

JavaFX : une nouvelle manière de construire des applications Swing



Annoncé en fanfare lors du dernier JavaOne (mai 2007), JavaFX a suscité, depuis, moult commentaires, une controverse sur sa pertinence et quelques espoirs. JavaFX a été présenté comme l'alternative Java aux solutions Flex d'Adobe ou Silverlight de Microsoft. En moins abouti. D'où la controverse.

Cette vision de JavaFX n'est que partiellement juste : JavaFX n'est pas à proprement parler une solution RIA, c'est-à-dire un outil pour projeter des appliquettes sur un navigateur client, à l'instar de Flex/Flash. Fondamentalement, JavaFX est un nouvel outil pour construire des applications Swing. Quelles qu'elles soient. JavaFX est une gamme d'outils qui est destinée à s'enrichir dans les mois et années qui viennent. Aujourd'hui, il y a deux outils dans la gamme :

- Un langage de scripting (JavaFX Script) avec son environnement de développement associé (plug-in Eclipse ou Netbeans)
 - Un système logiciel pour matériel mobile (comme un téléphone portable) capable d'exécuter des applications Java et donc JavaFX Script.
- Cet article ne traitera que de JavaFX Script, qui est l'outil censé être la solution RIA de la plate-forme Java et à ce titre, concurrente d'Adobe Flex/Flash et Silverlight. JavaFX Script est un interpréteur. A partir d'un fichier source JavaFX (extension .fx), cet interpréteur construit une interface homme machine (IHM) basée sur Swing et lance son exécution. L'originalité de JavaFX est que le style de programmation proposé est déclaratif et non impératif : l'IHM à construire est décrite (à l'instar de ce qui est fait classiquement en HTML par exemple) alors que traditionnellement, une application Swing est spécifiée comme une suite d'instructions à exécuter. Cette démarche n'est pas nouvelle. Elle est au cœur de Flex et de Silverlight. Avec Flex ou Silverlight, l'interface à construire est décrite en utilisant un format basé sur XML. Ce sera MXML dans le cas de Flex et XAML dans le cas de Silverlight. JavaFX utilise un format dérivé de JSON, mais le principe reste identique.

La programmation déclarative

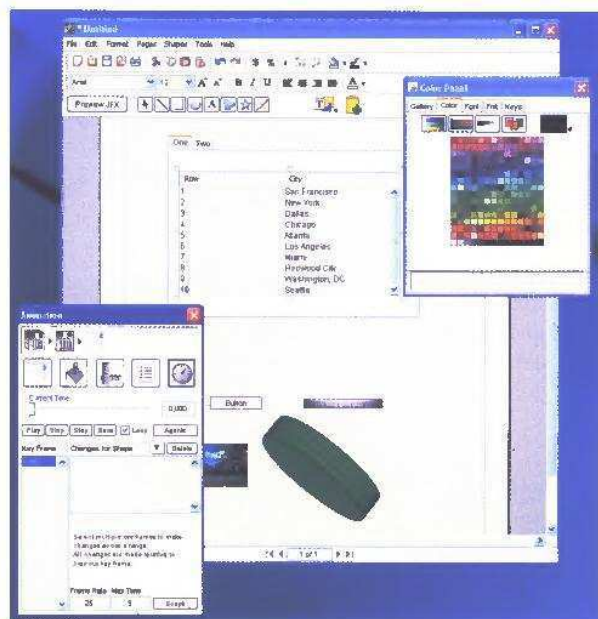
Illustrons ce nouveau paradigme de programmation par un exemple : Commençons, en utilisant JavaFX, par créer une frame contenant un bouton :

```
var myBean = new ActionBean();
Frame {
  content: Button {
    text: "Click here"
    action: operation() {
      myBean.dolt();
    }
  }
  visible: true
}
```

Créons le même frame en utilisant directement Swing :

```
public class MiniGui {

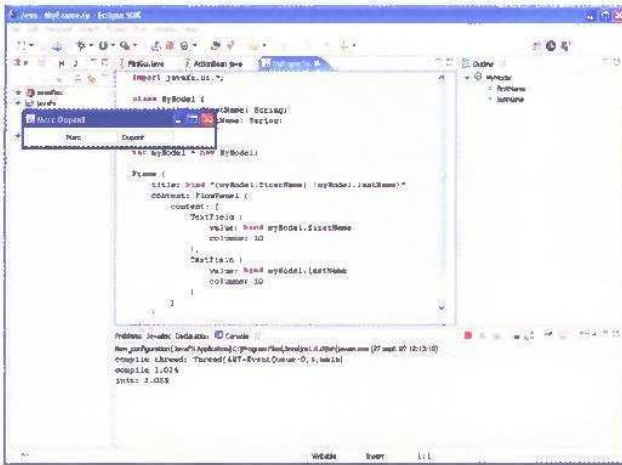
  public static void main(String argv[]) {
```



```
    JButton button = new JButton("Click here");
    button.addActionListener(new ActionListener() {
        ActionBean actionBean = new ActionBean();
        public void actionPerformed(ActionEvent evt) {
            actionBean.dolt();
        }
    });
```

```
    JFrame frame = new JFrame();
    frame.getRootPane().setLayout(new BorderLayout());
    frame.getRootPane().add(button);
    frame.pack();
    frame.setVisible(true);
}
```

Au-delà de la simplification de l'écriture en JavaFX, ce qu'il convient de noter est la différence d'approche. Avec JavaFX, on décrit ce que l'on désire obtenir et non comment on l'obtient. La programmation impérative ne perd pas tous ses droits. Mais elle reste confinée à l'implémentation de réflexes, comme " l'opération " qui sera exécutée lorsque le bouton est cliqué.



