# Cours R2.02
# Introduction à l'Interaction Humain-Machine

## Cours 4 : Widgets et événements (2/2)

Géry Casiez https://gery.casiez.net
Département informatique - IUT de Lille - Université de Lille

# Plan du cours en 9 semaines



1. Introduction à l'interaction, placement
2. Programmation événementielle
3. Widgets et événements (1/2)
4. **Widgets et événements (2/2)**
5. Conception et prototypage (1/2)
6. Conception et prototypage (2/2)
7. Heuristiques et recommandations
8. Modèles et théories
9. Méthodes d'évaluation des IHM

# Objectifs



Introduction à MVC

Utilisation des ListView

# Une brève introduction à MVC



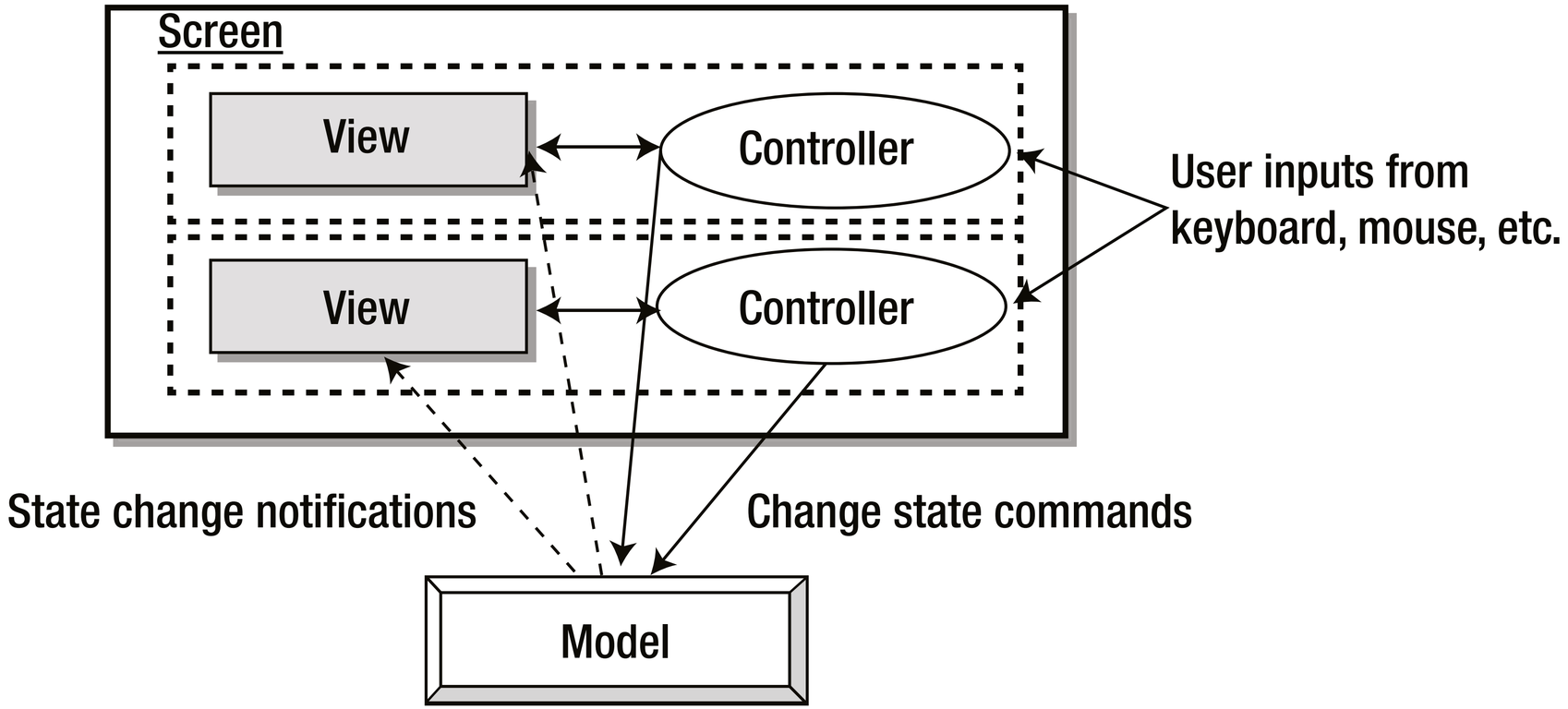
MVC : Modèle Vue Contrôleur

Idée : séparer les données d'une application de leur présentation

Le modèle informe la ou les vues de se mettre à jour

# Une brève introduction à MVC

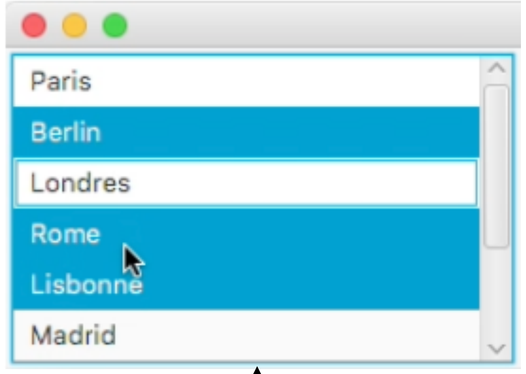




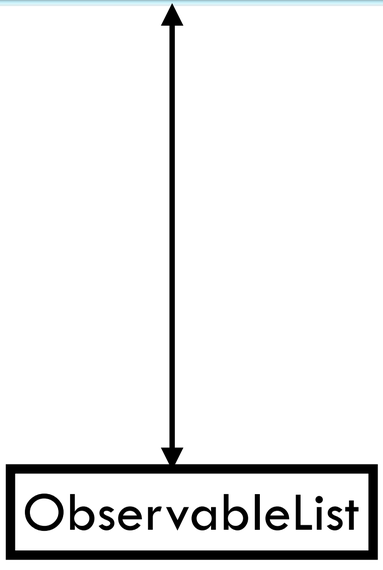
# Exemple de ListView

ListView






Vue




Modèle

`ObservableList<T>` `getItems()`

Returns an `ObservableList` that contains the items currently being shown to the user.

# ObservableList

ObservableList



javafx.collections

## Interface ObservableList<E>

**Type Parameters:**

`E - the list element type`

**All Superinterfaces:**

`Collection<E>, Iterable<E>, List<E>, Observable`

**All Known Subinterfaces:**

`ObservableListValue<E>, WritableListValue<E>`

**All Known Implementing Classes:**

`FilteredList, ListBinding, ListExpression, ListProperty, ListPropertyBase, ModifiableObservableListBase, ObservableLi`
`SimpleListProperty, SortedList, TransformationList`

---

```
public interface ObservableList<E>
extends List<E>, Observable
```

A list that allows listeners to track changes when they occur.

**Since:**

`JavaFX 2.0`

