JavaFX: เทคโนโลยีใหม่สำหรับสร้าง GUI

อ. เนรมิต ชุมสาย ณ อยุธยา*

Abstract:

Programmers who use Swing for application development often time find GUI building as a tedious and annoying task when come to coding manually. Later, drag-and-drop features such as Matisse GUI builder were incorporated into Netbeans which makes it easier to create GUIs. But for those who prefer to code without the help of visual designer find it's still too difficult to create even simple GUI. However, recently at Java One 2007 conference in May, Sun introduced new technology called: JavaFX Scripting Language, a language which helps make GUI building simpler. The aim of this article is to explore and introduce JavaFX as an alternative tool for GUI creation as compare to Swing.

คำสำคัญ

JavaFX (Computer program language)

บทนำ

สำหรับ programmer ที่ใช้ Java เป็นภาษาในการพัฒนานั้นการสร้าง Graphical User Interface (GUI) เป็นเรื่องไม่ง่ายนัก ถึงแม้ว่าจะมี Netbeans ที่ใช้ Matisse เป็นตัวช่วยสร้าง (ก่อนที่จะมี Matisse ยิ่งไม่ต้องพูดถึงเราต้องเขียน code เท่านั้นไม่มีการ drag and drop) ยิ่งเป็น programmer ใหม่แล้วการศึกษาถึงองค์ประกอบของ Swing (สำหรับสร้าง GUI) นั้นต้องใช้เวลามาก แต่เมื่อเดือนพฤษภาคม ที่ผ่านมาในงาน JavaOne 2007 ที่ประเทศสหรัฐอเมริกา JavaFX ได้ถูกนำมาเปิดเผยเป็นครั้งแรก สร้างความตื่นเต้นและสงสัยให้กับนักพัฒนาโปรแกรมหลายๆ คนที่เข้าร่วมงานเป็นอย่างมาก มีการถกเถียงและอภิปรายถึงขีดความสามารถของ JavaFX กันในหลายแง่มุม อย่างไรก็ตามเราคงไม่เข้าไปดูว่าเขาได้ถกเถียงกันอย่างไร แต่เราจะมาดูว่า JavaFX ตัวนี้สามารถนำมาใช้ในการสร้าง GUI ได้อย่างไร

JavaFX Scripting Language

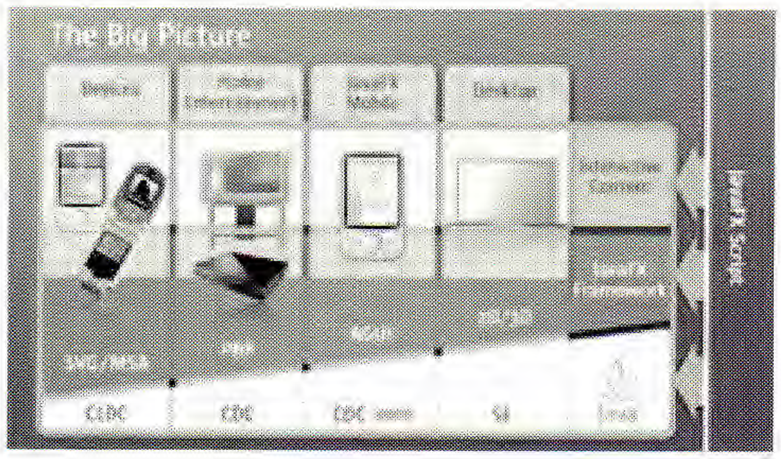
JavaFX เป็นภาษาที่ถูกเรียกว่า declarative and statically typed language ซึ่งมีความหมายอย่างหยาบๆ ว่าถ้าเราประกาศตัวแปรให้เป็นชนิดอะไรแล้วเราไม่สามารถกำหนดค่าที่เป็นชนิดอื่นให้กับตัวแปรตัวนี้ได้[1] ถ้าผู้อ่านเขียนภาษาที่เป็น script มาก่อนก็จะรู้ว่าเราสามารถเปลี่ยนชนิดของข้อมูลที่เก็บอยู่ในตัวแปรได้ (เช่น JavaScript เป็นต้น) ข้อแตกต่างระหว่างภาษาที่เป็น declarative กับภาษาที่เป็น procedural คือ declarative language จะเป็นภาษาที่ programmer ไม่จำเป็นต้องเขียน code รองรับการทำงาน (ในรูปแบบของ sequential instructions) เองแต่ต้องรู้ว่าองค์ประกอบต่างๆ ของภาษามีความสัมพันธ์กันอย่างไรเพื่อให้การเรียกใช้เป็นไปอย่างถูกต้อง ส่วน procedural language นั้นประกอบไปด้วยประโยคหลายๆ ประโยคที่มีการประมวลผลเป็นระดับๆ (sequential execution) และ programmer จะเป็นผู้เขียน code รองรับการทำงานต่างๆ ที่ต้องการเอง อย่างไรก็ตาม JavaFX ก็รองรับการเขียน code ทั้งสองรูปแบบ แต่ถ้าจะให้ดีแล้วเราควรเลือกใช้การเขียนที่เป็น declarative มากที่สุดเท่าที่โอกาสจะเอื้ออำนวย

บริษัท Sun Micros Systems วางแผนที่จะให้ JavaFX script เป็นอีกทางเลือกหนึ่งสำหรับการพัฒนาโปรแกรมบนอุปกรณ์ต่างๆ แต่ ณ เวลานี้ยังคงรองรับแค่ mobile device และ JavaFX script สำหรับ Rich Internet Application (RIA) เท่านั้น - ดูภาพ

---

* คณบดีคณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยฟาร์อีสเทอร์น

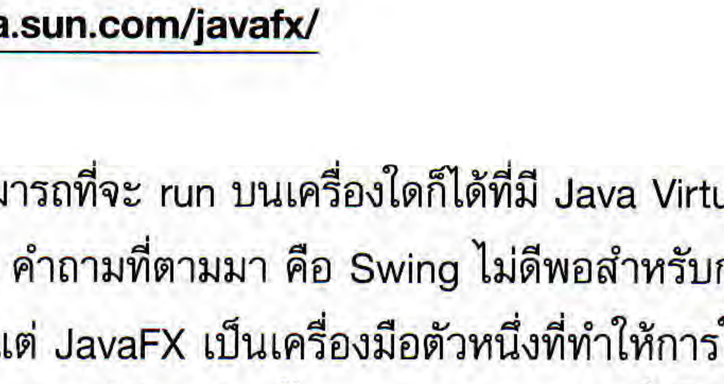
[1] Wikipedia - declarative program คือโปรแกรมที่บอกถึงว่าสิ่งที่ต้องการเป็นอะไร (what) ไม่ใช่บอกว่าทำอย่างไร (how) เช่น code ใน HTML จะบอกถึงหน้าตาของ web page ว่าควรเป็นอย่างไร แต่ไม่ได้บอกว่าต้องทำอย่างไร



