JavaFX Mobile

1. JavaFX Mobile とは

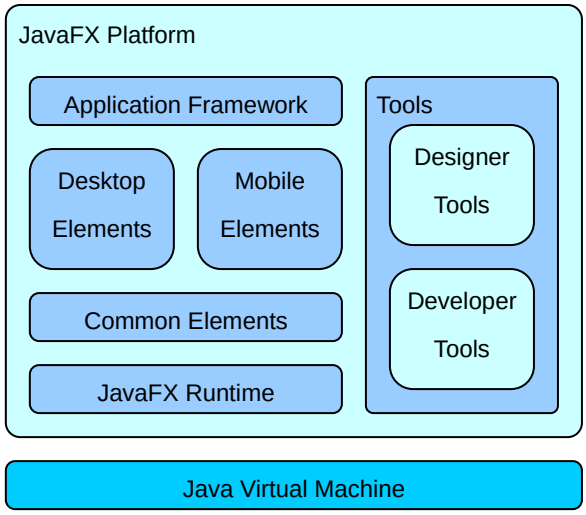
JavaFX は、RIA（リッチインターネットアプリケーション）開発プラットフォームです。すでに 2008 年 12 月 4 日に PC 向けの「JavaFX Desktop」がリリースされていましたが、2009 年 2 月 12 日に携帯電話向けの「JavaFX Mobile」がリリースされました。

JavaFX Mobile は、携帯電話向けの表現豊かな GUI アプリケーションを開発することができます。JavaFX Mobile は、Java ME プラットフォーム上に実装されているため、JavaFX Mobile から Java ME を使用して携帯電話の機能であるアドレス帳、Bluetooth、カメラ、GPS 等を利用したアプリケーションを開発することができます。また、既存の Java ME アプリケーションを JavaFX Mobile に対応させることによって、表現豊かなアプリケーションへ容易に変えることができます。

1.1. JavaFX プラットフォーム

JavaFX プラットフォームは、図 1-1 のような構成になっています。

図 1-1：JavaFX プラットフォーム



現在リリースされているデスクトップとモバイルだけでなく、テレビ等の機器に対応したバージョンがリリースされる予定です。

1.2. JavaFX Script

JavaFX アプリケーションの開発は、JavaFX Script プログラミング言語を使用します。JavaFX Script は、GUI プログラミングを容易にするために新しく設計されたプログラミング言語です。しかし、JavaScript と似た文法であることから JavaScript を使用している方には敷居は低いです。

JavaFX Script は、以下の特長を持っています。

- 宣言的文法
- データバインディング
- 置換トリガー

1.2.1. 宣言的文法

JavaFX Script を使用して GUI を作成する場合に宣言的文法を使用することができます。

図 1-2 のような GUI を作成する場合、宣言的文法を使用するとソースコードはリスト 1-1 のようになります。

図 1-2 JavaFX Mobile アプリケーションの例：

リスト 1-1：宣言的文法を使用した例

```
Stage {
    title: "Application title"
    width: 250
    height: 80
    scene: Scene {
        content: Text {
            font : Font {
                size : 16
            }
            x: 10, y: 30
            content: "Application content"
        }
    }
}
```

しかし、これまで Java の GUI である Swing のような記述をするとリスト 1-2 のようになります。

リスト 1-2：宣言的文法を使用しない例

```
Stage stage = new Stage();
stage.setTitle( “Application title” );
stage.setWidth(250);
stage.setHeight(80);
Scene scene = new Scene();
Text text = new Text();
text.setX(10);
text.setY(30);
text.setContent( “Application content” );
Font font = new Font();
font.setSize(16);
text.setFont(font);
scene.setContent();
stage.add(scene);
```

```
scene.setContent(text);
```

宣言的文法を使用したリスト 1-1 は、宣言的文法を使用しないリスト 1-2 に比べ、GUI の構造が分かり易く、かつ、コード量も少なくなります。

1.2.2. データバインディング

データバインディングは、bind キーワードを使用して変数の値と基本型の値、オブジェクト、関数の結果、式の結果を関連付けることができます。

簡単な例としてリスト 1-3 を記載しました。リスト 1-3 は、変数 x を変更すると自動的に変数 y が更新されます。リスト 1-3 の実行結果はリスト 1-4 になります。

リスト 1-3：データバインディングの例

```
var x = 0;
def y = bind x;
x = 1;
println(y);
x = 47;
println(y);
```

