IUT de Paris - Rives de Seine
Université Paris Cité

R4.Real.11 – Développement pour applications mobiles

– Cours 3 –

Listes et vues à adaptateur

Pr Chaouche A.-C.
ac.chaouche@gmail.com

Résumé

Prérequis

- Gestion des vues et des layouts
- Gestion des évènements

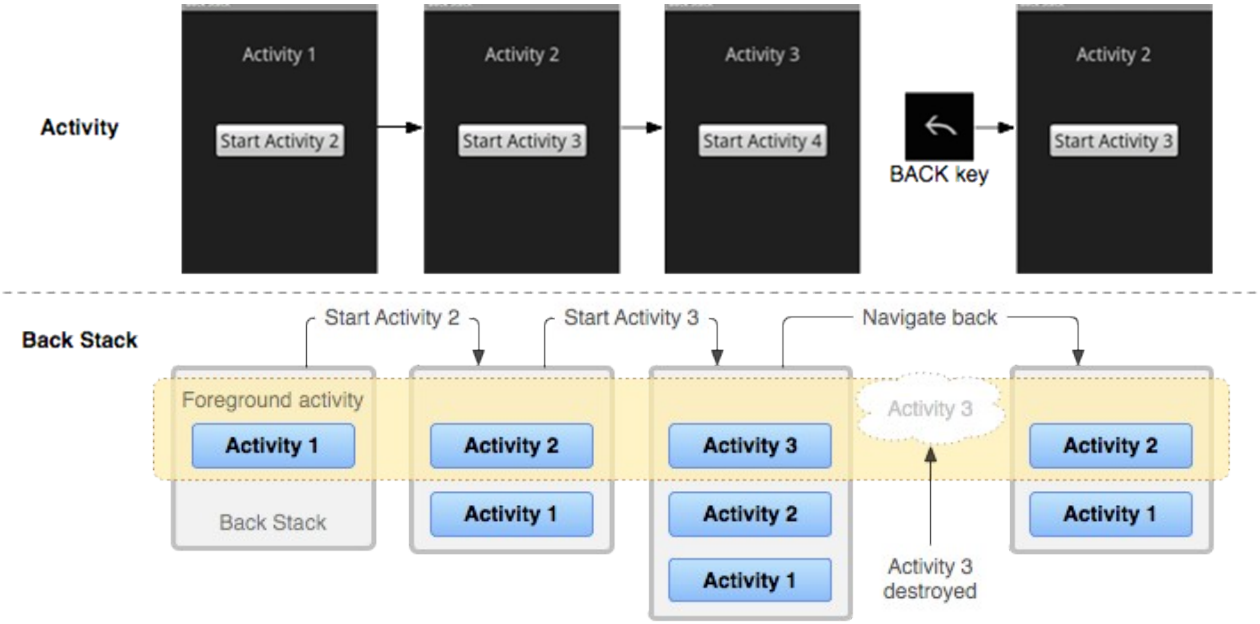
Objectifs du cours

- Passer d'une activité à une autre
- Créer et manipuler des vues à adaptateur (ListView, GridView, …)

Pile des activités

Les activités sont empilées/dépilées

- **Empilée** quand une activité démarre
- **Dépilée** (détruite) quand on presse le bouton **"BACK"**
- Une pression sur le bouton **"HOME"** ne dépile pas l'activité (passe simplement en arrière plan)



Intentions

- Gérer l'envoi et la réception de messages afin de faire coopérer les activités (ou même les applications)
- Déléguer une action à un **composant**, une **application** ou une **activité de l'application courante.**

3 cas d'usage principaux des intents

Pour démarrer une activité :

- en utilisant **`startActivity(Intent)`**, l'intent décrit l'activité et ses paramètres

Pour démarrer un service :

- en appellant **`startService(Intent)`**, démarrer un service (application sans IHM)