ภาพจาก http://java.sun.com/javafx/

เช่นเดียวกันกับ Java โปรแกรมที่สร้างขึ้นจาก JavaFX สามารถที่จะ run บนเครื่องใดก็ได้ที่มี Java Virtual Machine (JVM) อยู่ JavaFX ช่วยให้การสร้าง UI และ 2D Graphics ทำได้ง่ายขึ้น คำถามที่ตามมา คือ Swing ไม่ดีพอสำหรับการสร้าง UI หรือ ? ที่จริงแล้ว Swing เป็นเครื่องมือที่ใช้สร้าง UI ได้ดีที่สุดของ Java แต่ JavaFX เป็นเครื่องมือตัวหนึ่งที่ทำให้การใช้ Swing เป็นไปได้ง่ายขึ้น โครงสร้างของ code ใน JavaFX ใกล้เคียงกับ layout ของ UI ที่ผู้ใช้สร้างขึ้น และ JavaFX ยังเอื้อให้ผู้ใช้สามารถเรียกใช้ Java API ได้อย่างง่าย ๆ โปรแกรมตัวอย่างแรกของเราเป็นโปรแกรมง่ายๆ ที่แสดงคำว่า 'Hello World' ไปยังหน้าต่าง ดังตัวอย่างที่ 1

```
import javafx.ui.*;

Frame {
    title: "Hello"
    width: 200
    height: 50

    content: Label {
        text: "Hello World"
    }

    visible: true
}
```



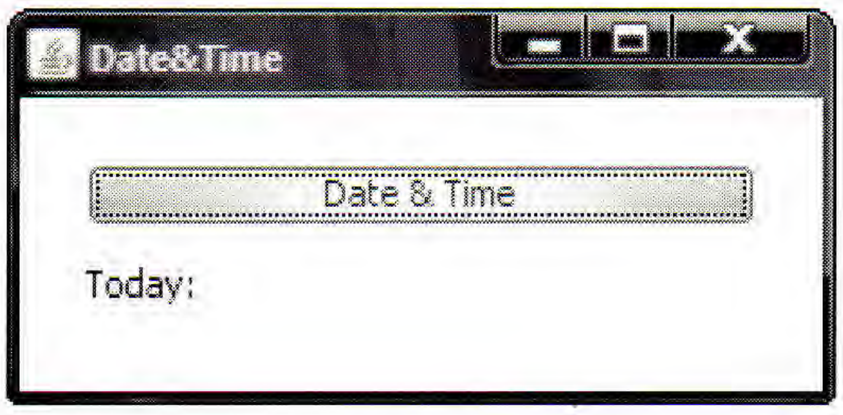
ตัวอย่างที่ 1 แสดงให้เห็นคำว่า Hello World ที่หน้าต่าง

ผู้อ่านจะเห็นว่าโครงสร้างของ code ที่เราเขียนขึ้นดูเข้าใจง่าย ชุดคำสั่งที่ใช้สร้าง UI อยู่ในรูปแบบที่กระชับและสอดคล้องกับหน้าต่างที่เกิดขึ้น (ถ้าเราใช้ Swing เขียนชุดคำสั่งหลายๆ ตัวดูแล้วไม่บอกถึงสิ่งที่จะเกิดขึ้นเมื่อ code ได้รับการประมวลผล - ดู code ของ Swing ด้านล่างที่สร้างหน้าต่างแบบเดียวกัน) จาก code ที่เขียนขึ้นเรากำหนดให้มีหน้าต่าง (Frame) ที่มีสัดส่วนเป็น 200 x 50 (width x height) พร้อมกับกำหนดให้ title ของหน้าต่างเป็น "Hello" และมีข้อความในตัวหน้าต่างเป็น "Hello World" ผู้อ่านจะสังเกตเห็นว่าเราไม่ต้องเขียนชุดคำสั่งแบบที่เป็น procedural เลยเราเพียงแค่ประกาศว่าเราต้องการ Frame ที่มี attribute ตามที่เราต้องการเท่านั้นเอง ภาษาที่เป็น declarative เช่น JavaFX นั้นจะสร้าง instance (หรือที่เรียกกันทั่วไปว่า object) ของ class ทุกตัวที่มีอยู่ให้โดยอัตโนมัติ และจะทำการกำหนดค่าให้กับ attribute ต่างๆ ที่มีอยู่ เช่น กำหนดให้ title ของ object ที่เกิดจาก class Frame มีค่าเป็น "Hello" เป็นต้น การกำหนดค่าก็ต้องทำผ่านชื่อของ attribute ตามด้วยเครื่องหมาย ':'และตามด้วยค่าที่ต้องการกำหนด ส่วนทางด้าน Swing นั้นเราต้องสร้าง Frame ขึ้นมาก่อนแล้วจึงเรียกใช้ method สำหรับสร้างส่วนอื่นๆ ที่เหลืออยู่ ดังที่แสดงจาก code นี้

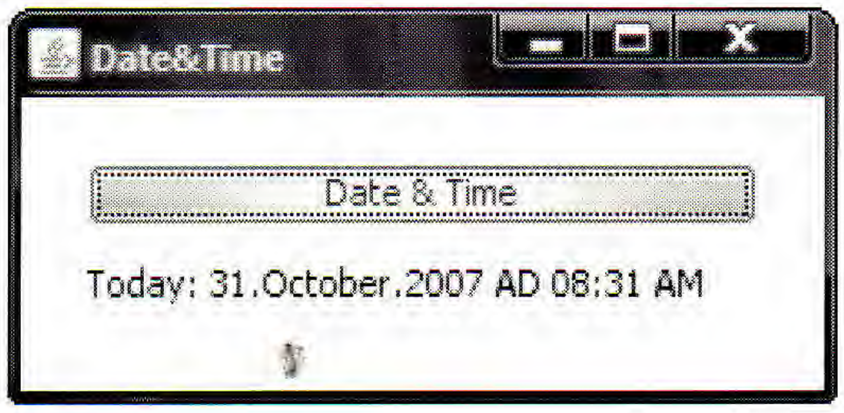
```
import javax.swing.JFrame;
import javax.swing.JLabel;

public class HelloWorldSwing {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Hello");
    frame.setSize(200, 50);
    final JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

ตัวอย่างที่ 2 การใช้ Swing ในการเขียนโปรแกรม

ผู้อ่านควรสังเกตว่าเราต้องสร้าง Label ด้วยค่าที่ต้องการก่อนการนำเอา JLabel เข้าสู่ JFrame ซึ่งต้องทำด้วยการเรียก getContentPane() ซึ่งเป็นสิ่งที่เราไม่ต้องทำเลยใน JavaFX เราเพียงแต่กำหนดให้ content ของ Frame เป็น Label เท่านั้นเอง เมื่อเปรียบเทียบ code ทั้งสองแล้วผู้อ่านคงจะพอเห็นแล้วว่า JavaFX น่าจะเป็นตัวสร้าง UI ที่ง่ายกว่า Swing จริงๆ อย่างไรก็ตาม ตัวอย่างที่เราแสดงให้ดูเป็นเพียงการสร้าง UI อย่างง่ายๆ ผู้อ่านอาจมีคำถามว่าถ้าเราต้องการให้ตัวโปรแกรมรองรับเหตุการณ์ (event) จากผู้ใช้ JavaFX มีความยากง่ายในการใช้อย่างไร เรามาลองดูตัวอย่างนี้กัน



หน้าต่างก่อนการกดปุ่ม



หน้าต่างหลังการกดปุ่ม

