JavaFX

Thorbiörn Fritzon
Lead Technologist
Sun Nordic

So, What’s This?

```
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.image.*;

Stage {
    scene: Scene {
        content: ImageView {
            image: Image { url: "... /jfx.002.png" }
        }
    }
}
```

Presentations Should be Fullscreen

```
Stage {
    fullScreen: true
    scene: Scene {
        content: ImageView {
            image: Image { url: "... /jfx.003.png" }
        }
    }
}
```

Presentations Are Multi-Page

```
var images = for( i in [1..63] ) Image {
    url: "... /jfx.{%03d i}.png"
};
var idx= 0;

Stage {
    fullScreen: true
    scene: Scene {
        content: ImageView {
            image: bind images[idx]
            onKeyPressed: function(e:KeyEvent) {
                idx++;
            }
        }
    }
}
```

Presentations Have Transition Effects

```
var fr= ImageView { opacity: 0.0;
          image: bind images[idx-1] };
var fd=Timeline { keyFrames:[ at(0s) { fr.opacity => 1.0},
                              at(1s) { fr.opacity => 0.0}]};

Stage {
    fullScreen: true
    scene: Scene {
        content: [ ImageView {
            image: bind images[idx]
            onKeyPressed: function(e:KeyEvent) {
                idx++;
                fd.playFromStart();
            }
        }, fr ]
    }
}
```

In Short

- Rich Internet Application Environment
- Declarative Programming Language for GUIs
- Scene-graph based Presentation Model:
  - Stage/Scene
  - Nodes
  - Effects/Transforms
  - Timelines/Transitions
- Three Execution Environments/Profiles:
  - JavaFX Desktop
  - JavaFX Mobile
  - JavaFX TV

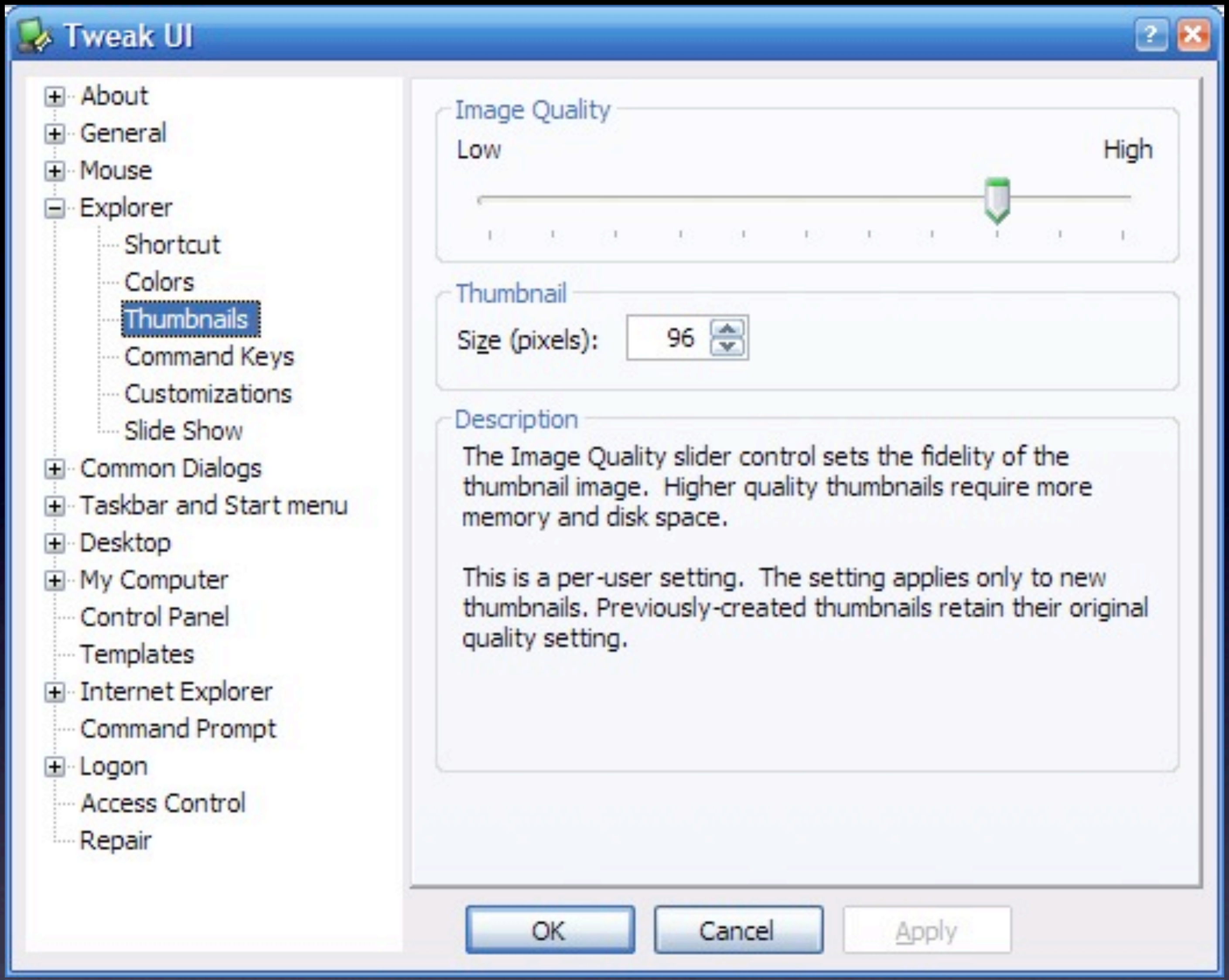
“Now, Why on Earth Would We Want This?”