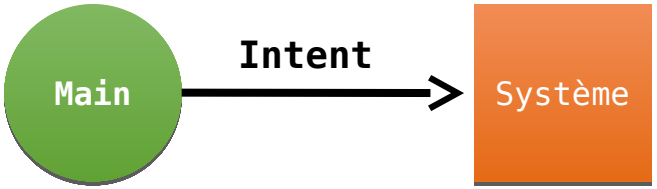
Pour envoyer un broadcast :

- en utilisant **`sendBroadcast(Intent)`**, un broadcast est un message que toute application peut recevoir

Types d'intentions (1/2)

Intent implicite

- Donner le nom d'une action générale
- Un composant d'une autre application peut traiter l'action
- Le système trouve la bonne application en utilisant les **intent-filters** déclarés dans le **manifest**



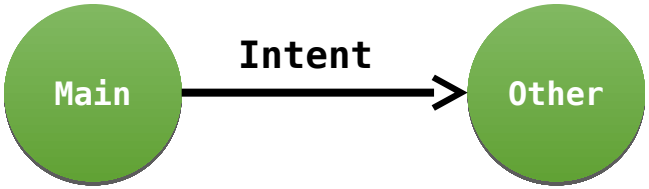
/java/MainActivity.java

```
...
Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:06xxx"));
startActivity(intent);
```

Types d'intentions (2/2)

Intent explicite

- Fournir le nom de la classe de l'activité à démarrer
- Les activités doivent être de la même application



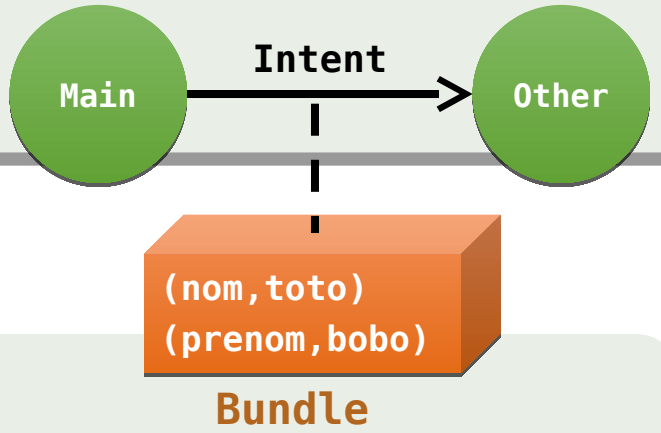
/java/MainActivity.java

```
...
Intent intent = new Intent(this, OtherActivity.class);
startActivity(intent);
```

Transfert de données

/java/MainActivity.java

```
Intent intent = new Intent(this, OtherActivity.class);
Bundle bundle = new Bundle();
bundle.putString("nom", "toto");
bundle.putString("prenom", "bobo");
intent.putExtras(bundle);
startActivity(intent);
```



/java/OtherActivity.java

```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
String nom = bundle.getString("nom");
String prenom = bundle.getString("prenom");
```

- Les types complexes (c-à-d les objets) doivent implémenter l'interface **Parcelable**, ou **Serializable**

Transfert d'objets complexes (1/2)

- Passage d'un type complexe (objet) est réalisé à travers la sérialisation
- **La sérialisation (Marshaling)** permet de rendre un objet persistant pour un stockage ou un échange
- Sérialiser un objet consiste à le convertir en un tableau d'octets, qui pourra être reconstitué à l'identique à la réception
- Sa reconversion vers sa représentation initiale (dans la mémoire) est appelée **désérialisation (unmarshaling)**
- La classe de l'objet à sérialiser doit implémenter l'interface `Serializable`

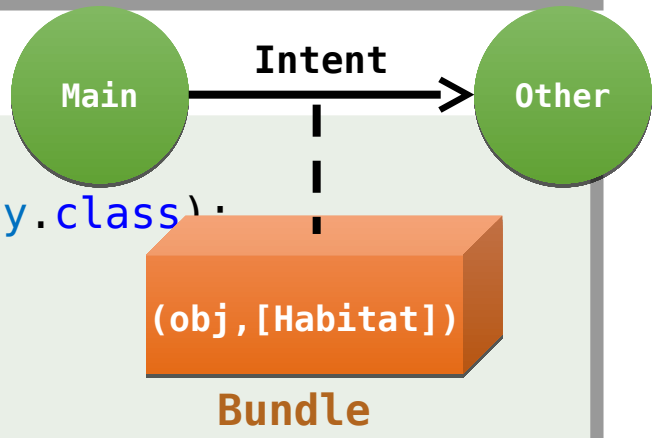
Transfert d'objets complexes (2/2)