```
class TimeModel {
    attribute time: String;
}

var model = new TimeModel();

var win = Frame {
    title: "Date&Time"
    width: 250
    height: 120
    content: GridPanel {
        border: EmptyBorder {
            top: 20
            left: 20
            bottom: 20
            right: 20
        }
        rows: 2
        columns: 1
        vgap: 10
        cells:
            [Button {
                text: "Date & Time"
                action: operation() {
                    var d = new Date();
                    model.time = d format as <<dd.MMMMM.yyyy GGG hh:mm aaa>>;
                }
            },
            Label {
                text: bind "Today: {model.time}"
            }]
    }
    visible: true
};
```

ตัวอย่างที่ 3 การรองรับเหตุการณ์จาก user

การสนองตอบต่อ event ของ Swing นั้นเราต้องกำหนดให้มี event listener ให้กับ component ที่เราต้องการ เช่น button.addActionListener() และต้องเขียน code รองรับ event นั้น แต่ใน JavaFX เราไม่จำเป็นต้องกำหนดให้มีการคอยฟัง event ที่จะเกิดขึ้นเพราะ JavaFX จะทำหน้าที่ดังกล่าวอยู่แล้ว สิ่งที่เราต้องทำก็คือ จะสนองตอบต่อเหตุการณ์นั้นๆ อย่างไรเท่านั้นเอง ดังเช่นที่เราได้แสดงให้ดูจากโปรแกรมตัวอย่างที่เห็น

เราจะให้โปรแกรมของเราทำการแสดงวันและเวลาไปยังหน้าต่างเมื่อผู้ใช้กดปุ่ม Date&Time ด้วย operation ที่เราสร้างขึ้นให้กับปุ่มดังนี้

```
action: operation() {
    var d = new Date();
    model.time = d format as <<dd.MMMMM.yyyy GGG hh:mm aaa>>;
}
```

เราดึงเอาเวลาในเครื่องมาเก็บไว้ที่ตัวแปร d หลังจากนั้นเราก็เปลี่ยน format ของเวลาที่ดึงมาให้อยู่ในรูปแบบของ String ที่เราต้องการพร้อมกับนำไปเก็บไว้ในตัวแปร time ที่เป็น attribute ของ class TimeModel ซึ่งเป็นที่ที่เราใช้เก็บเวลาจากเครื่อง เพราะฉะนั้นเราจึงต้องสร้าง object จาก class TimeModel ขึ้นมาใช้เหมือนกับที่เราทำใน Java ดังนั้นการเรียก attribute ตัวนี้จึงต้องทำผ่าน object ที่ถูกสร้างขึ้นจาก class นี้ (model.time)

มาถึงตอนนี้ผู้อ่านคงสงสัยว่าแล้วทำไมเมื่อกดปุ่ม เวลาจึงมาปรากฏในที่ๆ ได้กำหนดไว้ (ทั้งๆ ที่ไม่มี code สำหรับการแสดงผลเลย) คำตอบอยู่ที่ attribute: text ของ Label ที่เราสร้างขึ้น

```
Label {
    text: bind "Today: {model.time}"
}
```

เมื่อค่าของ model.time ได้ถูกกำหนดขึ้น (จากการกดปุ่ม) ค่าของ text ที่อยู่ใน Label ก็จะได้รับค่านี้ด้วยเพราะเราได้กำหนดให้มีการ "bind" text เข้ากับค่าของ model.time

การ bind คือการที่ object ตัวหนึ่งมีพฤติกรรม (behavior) ที่ขึ้นอยู่กับ object อีกตัวหนึ่ง เช่น ถ้า object x มีการ bind กับ object y การเปลี่ยนแปลงใด ๆ ของ y จะมีผลกับ x เช่นเดียวกันถ้า object y ได้ถูก bind เข้ากับ object x การเปลี่ยนแปลงของ x ก็จะมีผลกับ y เช่นกัน การใช้ bind ช่วยให้การ update เป็นไปได้ง่ายขึ้น ไม่ว่าจะเป็นค่าที่อยู่ภายใน object เองหรือคุณลักษณะ (behavior) ของ object เอง (เราได้แสดงการ bind ที่เปลี่ยนคุณลักษณะของ object ในเรื่องของการสร้างภาพเคลื่อนไหวในตอนท้ายของบทความนี้)

ในทางตรงกันข้ามถ้าเราใช้ Swing เป็นตัวสร้างแล้วเราก็ต้องเขียน code รองรับการสนองตอบต่อการกดปุ่มของผู้ใช้ เช่นที่เห็นนี้

```
public class DateTime {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Date&Time");
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(2, 1));
    panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
    panel.add(button);
    panel.add(label);
    JLabel label = new JLabel("Today: ");
    JButton button = new JButton("Date&Time");
    button.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
         Date today = new Date();
         SimpleDateFormat fm = new SimpleDateFormat("dd.MMMMM.yyyy GGG hh:mm aaa");
         String date = fm.format(today);
         label.setText("Today: " + date);
       }
    });

    frame.setSize(280, 120);
    frame.getContentPane().add(panel);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

ผู้อ่านคงเห็นแล้วว่าการสร้าง UI และการสนองตอบต่อเหตุการณ์ของ JavaFX นั้นทำได้อย่างเป็นระเบียบ (และง่ายกว่า) Java มาก (ผู้อ่านที่ชื่นชม Java อย่างมากก็สามารถมองในมุมต่างจากผู้เขียนได้) อย่างไรก็ตาม JavaFX ยังอยู่ในสภาวะที่เราเรียกว่าสภาวะเริ่มต้น ทั้งนี้ก็เพราะว่าภาษาอื่นๆ ที่มีอยู่ (รวมถึง Java) นั้นมีกระบวนการที่รองรับการทำงานอื่นๆ ที่ซับซ้อนมากกว่า JavaFX ยังต้องการการพัฒนาอีกมากกว่าจะเป็นที่ยอมรับสำหรับนักพัฒนาโปรแกรมหลายๆ คน

Trigger (สิ่งที่มาแทน Constructor และเครื่องมือสำหรับการ update ค่า)

