



# Real World Technologies: NetBeans GUI Builder, JRuby, JavaFX, and Java ME

[developers.sun.com/students/courses](http://developers.sun.com/students/courses)



# Topics

- Introduction to “Real World Technologies: NetBeans GUI Builder, JRuby, JavaFX, and Java ME – Earn a Certificate” 5-hour online course
- NetBeans GUI Builder
- JRuby & JRuby on Rails
- Java FX Programming
- Java ME Programming

# Real World Technologies 5-hour Online Course

- You can take it anytime – it is free!
- <http://developers.sun.com/students/courses>
- You get exposed to exciting and leading-edge real-life technologies
- You do the hands-on labs and homeworks of 4 topics
- If you submit all 4 homeworks, you will receive PDF version of certificate
- You can participate in the course forum



# NetBeans GUI Builder



# Problems with Traditional GUI Building

- Using Swing APIs and Layout managers is complex
- Resizing and alignment are hard
- Handling of locale is hard
- Providing Look and Feel of a particular OS is hard

# NetBeans GUI Builder Solves These Problems!

- Create professional forms without deep knowledge of Swing
- Good looking by default – spacing is handled according to underlying OS's Look and Feel
- GUI components are added and positioned by drag & drop, baseline support
- GUI components are grouped together, they react on neighbor changes
- Form behaves intelligently when resized
- GUI components alignment can be easily controlled

# NetBeans GUI Builder Features

- Simple and intuitive layout of GUIs without the complexity of Swing layout managers
- Drag and drop capability
- Automatic form alignment
- Visual guidelines for optimal spacing between components and alignment of components
- Support for both visual and non-visual forms
- Extensible Component Palette with pre-installed Swing and AWT components



# NetBeans GUI Builder Features

- Component Inspector showing a components tree and properties
- In-place editing of text labels of components (labels, buttons, textfields, etc).
- Full JavaBeans support - installing, using and customizing (properties, events, customizers)
- Visual JavaBean customization - ability to create forms from any JavaBean classes
- Built-in support for i18n and a11y
- In-place text label editing



# How Does NetBeans GUI Builder Work?

- Similar to Interface Builder on Mac or Visual Studio, but on Java platform
  - > Supports multiple OS'es
- New layout manager - **GroupLayout** layout manager

# **NetBeans GUI Builder Demo: These Are The Exercises You Will Do as Part of the Course**

# Demo 1: Building a Contact Editor

**E-mail Contacts**

**Name**

First Name:  Last Name:

Title:  Nickname:

Display Format:

**E-mail**

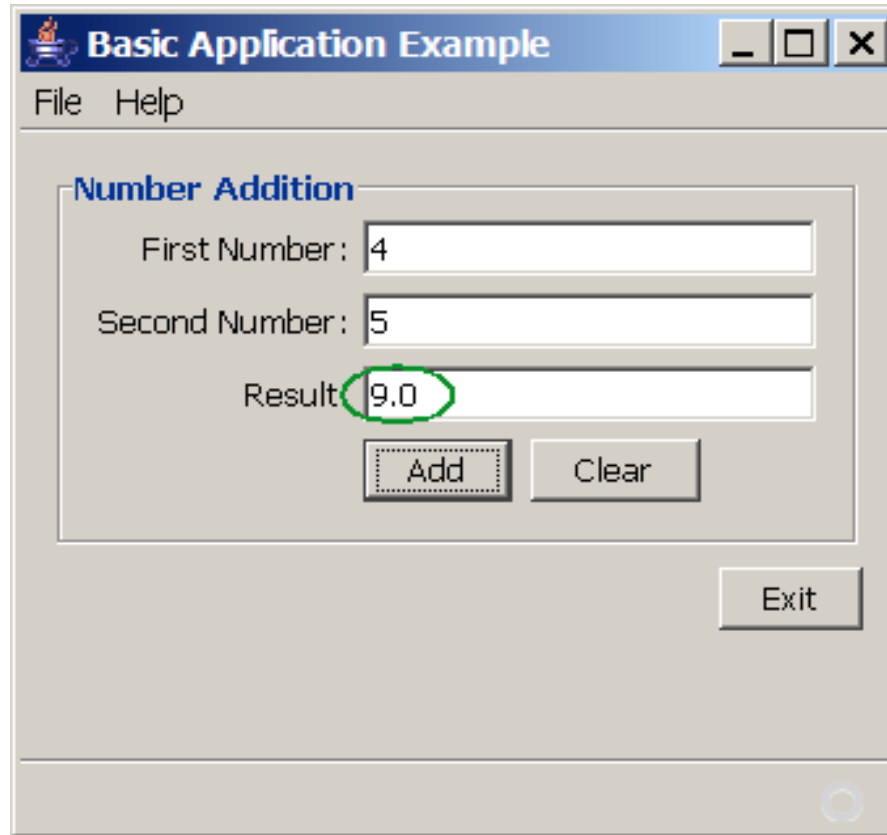
E-mail Address:

john.guy@xxxxxxx.yyy  
gui@yyyyyy.xxx

**Mail Format:**

HTML  Plain Text  Custom

# Demo 2: Adding Event Handlers





# JRuby & JRuby on Rails



# Topics

- What is and Why Ruby?
- What is and Why JRuby?
- What is “JRuby on Rails”?
- “Ruby on Rails” Principles
- Why “Ruby on Rails”?
- Why “JRuby on Rails”?