from:
Filthy Rich Clients

by Romain Guy

“Filthy Rich Clients [...] are so graphically rich that they ooze cool, they suck the user in from the outset and hang onto the user with a death grip of excitement, they make the user tell their friends about the applications.”

– Chet Haase, Sun Microsystems



Thumbnail

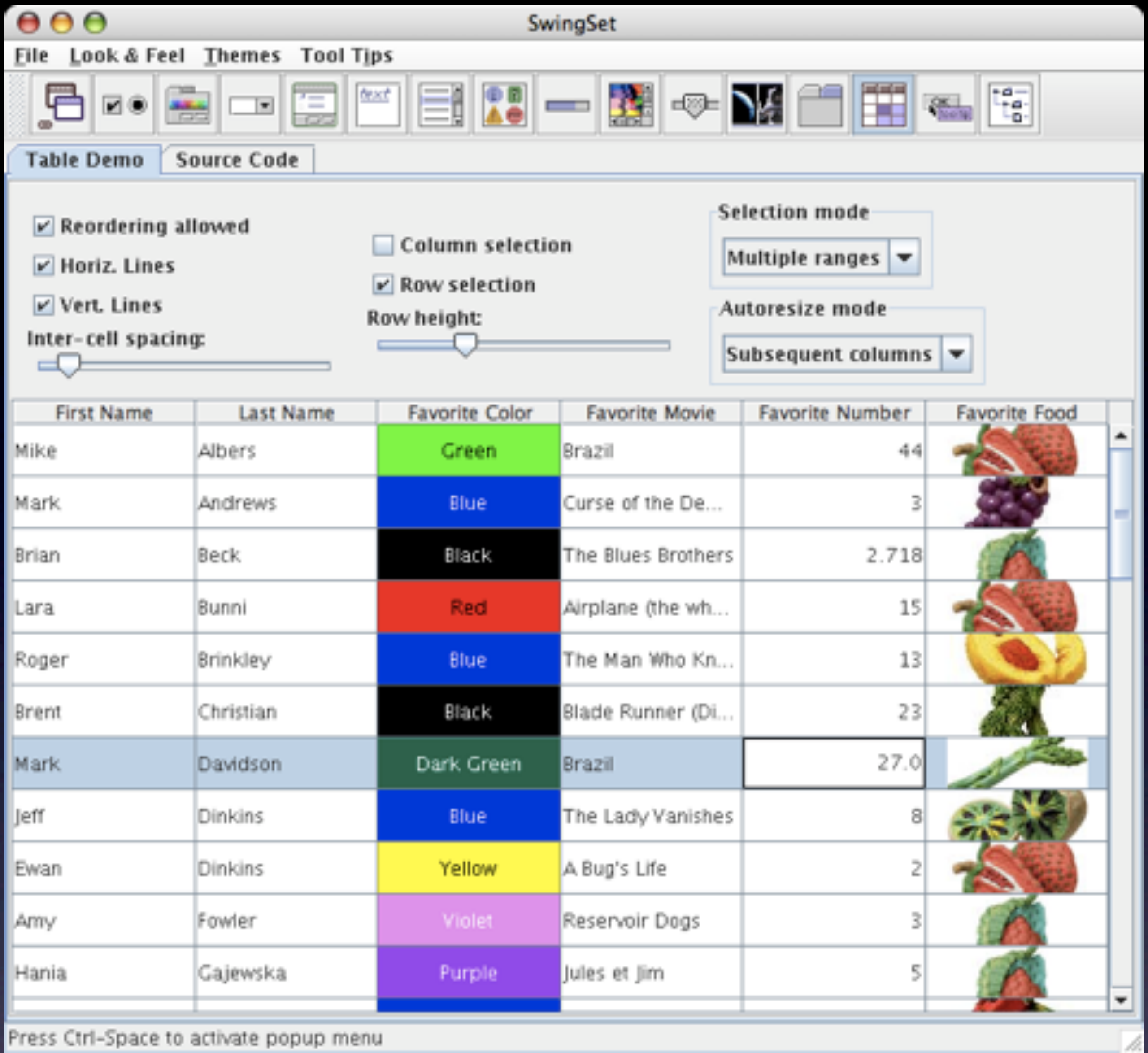
Size (pixels): 96

Description

The Image Quality slider control sets the fidelity of the thumbnail image. Higher quality thumbnails require more memory and disk space.

This is a per-user setting. The setting applies only to new thumbnails. Previously-created thumbnails retain their original quality setting.

What They Have



☑ Reordering allowed

☑ Horiz. Lines

☑ Vert. Lines

Inter-cell spacing:

□ Column selection

☑ Row selection

Row height:

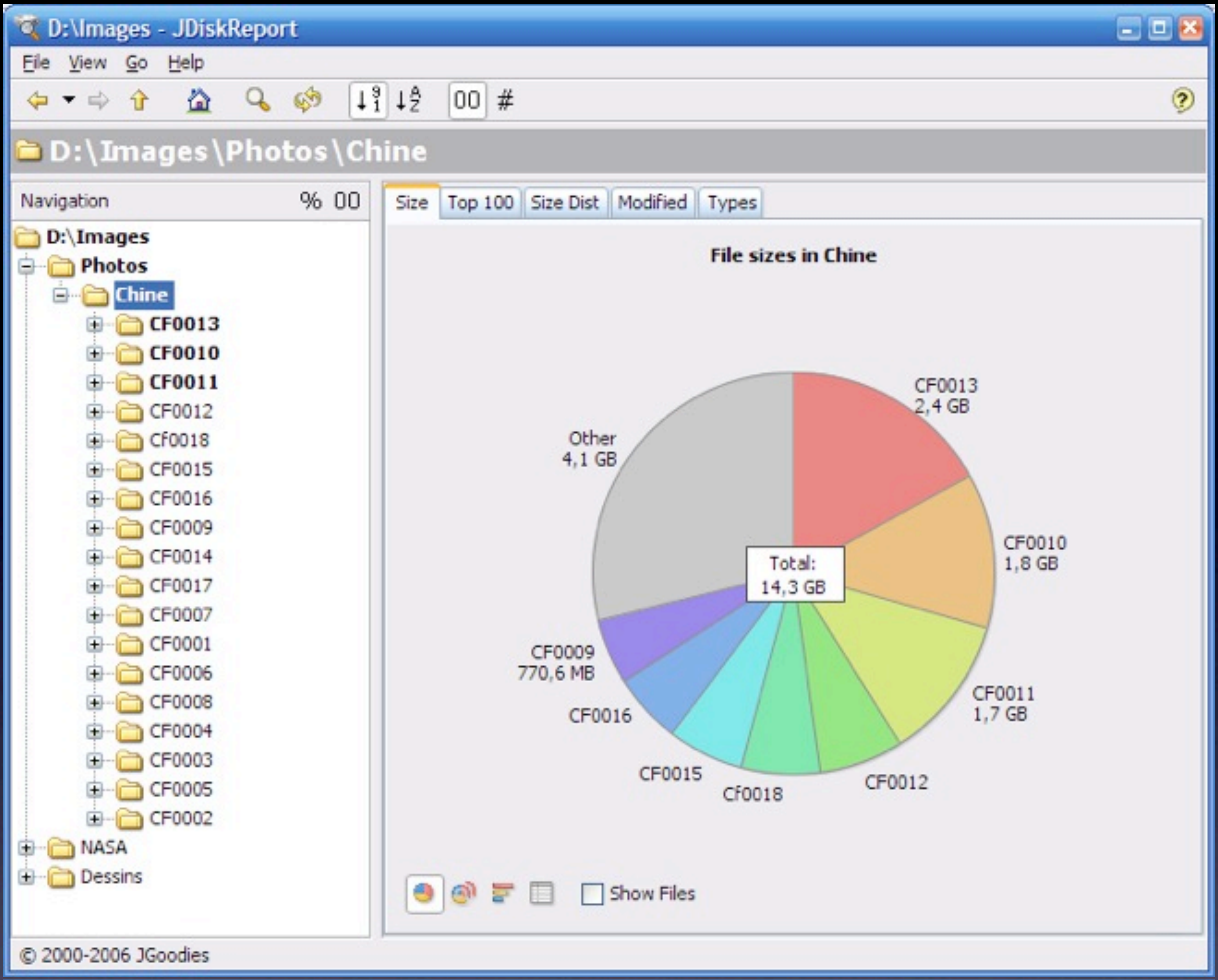
Selection mode: Multiple ranges

Autoresize mode: Subsequent columns

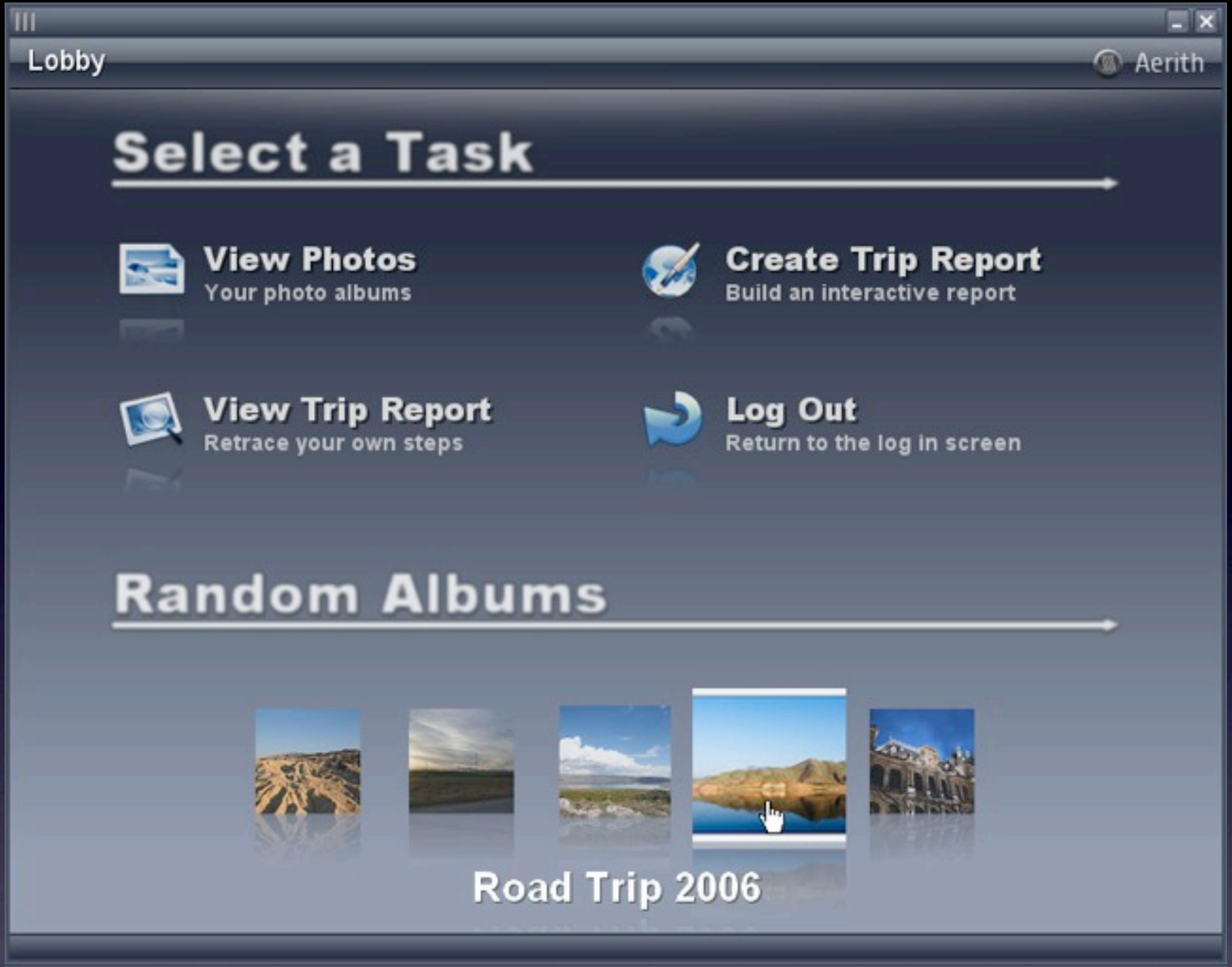
First Name	Last Name	Favorite Color	Favorite Movie	Favorite Number	Favorite Food
Mike	Albers	Green	Brazil	44	
Mark	Andrews	Blue	Curse of the De...	3	
Brian	Beck	Black	The Blues Brothers	2.718	
Lara	Bunni	Red	Airplane (the wh...	15	
Roger	Brinkley	Blue	The Man Who Kn...	13	
Brent	Christian	Black	Blade Runner (Di...	23	
Mark	Davidson	Dark Green	Brazil	27.0	
Jeff	Dinkins	Blue	The Lady Vanishes	8	
Ewan	Dinkins	Yellow	A Bug's Life	2	
Amy	Fowler	Violet	Reservoir Dogs	3	
Hania	Gajewska	Purple	Jules et Jim	5	

Press Ctrl-Space to activate popup menu

What We Have



What You Want



What You Will Have

Some Questions Pondered

- Why does it take a long time to write GUI programs?
- How can we avoid the “ugly Java technology GUI” stereotype?
- Why do Flash programs look different than Java platform programs?
- Why does it seem easier to write web apps than Swing programs?
- And how can we avoid having an enormous, writhing mass of listener patterns?

What Problem Does JavaFX Solve?

- Fundamentally: how can we make GUI development more efficient?
- GUI development is a collaboration between content designers, graphic artists, and programmers
- The main bottleneck in this process appears to be us, the programmers, and not the content designers or graphic artists
- But what exactly is making us inefficient? We'll explore that in subsequent slides

The “Ugly Java Technology GUI” Stereotype

- Part of the problem is, well, Swing
  - > AWT/Swing container/component hierarchy is a tree of rectangular (mostly gray) boxes
  - > If all you do is compose Swing components together →the result is typically the Ugly Java technology GUI
  - > Same problem exists with other toolkits, e.g., GTK, VB
- UI Designers and Swing programmers are using different building blocks
  - > UI Designers compose designs in tools like Photoshop and Illustrator
  - > The building blocks they use have direct analogs in Java 2D API, but not necessarily in Swing

