Desktop

JavaFX was hét grote nieuws tijdens de afgelopen JavaOne. Sun zet hiermee in op een platform dat gezien kan worden als een concurrent van Adobe Flash, Adobe Flex en Microsoft Silverlight. In dit artikel kijken we naar de mogelijkheden van de scripttaal en naar de verschillen met Java.

JavaFX Script voor Java developers

Floris Ouwendijk
werkt voor NCIM bij verschillende klanten waar hij voornamelijk aan Java-desktopapplicaties werkt.
f.ouwendijk@ncim.nl

Om dit inhoud te geven moet het platform draaien op het spectrum van mobiele devices tot PCs met hun variëteit aan besturingssystemen en zowel binnen de browser als ook als desktopapplicatie beschikbaar zijn. Ter vergelijking: Silverlight richt zich op Windows en OSX, de community werkt daarbij aan een versie voor Linux, en de content is voornamelijk gericht om binnen een browser te draaien. De tooling van Adobe is beschikbaar op een scala van platforms en biedt oplossingen voor binnen de browser als op de desktop. Om de portabiliteit van JavaFX te realiseren wordt Java gebruikt als onderliggend platform. Om de toepassing van de grafische capaciteiten van Java beter bruikbaar te maken introduceert men JavaFX script.

JavaFX script

Chris Oliver, werkzaam bij Sun, zag enkele bezwaren aan het ontwikkelen van user interfaces in Java: het kan makkelijker en het resultaat kan mooier.
Daartoe werkte hij aan F3 (Form Follows Function), een scripttaal met als doel meer uit Swing en Java2D te kunnen halen. Inmiddels is F3 omgedoopt naar JavaFX script.
JavaFX script wordt door Chris Oliver omschreven als "een declaratieve scripttaal met static typing voor goede IDE support en compile-time error controle (in tegenstelling tot JavaScript), type-afleiding, declaratieve syntax, automatische data-binding met volledige support voor 2d graphics en de standaard Swing componenten en declaratieve animatie. Daarnaast kun je Java classes importeren, Java objecten maken en hun methodes aanroepen en Java interfaces implementeren."

In het vervolg van dit artikel kijken we naar verschillende aspecten uit de taal.

Operators

JavaFX script kent een uitgebreidere set operators dan Java. De standaard operators voor vergelijken, Booleaanse bewerkingen en wiskundige operaties zijn:
`==, <>, <, >, <=, >=, and, or, not, +, -, *, /, %, +=, -=, *=, /= en %=`
Op een paar uitzonderingen na zijn deze identiek aan hun Java-equivalent. De uitzonderingen zijn <> en de Booleaanse bewerkingen.
Daarnaast komen de volgende operators overeen: new, . (punt voor member operaties en attributen), instanceof, this, [] (array access), () (grouping).
JavaFX voegt operators toe die het werken met lijsten eenvoudiger maken, te maken hebben met de declaratieve syntax, binding en threading. Deze zullen we verderop nader bekijken.

Types

De taal kent vier basis types: String, Boolean, Number en Integer. De betekenis van deze types ligt voor de hand en conversie van en naar Java types gaat automatisch. Een verschil met Java is wel dat deze types zich als objecten gedragen en ook enkele methodes kennen. Integers worden in de specificatie vergeleken met alle primitieve integer types in Java, maar ook met BigIntegers. Toch lijkt de huidige limiet op het formaat van een long te liggen, waarbij de operaties echter er wel voor zorgen dat dat er geen overflow optreedt (de waarde blijft dan op het maximum liggen).
Strings literals kunnen met een enkele quote of een dubbele quote gedefinieerd worden. In het

geval dat het met een dubbele quote wordt gedefinieerd, dan worden de delen binnen accolades ({}) geparsed als JavaFX script. Ook kunnen ze in het geval van dubbele quotes newlines bevatten.

