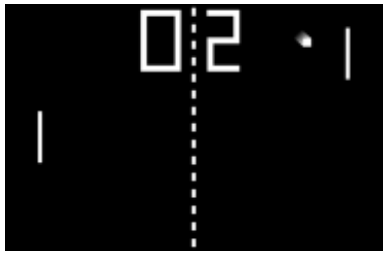
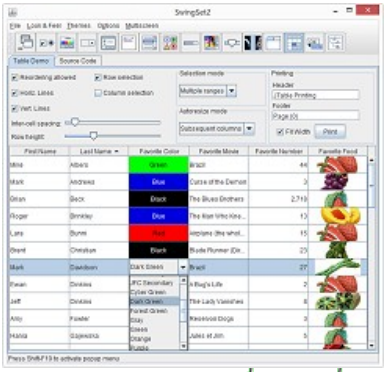
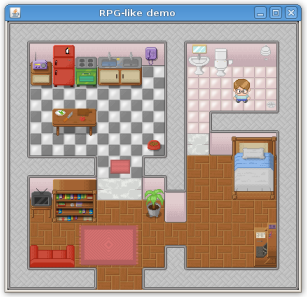
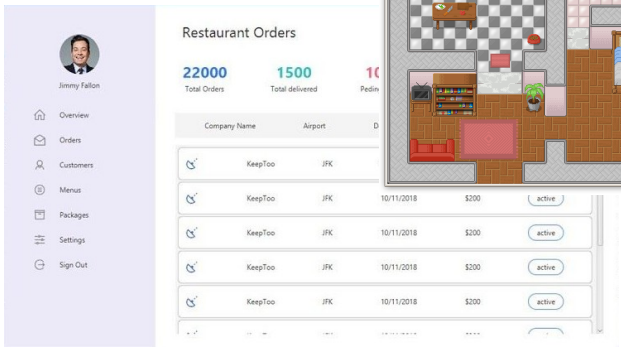
Interfaces Homme Machine

JAVA FX









Fabrice PELLEAU

Evenements… un peu de code

- Plusieurs handlers sur 1 event :
  - addEventHandler()
  - removeEventHandler()

```
Button b = new Button("Button");
b.addEventHandler(ActionEvent.ACTION, e -> System.out.println("Handler 1"));
b.addEventHandler(ActionEvent.ACTION, e -> System.out.println("Handler 2"));
```

- Un seul handler quoi qu'il arrive
  - setOn...()

```
Button b = new Button("Button");
b.setOnAction( e -> System.out.println("Handler qui va être remplacé"));
b.setOnAction( e -> System.out.println("C'est ce Handler qui prend la place"));
```

Beans Property

- Avec JavaFX il est courant d'utiliser les `Properties` pour tous les champs de votre classe. Une propriété nous permet, par exemple, d'être automatiquement averti lorsque la variable associée à un attribut a été modifiée.
- Ceci aide à maintenir la vue synchronisée avec les données.
  - Doc Oracle :
    - Using JavaFX Properties and Binding !
    - docs.oracle.com/javafx/2/binding/jfxpub-binding.htm
  - Tuto Devoloppez.com
    - JavaFX Binding de haut niveau