Java 2D API

- To match the designs of UI designers requires using Java 2D API
- But Java 2D API doesn't have compositional behavior
  - The barrier to entry for many Java code programmers is too high (i.e., other than Romain Guy)
- In addition to Swing Components, JavaFX Script includes SVG-like interfaces to Java 2D API as first-class elements which can be composed together into higher-level components.
- JavaFX Script allows declarative expression of this composition

Benefits of Declarative Syntax

- You can see it in Web applications
- For example, ease of composing styled text
  - HTML vs. JTextPane
- HTML Table using JSTL versus JTable
- JavaFX Script brings that same ease of use to Swing and Java 2D API programming

Benefits of Data Binding in JavaFX Script

- Cause and Effect—Responding to change
- The JavaFX Script bind operator—Allows dynamic content to be expressed declaratively
- Dependency-based evaluation **of any expression**
- Automated by the system—Rather than manually wired by the programmer
- You just declare dependencies and the JavaFX Script runtime takes care of performing updates when things change
- Eliminates listener patterns

History of JavaFX

- Originally it was kind of modeling language:
  - > The class declarations were based on the IDL used for object databases
  - > UML cardinality specifications for attributes
  - > UML object notation (NAME:CLASS) for constants
- Query language added:
  - > Java-like expressions
  - > Features from XQuery: sequences (arrays), predicates and list-comprehensions (foreach, Select)
  - > First-class functions and closures follow the syntax of ECMAScript to incorporate them into variable declarations

“Now, Why on Earth Would We Want This?”

… more than just the sandbox:

Java^{TM}SE Security

Java™ Platform Security

- Strong Data Typing
- Automatic Memory Management
- Bytecode Verification
- Secure Class Loading

- Type-Safe Reference Casting
- Structured Memory Access (no pointer arithmetic)
- Array Bounds Checking
- Checking References for `null`

Cryptography

- APIs:
  - digital signatures
  - message digests
  - ciphers (symmetric, asymmetric, stream & block)
  - message authentication codes
  - key generators
  - key factories
- RSA, DSA, AES, Triple DES, SHA, PKCS#5, RC2, and RC4
- PKCS#11 cryptographic token support

Authentication and Access Control

- JAAS
  - Open API for authentication and authorization
  - Large number of authentication sources

```
AuthProvider.login( Subject subject,
                    CallbackHandler handler )
```

- Java Policy Framework
  - 20 different permissions
    - on almost 100 target types

```
grant codeBase "file:/home/sysadmin/" {
    permission java.io.FilePermission "/tmp/abc", "read";
};
```

Secure Communications

- Transport Layer Security (TLS)
- Secure Sockets Layer (SSL)
- Kerberos
- Simple Authentication and Security Layer (SASL)
- Full support for HTTPS over SSL/TLS

Public Key Infrastructure

- Certificates and Certificate Revocation Lists (CRLs):
  - X.509
- Certification Path Validators and Builders:
  - PKIX (RFC 3280)
  - On-line Certificate Status Protocol (OCSP)
- KeyStores:
  - PKCS#11
  - PKCS#12
- Certificate Stores (Repositories):
  - LDAP
  - java.util.Collection

New in Java 6

- JSR 105, the XML Digital Signature API and implementation
- JSR 268, Smart Card I/O API
- Elliptic Curve Cryptography (ECC) in SunPKCS11
- Elliptic Curve CipherSuites in SunJSSE
- Access Network Security Services (NSS) using SunPKCS11
- FIPS 140 compliance for SunJSSE
- Pluggability restrictions have been removed from JSSE

New in Java 6, cont.

- Socket read timeouts are fully supported by SunJSSE SSLSockets
- Cipher Text Stealing (CTS) mode added to SunJCE block ciphers
- New PBKDF2WithHmacSHA1 Secretkeyfactory algorithm added to SunJCE
- Removed the 2048 RSA keysize limit from local_policy.jar
- New Certification Authority (CA) certificates added
- Support for AES Encryption Type in Java GSS/ Kerberos

New in Java 6, cont.