JavaFX ยังมีลูกเล่นอื่น ๆ ให้เราเรียกใช้ได้อีกมากมาย แต่หนึ่งในสิ่งที่ขาดไปก็คือ constructor ผู้อ่านที่ใช้ Java โดยส่วนใหญ่แล้วมักสร้าง constructor เพื่อใช้ในการกำหนดค่าให้กับ field ต่างๆ ที่มีอยู่ใน class นั้นๆ อย่างไรก็ตาม JavaFX มีโครงสร้างที่ดีตัวหนึ่งที่นำมาใช้แทน constructor ได้ นั่นก็คือ trigger เราจะแนะนำการใช้ trigger ด้วยการสร้าง Queue ที่มีการกำหนดค่าเบื้องต้นให้กับพื้นที่สำหรับเก็บข้อมูล (storage) และตัวนับจำนวนของข้อมูล (count) ที่มีอยู่ใน Queue ผ่านทาง trigger

```
class Queue {
   attribute storage: Number*;
   attribute count: Integer;
   operation add(item: Number);
   operation remove() : Number;
   operation print();
}

operation Queue.print() {
   var i = 0;
   System.out.print("[{this.storage[i++]}");
   while(i < this.count) {
      System.out.print(", {this.storage[i++]}");
   }
   System.out.println("]");
   System.out.println("count: {this.count}");
}

trigger on new Queue {
   this.storage = [1, 2, 3, 4];
   this.count = 4;
}

var queue = new Queue();
queue.print();
```

Class Queue ที่เราสร้างขึ้นนี้มี trigger เป็นตัวกำหนดข้อมูลเบื้องต้นให้กับ attribute: storage และ count โดยกำหนดให้ข้อมูลเบื้องต้นภายใน Queue มีค่าเป็น [1, 2, 3, 4] และค่าของ count เป็น 4

```
trigger on new Queue {
   this.storage = [1, 2, 3, 4];
   this.count = 4;
}
```

เมื่อเราสร้าง object: queue จาก class Queue ตัวนี้ (var queue = new Queue()) เราก็เรียกใช้ print() ผ่านทางประโยค queue.print() เพื่อตรวจสอบดูว่าค่าต่าง ๆ ได้ถูกกำหนดไว้จริง และเมื่อเรา run โปรแกรมผลลัพธ์ที่เราได้คือ

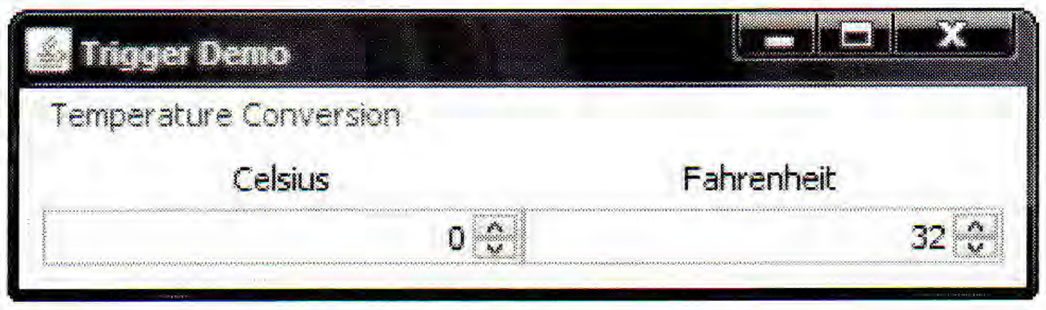
```
[1, 2, 3, 4]
count: 4
```

ซึ่งเป็นการยืนยันว่า trigger ได้ทำหน้าที่เป็นตัวกำหนดค่าเบื้องต้นให้กับเรา (แทนการใช้ constructor) ได้เป็นอย่างดี เราสามารถใช้ trigger ได้อีกหลายแบบ เช่น หักค่าของ count ออกหนึ่งค่าเมื่อมีการลบข้อมูลออกจาก Queue หรือเพิ่มค่าให้กับ count อีกหนึ่งค่าเมื่อมีการนำข้อมูลเข้าสู่ Queue ซึ่งเขียนได้ง่ายๆ ดังนี้

```
trigger on insert data into Queue.list {
    this.count++;
}

trigger on delete data from Queue.list {
    this.count--;
}
```

ผู้อ่านคงสังเกตเห็นถึงการใช้ภาษาที่คล้าย ๆ กับภาษา SQL ที่เราใช้กันในการ query ฐานข้อมูล ใช่แล้วครับ JavaFX มีคำสั่งที่คล้ายกับ SQL ในการทำงานกับโครงสร้างที่ใช้เก็บข้อมูล เช่น select n from n in [1..100] where n%2 == 0 อย่างไรก็ตามเราคงไม่พูดถึงโครงสร้างทั้งหมดของ JavaFX ในที่นี้เพราะเราต้องการแสดงให้ดูถึงศักยภาพของ JavaFX ในเรื่องของการสร้าง UI และ 2D Graphics เท่านั้น

โปรแกรมตัวอย่างต่อไปที่เราจะแสดงให้ดูเป็นโปรแกรมง่ายๆ ที่แสดงค่าของอุณหภูมิระหว่างองศา Celsius และ Fahrenheit ด้วยการใช้ trigger เป็นตัวปรับค่า




```
class Temp {
    attribute celsius: Number;
    attribute fahrenheit: Number;
}

trigger on Temp.celsius = value {
    fahrenheit = (9/5 * celsius + 32);
}

trigger on Temp.fahrenheit = value {
    celsius = (5/9 * (fahrenheit - 32));
}

var temp = Temp {
    fahrenheit: 32
};

Frame {
    title: "Trigger Demo"
    height: 100
    width: 350
    content: GridPanel {
        border: TitledBorder {
            title: "Temperature Conversion"
        }
        rows: 2
        columns: 2
        cells: [SimpleLabel {
            horizontalAlignment: CENTER text: "Celsius"
        },
        SimpleLabel {
            horizontalAlignment: CENTER text: "Fahrenheit"
        },
        Spinner {
            min: -60 max: 100 value: bind temp.celsius
        },
        Spinner {
            min: -76 max: 212 value: bind temp.fahrenheit
        }]
    }
    visible: true
    centerOnScreen: true
}
```

ตัวอย่างที่ 4 Trigger กับ UI

โปรแกรมตัวอย่างของเรากำหนดให้มี trigger สองตัวที่ทำหน้าที่ในการปรับเปลี่ยนค่าอุณหภูมิระหว่าง Celsius และ Fahrenheit ซึ่งในการออกแบบนั้นเราจะให้มีการคำนวณหาค่าใหม่ทุกครั้งที่ค่าของอุณหภูมิมีการเปลี่ยนแปลง (ไม่ว่าจะเป็น Celsius หรือ Fahrenheit)

```
trigger on Temp.celsius = value {
    fahrenheit = (9/5 * celsius + 32);
}


trigger on Temp.fahrenheit = value {
    celsius = (5/9 * (fahrenheit - 32));
}
```

