ViewBinding vs DataBinding

- *ViewBinding*
- *DataBinding*
- *@BindAdapters*

Tokio.

ViewBinding

- ***ViewBinding*** es principalmente una función que permite vincular las vistas del diseño XML con la vista que los controla (Activity-Fragment-CustomView).

- El *ViewBinding* viene a sustituir el conocido "*findViewById*" que se utiliza para obtener la referencia de la vista XML.

- Las principales ventajas son:

    1. **Seguridad nula**: El uso del *ViewBinding*, al crear una vinculación directa, evita el cruce de referencias y por ende el *Nullpointer* de la vista. Ej. Cuando nos traemos la referencia Textview de la actividad 1 en la actividad 2 => En este caso la app genera un *Nullpointer* al cruzar la referencia. Con *ViewBinding* evitaremos este problema.

    2. **Seguridad en los tipos**: Hace referencia a que la vinculación directa no necesita hacer un *cast* del tipo y evitamos así, el error de tipos en la definición.

- *ViewBinding* cuando se activa genera clases de vinculación, estas clases se encuentran en la *build*, por lo que se construyen dinámicamente.

ViewBinding – Implementación

- **Habilitar *ViewBinding***: En el archivo **build.gradle** del módulo, debemos de agregar:

```
android {
    ...
    viewBinding {
       enabled = true
    }
}
```

- **Invocación del objeto *ViewBinding*** : El nombre de la clase de vinculación se genera convirtiendo el nombre del archivo XML según la convención de mayúsculas y minúsculas, y agregando la palabra "**Binding**" al final.

Ejemplos:

Nombre layout: login_activity => objeto binding que se genera es **LoginActivityBinding**

Nombre layout: home_fragment => objeto binding que se genera es **HomeFragmentBinding**.

Nombre layout: ítem_main_product => objeto binding que se genera es **ItemMainProductBinding**

ViewBinding – Implementación *Activity*

1. Debemos de instanciar el objeto generado y guardarlo como una propiedad de la *activity*.
2. El objeto instanciado, tiene un método estático para inflar el *layout*.
3. Debemos de pasarle al "**setContentView**" de la *activity* el root del *layout*.

```
private ActivityMainBinding binding;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   // Debemos recuperar la vista-layout- a utilizar
   binding = ActivityMainBinding.inflate(getLayoutInflater());
   setContentView(binding.getRoot());
```

Tokio.

ViewBinding – Implementación *Fragment*

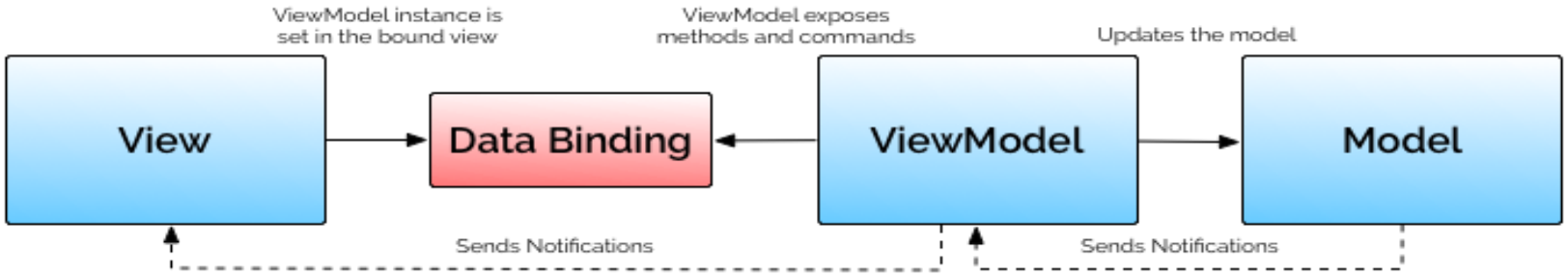
1. Debemos de instanciar el objeto generado y guardarlo como una propiedad del *Fragment*. Este se crea en el método sobrescrito "**onCreateView**"
2. El objeto instanciado, tiene un método estático para inflar el *layout*. A diferencia de una *Activity* debemos indicar el objeto *inflater* que viene como parámetro y sobre que *ViewGroup* se debe de inflar (también viene por parámetro), por ultimo pasamos *false*.
3. Debemos devolver el *root* de la vista.

```
//Variable del binding del Fragment
private FragmentViewbindingInflateBinding binding;

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
    //Crear la vista del Fragment
    binding = FragmentViewbindingInflateBinding.inflate(inflater, container, false);
    binding.txtInflateBinding.setText("ESTAMOS EN VIEWBINDING INFLATE");
    return binding.getRoot();
}
```

