Android

Interacción del usuario

Tema 4

RecyclerView

Contenido

- RecyclerView
- Implementar un RecyclerView
- OnClick en un elemento
- Extra: Libreria Glide

¿Qué es un RecyclerView?

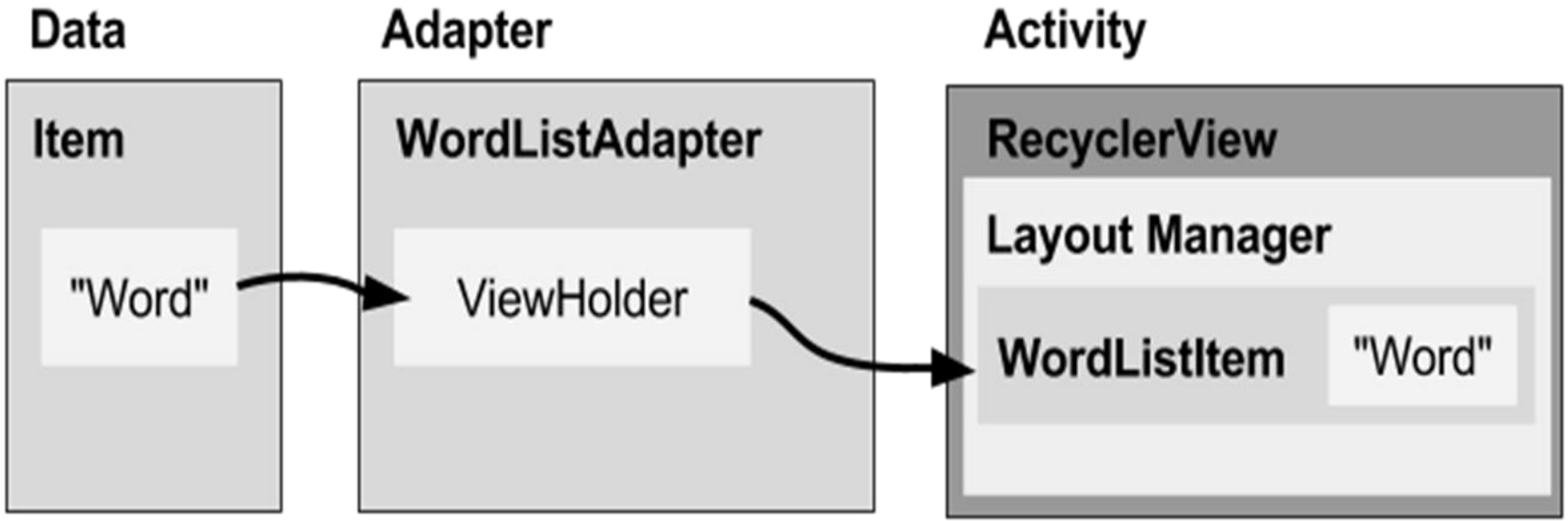
- RecyclerView es un contenedor de tipo scroll utilizado para mostrar muchos datos
- Eficaz
  - Utiliza y reutiliza un número ilimitado de elementos.
  - Actualizaciones de datos es inmediatata.

RecyclerView

Componentes

- **Datos**
- **RecyclerView** lista scroll para los elementos de la lista—`RecyclerView`
- **Layout** diseño de un elemento de la lista — fichero XML
- **Layout manage**r manejador de la visualizacion de los elementos del recyclerView —**`Recyclerview.LayoutManager`**
- **Adapter** conecta los datos con el `RecyclerView`—`RecyclerView.Adapter`
- **ViewHolder** informacion de como se visualiza un elemento del recyclerView—**`RecyclerView.ViewHolder`**

Comportamiento del RecyclerView

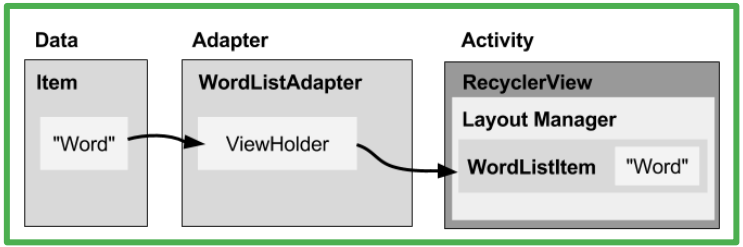


¿Qué es el layout manager?

- Se usa para posicionar los items dentro del `RecyclerView`
- Reutilizar los elementos que no son visibles para el usuario
- Gestores layout managers
  - `LinearLayoutManager`
  - `GridLayoutManager`
  - `StaggeredGridLayoutManager`

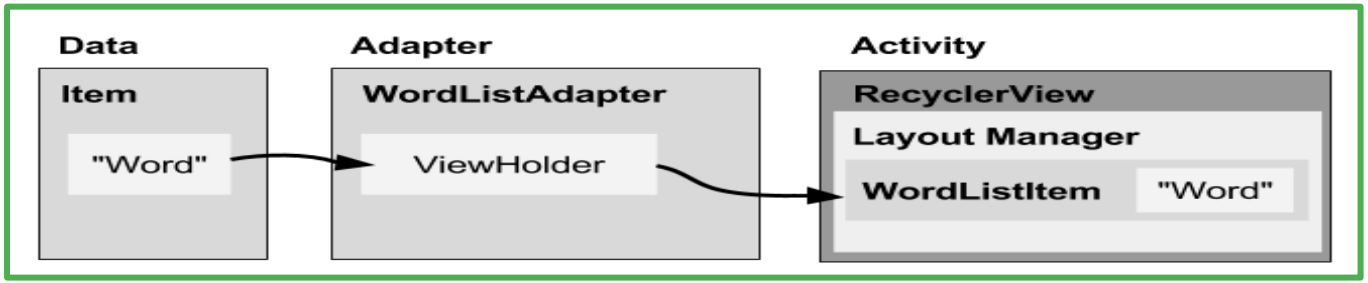
¿Qué es el adaptador?

