RECYCLERVIEW

CONTENU

- Composantes d'un RecyclerView
- Implémentation d'un RecyclerView

RECYCLERVIEW

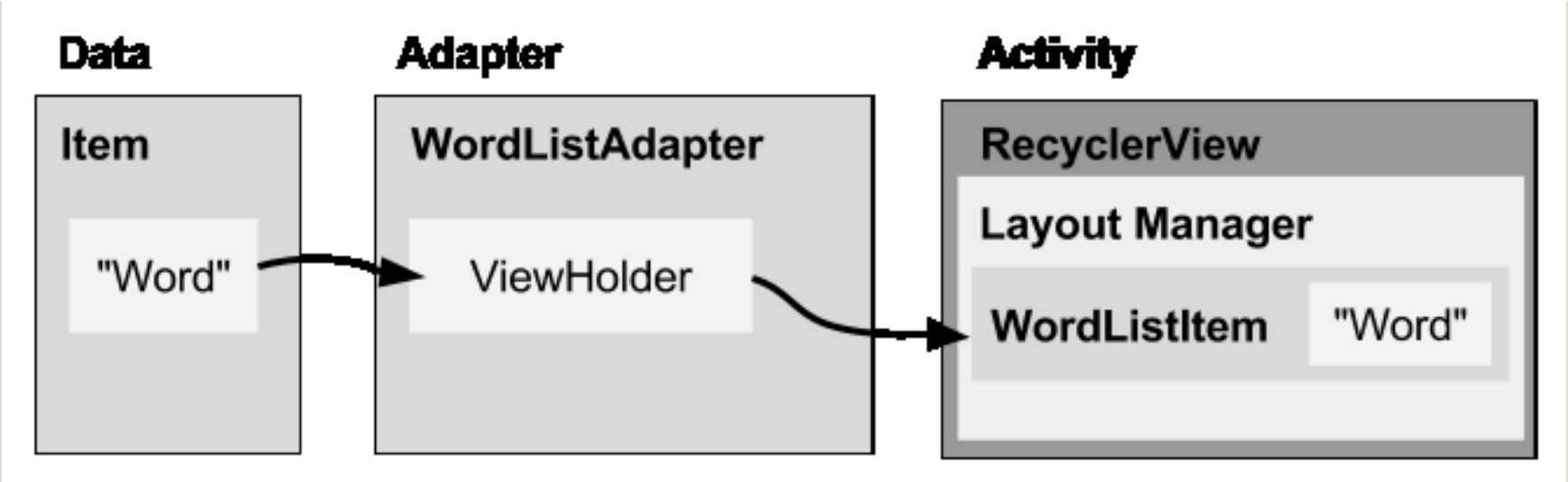
- `RecyclerView`: conteneur pour une grande liste de données
- Efficient
  - Utilise et réutilise un nombre de vues limité
  - Met à jour les données qui changent rapidement

LES COMPOSANTES D'UN RECYCLERVIEW

COMPOSANTES

- **Données**
- **RecyclerView** liste déroulante pour une liste d'items —`RecyclerView`
- **Layout** pour un élement de la liste —XML file
- **Layout manage**r s'occupe de l'organisation des éléments de la liste dans le RecyclerView—`Recyclerview.LayoutManager`
- **Adapter** alimente le `RecyclerView` en données provenant d'une source quelconque—`RecyclerView.Adapter`
- **ViewHolder** contient de l'information pour afficher un élément de la liste — `RecyclerView.ViewHolder`

COMPOSANTES

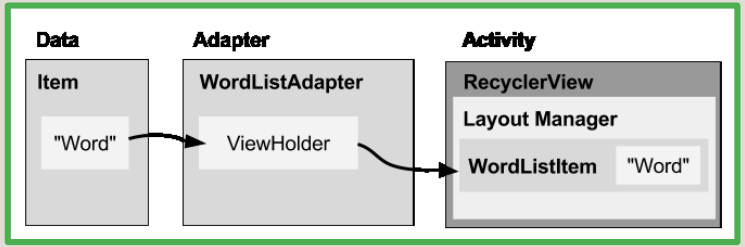


GESTIONNAIRE DE LAYOUT OU LAYOUT MANAGER

- Chaque `ViewGroup` contient un gestionnaire de layout
- Utilisé pour positionner les éléments dans le `RecyclerView`
- Réutilise les éléments de la liste (vue) qui ne sont plus visibles à l'utilisateur
- Layout managers qui viennent avec le RecyclerView
  - `LinearLayoutManager`
  - `GridLayoutManager`
  - `StaggeredGridLayoutManager`
- Hérite de `RecyclerView.LayoutManager`