/java/Habitat.java

```
public class Habitat implements Serializable {
    ...
}
```

/java/MainActivity.java

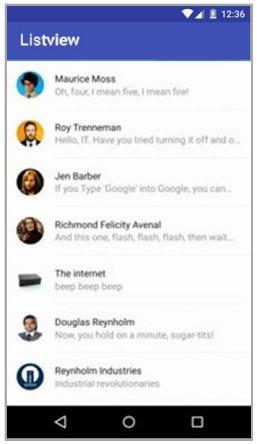
```
Habibtat h = new Habitat(...);
Intent intent = new Intent(this, OtherActivity.class);
Bundle bundle = new Bundle();
bundle.putSerializable("obj", h);
intent.putExtras(bundle);
startActivity(intent);
```

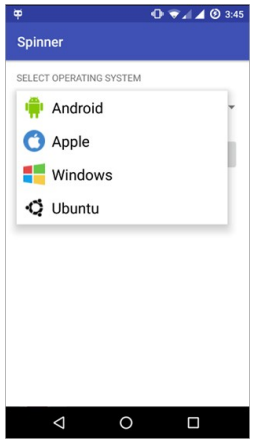


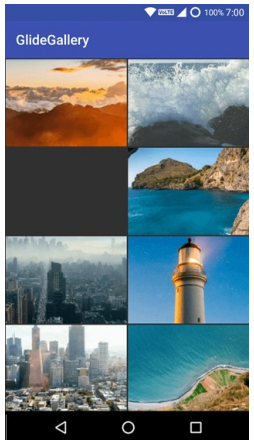
/java/OtherActivity.java

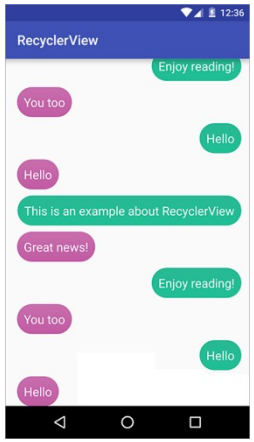
```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
Habitat h = (Habitat) bundle.getSerializable("obj");
```