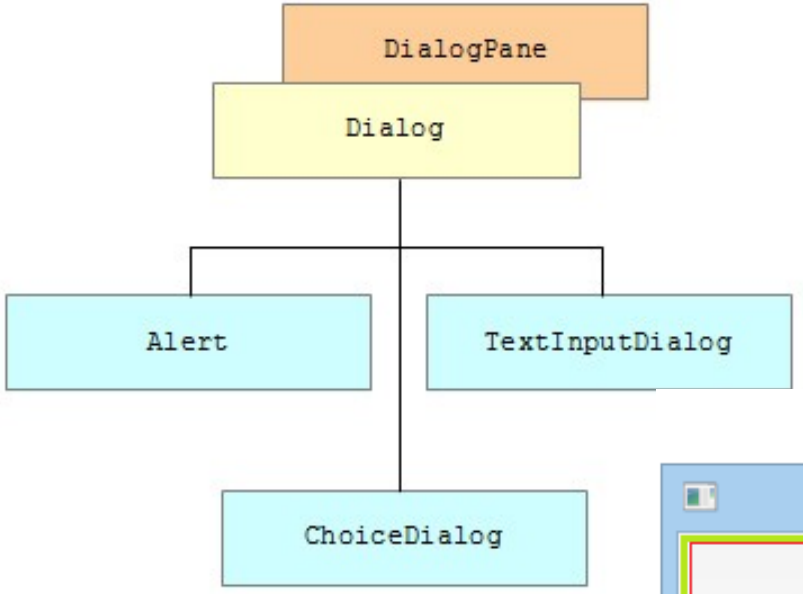
Exemple property + bind

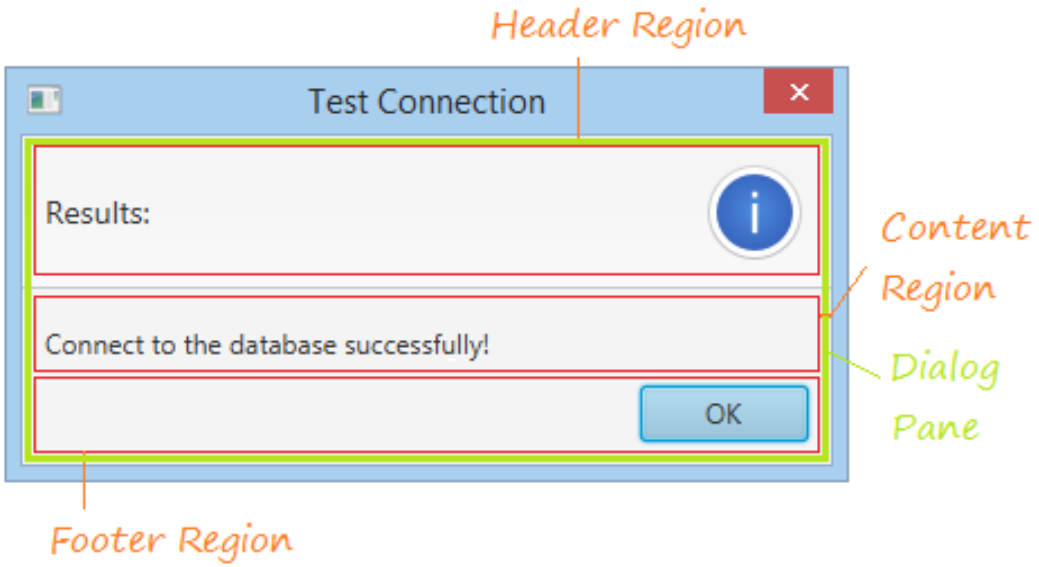
```
IntegerProperty aProperty = new SimpleIntegerProperty(-10);  // A
IntegerProperty bProperty = new SimpleIntegerProperty(10);   // B


bProperty.addListener(
  (observable, oldValue, newValue) →
         System.out.println("B : "+oldValue+" devient "+newValue));

bProperty.bind(aProperty);
        // B : 10 devient -10

aProperty.set(22);
        // B : -10 devient 22
```

Observable/Observer

- Les propriétés sont toutes Observables
  - Elles implémentent `ObservableValue`
  - `IntegerProperty`, `StringProperty`, etc.
- Il existe également les collections Observables
  - enrichissant le framework Collection
  - `ObservableList`, etc.
- En associant des données Observables aux composant JavaFX on automatise la prise en compte des modifications
  - Synchronisation des affichages