Trigger ที่เห็นทั้งสองตัวหมายความว่า "ทุกๆ instance (object) ของ class Temp เมื่อตัวแปร celsius (หรือ fahrenheit) ได้รับการกำหนดค่าใหม่ ให้ทำการประมวลผลประโยคที่อยู่ด้านในด้วยค่าใหม่ที่เกิดขึ้น" ด้วยเหตุนี้เอง เมื่อค่าของ celsius ถูกเปลี่ยนจาก Spinner ประโยค fahrenheit = (9/5 * celsius + 32) จึงถูกประมวลผล เช่นเดียวกันเมื่อค่าของ fahrenheit ถูกเปลี่ยนจาก Spinner ประโยค celsius = (5/9 * (fahrenheit - 32)) ก็ถูกประมวลผล ผู้อ่านอาจสงสัยว่าตัวแปร value ที่อยู่ด้านหลังเครื่องหมาย '=' นั้นมีความเป็นมาอย่างไร คำตอบง่ายๆ ก็คือ value ตัวนี้เป็นตัวแปรที่เก็บค่าใหม่ที่เกิดขึ้น (temporary variable) ซึ่งจะถูกนำไปใช้ในการคำนวณของประโยคที่อยู่ภายใน (เราสามารถกำหนดชื่อตัวแปรตัวนี้เป็นอะไรก็ได้) เช่น ถ้าค่าของ Temp.celsius ถูกเปลี่ยนค่าใหม่จะถูกเก็บไว้ใน value และจะถูกนำไปแทนที่ค่าของตัวแปร celsius เพื่อใช้ในการคำนวณหาค่าของ fahrenheit

และก็แน่นอนว่าค่าต่างๆ ที่เกิดขึ้นหลังจากการประมวลผลก็จะถูกนำไปแสดงใน Spinner ผ่านกระบวนการ bind ของตัวแปรทั้งสองกับตัว Spinner ดังนี้

```
Spinner {
    min: -60 max: 100 value: bind temp.celsius
},
Spinner {
    min: -76 max: 212 value: bind temp.fahrenheit
}
```

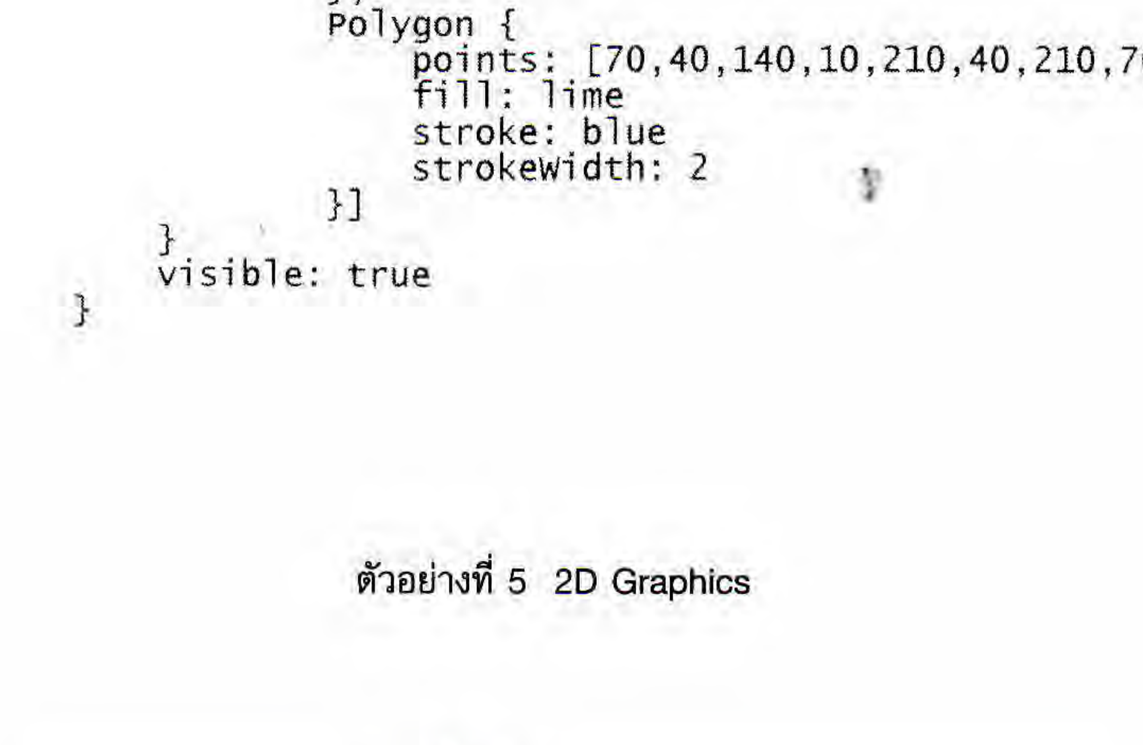
Trigger เป็นโครงสร้างตัวหนึ่งของ JavaFX ที่เอื้อให้การเขียนโปรแกรมของเราสนองตอบต่อเหตุการณ์ (event) ได้ง่ายขึ้น ทั้งยังสามารถนำมาใช้เป็นตัวกำหนดค่าเบื้องต้นให้กับ object ที่เราสร้างขึ้นได้เป็นอย่างดี

ภาพสองมิติ (2D Graphics)

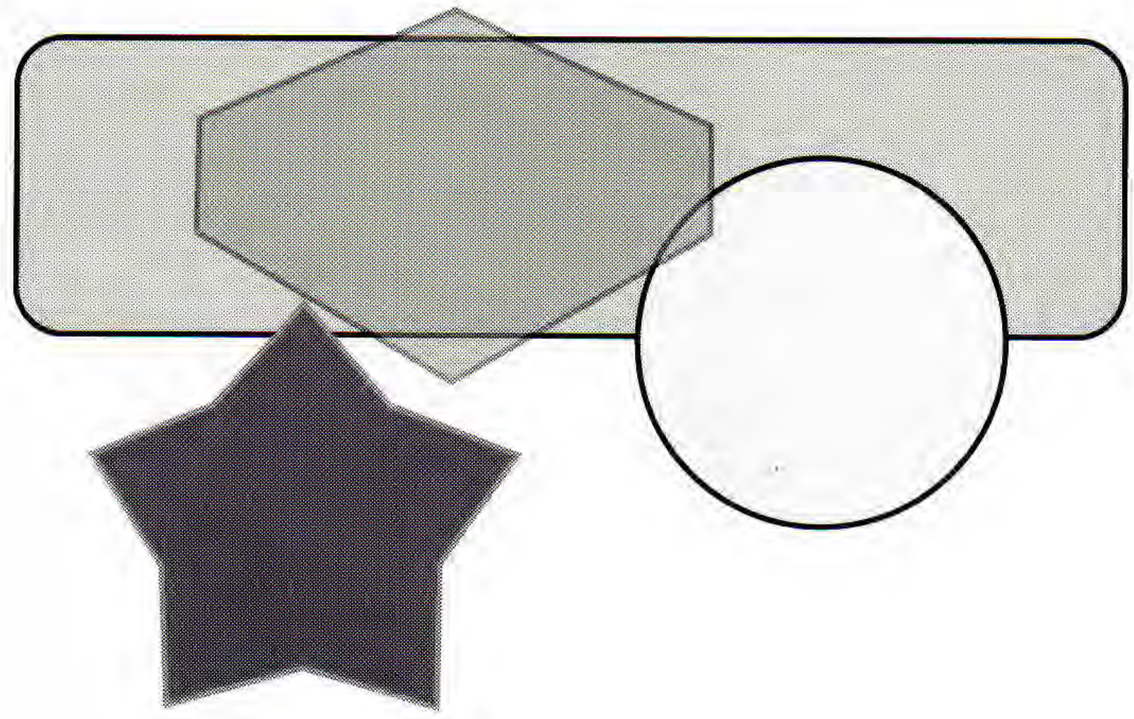
ดังที่ได้กล่าวไว้แล้วว่า JavaFX ช่วยให้การสร้างภาพสองมิติง่ายขึ้นเราจึงนำเอาวิธีการสร้างภาพสองมิติอย่างง่ายๆ มาแสดงให้ดู

```
Frame {
    title: "JavaFX 2D demo"
    height: 300
    width: 350
    content:
        Canvas {
            content: [Rect {
                x: 20
                y: 20
                height: 80
                width: 300
                arcHeight: 20
                arcWidth: 20
                fill: cyan
                stroke: purple
                strokeWidth: 2
            },
            Circle {
                cx: 200
                cy: 100
                radius: 50
                fill: yellow
                stroke: green
                strokeWidth: 2
            },
            Star {
                cx: 100
                cy: 150
                rin: 40
                rout: 60
                points: 5
                startAngle: 18
                stroke: green
                strokeWidth: 2
                fill: red
            },
                Polygon {
                    points: [70,40,140,10,210,40,210,70,140,110,70,70]
                    fill: lime
                    stroke: blue
                    strokewidth: 2
                }]
        }
        visible: true
}
```