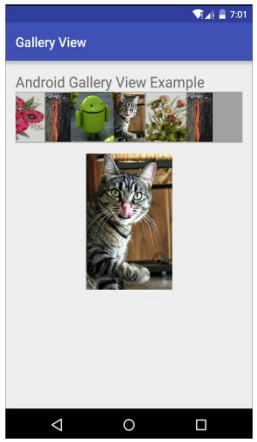
Vues à adaptateur

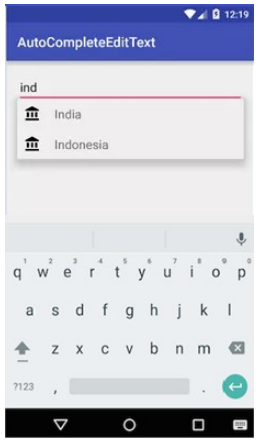










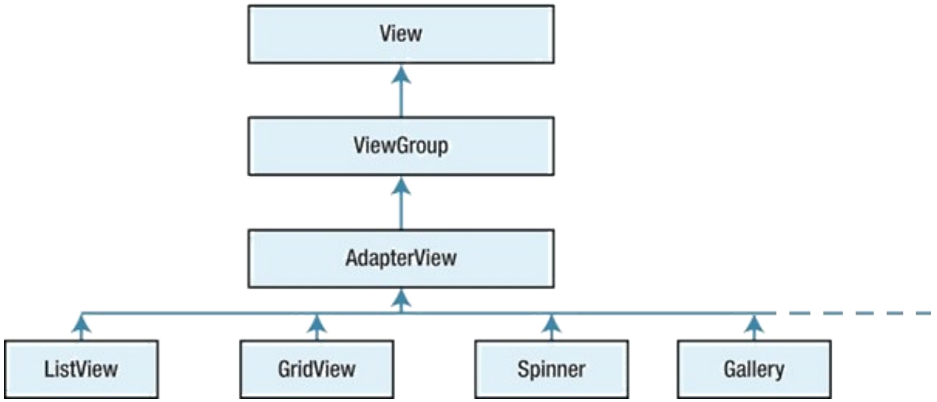


Vues à adaptateurs

- Une vue à adaptateur (**`AdapterView`**) est une sous-classe de **`ViewGroup`** où ses vues filles sont déterminées par un **`Adapter`** qui relie l'objet **`AdapterView`** aux données.

- Pour afficher des collections (**`List`**, **`Set`**, **`Map`**, ...)

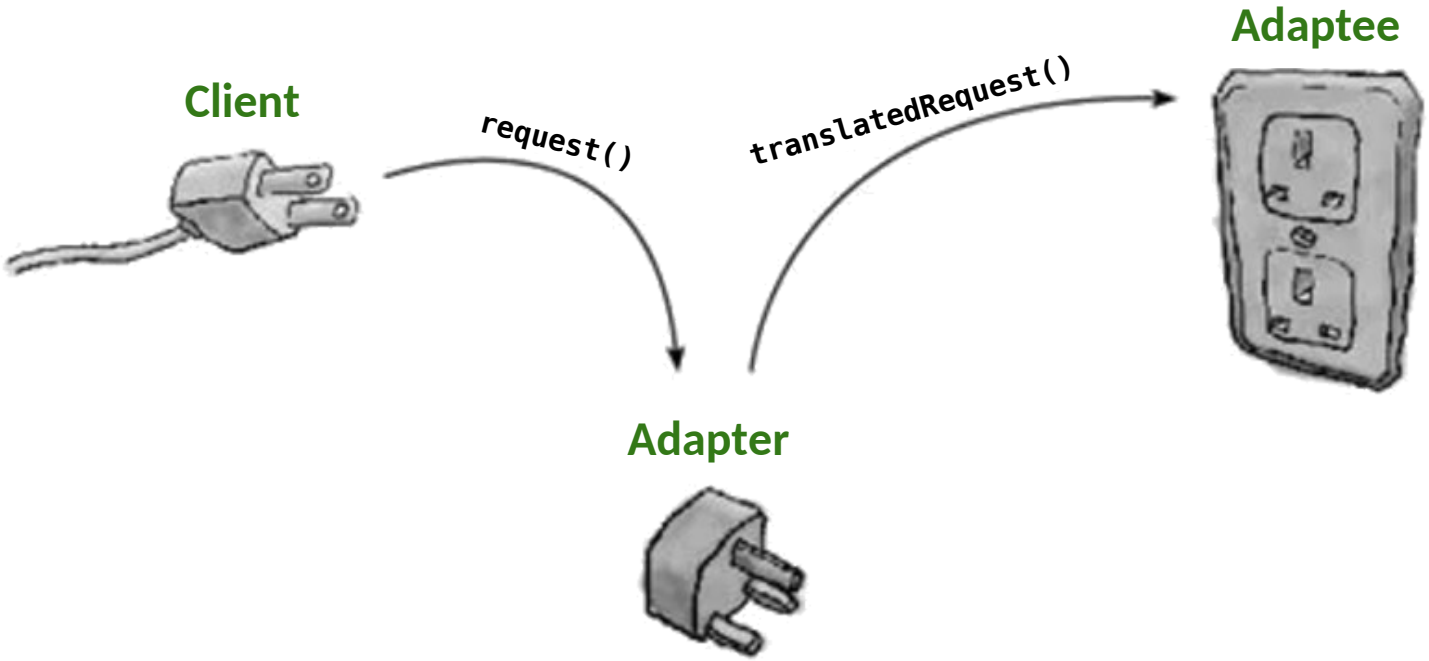
- Héritent de **`AdapterView`** :
  - **`ListView`**
  - **`Spinner`**
  - **`GridView`**
  - **`AutoCompleteTextView`**
  - **`Gallery`**
  - **`...`**