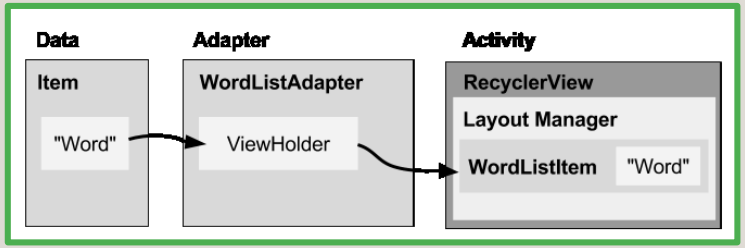
ADAPTEUR

- Peut aider dans la communication d'interfaces incompatibles
- Example: Récupère les données d'une base de données et prépare des chaines de caractères à insérer dans une vue
- Sert d'intermediaire entre les données et le RecyclerView ou Spinner ou autres
- Gère la création, la mise à jour, l'insertion, la suppression des éléments de concert avec l'évolution des données
- `RecyclerView.Adapter`



ADAPTEUR

- Utilisé par l'adapteur pour préparer un élément de la liste avec les données nécessaires
- Layout spécifié dans un fichier XML
- Peut contenir des éléments cliquables, touchables, etc.
- Sa disposition dans le RecyclerView est défini par le gestionnaire de layout
- `RecyclerView.ViewHolder`



IMPLÉMENTATION D'UN RECYCLERVIEW

IMPLÉMENTATION

1. Ajouter les dépendances par rapport à l'utilisation des `RecyclerView` dans le fichier to build.gradle au besoin
2. Ajouter le `RecyclerView` à votre fichier de layout (XML)
3. Créer le fichier de layout XML layout pour un élément de la liste
4. Créer votre adapteur héritant de `RecyclerView.Adapter`
5. Créer un ViewHolder héritant de `RecyclerView.ViewHolder`
6. Dans la fonction onCreate() pour une activité, créer le `RecyclerView, son adapteur et son` layout manager

AJOUT DE DÉPENDANCE (APP/BUILD.GRADLE)

Ajoutez la dépendance au fichier build.gradle au besoin

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

LAYOUT (ACTIVITÉ OU FRAGMENT)

AJOUTER LE RECYCLERVIEW AU LAYOUT (XML)

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

LAYOUT PAR ÉLÉMENT

CRÉER UN LAYOUT POUR UN ÉLÉMENT DE LA LISTE

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

CRÉATION D'UN ADAPTEUR

```
private inner class StudentAdapter(var students:
List<Student>):RecyclerView.Adapter<StudentViewHolder>(){


}
```

ADAPTEUR: ONCREATEVIEWHOLDER()

- `onCreateViewHolder()`
- `onBindViewHolder()`
- `getItemCount()`

ADAPTEUR: ONCREATEVIEWHOLDER()

```
@Override fun onCreateViewHolder: StudentHolder(
parent: ViewGroup, viewType: Int) {
        // Create view from layout
         val view = layoutInflater.inflate(
             R.layout. List_item_student, parent, false);
        return StudentHolder(view);
}
```

ADAPTEUR: ONBINDVIEWHOLDER()

```
override fun onBindViewHolder(holder: StudentHolder, position: Int)
{
    val student = students[position]
    holder.apply {
        titleTextView.text = student.title
        dateTextView.text = student.birthDate.toString()
    }
}
```

ADAPTEUR: GETITEMCOUNT()

```
override fun getItemCount() = students.size
```

ADAPTEUR: LA CLASSE VIEWHOLDER

```
private inner class StudentViewHolder(view:View)
    : RecyclerView.ViewHolder(view), View.OnClickListener {
```

VIEWHOLDER: CONSTRUCTEUR

```
private inner class StudentViewHolder(view: View)
    : RecyclerView.ViewHolder(view), View.OnClickListener {
    private lateinit var student: Student
    val titleTextView: TextView = view.findViewById(R.id.student_title)
    val dateTextView: TextView = view.findViewById(R.id.student_birthdate)
    private val studentImageView: ImageView = view.findViewById(R.id.good_student)

    init {
        view.setOnClickListener(this)
    }
```

VIEWHOLDER: GESTION D'ÉVÉNEMENTS

```
    override fun onClick(p0:View?) {
        Toast.makeText(context, "${student.title} pressed!",Toast.LENGTH_SHORT).show()
    }
}
```

CRÉATION D'UN RECYCLERVIEW

CRÉER LE RECYCLERVIEW DANS ONCREATEVIEW()

```
// Inflate the layout for this fragment
val view = inflater.inflate(R.layout.fragment_student, container, false)

adapter = StudentAdapter(students)

studentRecyclerView.adapter = adapter
```

RÉFÉRENCES

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