**See Also:**

`ListChangeListener, ListChangeListener.Change`

### *Method Summary*

All Methods | Instance Methods | Abstract Methods | Default Methods

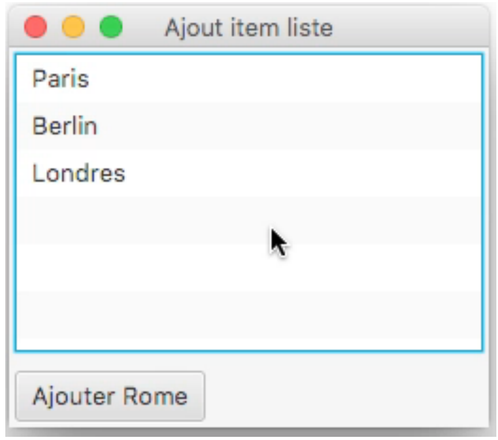
| Modifier and Type | Method and Description |
|---|---|
| boolean | **addAll**(E... elements)<br>A convenient method for var-arg adding of elements. |
| void | **addListener**(ListChangeListener<? super E> listener)<br>Add a listener to this observable list. |

# ListView

cours4/ListeViewAdditem.java



```
public class ListeViewAddItem extends Application {
    public void start(Stage stage) {
        ListView<String> list = new ListView<String>();
        list.getItems().addAll("Paris", "Berlin", "Londres");
        Button button = new Button("Ajouter Rome");
        button.setOnAction(e -> list.getItems().add("Rome"));

        VBox root = new VBox();
        root.setSpacing(10.0);
        root.setPadding(new Insets(3, 3, 3, 3));
        root.getChildren().addAll(list, button);

        Scene scene = new Scene(root);
        stage.setTitle("Ajout item liste");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        Application.launch(args);
    }
}
```



# ObservableList




javafx.collections

## Interface ObservableList<E>

**Type Parameters:**
`E - the list element type`

**All Superinterfaces:**
`Collection<E>, Iterable<E>, List<E>, Observable`

**All Known Subinterfaces:**
`ObservableListValue<E>, WritableListValue<E>`

**All Known Implementing Classes:**
`FilteredList, ListBinding, ListExpression, ListProperty, ListPropertyBase, ModifiableObservableListBase, ObservableLi`
`SimpleListProperty, SortedList, TransformationList`

---

`public interface ObservableList<E>`
`extends List<E>, Observable`

A list that allows listeners to track changes when they occur.