# What is Ruby?



# What is Ruby?

- Object oriented language
- Dynamically typed
- Extensible by Ruby Gems
- Released in 1995
- Gains popularity by a Killer application “Ruby on Rails”

# What is JRuby?



**Open source**  
**Java**  
**implementation of the**  
**Ruby**  
**language**

# What Is JRuby?

- Started in 2002
- Java platform implementation of Ruby language
- Open source, many active contributors
- Aiming for compatibility with current Ruby version
- Integrates with Java technology
  - > Call to Ruby from Java technology via Java Specification Request (JSR) 223, BSF, Spring
  - > Use Java class files from Ruby (e.g., Script Java)
- Growing set of external projects based on JRuby
  - > JRuby-extras (GoldSpike, ActiveRecord-JDBC,...)

# JRuby: Java Technology Integration

- Script Java
  - > Power of Java technology with the Syntax of Ruby

```
require 'java'
```

```
list = java.util.ArrayList.new
```

```
list << 1
```

```
list << 3
```

```
list << 2
```

```
list.sort
```

# What is “JRuby on Rails”?

# What Is “Ruby on Rails”?

- A full-stack MVC web development framework
- Open source, many contributors
- Written in Ruby
- First released in 2004 by David Heinemeier Hansson
- Gaining popularity
  - > Ruby and Rails book sales are outselling Perl book sales



# **“Ruby on Rails” Principles**

# “Ruby on Rails” Principles

- Convention over configuration
  - > Why punish the common cases?
  - > Encourages standard practices
  - > Everything simpler and smaller
- Don't Repeat Yourself (DRY)
  - > Framework written around minimizing repetition
  - > Repetitive code harmful to adaptability
- Agile development environment
  - > No recompile, deploy, restart cycles
  - > Simple tools to generate code quickly
  - > Testing built into the framework

# Why “Ruby on Rails”?

- Greatly simplified web development
  - > Instant applications: working code in minutes
  - > “Less Rails code than Java application configuration”
- Growing community of developers
- Makes small apps trivial to create
- Ruby is an excellent language

# Why “JRuby on Rails” over “Ruby on Rails”?

- Deployment to Java application servers
- Java technology production environments pervasive
  - > Easier to switch framework vs. whole architecture
  - > Lower barrier to entry
- Broader, more scalable database support
- Integration with Java technology libraries, legacy services
- No need to leave Java technology servers, libraries, reliability

# “JRuby on Rails”: Java EE Platform

- Pool database connections
- Access any Java Naming and Directory Interface™ (J.N.D.I.) API resource
- Access any Java EE platform TLA:
  - > Java Persistence API (JPA)
  - > Java Management Extensions (JMX™)
  - > Enterprise JavaBeans™ (EJB™)
  - > Java Message Service (JMS) API
  - > SOAP/WSDL/SOA

# “JRuby on Rails”: Deployment Platforms

- WEBrick
- Mongrel
- War file deployment to:
  - > GlassFish™ V2
  - > GlassFish™ V3
  - > Servlet-based app server

# JRuby on Rails: Futures

- Improve Java EE platform support for Rails
  - > Rubify Java EE platform APIs
  - > Create Rails plug-ins to make installable units
- Use Java technology native equivalents
  - > Unicode support
  - > XML support
- Port Ruby native libraries
  - > RMagick