ตัวอย่างที่ 5 2D Graphics



จาก code ที่เห็นเรากำหนดให้ Canvas เป็นที่เก็บ object สี่ตัวคือ Rect, Circle, Star, และ Polygon โดยเราจะทำการสร้าง Rect ก่อนด้วยจุดเริ่มต้นที่ (20, 20) มีความยาวเท่ากับ 300 และความสูงเท่ากับ 80 พร้อมทั้งกำหนดให้มุมทั้งสี่เป็นมุมโค้ง เส้นกรอบเป็นสีม่วงและพื้นเป็นสีฟ้า ต่อมาเราก็สร้าง Star ที่ (100, 150) ที่มียอดอยู่ห้ายอด และสุดท้ายเราสร้าง Circle ที่ (200, 100) มีรัศมีเท่ากับ 50 สีพื้นเป็นสีเหลือง และเส้นกรอบเป็นสีเขียว หลังจากนั้นเราสร้าง Polygon ด้วยจุดที่กำหนดไว้ด้วยสีพื้นที่เป็นสีเขียว มีเส้นกรอบเป็นสีน้ำเงิน ขนาดของเส้นเป็น 2



Java FX ยังเอื้อให้การ transform เป็นไปได้ง่ายขึ้น เช่นถ้าเราต้องการเคลื่อนย้ายรูปสองมิติที่เราสร้างขึ้นจากโปรแกรมตัวอย่างก่อนหน้านี้ เราก็สามารถทำได้ง่ายๆ เราจะแสดงการย้าย Circle ออกไปทางขวาของหน้าต่างอีก 40 pixels ด้วยการเพิ่มคำสั่ง transform: [translate(40, 0)] ให้กับ Circle พร้อมกันนี้เราก็ทำให้ Polygon มีความโปร่งแสง (Opacity) ด้วยการกำหนดค่า opacity ให้เป็น 0.3

```
Circle {
     transform: [translate(40, 0)]
     cx: 200
     cy: 100
     radius: 50
     fill: yellow
     stroke: green
     strokeWidth: 2
}

Polygon {
     opacity: 0.3
     points: [70,40,140,10,210,40,210,70,140,110,70,70]
     fill: lime
     stroke: blue
     strokeWidth: 2
}
```

มาถึงตอนนี้ผู้อ่านที่สนใจในเรื่องของการวาดภาพก็คงมีคำถามถึงเรื่องของการสร้างภาพเคลื่อนไหวใน JavaFX ใช่แล้วครับ เราสามารถสร้างภาพเคลื่อนไหวใน JavaFX ได้เช่นเดียวกันกับที่เราทำได้ใน Java

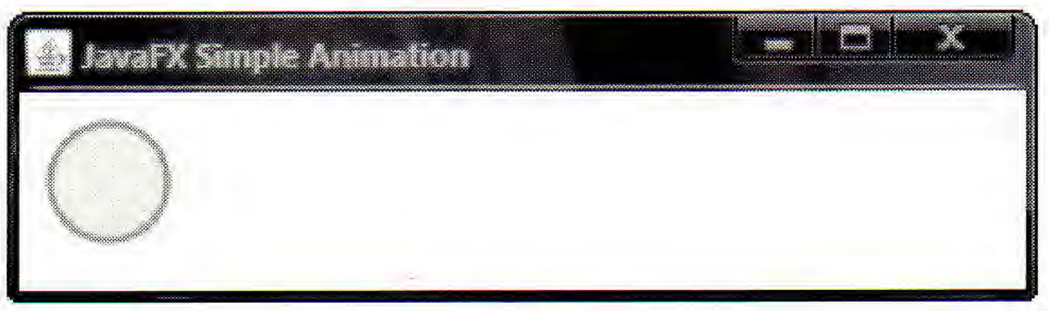
การสร้างภาพเคลื่อนไหวอย่างง่าย (Simple Animation)

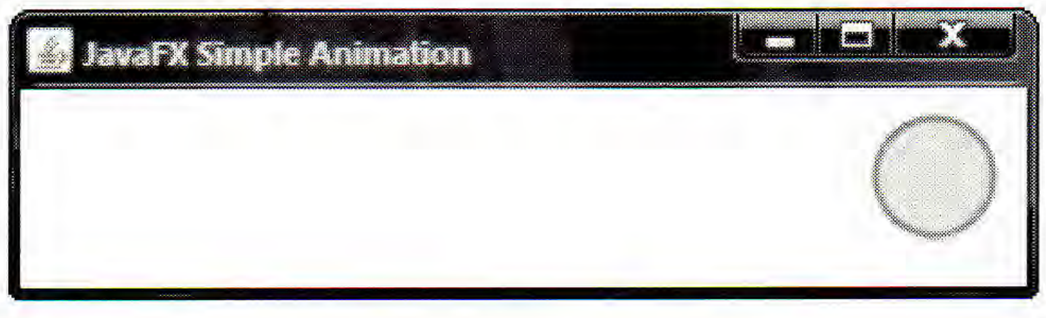
JavaFX มีคำสั่ง 'dur' (duration operator) ที่เราสามารถนำมาใช้ในการกำหนดช่วงเวลาผ่านทางอาร์เรย์สำหรับการสร้าง animation ได้อย่างง่ายๆ เช่น

var t = [0..100] dur 10 motion LINEAR;