リスト 1-4：リスト 1-3 の実行結果

```
1
47
```

なお、変数 y は def キーワードを使用して宣言しているため、直接変数 y を変更することはできません。

1.2.3. 置換トリガー

置換トリガーは、on replace キーワードを使用して変数の値が変更される

と実行される処理を記述することができます。

簡単な例としてリスト 1-5 を記載しました。リスト 1-5 は、変数 password の値が変更されると変更前の値が変数 oldValue に格納され、変数 oldValue の後のブロックが実行されます。リスト 1-5 の実行結果はリスト 1-6 になります。

リスト 1-5：置換トリガーの例

```
var password = "foo" on replace oldValue {
    println("¥nALERT! Password has changed!");
    println("Old Value: {oldValue}");
    println("New Value: {password}");
};

password = "bar";
```

リスト 1-6：リスト 1-5 の実行結果

```
ALERT! Password has changed!
Old Value:
New Value: foo

ALERT! Password has changed!
Old Value: foo
New Value: bar
```

リスト 1-5 の置換トリガーは、変数 password への「foo」に初期化したときと「bar」に変更したときの 2 度発生します。

2. 開発環境

JavaFX アプリケーションの開発環境として http://javafx.com/から NetBeans IDE 6.5.1 for JavaFX 1.1.1 をダウンロードすることができます。

NetBeans IDE 6.5.1 for JavaFX は、開発に必要な環境が揃っており、Windows 版と Macintosh 版があります。Windows 版のみ JavaFX Mobile アプリケーションを PC で実行するために必要なエミュレータ JavaFX Mobile Emulator が付属しています。

動作する JDK は、JDK 6 Update 7 以降ですが JDK 6 Update 13 が推奨されています。

Windows への開発環境のセットアップ手順は以下の通りです。

2.1. JDK 6 Update 13 のインストール

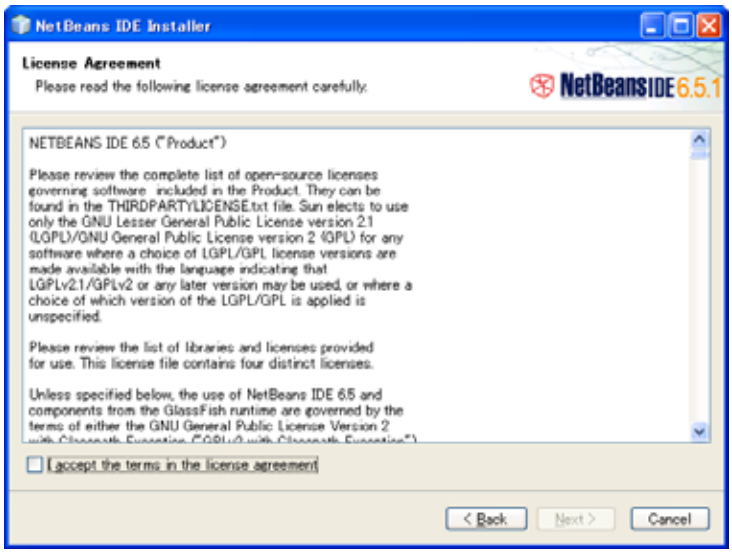
詳細な説明は割愛いたします。

2.2. NetBeans IDE 6.5.1 for JavaFX 1.1.1 のインストール

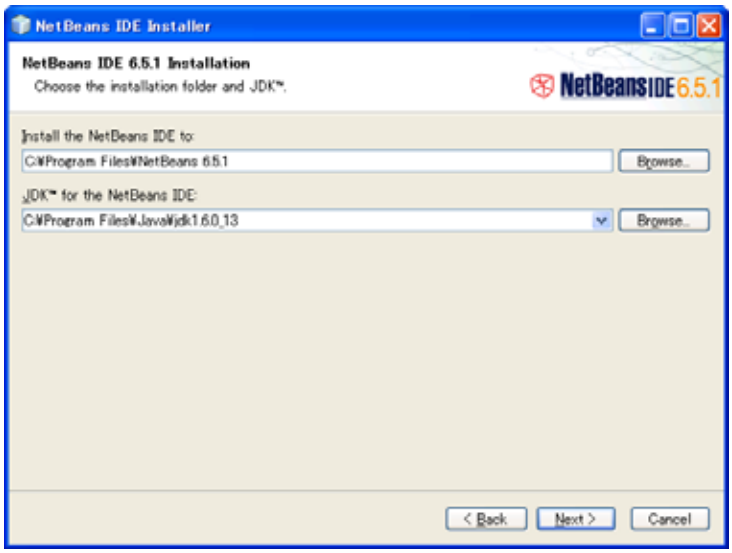
① http://javafx.com/より netbeans-6_5_1-javafx-1_1_1-windows.exe をダウンロードします。