La magie du binding

La programmation déclarative est une idée intéressante, mais qui n'apporterait qu'un avantage mineur au développeur si elle ne comportait pas un mécanisme qui établisse une correspondance entre des données et les widgets qui les présentent à l'utilisateur. Ce mécanisme est le "binding". Un binding est un lien entre un attribut d'un artefact graphique (le contenu d'un champ de saisie, le titre d'une fenêtre mais aussi la couleur de fond d'un panel, etc.) et une propriété d'un objet (un JavaBean par exemple). Ce lien implique une synchronisation automatique entre l'attribut et la propriété. Si nous avons, par exemple, un champ de saisie lié à la propriété name du bean myBean :

```
TextField {
    value: bind myBean.name
},
```

Lorsqu'un utilisateur saisit dans le champ textuel un nouveau nom, ce nom sera automatiquement reporté dans la propriété " name " de " myBean ". Inversement, si pour une autre raison, la propriété " name " est modifiée, sa nouvelle valeur est automatiquement affichée par le champ de saisie. Il est possible de lier une propriété à plusieurs attributs et obtenir ainsi des chaînes de synchronisations :

```
import javafx.ui.*;

class MyModel {
    attribute firstName: String;
    attribute lastName: String;
}

var myModel = new MyModel;

Frame {
    title: bind "{myModel.firstName} {myModel.lastName}"
    content: FlowPanel {
        content: [
            TextField {
                value: bind myModel.firstName
                columns: 10
```

```
},
    TextField {
        value: bind myModel.lastName
        columns: 10
    }
]
}
visible: true
}
```

Dans l'exemple qui précède, nous avons défini une fenêtre contenant deux champs de saisie. Ces deux champs sont liés aux deux propriétés du même bean " MyModel ". Ces deux propriétés sont toutes les deux liées avec le titre de la fenêtre (notez au passage l'utilisation des guillemets et des accolades pour intégrer la valeur de ces deux propriétés). Dès que la saisie d'un champ est validée (en passant au suivant par exemple) le titre de la fenêtre est automatiquement mis à jour. Les possibilités offertes par ce mécanisme sont nouvelles, originales et demandent un savoir-faire encore peu répandu. Je ne peux résister à présenter ici un exemple d'implémentation du " glisser-déposer " (drag and drop) réalisé à l'aide de quelques " binding " adroitement placés. Cet exemple est extrait du " JavaFX Script 2D Graphics Tutorial " :

```
Group {
    content:
    [Line {
        x1: bind x1
        y1: bind y1
        x2: bind x2
        y2: bind y2
        stroke: bind stroke
        strokeWidth: 2
    },
    Circle {
        cx: bind x1
        cy: bind y1
        radius: 5
        onMouseDragged: operation(e:CanvasMouseEvent) {
            x1 += e.localDragTranslation.x;
            y1 += e.localDragTranslation.y;
        }
    },
    Circle {
        cx: bind x2
        cy: bind y2
        radius: 5
        onMouseDragged: operation(e:CanvasMouseEvent) {
            x2 += e.localDragTranslation.x;
            y2 += e.localDragTranslation.y;
        }
    }
    ]
}
```

Un groupe rassemble un trait et deux cercles dont les centres sont situés à chaque extrémité du trait. Les extrémités du trait et les centres du cercle sont liés à 4 valeurs (x1 y1 x2 y2) définies par ailleurs dans le



fichier ".fx". A chaque cercle, un réflexe est défini lorsqu'un événement souris de type "déplacement" est reçu. Cet événement modifie les valeurs liées au centre du cercle, pour les faire correspondre à l'emplacement actuel de la souris. Du fait des binding, cette mise à jour entraîne automatiquement la mise à jour des coordonnées du centre du cercle ainsi que celles d'une extrémité du trait. L'utilisateur a l'impression de déplacer le trait à l'aide d'un cercle qui sert de poignée. Le tour est joué.