Een variabele wordt als volgt gedefinieerd:

```
var variableNaam : optioneelType = optioneleWaarde;
```

Bijvoorbeeld:

```
var answer = true;
var s = "The answer is {if answer then "Yes" else
"No"}";
```

In beide gevallen wordt het type van de variabele automatisch afgeleid,wat betekent dat er compiler fouten getoond zullen worden wanneer je bijvoorbeeld probeert te rekenen met bovenstaande variabelen, ze meegeeft aan een operatie die een ander type verwacht, of methodes op het object probeert aan te roepen die het type niet kent.
Achter het type kan ook nog een extra operator geplaatst worden die de kardinaliteit van het type bepaalt: '?' Indien er null toegekend mag worden aan de variabele (let op, niet geïnitialiseerd heeft die toch als waarde 'null'), '+' indien het een lijst moet zijn met minstens een waarde en '*' voor lijsten met nul of meer waarden.

Lijsten

JavaFX script kent geen directe support voor de Java Collections classes, maar heeft wel zeer sterke support voor een eigen type lijst. De volgende code geeft wat mogelijkheden:

```
var x = [1,2,3];
insert 4 into x; // geeft [1,2,3,4]
insert 5 as first into x; // geeft [5,1,2,3,4]
insert [6,7] as last into x; // geeft
[5,1,2,3,4,6,7]
sizeof x; //geeft 7
var y = reverse x; //geeft [7,6,4,3,2,1,5]
select n*n from n in [1..5] where n%2==1 // geeft
[1,9,25]
select n*n from n in [5,3..1] // geeft [25,9,1]
```

Lijsten zijn slechts ééndimensionaal: een lijst in een lijst zorgt voor een langere lijst.

Statements

If en while statements zijn gelijk aan die in Java, maar de accolades zijn verplicht. Throw en try/catch/finally zijn ook vergelijkbaar: het type in het catch blok wordt echter wel met JavaFX syntax (naam:type) gespecificeerd. Daarnaast kan ieder object gethrowd en gecatched worden (niet alleen throwables).
Het for statement kent de grootste afwijking omdat het op lijsten gebaseerd is:

```
for (i in [0..10]) { println(i); } //print 11
regels
for (i in [0..10] where i % 2 == 0) { println(i); }
//print 6 regels
for (i in [0..10], j in [0..10]) { println(i+j); }
//print 121 regels
```

Verder zijn er het standaard return statement, break en continue (deze laatsten echter zonder labels).

Operaties en functies

JavaFX script kent operaties en functies. Operaties zijn als Java methoden en kennen weinig beperkingen qua inhoud. Functies moeten gezien worden als wiskundige functies: ze kunnen alleen variabele declaraties en een return statement bevatten. Tegenover de nadelen van het niet kunnen gebruiken van veel gebruikelijke constructies binnen een functie staan de potentiële voordelen dat functies beter te paralleliseren zijn en goed samenwerken met de bind operator (waarover later meer).
Functies en operaties zijn ook als object toe te kennen aan een variabele of mee te geven aan een operatie:

```
var add1 = function(a:Number) { return a+1; };

var printOperation = operation(op:function(a:Number)
, n:Number) {
    println(op(n));
};

printOperation(add1, 5);
```

Let in het voorbeeld op de attribuut definities: overal is aangegeven dat het type Number is. Als we het type weglaten verandert er niets, behalve dat de type-safety niet meer compile-time gecontroleerd kan worden, met potentiële runtime problemen als gevolg.