② netbeans-6_5_1-javafx-1_1_1-windows.exe を実行します。

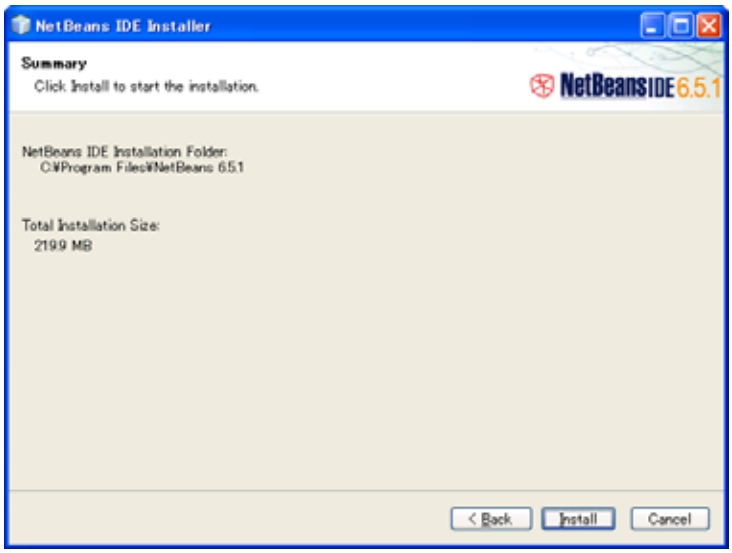
③ [Next]ボタンを押下します。



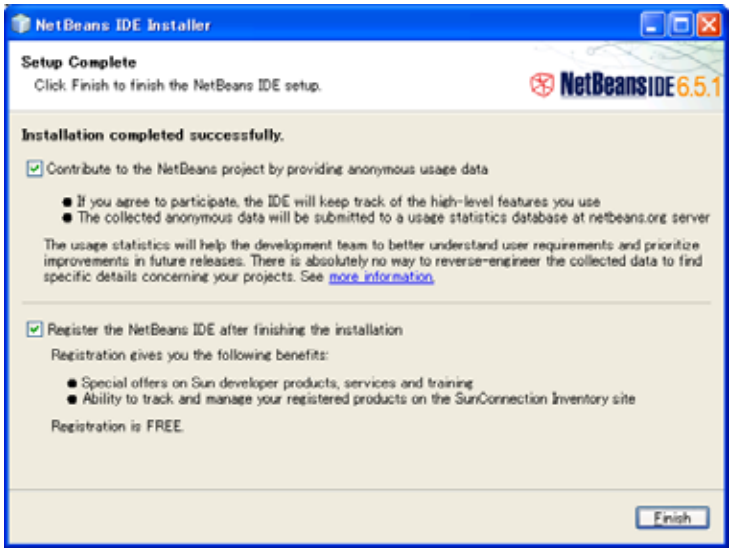
④ [I accept the terms in the license agreement]チェックボックスをチェックし、[Next]ボタンを押下します。



⑤ [Install the NetBeans IDE to]テキストボックスにインストールする任意のフォルダを、[JDK for the NetBeans IDE]テキストボックスに NetBeans で使用する JDK がインストールされているフォルダを入力し、[Next]ボタンを押下します。



⑥ [Install]ボタンを押下します。



⑦ NetBeans プロジェクトへ使用状況データを提供する場合は、[Contribute to the NetBeans project by providing anonymous usage data]チェックボックスをチェックします。また、ユーザ登録する場合は、[Register the NetBeans IDE after finishing the installation]チェックボックスをチェックします。