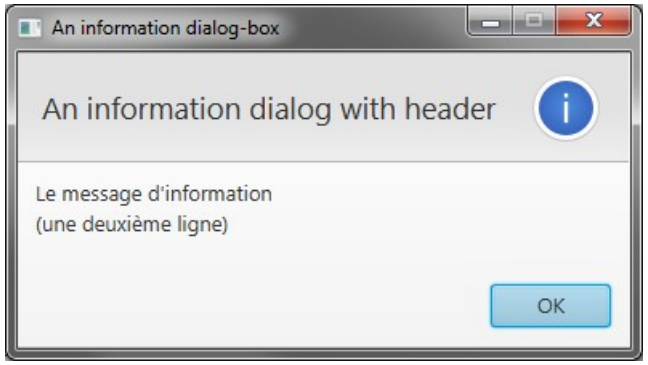
Boites de dialogue



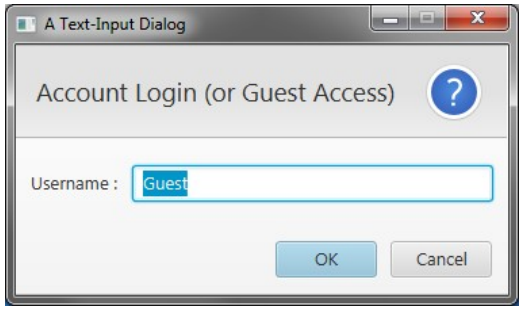


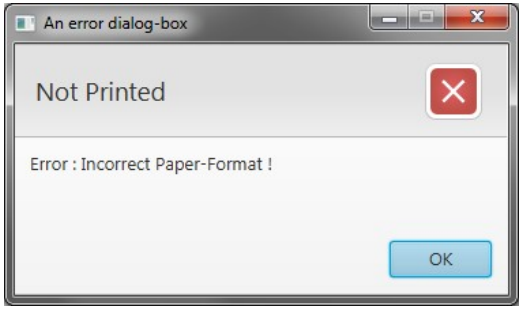
Exemple de boite de dialogue

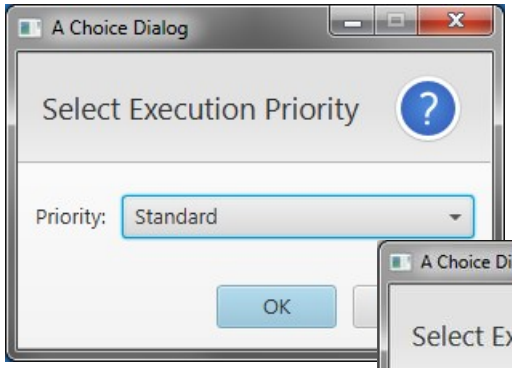
```
Alert dialog = new Alert(AlertType.INFORMATION);

dialog.setTitle("An information dialog-box");
dialog.setHeaderText("An information dialog with header");
dialog.setContentText("Le message d'information\n" + "(une deuxième ligne)");
dialog.showAndWait();
```

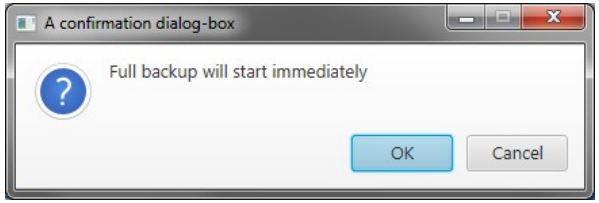


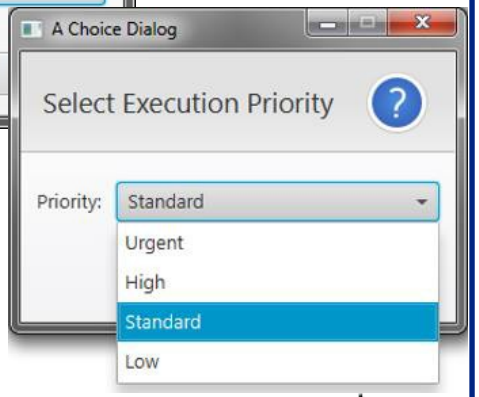
Variantes multiples...