Classes

De voorgaande constructies zijn te gebruiken als elementen in een script, zonder classes te creëren. Een aantal andere constructies zijn alleen toepasbaar op attributen, die onderdeel uitmaken van classes.
Classes kennen een wat andere syntax dan Java classes. Het volgende voorbeeld illustreert dit:

```
class TestClass {
    attribute someAttribute: Number?;
    operation m(a:Boolean):String;
}
attribute TestClass.someAttribute = 4;
operation TestClass.m(a) {
  if (a) {
        return "yes";
  } else {
        return "no";
  }
}
```

De class bevat definities voor attributen en operaties, maar de invulling daarvan vind daarbuiten

plaats, waarbij een verwijzing naar de class in de methode naam is opgenomen. In dit voorbeeld zijn bij het invullen van de methode de types weggelaten. Het is zelfs mogelijk om in de definitie het type weg te laten (zelfs het return type!), of het bij de definitie weg te laten en bij de implementatie juist te specificeren. Om het nog gekker te maken is het ook mogelijk om de operatie definitie weg te laten als de methode alleen binnen de class zelf gebruikt wordt.

Classes ondersteunen ook inheritance. Net als in Java kan met 'extends' aangegeven worden wat de superclass is. Een grote wijziging is echter dat er een komma gescheiden lijst van classes opgegeven kan worden: jawel, JavaFX ondersteunt multiple-inheritance! Indien een operatie in meerdere superclasses voorkomt wordt voorrang gegeven aan de class die als eerste genoemd is.
Interfaces (en het 'implements' keyword) worden niet ondersteund, evenals method overriding: een methode met andere argumenten wordt niet gezien als een andere methode en levert een compileer fout op.

Constructors kent de taal niet als je afgaat op de nu beschikbare specificatie. In plaats daarvan is een trigger beschikbaar (zie volgende sectie). Maar qua implementatie kun je WEL een methode gebruiken met dezelfde naam als de class (en optioneel enkele argumenten), die aangeroepen wordt indien je een object maakt. Constructors met meerdere argumenten worden echter niet uitgevoerd wanneer je een object maakt op basis van de declaratieve syntax en het creëren van verschillende constructors is niet mogelijk omdat overriding niet is toegestaan.
Declaratieve constructie syntax zal veel developers aanspreken. Vergelijk de volgende constructies:

```
var a = new TestClass();
a.someAttribute = 5;

var b = TestClass {
    someAttribute: 5
};
```

De resultaten zijn hetzelfde, maar de intentie komt duidelijker over in het tweede voorbeeld. Dit werkt nog sterker wanneer hiërarchieën van objecten gemaakt worden, zoals in user interfaces gebeurt:

```
Frame {
    width: 400, height: 300, visible: true
    content: BorderPanel {
        top: Label {
            text: "Hi there"
            foreground: red
        }
        center: TextArea {}
    }
}
```

Triggers

Triggers zijn het antwoord van JavaFX op zowel listeners als setters. De taal kent de volgende varianten:

```
trigger on new MyClass { doSomething(); }
trigger on insert val into MyClass.aListAttribute {
doSomething(val); }
trigger on delete val from MyClass.aListAttribute {
doSomething(val); }
trigger  on  MyClass.aAttribute[oldVal]  =  newVal  {
doSomething(oldVal, newVal); }
```

Alle varianten worden uitgevoerd binnen de context van de instantie van de class die de trigger veroorzaakt. Het is ook alleen mogelijk om triggers te plaatsen op attributen van een class, niet op variabelen. In de laatste variant is het deel "[oldVal]" optioneel. Daarnaast kun je in de insert en delete triggers, maar ook de replace trigger indien toegepast op een lijst, met 'indexof val' de index bepalen van het toegevoegde/verwijderde/aangepaste element binnen de lijst.
Een beperking bij het gebruik van triggers als setter, is dat je er voor moet zorgen niet (indirect) de waarde van het attribuut weer aan te passen, gezien dit, zonder verdere maatregels, tot een StackOverflowError leidt. Het voornaamste gebruik is dan ook als een listener in de situaties waar bind geen oplossing biedt of niet toepasbaar is.