⑧ [Finish]ボタンを押下します。

3. 開発手順

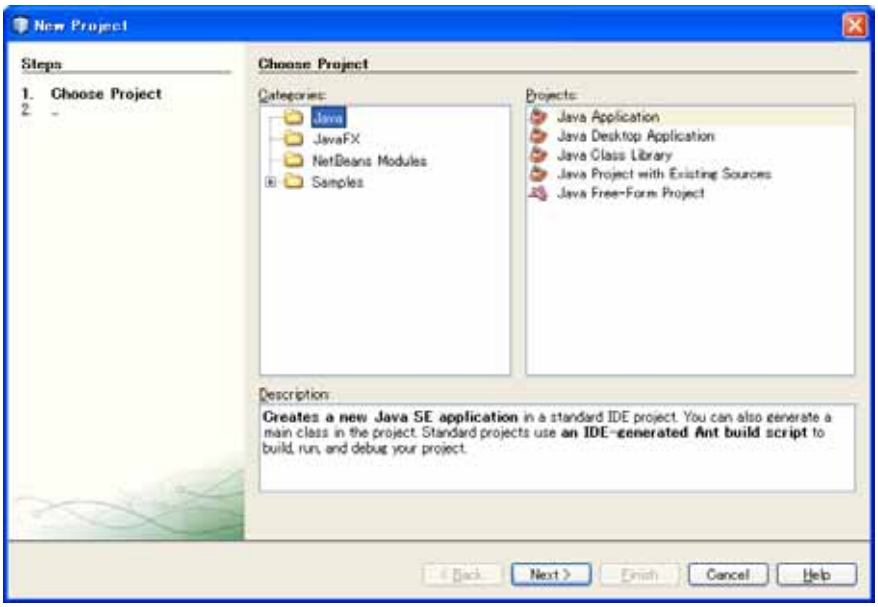
JavaFX Mobile アプリケーションの開発手順について説明します。

3.1. JavaFX Script Application プロジェクトの作成

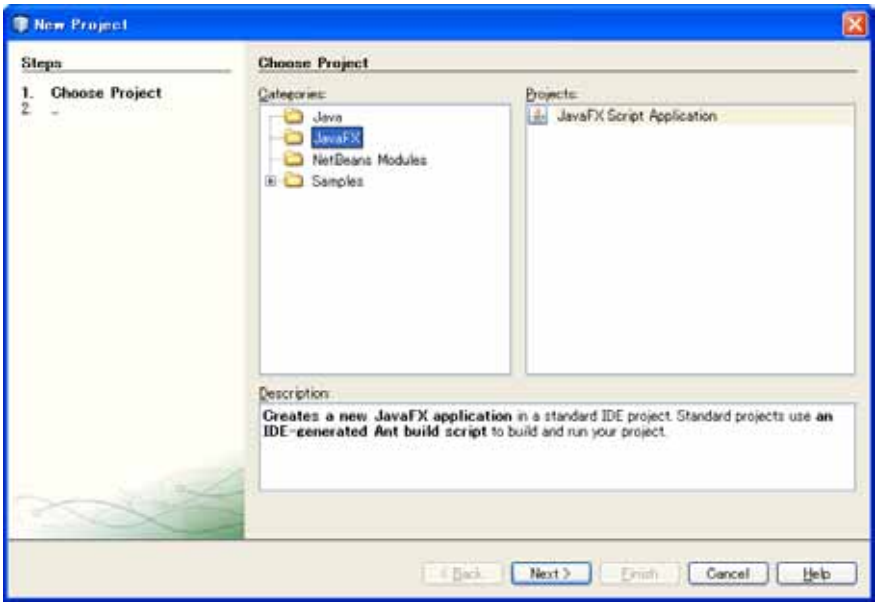
デスクトップ・モバイルに関わらず、JavaFX アプリケーションを開発するには JavaFX Script Application プロジェクトを作成する必要があります。