- Ces vues gèrent leurs données via des adaptateurs

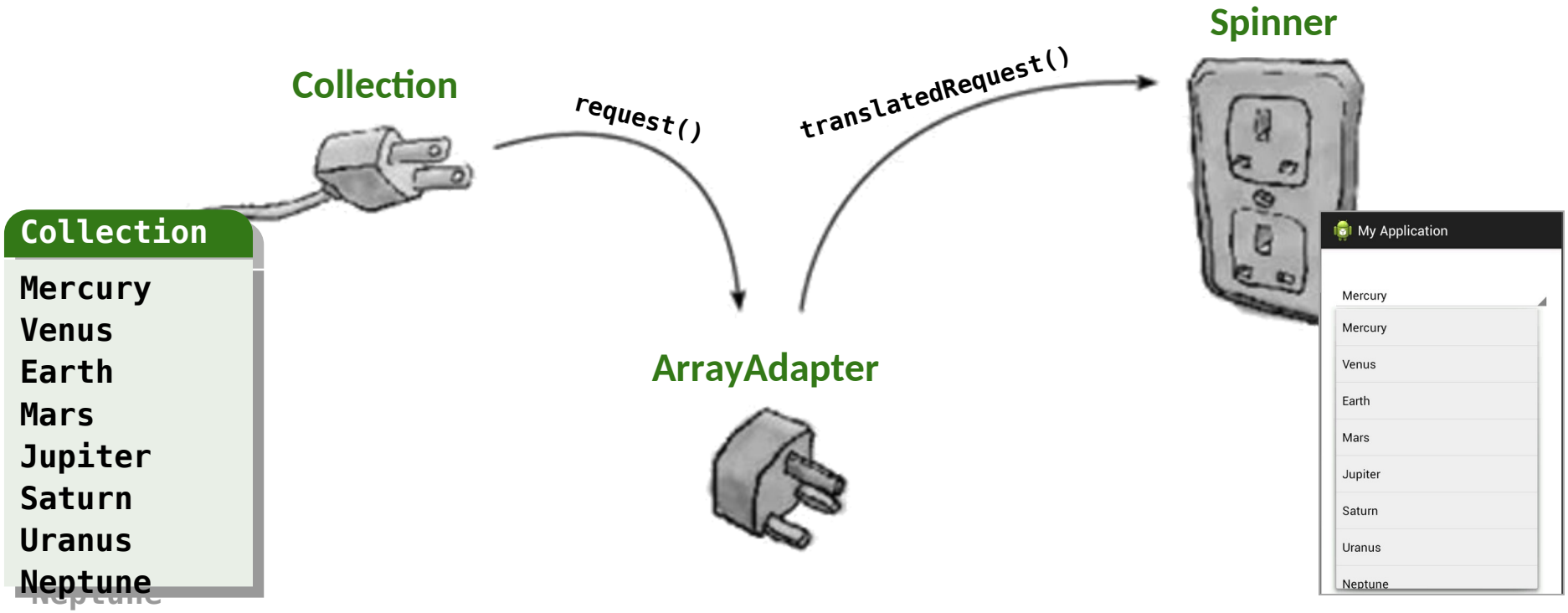
Vues à adaptateurs

Design pattern : Adapter (1/2)