Bind

Bind is een van de krachtigste constructies uit de taal en ook de constructie waar het meeste mis kan gaan zonder dat de compiler je helpt. Met bind kun je een expressie binden aan een attribuut, waarbij wijzigingen in waarden in de expressie leiden tot een wijziging in de waarde van het attribuut en soms ook omgekeerd.
Het definiëren van een binding moet dan ook plaatsvinden als deel van de class-declaratie of in de object literal.
Enkele voorbeelden:

```
class TestClass {
    attribute att1:Integer;
    attribute att2:Integer;
}
attribute TestClass.att1 = bind att2+3;
var a = TestClass{
    att1: 5
};
println("{a.att1} {a.att2}"); //print 5 2
a.att1 = 7;
println("{a.att1} {a.att2}"); //print 7 4
a.att2 = 9;
println("{a.att1} {a.att2}"); //print 12 9


class Model {
    attribute name: String;
}
var m = new Model();
Frame {
    width: 400, height: 300, visible: true
    content: BorderPanel {
        top: Label { text: bind "Hi {m.name}" }
```

Jawel, JavaFX ondersteunt multiple inheritance!

```
        center: TextField { text: bind m.name }
    }
}
```

Het eerste voorbeeld zorgt ervoor dat de twee attributen altijd een verschil van drie houden. Het tweede voorbeeld toont hoe het label altijd overeenkomt met de in het textfield getikte tekst.
In de bind expressie kunnen ook operaties en functies gebruikt worden. Hier treedt een belangrijk verschil op tussen de twee: wijzigingen in waarden die binnen functies gebruikt worden zorgen wel voor een update van de expressie, terwijl dit voor operaties niet geldt.

Threads/animatie

Support voor threads en animatie is vanuit de taal ondersteund met enkele operaties. Het is echter nog niet duidelijk of de huidige syntax definitief is.
Het uitvoeren van een blok code op een nieuwe thread gaat eenvoudig door dit binnen een "do {}" blok te plaatsen. Dit zorgt ervoor dat de huidige thread gepauzeerd wordt totdat de code binnen de accolades is uitgevoerd. Als de huidige thread een UI thread is, dan betekent dat dat er een nieuwe UI thread gecreëerd wordt die er voor zorgt dat de applicatie blijft reageren. Het elegante in deze constructie is dat excepties doorgegeven worden aan de originele thread. Daarmee worden constructies om excepties door te geven van thread naar thread overbodig. Daarnaast is de code ook goed leesbaar omdat de code na het 'do' blok pas uitgevoerd wordt nadat het blok verlaten wordt.
Animatie is een punt dat sterk naar voren geschoven wordt, maar waarvan de syntax en het gebruik erg slecht zijn gedefinieerd. Twee manieren zijn:

```
num = [1..10] dur 1000;
trigger on (i = [1..10] dur 100) {
    println(i);
}
```

Hierin in het 'dur' keyword verantwoordelijk voor de animatie, waarbij over een periode van 1 seconde opeenvolgende nummers aan de 'num' variabele worden toegekend. Om er iets zinnigs mee te doen kun je een attribuut van een class binden aan de variabele, denk aan een coördinaat of een omzetting van de waarde naar een kleur. Het voorbeeld op basis van de trigger is handig wanneer je een bepaald stuk code uit wil voeren. Let we op dat beide voorbeelden de huidige thread niet pauzeren. Ook zul je zien dat het tweede voorbeeld niet alle waarden print: over de periode van 100ms worden waarden genomen die kloppen bij het tijdstip waarop de timer ontwaakt.
De default van het 'dur' keyword is ook om meer tijd te geven aan de eerste en laatste elementen, zodat een fade-in, fade-out wordt bewerkstelligd. Voor een lineair effect kun je 'linear' achter 'dur' plaatsen, of zelfs je eigen functie om een bepaald effect te bereiken.