**Since:**
`JavaFX 2.0`

**See Also:**
`ListChangeListener, ListChangeListener.Change`

***Method Summary***

| All Methods | Instance Methods | Abstract Methods | Default Methods |
|---|---|---|---|

| Modifier and Type | Method and Description |
|---|---|
| boolean | `addAll(E... elements)`<br>A convenient method for var-arg adding of elements. |
| void | `addListener(ListChangeListener<? super E> listener)`<br>Add a listener to this observable list. |

Permet de s'abonner aux changements d'états de la liste

# Interface ListChangeListener




## Interface ListChangeListener<E>

**Type Parameters:**
E - the list element type

**All Known Implementing Classes:**
WeakListChangeListener

**Functional Interface:**
This is a functional interface and can therefore be used as the assignment target for a lambda expression or me

---

```
@FunctionalInterface
public interface ListChangeListener<E>
```

Interface that receives notifications of changes to an ObservableList.

**Since:**
JavaFX 2.0

**See Also:**
ListChangeListener.Change

### *Nested Class Summary*

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static class | **ListChangeListener.Change<E>**<br>Represents a report of a changes done to an Observablelist. |

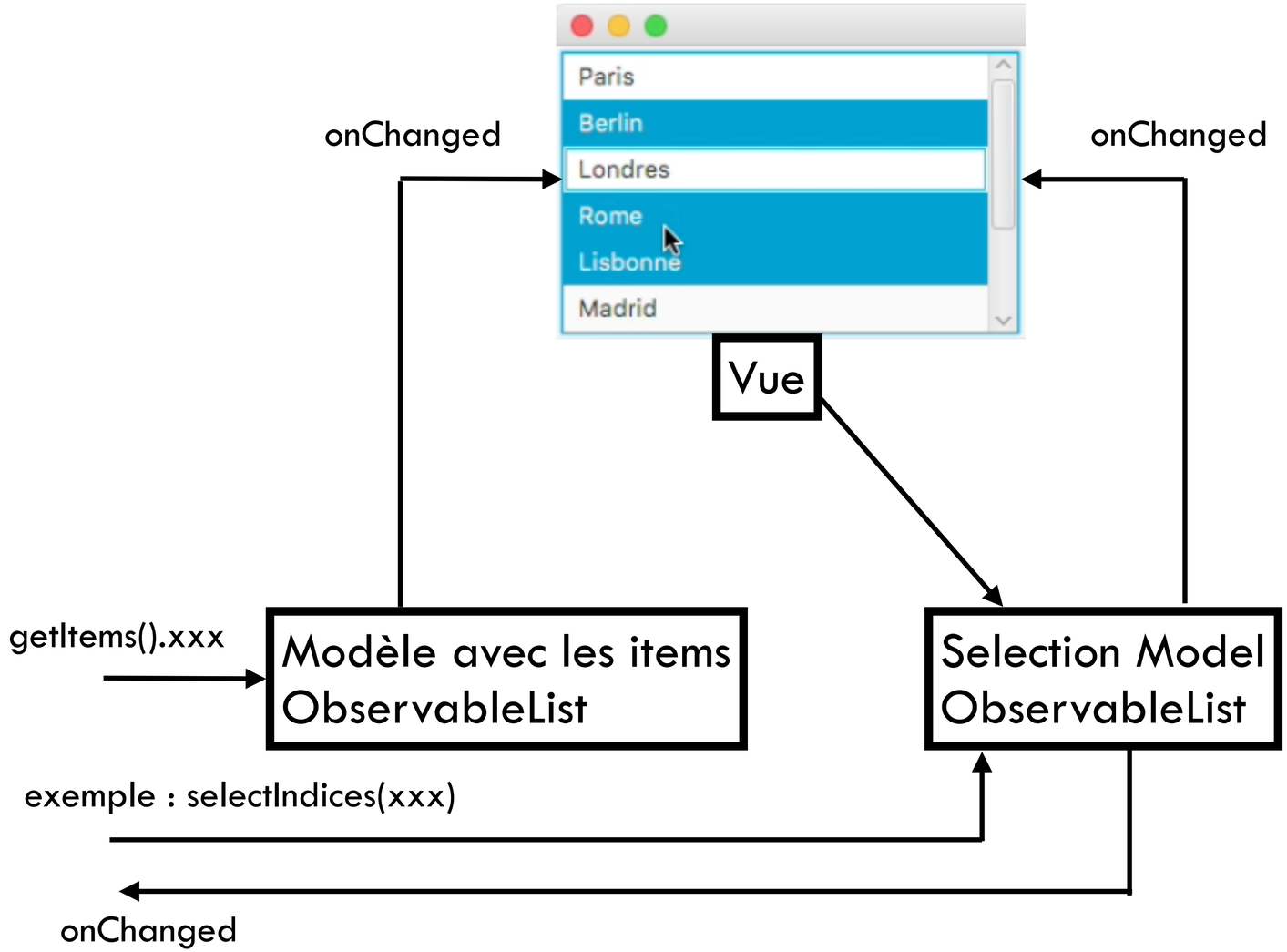
### *Method Summary*

**All Methods** | **Instance Methods** | **Abstract Methods**

| Modifier and Type | Method and Description |
|---|---|
| void | **onChanged(ListChangeListener.Change<? extends E> c)**<br>Called after a change has been made to an ObservableList. |

