# RecyclerView sous Android

Le **RecyclerView** est un widget Android utilisé pour afficher une grande quantité de données dans une liste ou une grille, de manière performante. Les méthodes associées au **RecyclerView** et à son **Adapter** sont essentielles pour comprendre son fonctionnement. Voici une explication détaillée des principales méthodes utilisées :

## Méthodes dans RecyclerView

### 1. setLayoutManager()

Définit comment les items doivent être disposés (liste verticale, grille, etc.).

```
// Liste verticale
recyclerView.setLayoutManager(new LinearLayoutManager(this));
// Grille avec 2 colonnes
recyclerView.setLayoutManager(new GridLayoutManager(this, 2));
```

**Options disponibles** :

- **LinearLayoutManager** : Disposition verticale ou horizontale.
- **GridLayoutManager** : Affiche les items sous forme de grille.
- **StaggeredGridLayoutManager** : Grille avec des hauteurs/largeurs variables.

### 2. setAdapter()

Associe un Adapter au RecyclerView. L'adaptateur est responsable de fournir les données et de créer les vues pour les items.

```
recyclerView.setAdapter(new MyAdapter(dataList));
```

### 3. addItemDecoration()

Ajoute des décorations (comme des séparateurs ou des marges) entre les items.

```
recyclerView.addItemDecoration(new DividerItemDecoration(this, DividerItemDecoration.VERTICAL));
```

### 4. setHasFixedSize()

Optimise les performances en indiquant que la taille du RecyclerView ne changera pas.

```
recyclerView.setHasFixedSize(true);
```

## Méthodes dans l'Adapter

### 1. onCreateViewHolder(ViewGroup parent, int viewType)

Crée une nouvelle instance de ViewHolder pour chaque item visible.

```
@Override
public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_layout,
parent, false);
    return new MyViewHolder(view);
}
```

### 2. onBindViewHolder(ViewHolder holder, int position)

Lie les données à la vue associée à une position spécifique.

# RecyclerView sous Android

```
@Override
public void onBindViewHolder(MyViewHolder holder, int position) {
    holder.textView.setText(dataList.get(position));
}
```

### 3. getItemCount()

Retourne le nombre total d'éléments dans la liste.

```
@Override
public int getItemCount() {
    return dataList.size();
}
```

### 4. Méthodes Spécifiques pour Notifications

Ces méthodes permettent de notifier le RecyclerView d'un changement dans les données.

- **notifyItemInserted(int position)** : Notifie qu'un nouvel item a été ajouté.
- **notifyItemRemoved(int position)** : Notifie qu'un item a été supprimé.
- **notifyItemChanged(int position)** : Notifie qu'un item a été modifié.
- **notifyDataSetChanged()** : Notifie que toute la liste a changé.

## Méthodes dans le ViewHolder

### 1. ViewHolder

Contient les références aux vues des items, afin de les réutiliser (RecyclerView réutilise les vues pour économiser des ressources).

```
public static class MyViewHolder extends RecyclerView.ViewHolder {
    TextView textView;
    ImageView imageView;

    public MyViewHolder(View itemView) {
        super(itemView);
        textView = itemView.findViewById(R.id.text);
        imageView = itemView.findViewById(R.id.image);
    }
}
```

**Ordre d'Exécution**

1. **onCreateViewHolder** : Appelé pour créer un nouveau ViewHolder lorsque nécessaire.
2. **onBindViewHolder** : Appelé pour associer les données au ViewHolder existant.
3. **getItemCount** : Détermine combien d'items le RecyclerView doit afficher.