Standaard library

Dat JavaFX vooral gericht is op het zijn van een alternatief voor de constructie van een user interface blijkt het sterkst wanneer we kijken naar de meegeleverde classes: ze zijn allemaal UI gerelateerd!
Veel van de classes zijn wrappers om Swing componenten, met de nadruk om het gebruik eenvoudiger te maken. In de sectie over classes zagen we al het BorderPanel: dit is feitelijk een JPanel met BorderLayout, maar de toevoeging van de attributen 'top', 'left', 'right', 'bottom' en 'center' maken de hiërarchische opbouw mogelijk.
Naast de Swing gebaseerde classes is er een aantal classes om Java2D eenvoudiger te maken: er zijn classes voor vormen, operaties op vormen (een cirkel tekenen waaruit een vierkant is weggesneden), vulling van elementen met kleuren, gradiënten en patronen, en zelfs 'morphing', de transformatie van de ene vorm naar de andere, wordt ondersteund!

Java-integratie

De binding tussen Java en JavaFX script is vanzelfsprekend sterk. Vanuit JavaFX script kunnen instanties van Java classes gecreëerd en gebruikt worden. Daarbij zorgt de interpreter er zelfs voor dat getters en setters benaderbaar zijn alsof het attributen zijn. Helaas werkt bind (nog) niet samen met die attributen (ook niet wanneer JavaBean conventies en PropertyChangeEvents gebruikt worden).
Een JavaFX class kan ook achter 'extend' een Java class of interface opgeven, zonder dat de compiler klaagt. Het gebruik van functionaliteit uit die classes lijkt echter niet aanwezig: aanwezige functies worden herkend, maar geven een runtime exceptie of null indien ze gebruikt worden.
De meest eenvoudige combinatie van Java en JavaFX script is door het script verantwoordelijk te maken voor het initialiseren van de benodigde Java objecten en het geheel te starten door de JavaFX interpreter te starten met het script. Andersom is echter ook mogelijk: met JSR-223 (de script engine) kun je een JavaFX script engine initialiseren en die gebruiken om stukken script uit te voeren.

Vooruitblik

De basis van het platform is gelegd, maar het is nog niet af. Performance en een onvolledige en schuivende specificatie zijn enkele praktische zaken die JavaFX uit de productieomgeving houden.

Het OpenJFX compiler project werkt aan een JavaFX script naar bytecode compiler, zodat je scripts aanzienlijk sneller uitgevoerd kunnen worden. Dit zal zeker de gebruiksvriendelijkheid ten goede komen. Men staat echter pas aan de start van het project, waarbij wat eenvoudige taalconstructies omgezet kunnen worden, maar voor de complexere constructies die de taal zo bruikbaar maken zal nog wel enige tijd nodig zijn.
De geruchten gaan dat Sun werkt aan een op Netbeans gebaseerde tool voor designers, wat JavaFX binnen het bereik moet brengen van developers en designers die momenteel met tools van de concurrentie werken. De community zit echter ook niet stil en het bedrijf ReportMill heeft inmiddels al een JFXBuilder tool gemaakt.
Al met al is de verwachting dat in het eerste half jaar van 2008 meer details over de verschillende ontwikkelingen naar buiten zullen komen en dat de volgende JavaOne weer een sterke nadruk op JavaFX zal kennen.

Resources

https://openjfx.dev.java.net/ is de homepage van het OpenJFX project. Hier vind je de sources, een forum, en de issue tracker.
http://jfx.wikia.com/ Planet JFX wordt onderhouden door een community van early adopters. Hier staan ervaringen, voorbeelden en door de community gemaakte componenten.
http://blogs.sun.com/chrisoliver/ de weblog van Chris Oliver
http://www.reportmill.com/jfx/ pagina met informatie over JFXBuilder van ReportMill

Rags: Tools komen eraan

Net zoals Rich Green (zie redactioneel Java Magazine 2/2007) ziet Raghavan 'Rags' N. Srinivas, CTO Technology Evangelism bij Sun Microsystems, JavaFX als een noodzaak om Java op de client een nieuwe impuls te geven.
Rags: "Java is erg geconcentreerd geweest op de server kant, het mist zijn 'flash'. FX vult dat heel goed in. Er bestaat geen twijfel over dat enterprise nog steeds een heel belangrijk aspect van Java is, maar iedereen is ervan overtuigd dat JavaFX heel belangrijk moet worden."