DataBinding

- La biblioteca de vinculación de datos es una biblioteca de compatibilidad que permite vincular los componentes de la IU de tus diseños a las fuentes de datos de tu app usando un formato declarativo en lugar de la programación.
- ***DataBinding*** es el proceso que establece una conexión entre la UI y la lógica de negocio **-modelo de Dominio-**. Cuando una variable se modifica desde la UI, el modelo de negocio se actualiza automáticamente y viceversa.
- En Android, *DataBinding* es ofrecido por *Jetpack architecture* y su uso es recomendable en una arquitectura de MVVM. *DataBinding* esta desarrollado con un patrón *observer* y por ende cualquier cambio es notificado a sus suscriptores.



Tokio.

DataBinding vs ViewBinding

- Si bien el proceso de clases generadas y su uso es muy similar entre ambos existen alguna diferencia. Entendemos como vinculación de vista al *ViewBinding* y vinculación de dato al *DataBinding*.
- **Según Android *Developers*** "La biblioteca de vinculación de datos es una biblioteca de compatibilidad que permite vincular los componentes de la IU de tus diseños a las fuentes de datos de tu app usando un formato declarativo en lugar de la programación."
- Por lo tanto la ventaja de *ViewBinding* es:

  Compilación más rápida.

  Facilidad en su uso.

- La ventaja de *DataBinding* sobre *ViewBinding* es:

  Vincular el dato en formato declarativo en el archivo XML.

  Vinculación del dato bidireccional.

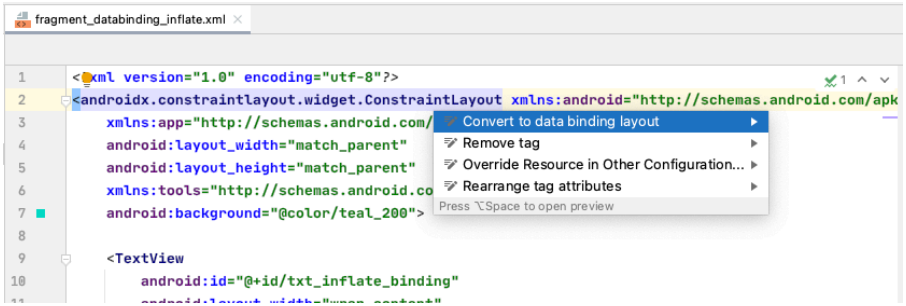
- Teniendo las limitaciones claras, Android *Developer* nos indica "Puedes usar la vinculación de datos en diseños que requieren funciones avanzadas y usar la vinculación de vista en diseños que no lo requieren."