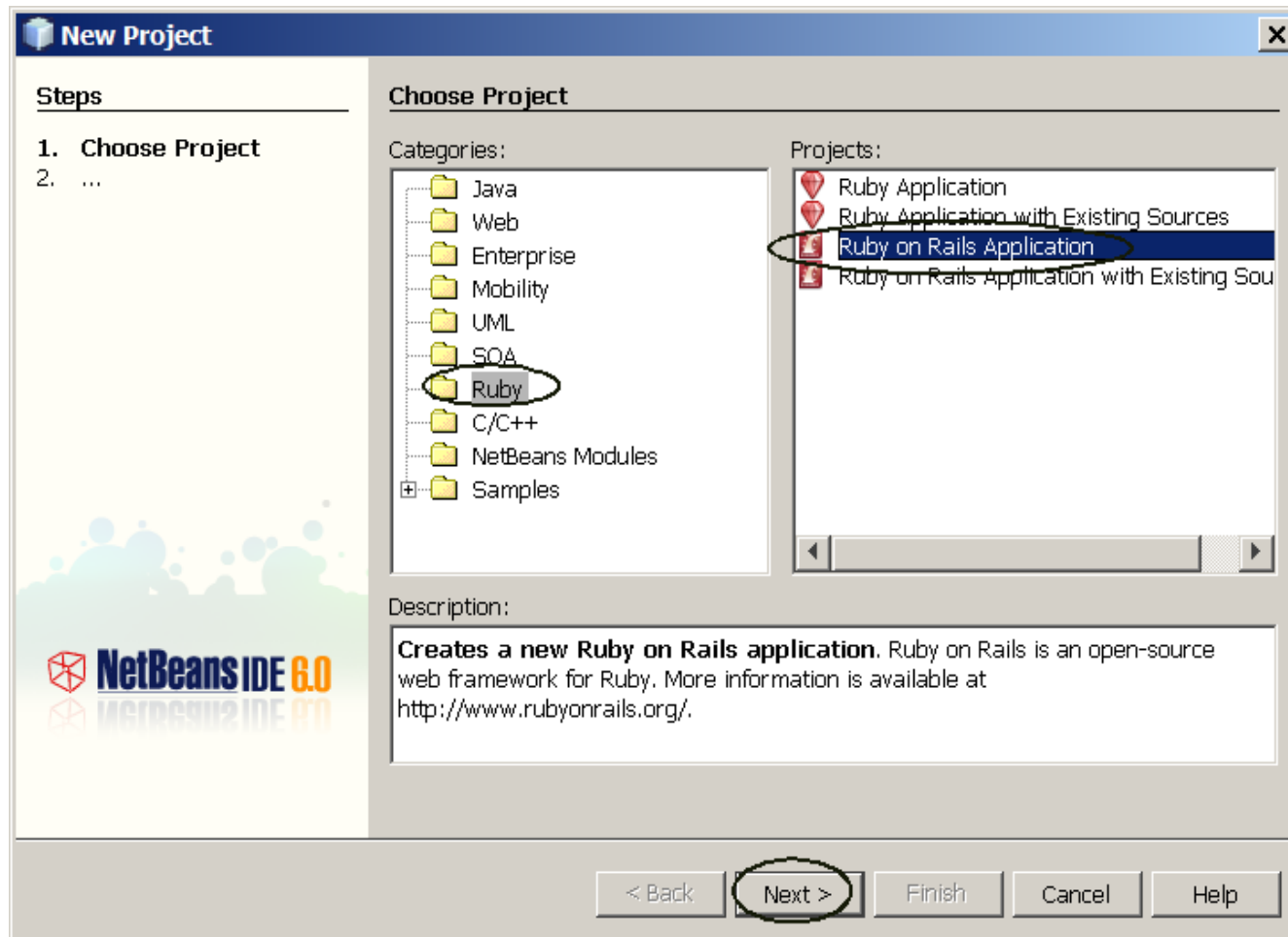


# Step By Step Process of Building “Hello World” Rails Application

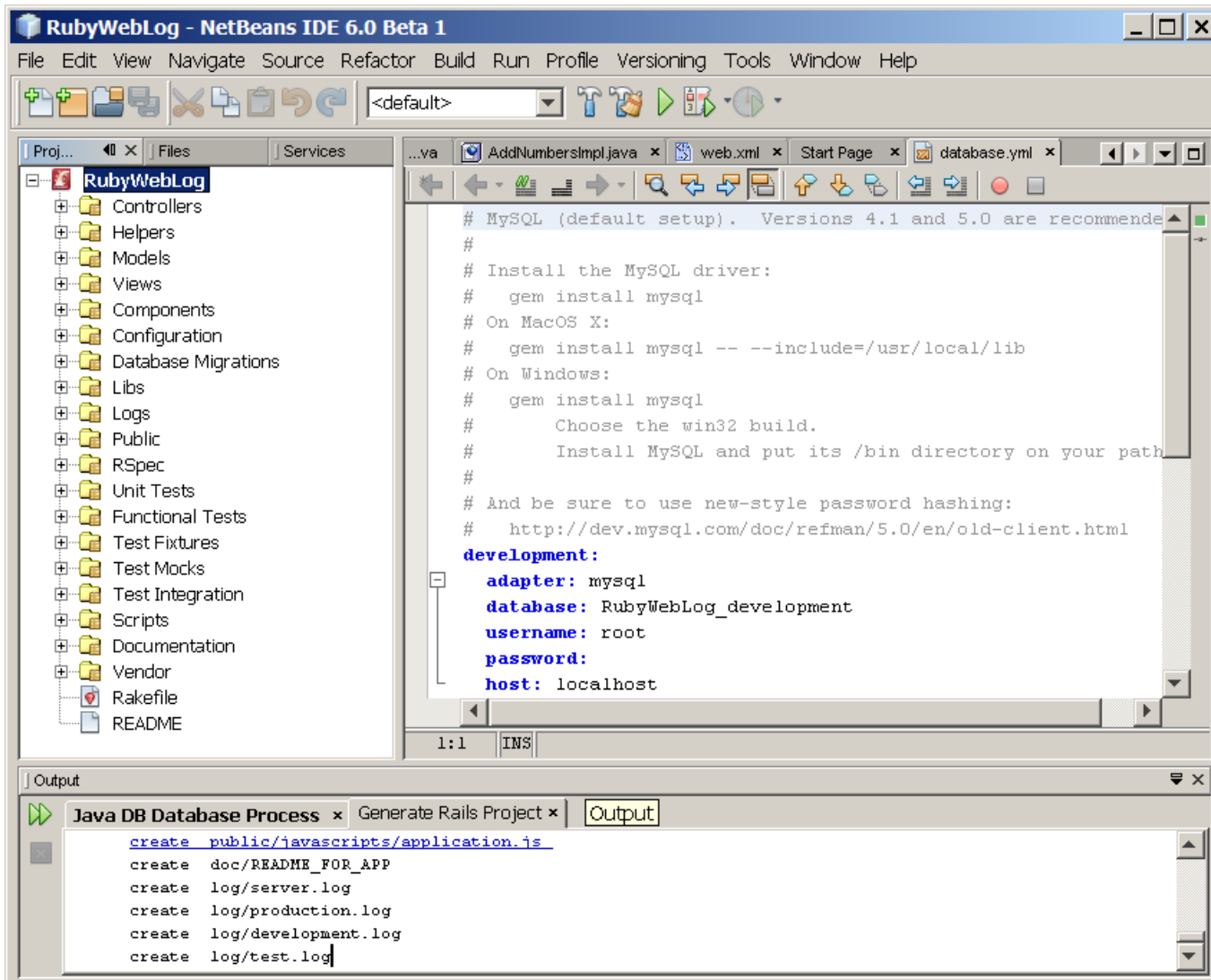
# Steps to Follow

1. Create “Ruby on Rails” NetBeans project
  - > IDE generate necessary directories
2. Create Models (through Rails Generator)
  - > Migrate database
3. Create Controllers
  - > Scaffolding
4. Create Views
5. Set URL Routing

# 1. Create “Ruby on Rails” NetBeans Project



# NetBeans Generates Rails Directories



The screenshot shows the NetBeans IDE interface for a project named "RubyWebLog". The left sidebar displays a project tree with the following structure:

- Controllers
- Helpers
- Models
- Views
- Components
- Configuration
- Database Migrations
- Libs
- Logs
- Public
- RSpec
- Unit Tests
- Functional Tests
- Test Fixtures
- Test Mocks
- Test Integration
- Scripts
- Documentation
- Vendor
- Rakefile
- README

The main editor window shows the content of the `database.yml` file, which includes instructions for installing the MySQL driver and configuring the database connection for the development environment:

```
# MySQL (default setup). Versions 4.1 and 5.0 are recommended.
#
# Install the MySQL driver:
#   gem install mysql
# On MacOS X:
#   gem install mysql -- --include=/usr/local/lib
# On Windows:
#   gem install mysql
#     Choose the win32 build.
#     Install MySQL and put its /bin directory on your path.
#
# And be sure to use new-style password hashing:
#   http://dev.mysql.com/doc/refman/5.0/en/old-client.html
development:
  adapter: mysql
  database: RubyWebLog_development
  username: root
  password:
  host: localhost
```

The Output window at the bottom shows the execution of the `generate rails project` command, listing the files created:

```
create public/javascripts/application.js
create doc/README_FOR_APP
create log/server.log
create log/production.log
create log/development.log
create log/test.log
```

## 2. Create Models

**Rails Generator** [X]

Generate:

Arguments:

When file already exists:  Skip  Overwrite

Preview Changes Only

Description:

The model generator creates stubs for a new model.

The generator takes a model name as its argument. The model name may be given in CamelCase or under\_score and should not be suffixed with 'Model'.

As additional parameters, the generator will take attribute pairs described by name and type. These attributes will be used to prepopulate the migration to create the table for the model and give you a set of predefined fixture. You don't have to think up all attributes up front, but it's a good idea of adding just the baseline of what's needed to start really working with the resource.

The generator creates a model class in `app/models`, a test suite in `test/unit`, test fixtures in `test/fixtures/singular_name.yml` and a migration in `db/migrate`.