作成手順は以下の通りです。

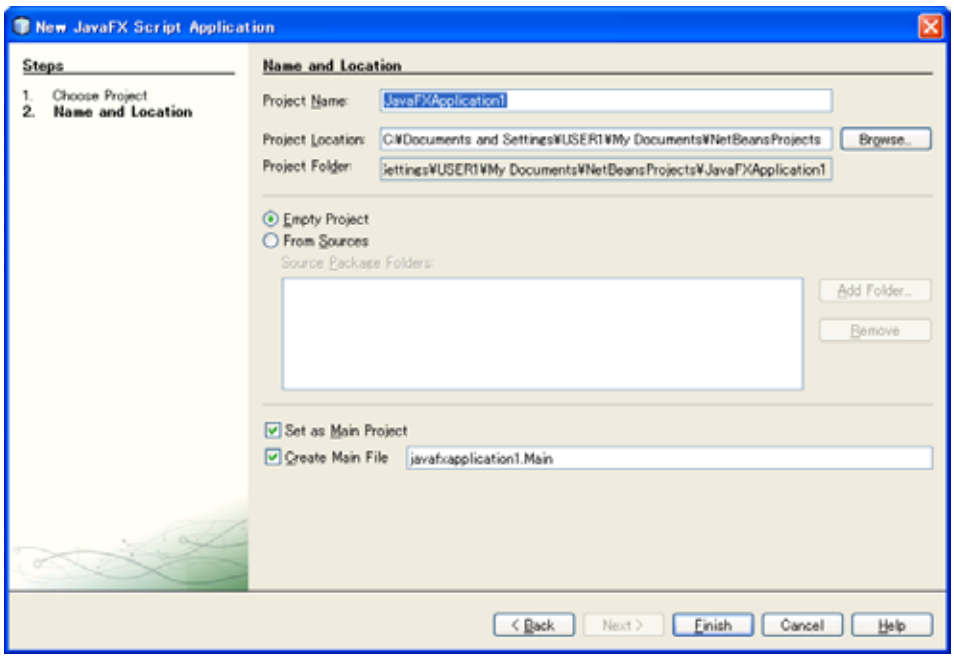
① メニューから[File]-[New Project...]を選択します。



② 表示された[New Project]ダイアログの[Categories]ツリーから[JavaFX]を選択し、[Projects]リストから[JavaFX Script Application]を選択します。



③　[Next >]ボタンを押下します。



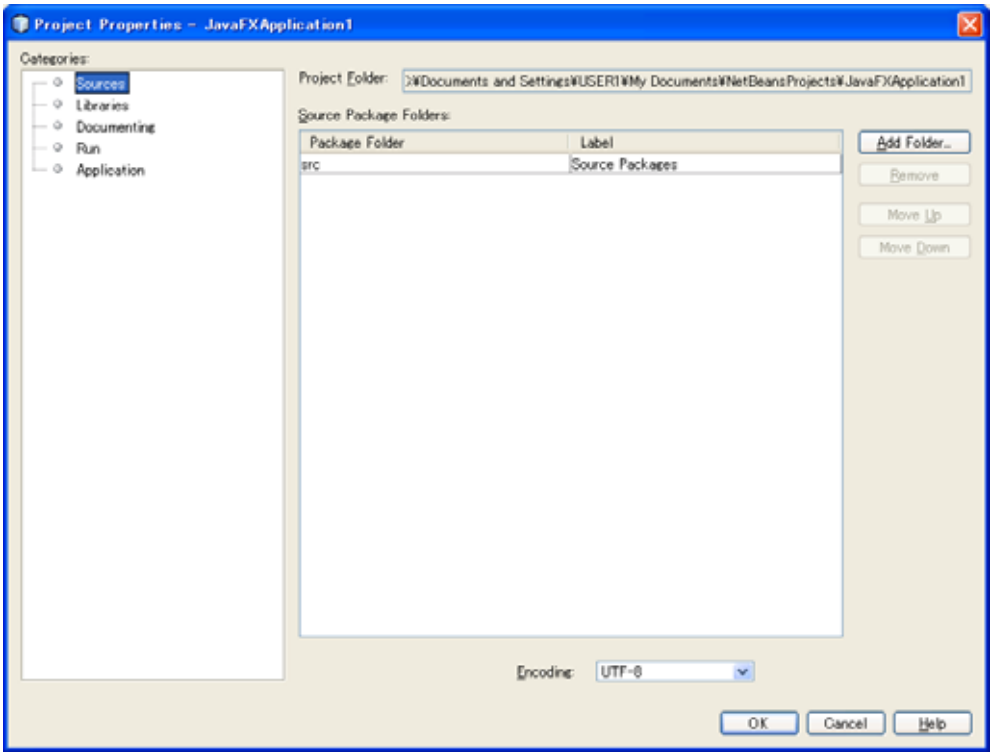
④　表示された[New JavaFX Script Application]ダイアログで[Project Name]にプロジェクト名、[Create Main File]にパッケージ名が含まれたファイル名（拡張子は不要）を入力し、[Finish]ボタンを押下します。