ประโยคที่เห็นด้านบนนี้กำหนดให้ค่าของ t เปลี่ยนไปจาก 0 ถึง 100 (ทีละ 1 ค่า) ทุก ๆ 10 เสี้ยววินาที (millisecond) ส่วนคำว่า 'LINEAR' บอกถึงการปรับค่าแบบเส้นตรงหรือปรับแบบคงที่ (linear interpolation) ซึ่งในการปรับค่านั้น JavaFX มีค่าของ motion อยู่สี่แบบคือ easein, easeout, easeboth, และ linear เพื่อให้เห็นภาพชัดเจนขึ้นเราจะแสดงการสร้างภาพเคลื่อนไหวอย่างง่ายๆ ให้ผู้อ่านดู โดยเราจะใช้ 'dur' และการ translate เป็นองค์ประกอบหลักของการย้าย object 2D

```
Frame {
    title: "JavaFX Simple Animation"
    height: 100
    width: 350
    content:
        Canvas {
        content: Circle {
            cx: 30
            cy: 30
            radius: 20
            fill: yellow
            stroke: green
            strokeWidth: 2
            var x = 0
            transform: bind [translate(x, 0)]
            onMouseClicked: operation(e) {
                x = [0..280] dur 1000 motion LINEAR;
            }
        }
    }
    visible: true
    centerOnScreen: true
}
```





ตัวอย่างที่ 6 Translation

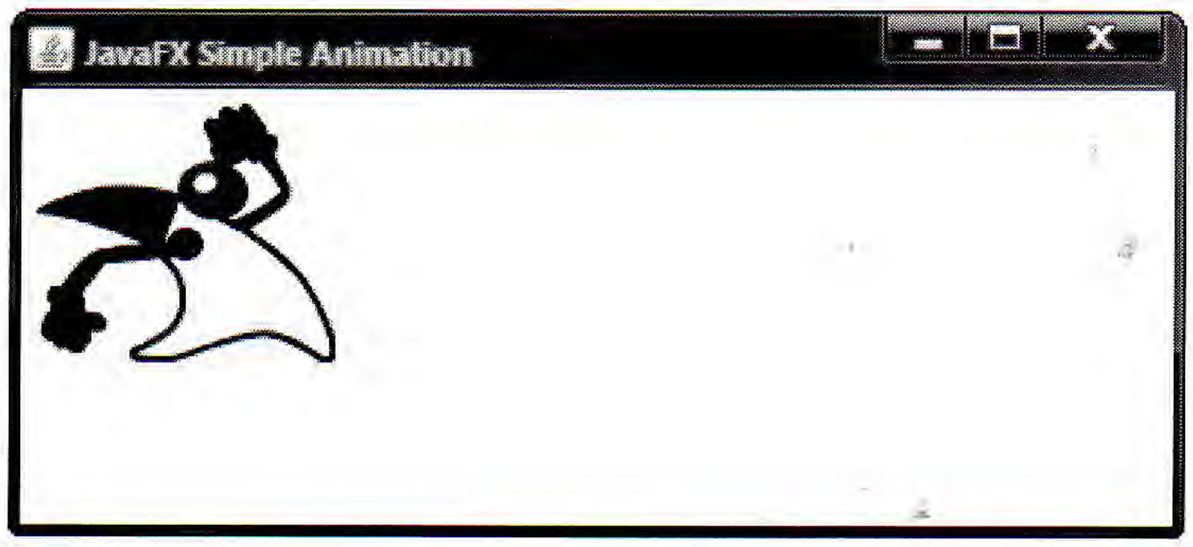
Code ที่อยู่ในกรอบกำหนดให้วงกลมที่เราสร้างขึ้นย้ายตำแหน่ง (transform) จากจุดเริ่มต้นไปทางขวาด้วยค่า x เริ่มต้นที่ 0 (ค่า y เป็น 0 ตลอด) และจะเปลี่ยนไปทุกๆ 1 วินาทีจนกว่าค่า x จะเป็น 280 ด้วยการเปลี่ยนค่าที่คงที่ (linear) หลังจากที่ผู้ใช้กดปุ่มบน mouse

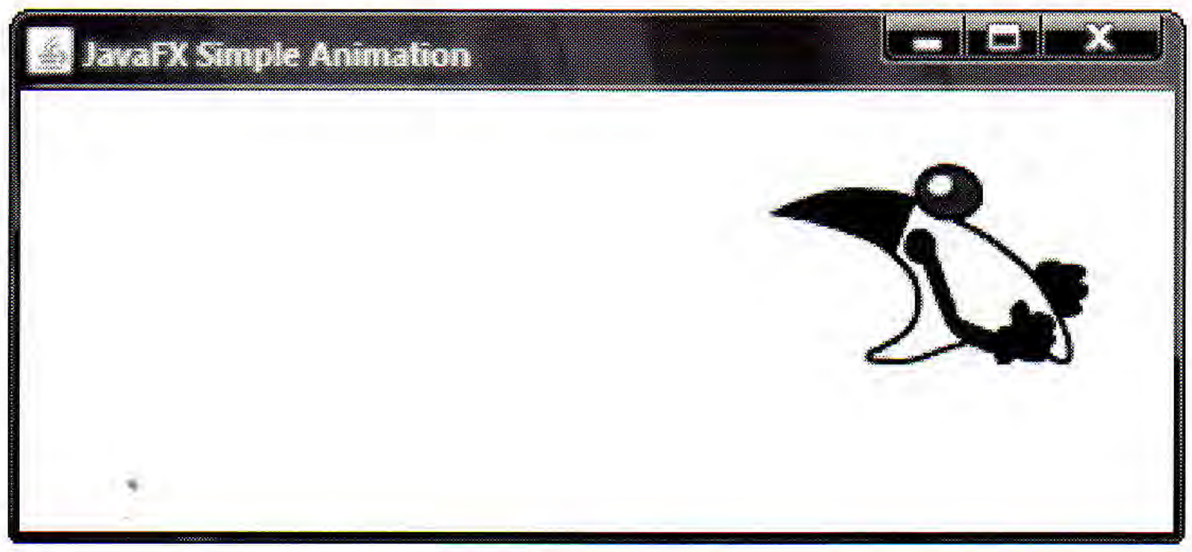
จากประสบการณ์ของผู้เขียนเองแล้วนี่เป็นการสร้างภาพเคลื่อนไหวที่ง่ายที่สุด (เมื่อเปรียบเทียบกับ Java) จากตัวอย่างการย้ายภาพก่อนหน้านี้เราย้ายด้วยการเรียก transform: [translate(40, 0)] ซึ่งเมื่อได้รับการ execute ภาพก็จะไปอยู่ในตำแหน่ง (40, 0) ทันที แต่จากตัวอย่างของการสร้างภาพเคลื่อนไหวเราทำการย้ายด้วยการเรียก transform: bind [translate(x, 0)] ซึ่งเป็นการย้ายภาพทีละค่าของ x จาก 0 ถึง 280 จึงทำให้เราเห็นการย้ายภาพแบบต่อเนื่องเสมือนหนึ่งว่าวงกลมเคลื่อนที่ไปอย่างช้าๆ ในหน้าต่างที่เราสร้างขึ้น ผู้อ่านอย่าลืมว่าถ้าเราไม่ใช้ bind กับการ translate เราก็จะไม่เห็นการเคลื่อนไหวของวงกลม ทั้งนี้ก็เพราะว่าการ transform เกิดจากค่าใหม่ที่มาจากการ bind เข้ากับการ translate ด้วยค่า x ที่เปลี่ยนไปทุกๆ 1 วินาที

