:androidannotations:$AAVersion"
- implementation "org.androidannotations:androidannotations-api:$AAVersion"

Donde `AAVersion` es una variable con la versión de AndroidAnnotations

- def AAVersion = '4.5.2'

Finalmente hicimos un hello world sin AndroidAnnotations y con AndroidAnnotations para validar el ahorro de código que supone así como un código más limpio.

Primeras anotaciones

En este tema vimos las primeras anotaciones que nos permiten crear aplicaciones con AndroidAnnotations. Entre las que destacamos:

- `@EActivity` -> Nos permite mejorar una actividad con AndroidAnnotations y permite especificar su layout. Si no se especifica se deberá emplear `setContentView`.
- `@EApplication` -> Nos permite definir una nueva aplicación dentro del proyecto. Recordar que la clase se debe registrar en el manifest del projecto con la opción "android:name".
- `@EBean` -> Nos permite definir un bean empleando AndroidAnnotations.
- `@EView` -> Nos permite crear un objeto personalizado de UI.
- `@EViewGroup` -> Nos permite crear un objeto personalizado de UI que a su vez contenga otros subelementos.
- `@Fullscreen` -> Nos permite poner una actividad a pantalla completa.
- `@WindowFeature` -> Permite configurar elementos de la vista de una actividad.

Inyección

En esta sección se vieron distintas anotaciones que permiten la instanciación de clases:

- `@Bean` -> Permite instanciar la anotación `@EBean`. Permite instanciar de tres maneras diferentes (1) directamente sobre el objeto `@Bean MyClass myClass;` (2) en un método `@Bean void setOneBean(MyClass myClass)` ó `void setMultipleBeans(@Bean MyClass myClass, @Bean MyOtherClass myOtherClass)` (3) empleando una interfaz `@Bean(MyImplementation.class) MyInterface myInterface;`.
- `@Extra` -> Permite recibir una información enviada por un intenet, para lo

cual empleamos

```
<CLASS>_.intent(<CONTEXT>).<VAR_WITH_EXTRA>(<MESSAGE>).start() ;
```

Permite instanciar de dos maneras diferentes (1) directamente sobre el objeto `@Extra MyClass myClass;` (2) en un método `@Extra void setOneBean(MyClass myClass)` ó `void setMultipleExtras(@Extra MyClass myClass, @Extra MyOtherClass myOtherClass).` Aunque en la respectiva clase no lo vimos permite de forma adicional configurar un valor por defecto `@Extra("myDateExtra") Date myDateExtraWithDefaultValue = new Date();`.

- `@App` -> Permite instanciar la anotación `@EApplication`. Permite ser configurado tanto a nivel de objeto como de método.
- `@RootContext` -> Permite obtener el contexto de la aplicación. Permite ser configurado tanto a nivel de objeto como de método.

Ciclo de vida de una app

Las anotaciones del ciclo de vida nos permiten hacer operaciones en determinados momentos de creación. Las anotaciones vistas fueron:

- `@AfterExtras` -> Permite hacer operaciones tras realizar las inyección de los extra.
- `@AfterInject` -> Permite hacer operaciones tras realizar las inyección de las inejcciones.
- `@AfterViews` -> Permite hacer operaciones tras realizar las inyección de las vistas.

Recursos de aplicación

Permite obtener recursos de la aplicación:

- `@StringRes` -> Permite obtener los recursos de `strings.xml`. Se puede emplear de dos maneras diferentes, especificando el nombre del recurso que se va a instancar por medio de una variable a la anotación `@StringRes(R.string.xxx)` o si no se especifica dicho nombre del recurso, entonces se buscará en los recursos de `strings.xml` el nombre de la variable Java.
- `@ColorRes` -> Permite obtener los recursos de `colors.xml`. Se puede emplear de dos maneras diferentes, especificando el nombre del recurso que se va a instancar por medio de una variable a la anotación "@ColorRes(R.color.xxx)" o si no se especifica dicho nombre del recurso, entonces se buscará en los recursos de `colors.xml` el nombre de la variable Java.
- `@BooleanRes` -> Permite obtener los recursos de `booleans.xml`. Se puede

emplear de dos maneras diferentes, especificando el nombre del recurso que se va a instancar por medio de una variable a la anotación `@BooleanRes(R.bool.xxx)` o si no se especifica dicho nombre del recurso, entonces se buscará en los recursos de `booleans.xml` el nombre de la variable Java.

- `@AnimationRes` -> Permite obtener los recursos de `anim.[animation].xml`. Se puede emplear de dos maneras diferentes, especificando el nombre del recurso que se va a instancar por medio de una variable a la anotación `@AnimationRes(R.anim.xxx)` o si no se especifica dicho nombre del recurso, entonces se buscará en los recursos de `anim.[animation].xml` el nombre de la variable Java.
- `@IntArrayRes` -> *Práctica final* -> Permite objeter los recursos en array. Se puede emplear de dos maneras diferentes, especificando el nombre del recurso que se va a instancar por medio de una variable a la anotación `IntArrayRes(R.yyy.xxx)` o si no se especifica dicho nombre del recurso, entonces se buscará en los recursos el nombre de la variable Java. Un array en los res deberá estar definido como:

```
<array name="hulk_colors">
    <item>@color/hulk_1</item>
    <item>@color/hulk_2</item>
    <item>@color/hulk_3</item>
    <item>@color/hulk_4</item>
    <item>@color/hulk_5</item>
</array>
```