3.2. プロジェクトのプロパティの変更

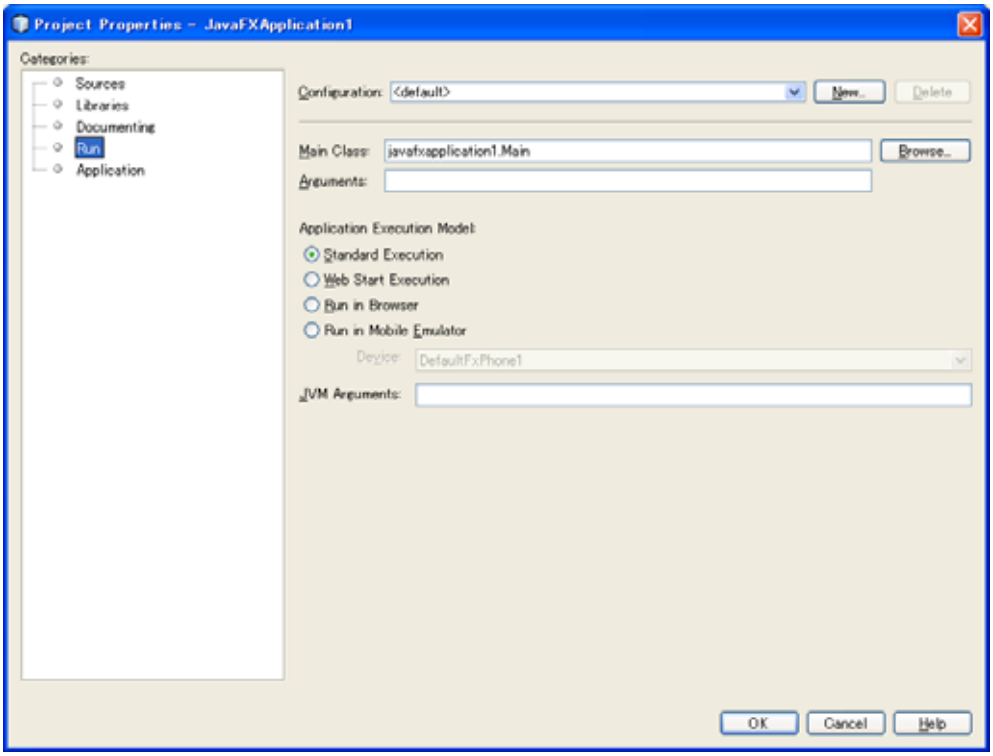
JavaFX Mobile アプリケーションをエミュレータで実行するにはプロジェクトのプロパティを変更する必要があります。

変更手順は以下の通りです。

① メニューから[File]-[Project Properties (プロジェクト名)]を選択します。



② 表示された[Project Properties]ダイアログの[Categories]リストから[Run]を選択します。



③ [Application Execution Model]ラジオボタンから[Run in Mobile Emulator]を選択し、[Device]コンボボックスから[DefaultFXPhone1]もしくは[DefaultFxTouchPhone1]を選択します。

[DefaultFXPhone1]を選択し、アプリケーションを実行すると通常の携帯電話のようなエミュレータとなります（図 3-1）。

図 3-1：DefaultFXPhone1 を選択した場合

[DefaultFxTouchPhone1]を選択し、アプリケーションを実行するとiPhoneのようなタッチパネル式の携帯電話のエミュレータとなります(図3-2)。

図 3-2：DefaultFxTouchPhone1 を選択した場合

④ [OK]ボタンを押下します。

3.3. アプリケーションの実行

JavaFX Mobile アプリケーションを実行するにはメニューから[Run]-[Run Project (プロジェクト名)]を選択するだけです。

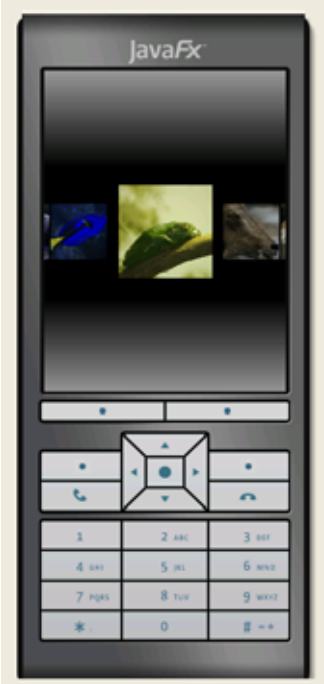
4. GUI アプリケーションの開発

JavaFX Mobile による GUI アプリケーション開発のすべての要素を説明することができないため、NetBeans IDE 6.5.1 for JavaFX 1.1.1 に付属するサンプル「Display Shelf Mobile」を使用し以下について説明します。

- Stage と Scene
- UI コンポーネント
- アニメーション

「Display Shelf Mobile」サンプルは、[New Project]ダイアログで[Categories]ツリーから[Samples] - [JavaFX]を選択し、 [Projects]リストから[Display Shelf Mobile]を選択することにより取得することができます。