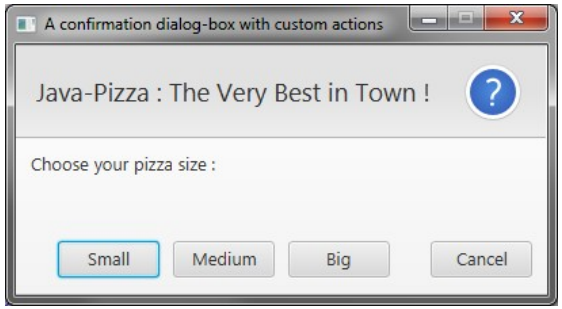




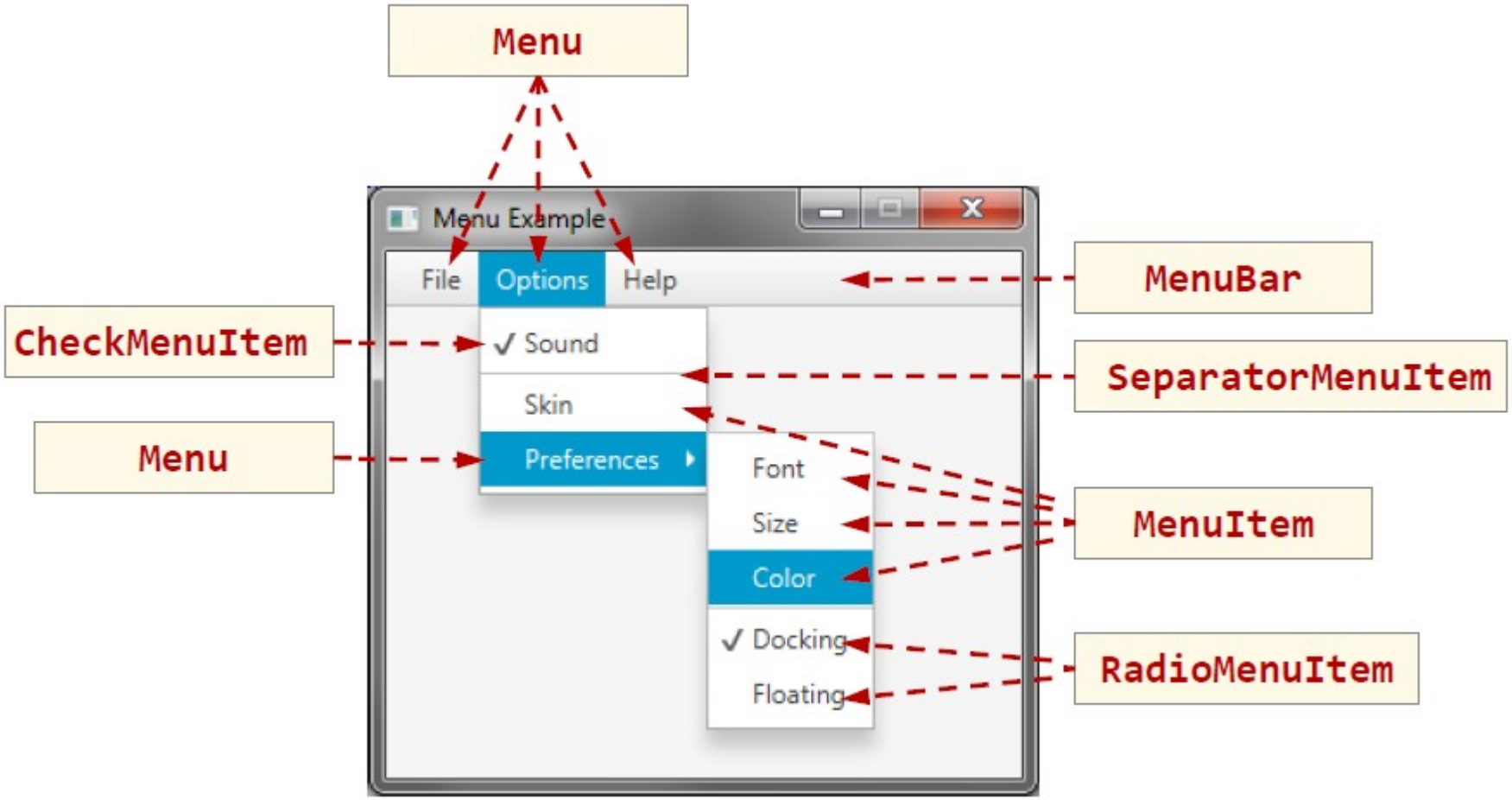








Gestion des menus

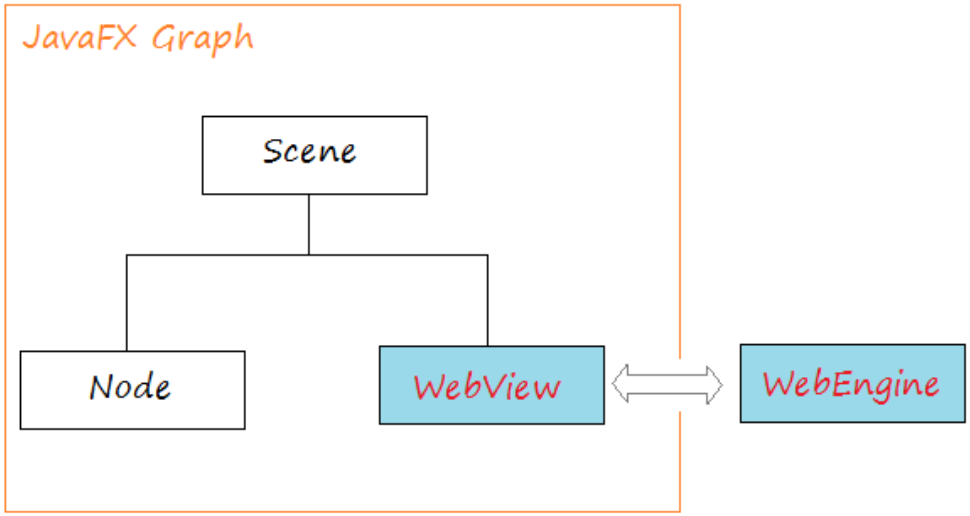


HTML et Javascript

- JavaFX `WebView` est un navigateur intégré basé sur WebKit.
- Il prend en charge HTML5, CSS, SVG, JavaScript et DOM.
- La combinaison de `WebView` et `WebEngine` permet :
  - D'interpréter le code HTML,CSS,JS, etc.
  - D'exécuter du JavaScript depuis Java
  - D'exécuter du Java depuis JavaScript

WebView et WebEngine

- WebEngine interprète le code HTML et interagit avec l'application
- WebView est un Node JavaFX qui affiche le résultat



WebView et WebEngine

```
// Créer un WebView et charger une page
WebView webView = new WebView();
webView.getEngine().load("http://eclipse.com"));
```

```
// Executer du code JavaScript en Java
webView.getEngine().executeScript("maFonctionJS()");
```

```
// Executer du code Java en JavaScript
public class Bridge {                // la classe doit être publique
    public void printInJava() {
        System.out.println("Hello Java");
    }
}
JSObject jsWindow = (JSObject) webEngine.executeScript("window");
jsWindow.setMember("myJavaMember", new Bridge());
```

```
<!-- Dans la page HTML -->
<button onclick='myJavaMember.showTime();'>Call To JavaFX</button>
```

JavaFX Application Thread