DataBinding – Implementación

- **Habilitar *DataBinding***: En el archivo **build.gradle** del módulo, debemos de agregar:

```
android{
.....
  buildFeatures {
   dataBinding = true
   }
}
```

- **Invocación al objeto *DataBinding* y uso:** idéntico al *ViewBinding*.
- **Layout XML**: Deberemos activar *DataBinding* y nos genera un *tag* a nivel *root* como **<layout>**



DataBinding – Implementación

- **Java-class**: Al igual que *ViewBinding* debemos de generar el objeto generado y a posteriori podremos setear el valor del **<variable>** que hemos definido dentro del *tag* **<data>.**

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="user"
            type="com.tokioschol.tp8_viewbinding_databinding.databinding.domain.UserModel" />

    </data>
```

```java
public class DataBindingFragment extends Fragment {

    private FragmentDatabindingInflateBinding binding;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        //Crear la vista del Fragment
        binding = FragmentDatabindingInflateBinding.inflate(inflater, container, attachToRoot: false);
        //Set data al layout-objeto binding-
        binding.setUser(provideUser());
        return binding.getRoot();
    }
```

Tokio.

DataBinding – Implementación

- **Java-class**: Al igual que *ViewBinding* debemos de generar el objeto generado y a posteriori podremos setear el valor del **<variable>** que hemos definido dentro del *tag* **<data>.** Una vez se ha realizado el set del dato ya se puede utilizar el mismo de forma declarativa.

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="user"
            type="com.tokioschol.tp8_viewbinding_databinding.databinding.domain.UserModel" />

    </data>
```

```java
public class DataBindingFragment extends Fragment {

    private FragmentDatabindingInflateBinding binding;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        //Crear la vista del Fragment
        binding = FragmentDatabindingInflateBinding.inflate(inflater, container, attachToRoot: false);
        //Set data al layout-objeto binding
        binding.setUser(provideUser());
        return binding.getRoot();
    }
```

```xml
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/teal_200">

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/databinding_username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@{user.name}"
        android:textColor="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="DATABINDING INFLATE" />
```

Tokio.

DataBinding – Expresiones

- matemáticos + - / * %
- de concatenación de strings +
- lógicos && ||
- binarios & | ^
- unarios + - ! ~
- mayúscula >> >>> <<
- de comparación == > < >= <= (ten en cuenta que < debe tener el formato de escape <)
- Instanceof
- agrupación ()
- literales: caracteres, strings, números, null
- de transmisión
- de llamadas a métodos
- de acceso de campo
- de acceso de matriz []
- operadores ternarios ?:

DataBinding – Adapters

En Android, cuando se utilizan *DataBindings*, el *framework* se encarga de hacer el set con la propiedad del SDK. Ahora bien, es posible que dicha propiedad no esté creada o estemos interesados en construir una adaptación propia. Para ello Android nos facilita la anotación *BindAdapter*.

Un *BindAdapter* no es más que la creación "*custom*" de un *setter* de una propiedad en XML. Veamos un ejemplo:

En este ejemplo vemos los nombres de la propiedad que se utilizará en XML"colorView"-"scrVectorDrawable" y la función que ejecuta cuando el *layout* los invoca.

Y el método debe de recibir como parámetros, el objeto desde el que se invoca y el tipo de valor a recibir.

```
@BindingAdapter(value = {"app:colorView","app:srcVectorDrawable"},requireAll = true)
public static void setSrcVector(AppCompatImageView view, @ColorRes int color, Drawable resource){
    if(view!=null){
        view.setImageDrawable(resource);
        view.setColorFilter(color);
    }
}
```

```
droidx.appcompat.widget.AppCompatImageView
 android:layout_width="75dp"
 android:layout_height="75dp"
 android:layout_marginTop="25dp"
 app:colorView="@{user.name != null ?@color/green:@color/red}"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/databinding_lastName"
 app:srcVectorDrawable="@{user.name != null ?@drawable/ic_satisfied:@drawable/ic_dissatisfied}"
 tools:srcCompat="@drawable/ic_dissatisfied" />
```

Tokio.

Referencias bibliográficas

- https://developer.android.com
- ***VIEWBINDING***: https://developer.android.com/topic/libraries/view-binding
- ***DATABINDING***: https://developer.android.com/topic/libraries/data-binding
- ***BINDADAPTERS***: https://www.develou.com/binding-adapters/
- ***BINDADAPTER ANDROID***: https://developer.android.com/topic/libraries/data-binding/binding-adapters