เราสามารถที่จะใช้ image แทนภาพ 2D ได้อย่างง่ายๆ ขั้นตอนก็ไม่ยุ่งยากเท่าใดนัก เราดัดแปลงโปรแกรมก่อนหน้านี้ให้รองรับภาพที่เก็บในรูปแบบของ gif ไฟล์ด้วยการใช้โครงสร้างของ JavaFX ที่เรียกว่า ImageView ซึ่งภายในตัว ImageView จะมี attribute: image ที่เป็นตัวบ่งบอกถึงภาพที่จะต้องแสดงออกทาง Canvas

```
Frame {
    title: "JavaFX Simple Animation"
    height: 180
    width: 400
    content:
        Canvas {
        content: Group {
            content: ImageView {
                var x = 0
                transform: bind [translate(x, 0)]
                image: Image {url: "{__DIR__}/duke.running.gif"}
                onMouseClicked: operation(e) {
                    x = [1..250] dur 1000 motion LINEAR;
                }
            }
        }
    }
    visible: true
    centerOnScreen: true
}
```

ตัวอย่างที่ 7 Image และ Translation





เราเปลี่ยน content ของ Canvas ให้เป็น ImageView เพื่อรองรับการนำเข้า image ที่เราต้องการผ่านทาง url การเคลื่อนที่ของ image จะเกิดขึ้นหลังจากที่เรากดปุ่มบน mouse (เช่นที่เราทำกับโปรแกรมก่อนหน้านี้) ถึงแม้ว่าเราไม่สามารถแสดงการเคลื่อนไหวให้ดูได้ แต่ผู้อ่านก็สามารถนำ code ที่แสดงให้ดูไปทำการทดลองเพื่อให้เห็นถึงผลลัพธ์แท้จริงที่เกิดขึ้น

สรุป

JavaFX เป็นเครื่องมือตัวใหม่ที่นักพัฒนาโปรแกรมด้วยภาษา Java สามารถนำมาใช้เพื่อเอื้ออำนวยให้การสร้าง GUI ด้วย Swing และ 2D ทำได้ง่ายขึ้น รูปแบบการสร้าง UI ดูแล้วเข้าใจง่าย กระชับและบอกถึงความเป็นอันหนึ่งอันเดียวกันของตัว code และตัว UI ที่สร้างขึ้น การสนองตอบต่อเหตุการณ์ก็ไม่ซับซ้อน เพราะมีโครงสร้างที่เรียกใช้ได้ง่ายเป็นตัวช่วย ภาพเคลื่อนไหวที่ต้องเขียนด้วย code ที่ดูแล้วยุ่งยากและซับซ้อนใน Swing ก็ทำได้ง่ายขึ้นใน JavaFX

ถึงแม้ว่าบทความนี้เป็นแค่การแนะนำ JavaFX ให้กับผู้อ่านประกอบกับ (1) ตัวอย่างที่เราได้นำเสนอเป็นเพียงตัวอย่างเล็กๆ ที่เราสามารถทำได้ด้วย JavaFX (2) ยังไม่มีข้อสรุปที่ชัดเจนว่า JavaFX ทำงานได้ดีกว่า Swing และ (3) ด้วยข้อจำกัดของเนื้อที่เราจึงไม่สามารถที่จะนำเอาองค์ประกอบทั้งหมดของ JavaFX มาแสดงให้ดูได้ ก็เพียงแต่หวังว่าบทความนี้คงจะจุดประกายให้ผู้อ่านหลายๆ ท่านให้ทำการศึกษาค้นคว้าถึงความเป็นไปได้ถึงการนำเอาเทคโนโลยีของ JavaFX มาใช้พัฒนา UI ต่อไป

เครื่องมือที่ใช้ในการทดสอบ JavaFX

1. JavaFX PAD เป็นโปรแกรมที่เขียนขึ้นด้วย JavaFX เพื่อใช้ศึกษาถึงองค์ประกอบต่างๆ ของ JavaFX ซึ่งจะแสดงผลทันทีที่มีการเปลี่ยนแปลงเกิดขึ้นภายในโปรแกรมทันที ในความเห็นของผู้เขียนเห็นว่าเป็นเครื่องมือที่เหมาะสมสำหรับการเริ่มต้นเรียน JavaFX

2. JavaFX 2D Tutorial - โปรแกรมสอนการสร้างภาพสองมิติและการปรับเปลี่ยนภาพ เช่น การ transform และการสร้างภาพเคลื่อนไหว

3. Netbeans 6.0 Beta และ JavaFX Plugin เครื่องมือตัวนี้เป็นเครื่องมือที่สูงขึ้นมาอีกระดับหนึ่ง สามารถใช้ในพัฒนาโปรแกรม (ในภาษา) อื่นๆ ได้ด้วย ผู้อ่านสามารถ download ได้ที่ http://www.netbeans.org/community/releases/60/index.html ส่วน JavaFX Plugin นั้นอยู่ที่ http://javafx.netbeans.org/

เอกสารอ้างอิง

เนรมิต ชุมสาย ณ อยุธยา. (2006) **เริ่มต้นกับ Java.** ค้นเมื่อ 7 พฤศจิกายน 2550, จาก: http://sci.feu.ac.th/faa/Java.intro/introToJava.html

Declarative programming. Retrieved October 31, 2007, from: http://en.wikipedia.org/wiki/Declarative__programming

Getting started with the JavaFX Script Language (for Swing Programmers). Retrieved October 5, 2007, from: https://openjfx.dev.java.net/Getting__ Started__With__JavaFX.html

The JavaFX Script Programming Language. Retrieved October 5, 2007, from: https://openjfx.dev.java.net/JavaFX__Programming__Language.html

JavaFX 2D Graphics Tutorial - Retrieved October 31, 2007, from: https://openjfx.dev.java.net/#tutorials

James Weaver's JavaFX Blog - Retrieved October 31, 2007, from: http://learningjavafx.typepad.com/weblog/2007/10/develop-and-run.html

Learning JavaFX Script Series Part 1. Retrieved October 8, 2007, from: http://java.sun.com/developer/technicalArticles/scripting/javafxpart1/

Planet JFX. Retrieved October 5, 2007, from: http://jfx.wikia.com/wiki/Main__Page

อื่นๆ

ไฟล์ duke.running.gif จาก web site ของบริษัท Sun Micro Systems