図 4-1：「Display Shelf Mobile」サンプル



4.1. Stage と Scene

JavaFX アプリケーションには、Stage および Scene が必要です。Stage はウィンドウ、Scene はページに相当します。「Display Shelf Mobile」サンプルの Main.fx では**リスト 4-1** のように記述しています。

リスト 4-1：「Display Shelf Mobile」サンプルの Stage と Scene（Main.fx の 103～130 行目）

```
var scene:Scene = Scene {

    // MOBL-59: As of b04, the gradient works on the device but not working on the
    // emulator yet.
    fill: LinearGradient {
        startX: 0 startY: 0
        endX: 0 endY: 1
        proportional: true
        stops: [
            Stop { offset: 0.0 color: Color.rgb(150,150,150) }
            Stop { offset: 0.3 color: Color.rgb(0,0,0)},
            Stop { offset: 0.7 color: Color.rgb(0,0,0)},
            Stop { offset: 1.0 color: Color.rgb(150,150,150)},
        ]
    }
    content: [
        shelf
    ]
};

Stage {
    title: "Display Shelf Mobile"
    visible: true
    resizable: true
    scene: scene
    width: 320
    height: 240
}
```

4.2. UI コンポーネント

JavaFX の UI 構造は Scene をルートとするツリー構造となり、各要素はノードと呼ばれます。ノードは、javafx.scene.Node クラスを継承します。UI コンポーネントは、シェープ、テキスト、画像等の様々な種類があります。

UI コンポーネントを使用した例として画像表示を「Display Shelf Mobile」サンプルを使用して説明します。

「Display Shelf Mobile」サンプルでは、画像ノードを直接表示せず、アプリケーション独自のノードでラッピングし表示しています。アプリケーション独自のノードを作成するには javafx.scene.CustomNode クラスを継承します（**リスト 4-2**）。

リスト 4-2：「Display Shelf Mobile」サンプルの画像表示（Item.fx の 41～65 行目）

```
public class Item extends CustomNode {
...
}
```

画像を表示するには、まず、画像を javafx.scene.image.Image オブジェクトとしてロードする必要があります（**リスト 4-3**）。

リスト 4-3：「Display Shelf Mobile」サンプルの画像表示（Main.fx の 85 行目）

```
            image:Image { url: "{__DIR__}photos/{i}" }
```

Image クラスは、ノードではないため Node クラスを継承した javafx.scene.image.ImageView にセットする必要があります（**リスト 4-4**）。

リスト 4-4：「Display Shelf Mobile」サンプルの画像表示（Item.fx の 55～57 行目）

```
                ImageView {
```

```
                image: image fitWidth: bind imageWidth fitHeight: bind imageWidth
            },
```

Item.fx の 56 行目の「image: image」で Main.fx の 85 行目（**リスト 4-3**）で生成した Image オブジェクトをセットしています。

4.3. アクション

UIコンポーネントへのイベントに対応するアクションを追加することができます。JavaFXのアクションは表4-1の通りです。

表4-1：アクション

アクション	説明
onKeyPressed	キーを押下したときのアクション
onKeyReleased	キーを押下し終えたときのアクション
onKeyTyped	キーによって文字を入力したときのアクション
onMouseMoved	カーソルを動かしたときのアクション
onMouseEntered	カーソルが領域内に入ったときのアクション
onMouseExited	カーソルが領域内から出たときのアクション
onMouseClicked	マウスボタンを押下したときのアクション
onMousePressed	マウスボタンを押し続けたときのアクション
onMouseReleased	マウスボタンを離したときのアクション
onMouseDaragged	ドラッグしたときのアクション
onMouseWheelMoved	マウスホイールを動かしたときのアクション

「Display Shelf Mobile」サンプルでは、キーを押下したときのアクション（リスト4-5）とマウスボタンを押下したときのアクション（リスト4-6）を定義しています。

リスト4-5：「Display Shelf Mobile」サンプルのキーアクション（Main.fxの92～99行目）

```
onKeyPressed:function(e:KeyEvent):Void {
    if(e.code == KeyCode.VK_LEFT or e.code == KeyCode.VK_SOFTKEY_0) {
        shelf.shift(1);
    }
    if(e.code == KeyCode.VK_RIGHT or e.code == KeyCode.VK_SOFTKEY_1) {
        shelf.shift(-1);
    }
}
```