# 3. Create Controllers

**Rails Generator**

Generate:

Name:

Views:

When file already exists:  Skip  Overwrite

Preview Changes Only

Description:

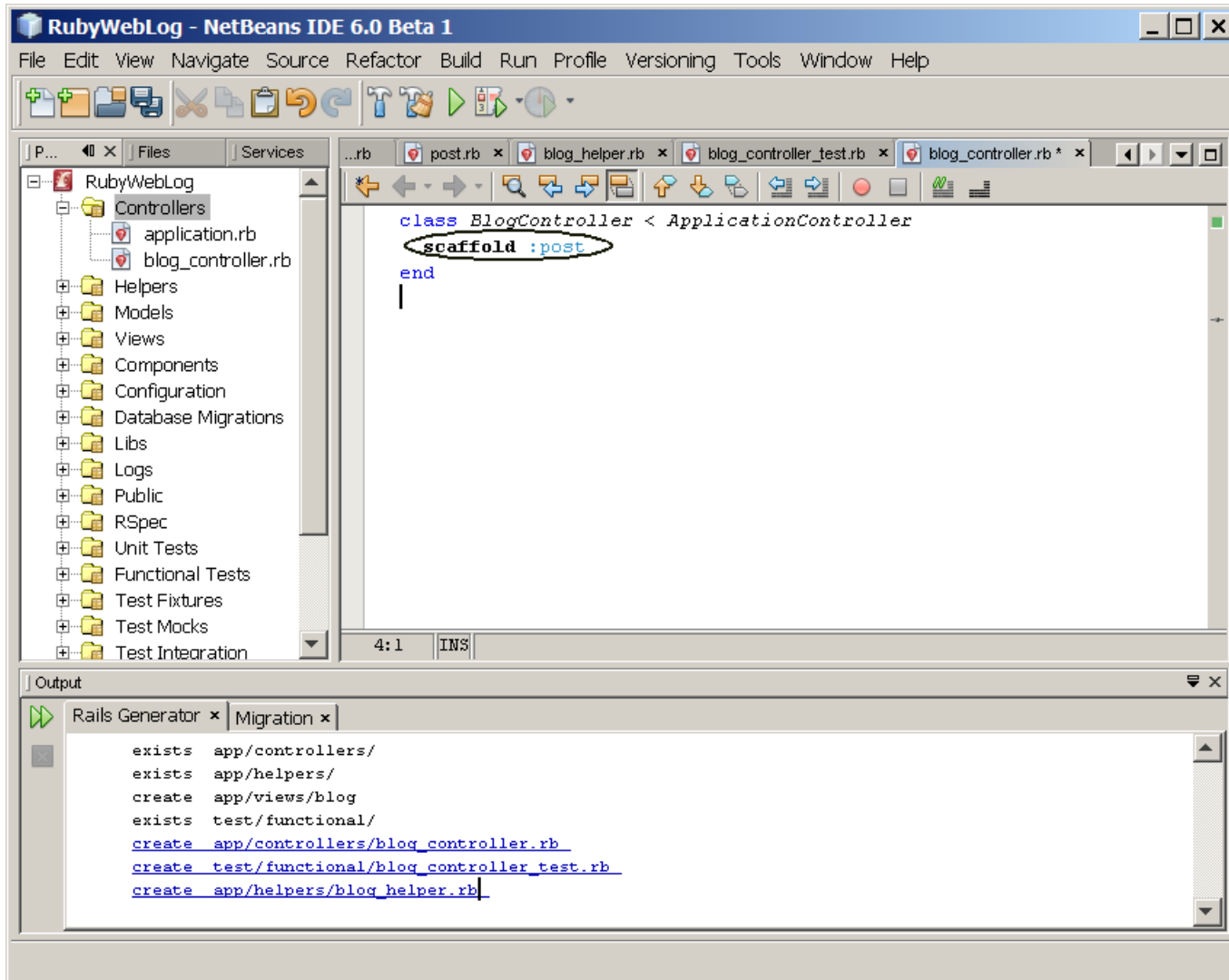
The controller generator creates stubs for a new controller and its views.

The generator takes a controller name and a list of views as arguments. The controller name may be given in CamelCase or under\_score and should not be suffixed with 'Controller'. To create a controller within a module, specify the controller name as 'module/controller'.

The generator creates a controller class in app/controllers with view templates in app/views/controller\_name, a helper class in app/helpers, and a functional test suite in test/functional.

