Android – Butterknife

Introducción

En este post trataremos con una librería muy útil. En este caso nos ayuda deshacernos de mucho código. De esta manera nuestra aplicación nos queda mucha más limpia y legible. Lo mejor de todo es que esta librería inyecta código en tiempo de compilación, es decir, el rendimiento de nuestra aplicación no se verá afectado por su uso.

- Configuración

- Ejemplo de uso con Activity
- Ejemplo de uso con Fragment
- Eventos
- Usando Resources
- Plugin ButterKnife Zelezny

Configuración

Para la instalación, tenemos que añadir el plugin android-apt a nuestro classpath en el fichero build.gradle. Este se encuentra en la raíz de nuestro proyecto.

```
dependencies{
  classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
}
```

Dentro del archivo app/build.gradle, debemos añadir el plugin antes de añadir las dependencias de Butterknife.

```
apply plugin : 'com.neenbedankt.android-apt'

dependencies {
  compile 'com.jakewharton:butterknife:8.0.1'
  apt 'com.jakewharton:butterknife-compiler:8.0.1'
}
```

Ejemplo de uso

Eliminaremos el uso de findViewById utilizando @BindView en los views de Android (TextView, Button, EditText…):

```
class MainActivity extends Activity {
  // Automatically finds each field by the specified ID.
  @BindView(R.id.title) TextView title;
  @BindView(R.id.name) EditText name;
  @BindView(R.id.btn_send_name) Button sendName;

  @Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_activity);
```

```
    ButterKnife.bind(this);
    // Some code
  }
}
```

Como podemos ver se simplifica el código y a su vez se hace más legible.

Ejemplo de uso con Fragment

Cuando utilicemos fragments tendremos que especificar en el método bind la vista con la que vamos a trabajar y en el evento onDestroyView utilizar unbind:

```
public class SimpleFragment extends Fragment {
  @BindView(R.id.txt_name) Button name;
  @BindView(R.id.btn_send_name) Button sendName;

  @Override public View onCreateView(LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
     View view = inflater.inflate(R.layout.simple_fragment,
container, false);
    ButterKnife.bind(this, view);
    // Some code
    return view;
  }

  // When binding a fragment in onCreateView, set the views to
null in onDestroyView.
   // Butter Knife has an unbind method to do this
automatically.
  @Override public void onDestroyView() {
    super.onDestroyView();
    ButterKnife.unbind(this);
  }
}
```

Eventos

Los eventos serán funciones con la anotación correspondiente.

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

También podemos agrupar vistas y asignarlas a un único evento

```
@OnClick({R.id.maint_btn_change_text,
R.id.main_btn_new_intent})
    void buttonClick(View v) {
        switch (v.getId()){
            case R.id.maint_btn_change_text:
                title.setText(name.getText().toString());
                break;
            case R.id.main_btn_new_intent:
                 Intent resourcesIntent =  new Intent(this,
ResourcesActivity.class);
                startActivity(resourcesIntent);
                break;
        }
    }
```

Podremos utilizar los siguientes eventos: OnClick, OnLongClick, OnEditorAction, OnFocusChange, OnItemClick, OnItemLongClick,OnItemSelected, OnPageChange, OnTextChanged, OnTouch, OnCheckedChanged.

Usando resources

Podemos hacer binding de resources fácilmente

```
@BindString(R.string.title) String title;
@BindDrawable(R.drawable.my_drawable) Drawable myDrawable;
@BindColor(R.color.red) int red;
```