リスト 4-6：「Display Shelf Mobile」サンプルのマウスアクション（Item.fx の 48～50 行目）

```
    override var onMousePressed = function(e:MouseEvent) {
        shelf.shiftToCenter(this);
    };
```

4.4. アニメーション

アニメーションは、時系列を表すタイムライン（javafx.animation.Timeline クラス）に、タイムライン上のとある瞬間を表すフレーム（javafx.animation.KeyFrame クラス）を定義し実現します。フレームとフレームの途中の画面は、JavaFX によって自動計算され描画されます。

「Display Shelf Mobile」サンプルでは、リスト 4-7 のように定義しています。

リスト 4-7：「Display Shelf Mobile」サンプルのアニメーション（DisplayShelf.fx の 103～146 行目）

```
public function doLayout() {

    var endKeyframes:KeyFrame[];
    var duration = 0.5s;
    var centerOffset = (scene.width-imageWidth)/2;

    for(n in left.content) {
        var it = n as Item;
        var newX = -left.content.size()*spacing +  spacing * indexof n + leftOffset;
        insert KeyFrame { time: duration values: [
                n.translateX => newX + centerOffset,
                n.scaleX => scaleSmall,
                n.scaleY => scaleSmall,
                it.angle => 45
            ] } into endKeyframes;
    }

    for(n in center.content) {
        var it = n as Item;
        insert KeyFrame { time: duration values: [
                n.translateX => centerOffset,
                n.scaleX => 1.0,
                n.scaleY => 1.0,
                it.angle => 90
            ] } into endKeyframes;
```

```
        }

        for(n in right.content) {
            var it = n as Item;
            var newX = right.content.size()*spacing -spacing * indexof n + rightOffset;
            insert KeyFrame { time: duration values: [
                    n.translateX => newX + centerOffset,
                    n.scaleX => scaleSmall,
                    n.scaleY => scaleSmall,
                    it.angle => 135
                ] } into endKeyframes;
        }

        var anim = Timeline {
            keyFrames: [endKeyframes]
        };
        anim.playFromStart();

    }
```

KeyFrame クラスは、アニメーションのスタートしてからのミリ秒を表す time プロパティとそのときの状態を表す values プロパティが設定します。

5. まとめ

JavaFX Mobile は、使用できる JavaFX API が JavaFX Desktop に比べ若干制限されているだけで、アプリケーションの開発方法については相違がありません。したがって、JavaFX Desktop、JavaFX Mobile のどちらかを学習することによって、デスクトップとモバイルの双方に対応したアプリケーションを容易に開発できるようになります。また、リリース予定のテレビに対応したアプリケーションも容易に開発できるようになるでしょう。

携帯電話における有力な RIA プラットフォームは、JavaFX Mobile、Microsoft Silverlight、Adobe AIR、さらに Google Android、Apple iPhone と群雄割拠になっています。しかし、OS によってある程度住み分けができると考えられます。Windows Mobile は Silverlight、Google Android は Android 独自の Java、Apple iPhone は Objective-C が OS との結び付きが強いため大勢を占めることになるでしょう。しかし、Windows Mobile、Android、iPhone 以外の OS では JavaFX Mobile と AIR の争いになると思われます。JavaFX Mobile と AIR は、ベースとなっている Java ME と Flash がすでに多くの携帯電話に搭載されています。しかし、携帯電話アプリケーションの実装では Java ME が大勢を占めていることから携帯電話 RIA アプリケーションでも JavaFX Mobile が優勢になると思います。

JavaFX Mobile に対応した端末がまだ出荷されていないので、Java エンジニアとしては早期に出荷されることを望みます。

6. 参考資料

- JavaFX
  http://javafx.com/

- JavaFX リファレンス - ドキュメント、チュートリアル、および API
  http://sdc.sun.co.jp/java/javafx/reference/docs.html

- JavaFX 1.1 API | APIs for Rich Internet Applications for Mobile and Desktop
  http://java.sun.com/javafx/1.1/docs/api/

- JavaFX チュートリアル ｜ Web 開発用の JavaFX Script について
  http://sdc.sun.co.jp/java/javafx/1/tutorials/core/index.html

- JavaFX チュートリアル ｜ デスクトップおよびモバイル用のアプリケーションの開発
  http://sdc.sun.co.jp/java/javafx/1/tutorials/ui/index.html

- ついにベールを脱いだ JavaFX
  http://gihyo.jp/dev/serial/01/javafx/

開発部
大森　洋行