Vues à adaptateurs

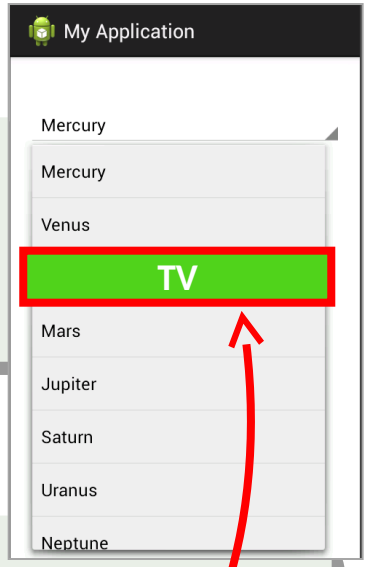
Design pattern : Adapter (2/2)



Spinner et ArrayAdapter

/res/layout/activity_main.xml

```
<Spinner
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/sp_planets"/>
```



/java/MainActivity.java

```
Spinner planetsSP = findViewById(R.id.sp_planets);
...
String[] items = {"Mercury", "Venus", "Earth", ... };
ArrayAdapter<String> adapter =
            new ArrayAdapter<>(this,
                               android.R.layout.simple_list_item_1,
                               items);
planetsSP.setAdapter(adapter);
...
```

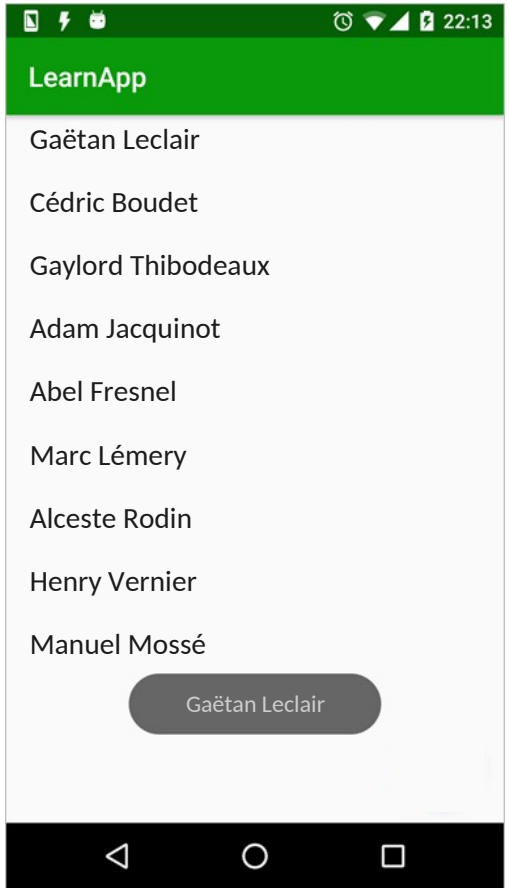
Exercice : Création d'une vue de liste

Modification de l'activité `HabitatActivity`

- Une `ListView` pour afficher la liste des résidents d'un bâtiment

- En cliquant sur chaque élément de la liste

→ Afficher le nom du résident dans un `Toast`



Listeners d'un AdapterView

Interagir avec un **AdapterView** à travers des **Listeners**

- **OnItemClickListener**
  - onItemClick(AdapterView<?> parent, View view, int position, long id)
- **OnItemLongClickListener**
  - onLongClick(AdapterView<?> parent, View view, int position, long id)
- **OnItemSelectedListener**
  - onItemSelected(AdapterView<?> parent, View view, int position, long id)
  - onNothingSelected(AdapterView<?> parent)

Méthodes de mise à jour des adaptateurs

android.widget.Adapter

```
void      add(T item)                 // ajouter un élément en fin de l'AdapterView
void      insert(T item, int index)   // insérer un élément à une position donnée
void      addAll(T... items)          // insérer plusieurs éléments
T         getItem(int index)          // récupérer l'élément d'une position donnée
int       getPosition(Object o)       // récupérer la position d'un élément donné
void      remove(T item)              // supprimer un élément donnée
void      clear()                     // supprimer tous les éléments
void      notifyDataSetChanged()      // notifie l'AdapterView des nouveaux
                                         changements pour se rafraîchir
```