Plugin ButterKnife Zelezny

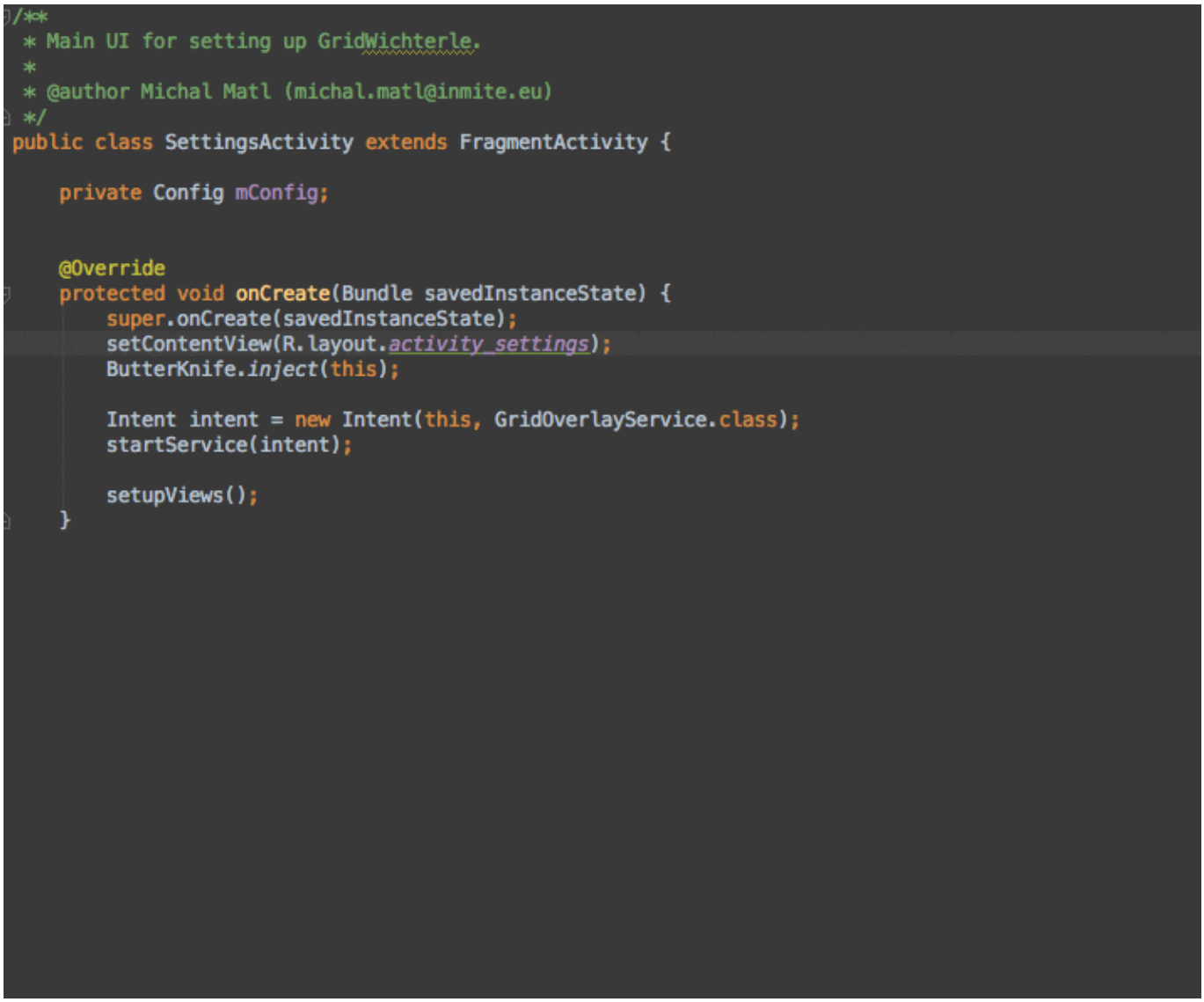
Este es un plugin para Android Studio muy útil. A partir de

una vista nos genera todos los bindviews para esta.

Primeramente deberemos instalar el plugin en Android Studio:

- Des de Android Studio: iremos a **File** -> **Settings** -> **Plugins** -> **Browse repositories** y buscaremos **ButterKnife Zelezny**
- Descargandolo: decargamos el plugin ButterKnife Zelezny y lo instalamos des de **File** -> **Settings** -> **Plugins** -> **Install plugin from disk**

Para hacer uso del plugin debemos incluir Butterknife tal y como hemos mostrado al principio del post. Finalmente, os dejo un gif sacado de la página del proyecto de GitHub donde se muestra su utilización.



```java
/**
 * Main UI for setting up GridWichterle.
 *
 * @author Michal Matl (michal.matl@inmite.eu)
 */
public class SettingsActivity extends FragmentActivity {

    private Config mConfig;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        ButterKnife.inject(this);

        Intent intent = new Intent(this, GridOverlayService.class);
        startService(intent);

        setupViews();
    }
```

Como nota final, decir que se generan los nombres de las variables a partir de los ids asignados a las vistas del layout seleccionado. Es decir, **main_btn_show_toast** se convertirá en **mainBtnShowToast**. En el cuadro de dialogo podremos modificar esos nombres, uno a uno claro. Si observamos atentamente un correcto “naming” en nuestro layout nos generará automáticamente variables con un “naming” adecuado. El problema es que nos encontramos con la anotación correspondiente al layout, **main**BtnShowToast. Solucionar esto es fácil, cogeremos como ejemplo las variables correspondientes a los Buttons del layout main:

- Seleccionamos el layout y la primera letra correspondiente a la vista que queremos modificar

```
@BindView(R.id.main_btn_resources) Button mainBtnResources;
@BindView(R.id.main_btn_disable_button) Button mainBtnDisableButton;
```

- Seguidamente con atajo de teclado **alt + j** seleccionaremos todas la variables correspondientes al layout y la inicial de la vista a modificar

```
@BindView(R.id.main_btn_resources) Button mainBtnResources;
@BindView(R.id.main_btn_disable_button) Button mainBtnDisableButton;
```

- Y finalmente, corregimos el nombre para todas las variables

```
@BindView(R.id.main_btn_resources) Button btnResources;
@BindView(R.id.main_btn_disable_button) Button btnDisableButton;
```

Observaciones

Butterknife nos ofrece una manera de mantener nuestro código limpio y legible. Algo que a largo plazo se hace indispensable, pues cualquier software se va a tener que mantener. Y que nuestro código sea limpio y legible es vital para esta tarea. Hemos mostrado también un plugin muy útil para utilizar la librería y al final hemos hablado un poco de naming. Esto daría para otro post y llegará. Por otro lado también hemos mostrado un atajo de teclado de Android Studio, los atajos de teclado para Android Studio dan para otro futuro post y así podríamos seguir y no parar nunca. Finalmente, dejo el enlace a la web de Butterknife (donde podréis encontrar ejemplos más avanzados), el enlace a la cuenta de GitHub del plugin ButterKnife Zelezny y un simple ejemplo en Github (el proyecto de Picasso refactorizado usando Butterknife).

- Butterknife
- ButterKnife Zelezny
- ButterKnifeTest