Une richesse à concrétiser

JavaFX Script propose d'ores et déjà un langage complet pour décrire des constructions (graphiques ou données) d'une part et pour implémenter des réflexes d'autres part. Il est clairement orienté objet avec la possibilité de définir des classes. L'héritage est proposé, y compris multiple, ce que Java ne permet pas ! La quasi-totalité des instructions de contrôle sont présentes (if, while, for, foreach, try...catch). JavaFX implémente une logique de collections très souples, servies par de nombreuses constructions syntaxiques. En voici une illustration, tirée de "Learning JavaFX Script" :

```
var x = [1,2,3];
insert 10 into x;      // yields [1,2,3,10]
insert 12 before x[1]; // yields [1,12,2,3,10]
delete x[. == 12];    // yields [1,2,3,10]
delete x[. >= 3];     // yields [1,2]
insert 5 after x[. == 1]; // yields [1,5,2];
insert 13 as first into x; // yields [13, 1, 5, 2];
delete x;              // clears the array and yields []
```

Sont aussi proposés, un mécanisme puissant de formatage, la possibilité d'armer des réflexes sur mise à jour d'un attribut ou d'une collection, ou encore d'exécuter un traitement en tâche de fond.

JavaFX autorise aussi la création et la manipulation d'objets Java. Il existe bien entendu quelques restrictions dont la plus sérieuse est que ces objets ne peuvent être atteints par le mécanisme de "binding". C'est une fonctionnalité précieuse qui ouvre à JavaFX d'immenses possibilités. En premier lieu, la communication avec un serveur, Java EE, RMI ou Web Service est immédiatement disponible. On peut, bien entendu, mélanger des instructions JavaFX et des appels à des méthodes d'objets Java au sein d'un même traitement JavaFX (i.e. operation). L'intégration des deux mondes est très naturelle.

La bibliothèque des objets proposés par JavaFX est d'ores et déjà conséquente : outre les Frames, TextField et Buttons déjà présentés, on y trouve les panels, les tables, des arbres, des onglets, les labels, les cases à cocher et boutons radio, les curseurs, etc.

Mais ce n'est pas tout : une importante partie de Java 2D est disponible sous la forme d'objets spécifiques de type ligne, cercle, rectangle, images. L'exemple du "glisser-déposer" vu plus haut en est une illustration. Une instruction spécifique appelée "dur" permet de créer facilement des animations comme en témoigne la snippet suivante :

```
attribute width: Number;
...
Rect {
    width: bind width
    height: bind height
```

```
stroke: darkblue
},
...
Button {
    text: "Width"
    action: operation() {
        width = [0..200] dur 1000;
    }
},
```

Quand on appuie sur le bouton, un réflexe est déclenché qui modifie la largeur du rectangle en la faisant passer de la valeur 0 à la valeur 200 en 1 seconde (1000 millisecondes).

Dans le monde rude du RIA, la place de l'outillage est prépondérante : qu'est ce que JavaFX peut opposer aux redoutables FlexBuilder et VisualStudio de ses concurrents ? Sur les IDE Java les plus connus, pas grand-chose encore. Il existe bien des plug-in pour Eclipse et NetBeans, mais ni l'un ni l'autre n'offrent encore de vue WYSIWYG de la page JFX. Cela devrait être rapidement corrigé. Aujourd'hui, c'est ailleurs qu'il faut chercher les premiers vrais compositeurs d'application JavaFX. Prenez le temps, par exemple, de jouer avec un outil plein de promesses : JFX-Builder. Il est disponible via JavaWeb Start sur le site : <http://www.reportmill.com/jfx/>

Perspectives

JavaFX est un produit finalement mal connu : il est présenté un peu partout sur le Web comme une solution RIA, mais c'est en fait une nouvelle solution pour construire des applications Swing, qu'elles soient RIA ou non. D'ailleurs, qu'est-ce qu'une application Swing RIA ? C'est une application graphique qui s'exécute sur une applet. Oui : les applets qui accompagnent Java depuis le tout début, celles-là même qui ont été tant décriées et qui reviennent aujourd'hui à l'honneur grâce à l'expérience – beaucoup plus réussie – de Flash.

La solution Java pour faire du RIA rassemble donc plusieurs technologies dont JavaFX n'est qu'une composante, d'ailleurs facultative : on peut continuer à programmer des applets en utilisant directement Java/Swing.

L'intérêt de JavaFX est donc à la fois ailleurs et plus large. Je pense qu'aujourd'hui toute équipe qui développe une application graphique basée sur Swing devrait s'intéresser à JavaFX : la programmation déclarative, le mécanisme de "binding" et le langage de scripting proposé, très bien intégré au reste de la plate-forme Java, sont de nature à améliorer considérablement la productivité.

De façon logique, l'avenir de JavaFX n'est pas lié au devenir de Java en tant que plate-forme RIA. JavaFX est une nouvelle brique qui vient accompagner les bibliothèques et outils Java pour fabriquer des applications riches et partagera le destin de cette plate-forme sur ce créneau.



■ Henri DARMET

Directeur Technique

Objet Direct / Homsys Group

Objet Direct, filiale à 100% de Homsys Group est une société de conseil, de services et de formation, spécialisée sur les technologies objet et Web. Conseil en méthodologie, en architecture et en urbanisation du SI, développement applicatif, édition et distribution de logiciels. www.objetdirect.com