Is FX nu op developers of op ontwerpers gericht?
Rags: "Contentdesigners zijn ook belangrijk. Met een flowbased tool waar je filters aan toe kunt voegen, kun je ze in het Java-kamp krijgen. Tegelijkertijd moet er onder iedere flashy website iets zitten, en dat is waar Java in beeld komt. Zo gauw Java developers dat zien zullen ze anders naar JavaFX gaan kijken. Simpel voorbeeld: probeer maar eens een menulijst te maken in Swing. Dat is heel erg proceduregebaseerd, het is omslachtig, niet intuïtief. In JavaFX is het heel simpel en alles wordt op een declaratie manier gedaan, op een manier die meteen duidelijk is. Als je dan naar Java kijkt lijkt het legacy – wat het niet is natuurlijk – maar op dit soort punten wel. "

Foto: Dré de Man

Bij Adobe hebben ze een heel sterke band met designers. Is het niet heel moeilijk voor JavaFX om bij die groep populair te worden?
Rags: "Als je mij vraagt of de content developers nu naar FX zulle overstappen, dan zeg ik nee. Ze zijn nog steeds tevreden met de tools die zie gebruiken, naarmate de tools in de JavaFX-wereld beter zullen worden, zullen er naar kijken gaan kijken en ze gaan gebruiken. Nu is het nog vooral geschikt voor ontwikkelaars die een betere user interface willen maken. We zulen heel spoedig ook voor contentauteurs goede tools aanbieden. Het doel is nu alles op alles te zetten om de fundamenten klaar te hebben. Daarna komen de tools. Je hebt nu al JFlex Builder, een heel simpel tool gemaakt door Report Mill. Het is een visueel tool dat drag en drop mogelijk maakt, en het bouwt alles visueel. Het is nog niet compleet maar het is er en op de achtergrond genereert het JavaFX voor je. JavaFX zal dus door de contentauteurs via dit soort tools gegenereerd worden en wordt dan uiteindelijk door Java-ontwikkelaars in webservices geïntegreerd. Die workflow is er nu nog niet, maar die zal er spoedig komen."

Tijdens JFall hield Angela M. Caicedo een sessie over Java ME en Ajax. Waar is FXmobile?
Rags: "Allereerst: Java ME gaat niet weg, Het zal de basis blijven voor FX Mobile, net zoals Java SE de basis is voor FX. ME is bezig de functionaliteit van SE te benaderen. Op een zeker punt zal de functionaliteit van FX die van SE overtreffen, dat is nu nog niet het geval. Maar om heel eerlijk te zijn: FX Mobile is nog steeds in de ontwerpfase. Ik kan niet er meer details over geven dat dit.

Wat zijn de sterke punten van Java FX?
Rags: "Schaalbaarheid, performance, portabiliteit. Dat zijn allemaal dingen waar we ervaring mee hebben, en dat werkt allemaal. Samenwerking met de Javagemeenschap is ook sterk punt. Als je dat allemaal bij elkaar optelt, vormt JavaFX een goed aanbod. Bovendien komen we ook met de consumer JRE (Java runtime engine). Als je een simpele Hello World applicatie wil uitvoeren, misschien geen Swing, waarom moet je dan 12 MB JRE downloaden? Je wil alleen maar de één of twee Mb die het werk doen. De rest kan dan op achtergrond gedownload worden voor een andere keer waarop je wel die Swing-applicatie wil draaien. Dat maak het gemakkelijker voor consumenten als mijn oma om al applicaties te gebruiken. Want alhoewel Java nu op 90% van de dekstop aanwezig is, is de gebruikerservaring niet goed. Dat zal veranderen, en dat zal uiteraard een grote impact hebben op Java FX. De consumer JRE is nu in bèta, de verwachting is dat hij begin 2008 gereleased wordt."