# Exemple de ListView





# Abonnement / notification

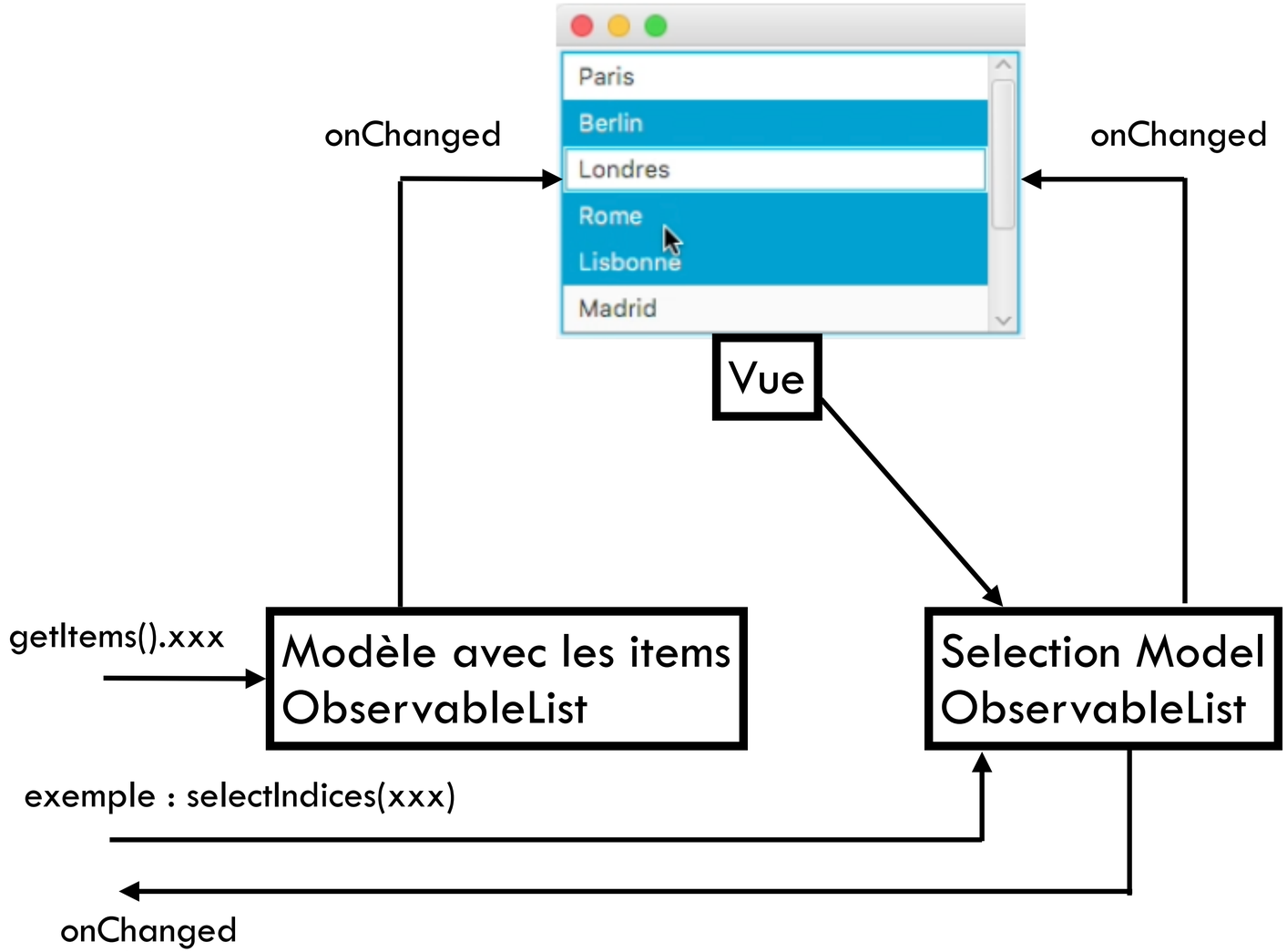


Abonnement aux changements d'état du modèle en utilisant **addListener**

Notification du changement d'état par l'appel de la méthode **onChanged**

# Exemple de ListView



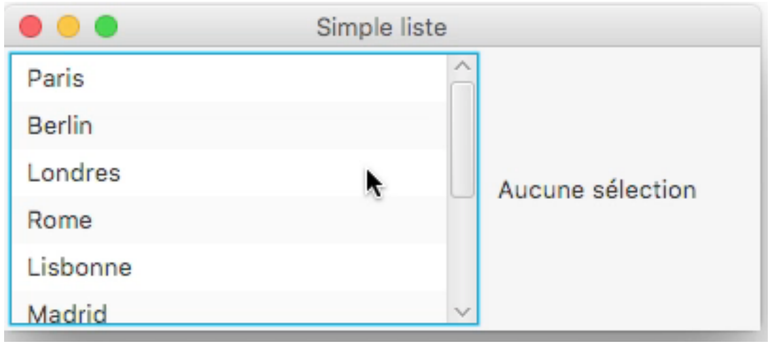


# ListView : gestion de la sélection simple

cours4/ListeSimple.java



```
public class ListeSimple extends Application {
   Label label;

   class MonListChangeListener implements ListChangeListener<String> {
      public void onChanged(Change<? extends String> report) {
         label.setText("Sélection de " + report.getList());
      }
   }

   public void start(Stage stage) {
      label = new Label("Aucune sélection");
      ListView<String> list = new ListView<String>();
      list.getItems().addAll("Paris", "Berlin", "Londres", "Rome", "Lisbonne", "Madrid", "New York", "Tokyo", "Pékin");
      list.getSelectionModel().getSelectedItems().addListener(new MonListChangeListener());

      HBox root = new HBox();
      root.setAlignment(Pos.CENTER_LEFT);
      root.setSpacing(10.0);
      root.setPadding(new Insets(3, 3, 3, 3));
      root.getChildren().addAll(list,label);

      Scene scene = new Scene(root, 400, 150);
      stage.setTitle("Simple liste");
      stage.setScene(scene);
      stage.show();
   }

   public static void main(String[] args) {
      Application.launch(args);
   }
}
```