- Ayuda a que las interfaces trabajen juntas
  - Ejemplo: coge los datos y los coloca en la View
- Intermediario entre los datos y la `View`
- Administra la creación, actualización, adicción y eliminación de los elementos a medida que cambia los datos
- `RecyclerView.Adapter`



¿Qué es un ViewHolder?

- Lo utiliza el adaptador para preparer el `View` con datos de la lista
- Se muestran en el Layout especificado en el fichero XML
- Puede tener elementos clickable



Implementar un RecyclerView

Pasos a realizar

1. Agregar la dependencia `RecyclerView` en build.gradle si es necesario.
2. Agregar el componente `RecyclerView` en el layout
3. Crear el layout XML para el item
4. Crear el Adapter extendiendo del `RecyclerView.Adapter`
5. Extender el adapter del `RecyclerView.ViewHolder`
6. En `Activity` `onCreate()`, trabajar con el `RecyclerView` , el adapter y layout manager

Agregar dependencia a app/build.gradle

Agregar dependencia `RecyclerView` al build.gradle si es necesario:

```
dependencies {
   ...
   compile 'com.android.support:recyclerview-v7:26.1.0'
   ...
   }
```

Agregar al Layout XML el RecyclerView

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recyclerview"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</android.support.v7.widget.RecyclerView>
```

Crear el layout para 1 list item

```
<LinearLayout …>
   <TextView
      android:id="@+id/word"
      style="@style/word_title" />
</LinearLayout>
```

Implementar el adaptador

```
public class WordListAdapter(private val words: List<Elemento>)
        :RecyclerView.Adapter<WordListAdapter.ViewHolder> {


        //crear la class interna ViewHolder
        class ViewHolder(view: View):RecyclerView.ViewHolder(view){


        }
    }
}
```

Adaptador tiene 3 metodos obligatorios

- `onCreateViewHolder()`
- `inBindViewHolder()`
- `getItemCount()`

Echemos un vistazo!

Adapter:onCreateViewHolder()

```
override fun onCreateViewHolder(parent: ViewGroup,
viewType: Int): ViewHolder {
    val view = LayoutInflater.from(parent.context)
          .inflate(R.layout.item_elemento, parent, false)

        //retorna el ViewHolder ya creado
        return ViewHolder(view)
    }
```

Adapter:onBindViewHolder()

```
override fun onBindViewHolder(holder: ViewHolder,
position: Int) {
    //dato a mostrar con la position del elemento
    holder.bind(words[position])

   val word = words.get(position)
   holder.bind(word)
}
```

Adapter:getItemCount()

```
override fun getItemCount(): Int {
        return words.size
    }
```

Adapter: ViewHolder Class

```
class WordViewHolder extends RecyclerView.ViewHolder { //.. }
```

View holder constructor

```
class ViewHolder(view: View) : RecyclerView.ViewHolder(view){

    //para trabajar con el binding dentro de la vista
    private val binding = ItemElementoBinding.bind(view)

    //crear una funcion donde indicamos los elemento a mostrar
    fun bind(elementoData: ElementoData){
        binding.textView.text = elementoData.word //recoger el dato
que tenemos en la lista
        }
    }
```

Crear el RecyclerView en Activity onCreate()

```
// asignar el layout
  binding.recycler.layoutManager = LinearLayoutManager(this)
// asignar el adaptador
  binding.recycler.adapter = UserAdapter(getUser())

//crear funcion getUser() para introducir los datos de la lista
```

Controlar el clic del ratón

Si desea controlar los clics del ratón:

Hay que trabajar con las siguientes clases:

- Construcción del Adapter
- La class `onBindViewHolder`
- `MainActivity`

Adaptador

```
class UserAdapter(private val users:
List<UserData>, private val listener: (UserData)
-> Unit):
RecyclerView.Adapter<UserAdapter.ViewHolder> (){

//lambda para el clic del ratón

//entrada:datos, salida:Unit

}
```

onBindViewHolder

```
override fun onBindViewHolder(holder: ViewHolder, position: Int){
        val user = users.get(position)
        holder.bind(user)
        //itemView Nos indica el numero de la view que se esta
visualizando
         holder.itemView.setOnClickListener { listener(user) }
    }
```

MainActivity

```
//en el adaptador
 binding.recycler.adapter = UserAdapter(getUser()) { user->
      //mostrar un toast
      Toast.makeText(this,user.getFullName(),
           Toast.LENGTH_SHORT).show()
       } // asignar el adaptador
```

Practica: RecyclerView

- Esto es bastante complejo con muchas piezas separadas. Por lo tanto, hay que praticar, implementa un RecyclerView que muestra una lista de elementos en las que se puede hacer clic..

- Muestra todos los pasos, uno a uno con una app completa

Aprende más

- RecyclerView
- RecyclerView class
- RecyclerView.Adapter class
- RecyclerView.ViewHolder class
- RecyclerView.LayoutManager class

Glide – librería

- Glide es una librería para cargar imágenes
- Hay que añadir la librería como una dependencia
- Se utiliza de la siguiente forma:

```
//cargar la imagen con Glide
    Glide.with(binding.imagen)
        .load(photo.url)
        .into(binding.imagen)
```

END

Parcelable

Añadir plugin

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-parcelize'
}
```

Implementar la interface

```
import android.os.Parcelable

import kotlinx.parcelize.Parcelize


@Parcelize

data class Photo(val title: String, val url: String) : Parcelable
```