- Support for RC4-HMAC Encryption Type in Java GSS/ Kerberos
- Support for SPNEGO in Java GSS
- Support for new Pre-Authentication Mechanisms
- Native Platform GSS Integration
- Access to native PKI and cryptographic services on Microsoft Windows
- Enhancements to the implementation of PKI Certificate Path Validation
- JAAS-based authentication using LDAP

The Language:

JavaFX Script

Scripts

```
var ten : Integer = 10;
java.lang.System.out.println("Twice {ten} is {2 * ten}.");

// Yields:
Twice 10 is 20.
```

Classes

```
class Rectangle {

    def sides: Integer = 4;
    var width: Integer;
    var height: Integer;

    function grow(): Void {
        grow(1);
    }

    function grow(amount: Integer): Void {
        width += amount;
        height += amount;
    }

}
```

Objects

```
Rectangle {

     width: 100
     height: 100

}

var myRect = Rectangle {

     width: 100
     height: 100
}
```

Sequences

```
var weekDays = ["Mon","Tue","Wed","Thur","Fri"];
var week = [weekDays, ["Sat","Sun"]];

var mon = week[0];
var wed = week[2];
var fri = week[4];

// returns true
days == ["Mon","Tue","Wed","Thur","Fri","Sat","Sun"];

1 == [1]; // returns true

var xs:Number[]; // sequence of Number
var strs:String[]; // sequence of String
```

Sequences (cont.)

```
var nums = [1..100];

sequence[variableName| booleanExp]

var nums = [1,2,3,4];
var numsGreaterThanTwo = nums[n|n > 2];

seq[a..b]  // the sequence between the indexes
              a and b inclusive
seq[a..<b] // the sequence between the indexes
              a inclusive and b exclusive
seq[a..]   // same as seq[a..<sizeof seq]
seq[a..<]  // for consistancy. This is the same
              as seq[a..<sizeof seq-1]
```

Sequences (cont.)

```
function factors(n:Number) {
     return for (i in [1 .. n/2] where n mod i == 0) i;
}

var nums = [1..5];
// returns 3,4,5:
var numsExceptTheFirstTwo = nums[n|indexof n > 1];

insert x into seq
insert x before seq[idx]
insert x after seq[idx]

delete seq
delete x from seq
delete seq[idx]
delete seq[a..b] // and all other slice forms
```

Data Binding

```
import javafx.application.Frame;
import javafx.application.Stage;
import javafx.scene.text.Text;

var myString = "Hello World!";

Frame {
     width: 50
     height: 50
     visible: true
     stage: Stage {
          content: Text {
               content: bind myString
          }
     }
}

// If some other part of code changes myString
// then the GUI's text will automatically change
// as well.
```

Data Binding (cont.)

```
var x= bind expr;
var sum= bind expr1 + expr2;

var y = 3;
function ten() : Integer { 10 }
var sum = bind ten() + y;
y = 7;

bind { var a = expr; var b = expr;
       var c = expr; expr }

var x = bind if (condExpr) expr1 else expr2;
```

Triggers

```
import java.lang.System;

ReplaceDemo {

     mySensitiveData: "Will anyone notice?"

}

class ReplaceDemo {
     var mySensitiveData: String
          on replace {
               System.out.println("I noticed a change!");
          };
}
```

Animation

```
var crossfade:Timeline = Timeline {
    repeatCount: Timeline.INDEFINITE
    keyFrames: [
        KeyFrame {
            time: 0s
            values: slide.opacity => 1.0
        },
        KeyFrame {
            time: 500ms
            values: slide.opacity => 0.0
            action: function() { idx++; }
        },
        KeyFrame {
            time: 1s
            values: slide.opacity => 1.0
        }
    ]
};
```

?

Thorbiörn Fritzon
<fritzon@sun.com>