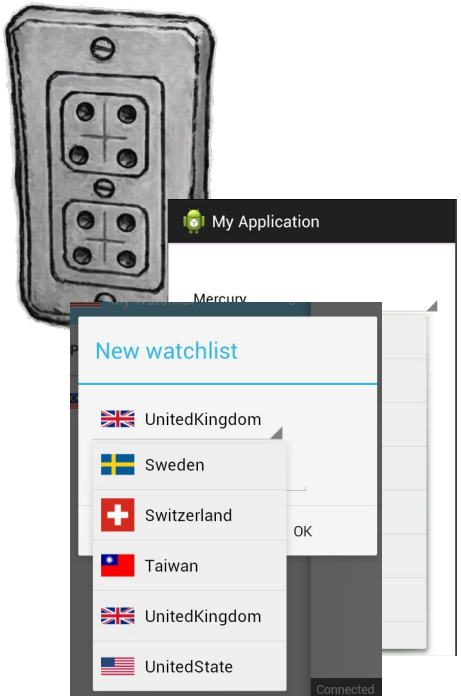
Collection

Collection

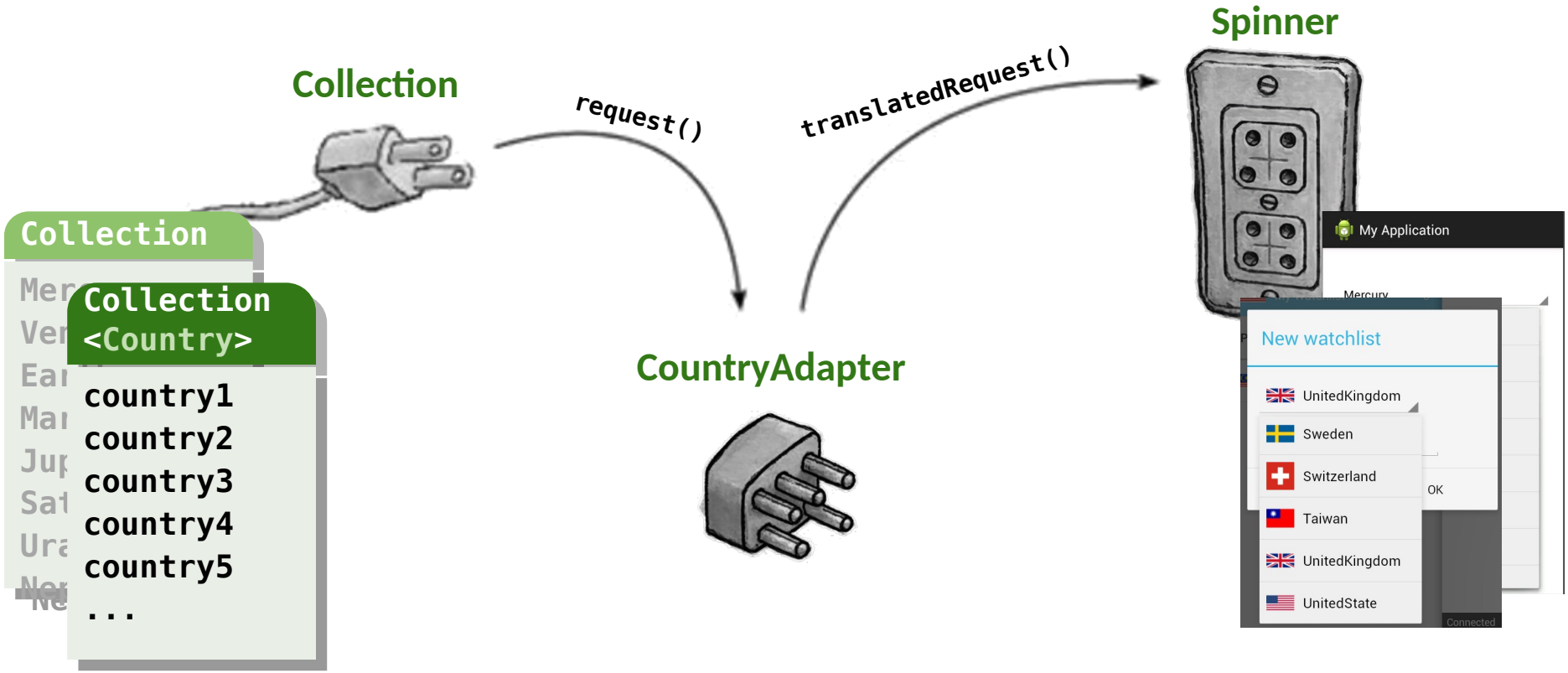
Collection
<Country>

country1
country2
country3
country4
country5
...