- La gestion des événements JavaFX est réalisée dans l'**Application Thread**
  - Équivalent de l'EDT Swing
- Pour garantir un bon fonctionnement **« thread safe »** de l'application 100 % des modifications de l'interface doivent être réalisées dans l'Application Thread.
- Contrairement à Swing, la majorité des modifications d'interface en dehors de l'Application Thread déclenchera une exception `IllegalStateException`
  - *Il est donc plus simple de sécuriser son application*

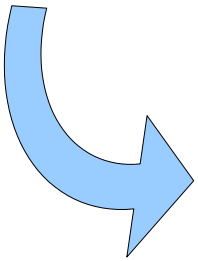
javafx.application.**Platform**

- `Platform` propose 2 méthodes pour interagir avec **JavaFX Application Thread**
- `runLater( Runnable )`
  - Re-injecter des instructions dans l'Application Thread
- `isFxApplicationThread()`
  - Savoir si on est déjà dans le Thread JavaFX
    - *Evite de lancer runLater pour rien*
- *Classe équivalente de SwingUtilities*

Thread vs Application Thread

- Action JavaFX exécutée en dehors de l'application Thread
  - ~~Thread Safe~~

```
itemSauvegarde.setOnAction(e->{
  new Thread (()-> {
    grosTraitementFichier();
    label.setText("Traitement terminé");
  }).start();
});
```



```
itemSauvegarde.setOnAction(e->{
  new Thread (()-> {
    grosTraitementFichier();
    Platform.runLater(()-> labelStatus.setText("Traitement terminé"));
  }).start();
});
```