# ListView : gestion de la sélection multiple

cours4/ListeSelectionMultiple.java



```
public class ListeSelectionMultiple extends Application {
   Label label;

   class MonListChangeListener implements ListChangeListener<String> {
      public void onChanged(Change<? extends String> report) {
         label.setText("Sélection de " + report.getList());
      }
   }

   public void start(Stage stage) {
      label = new Label("Aucune sélection");
      ListView<String> list = new ListView<String>();
      list.getItems().addAll("Paris", "Berlin", "Londres", "Rome", "Lisbonne", "Madrid", "New York", "Tokyo", "Pékin");
      list.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
      list.getSelectionModel().getSelectedItems().addListener(new MonListChangeListener());

      HBox root = new HBox();
      root.setAlignment(Pos.CENTER_LEFT);
      root.setSpacing(10.0);
      root.setPadding(new Insets(3, 3, 3, 3));
      root.getChildren().addAll(list,label);

      Scene scene = new Scene(root, 400, 150);
      stage.setTitle("Liste multi-sélection");
      stage.setScene(scene);
      stage.show();
   }

   public static void main(String[] args) {
      Application.launch(args);
   }
}
```

# Résumé



MVC = Modèle, Vue, Contrôleur

MVC est utilisé par de nombreux widgets de JavaFX

- ListView possède 2 modèles (ObservableList) et une vue

Le modèle informe de son changement d'état par un mécanisme de listener (sytème d'abonnement)

- Utilisation d'un ListChangeListener pour ListView