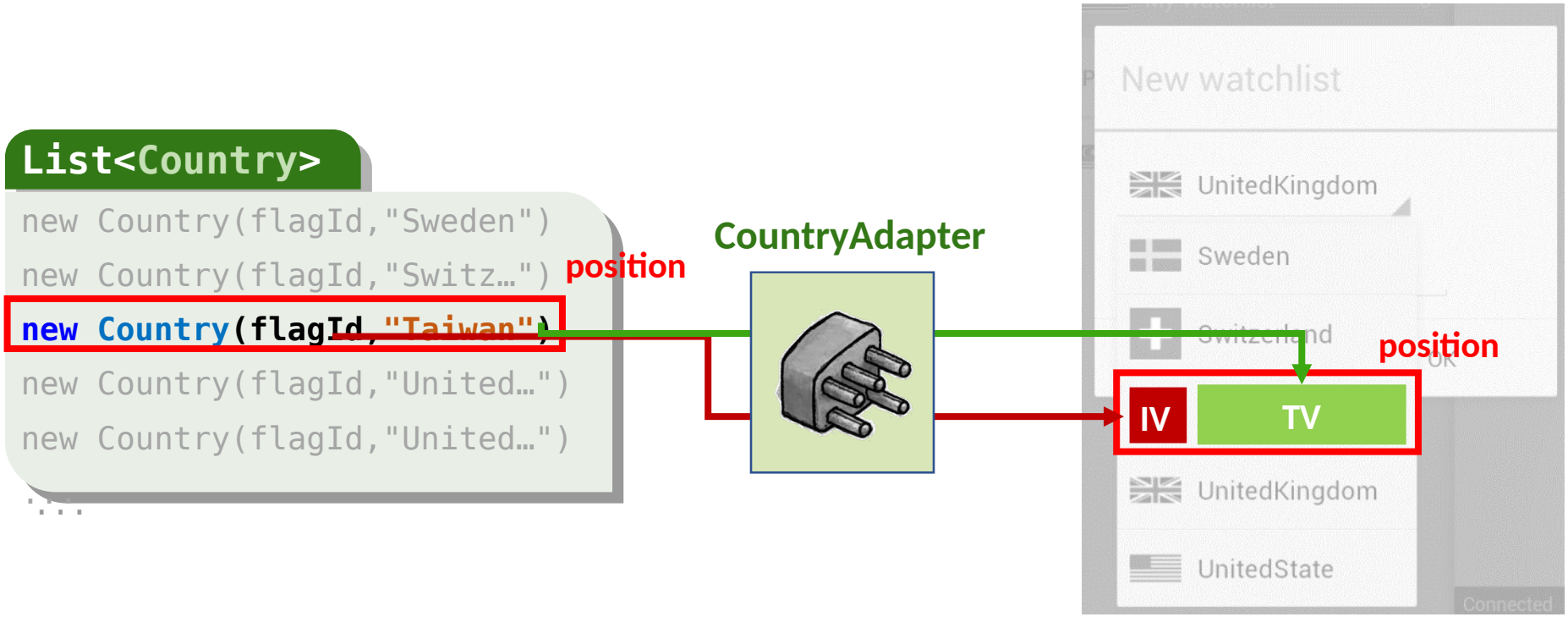
Spinner



Vues à adaptateur personnalisé (1/2)



Vues à adaptateur personnalisé (2/2)



Liste à adaptateur

Standard **vs.** personnalisé