Example:

# Scaffolding



The screenshot shows the NetBeans IDE interface for a project named 'RubyWebLog'. The left sidebar displays a project tree with folders for Controllers, Helpers, Models, Views, Components, Configuration, Database Migrations, Libs, Logs, Public, RSpec, Unit Tests, Functional Tests, Test Fixtures, Test Mocks, and Test Intearction. The main editor window shows the code for 'blog\_controller.rb', which contains the following code:

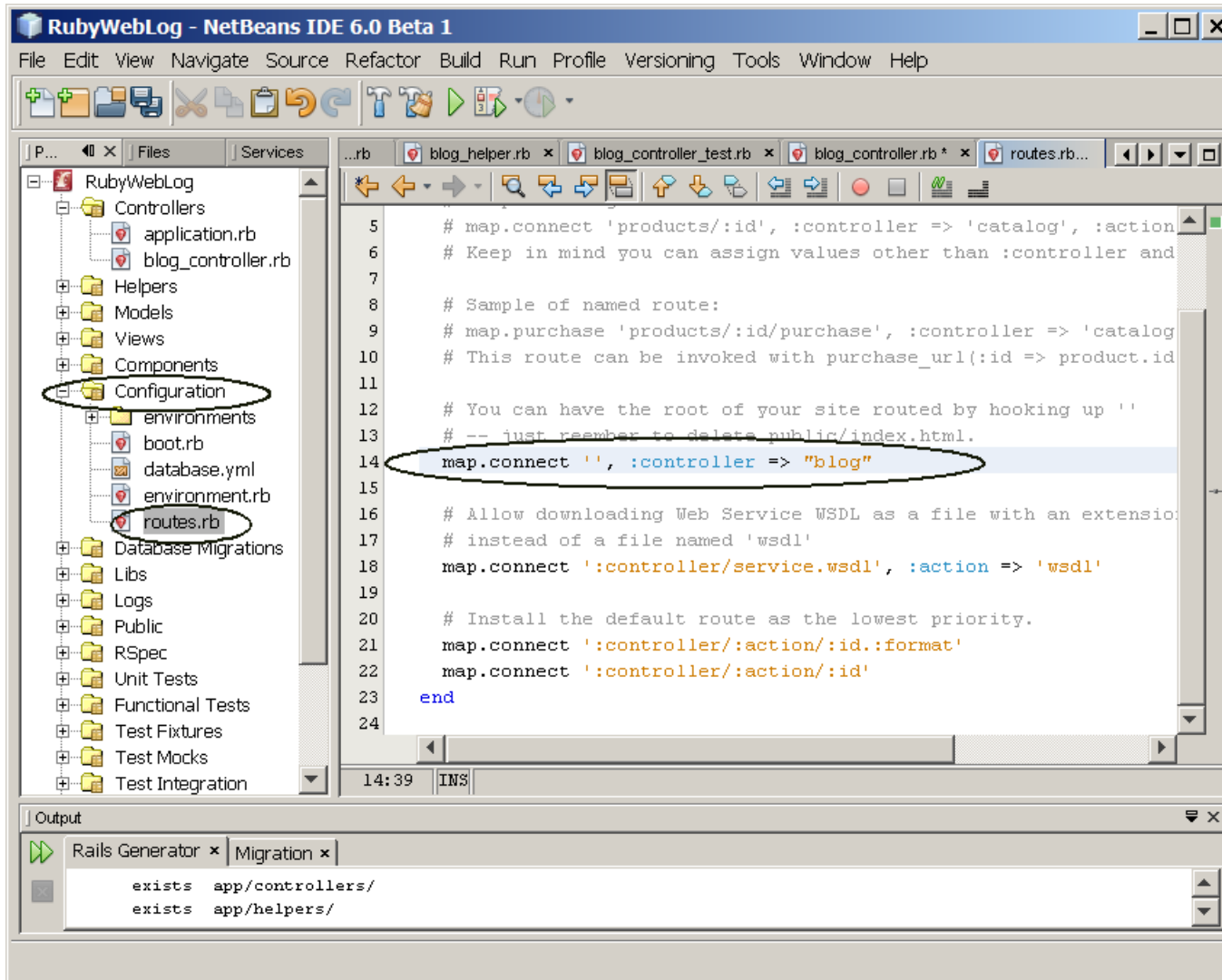
```
class BlogController < ApplicationController
  scaffold :post
end
```

The 'scaffold :post' line is circled in red. The bottom output window shows the output of the Rails Generator, indicating the creation of several files:

```
exists app/controllers/
exists app/helpers/
create app/views/blog
exists test/functional/
create app/controllers/blog_controller.rb
create test/functional/blog_controller_test.rb
create app/helpers/blog_helper.rb
```



# 5. Set Route



The screenshot shows the NetBeans IDE interface for a project named 'RubyWebLog'. The left sidebar displays a project tree with the following structure:

- RubyWebLog
  - Controllers
    - application.rb
    - blog\_controller.rb
  - Helpers
  - Models
  - Views
  - Components
  - Configuration
    - environments
    - boot.rb
    - database.yml
    - environment.rb
    - routes.rb
  - Database Migrations
  - Libs
  - Logs
  - Public
  - RSpec
  - Unit Tests
  - Functional Tests
  - Test Fixtures
  - Test Mocks
  - Test Integration

The 'Configuration' folder and its sub-items are circled in red. The 'routes.rb' file is also circled in red. The main editor window shows the content of 'routes.rb' with the following code:

```

5  # map.connect 'products/:id', :controller => 'catalog', :action => 'show'
6  # Keep in mind you can assign values other than :controller and :action
7
8  # Sample of named route:
9  # map.purchase 'products/:id/purchase', :controller => 'catalog', :action => 'purchase'
10 # This route can be invoked with purchase_url(:id => product.id)
11
12 # You can have the root of your site routed by hooking up ''
13 # -- just remember to delete public/index.html.
14 map.connect '', :controller => "blog"
15
16 # Allow downloading Web Service WSDL as a file with an extension
17 # instead of a file named 'wsdl'
18 map.connect ':controller/service.wsdl', :action => 'wsdl'
19
20 # Install the default route as the lowest priority.
21 map.connect ':controller/:action/:id.:format'
22 map.connect ':controller/:action/:id'
23 end
24

```

The line `map.connect '', :controller => "blog"` is circled in red. The status bar at the bottom shows the time 14:39 and the cursor position INS.

The Output window at the bottom shows the following text:

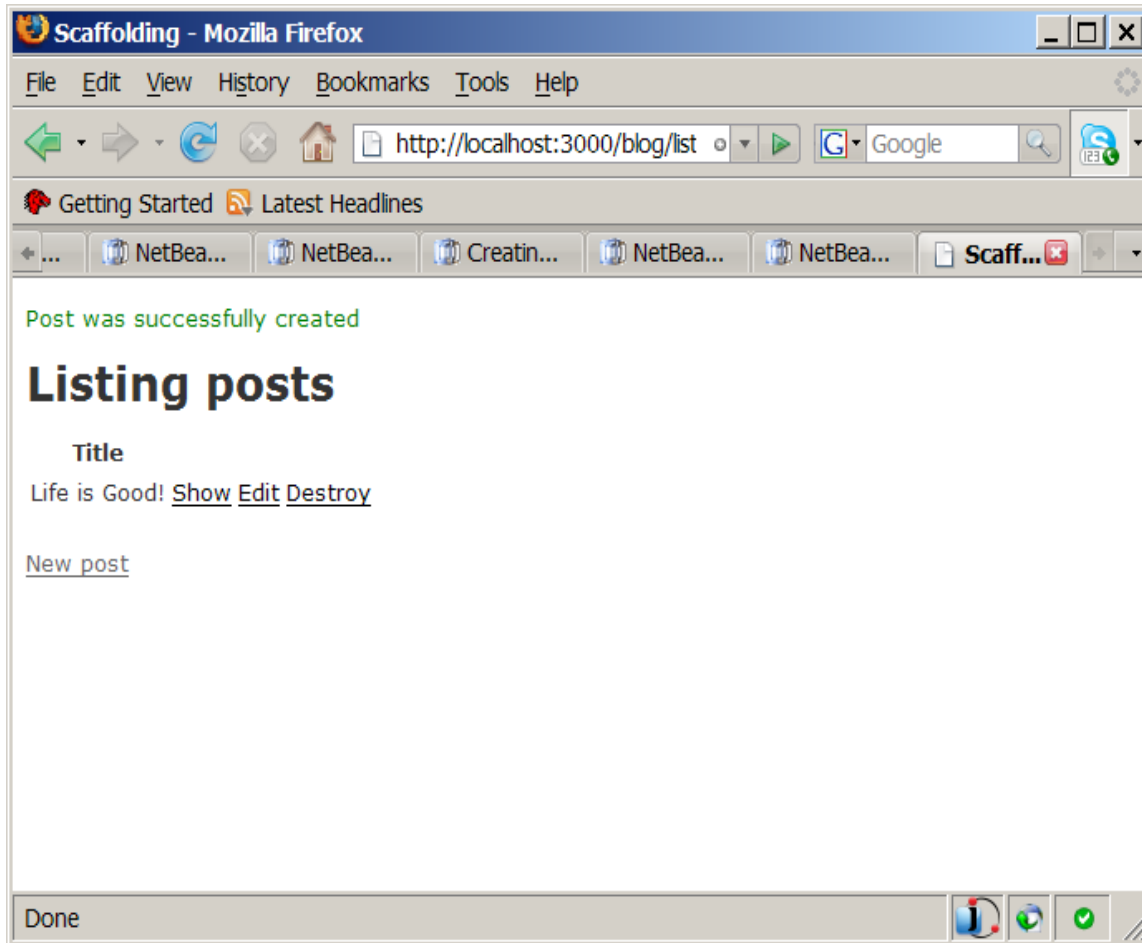
```

Rails Generator x Migration x
exists app/controllers/
exists app/helpers/

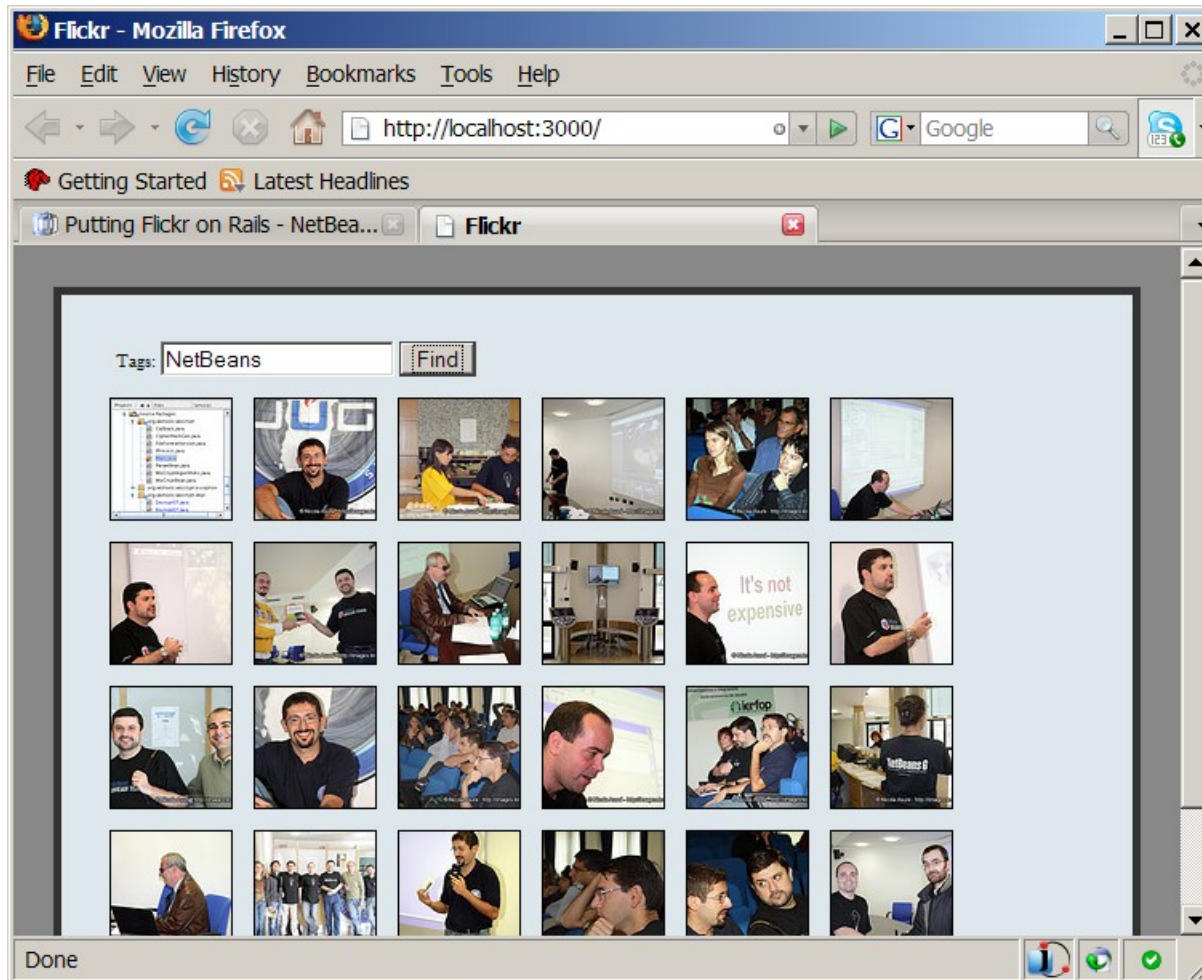
```

# **JRuby on Rails Demo: These Are The Exercises You Will Do as Part of the Course**

# Demo 1: Building WebLog Webapp



# Demo 2: Building Flickr Webapp





# JavaFX Programming



# Topics

- What is and Why JavaFX?
- JavaFX Product family and Architecture
- JavaFX Script
- JavaFX Script examples



# What is and Why JavaFX?

# Market Trends

- Personal & mobile computing are blurring
  - > Handsets becoming more advanced
  - > Higher-speed IP Networks
- Ecosystem shifting to “platforms”
  - > Full application environment including middleware, user experience, etc.
  - > Reduce development costs and improves device consistency
- Software stacks largely proprietary
  - > Industry looking for an Open alternative



# The GUI Quagmire

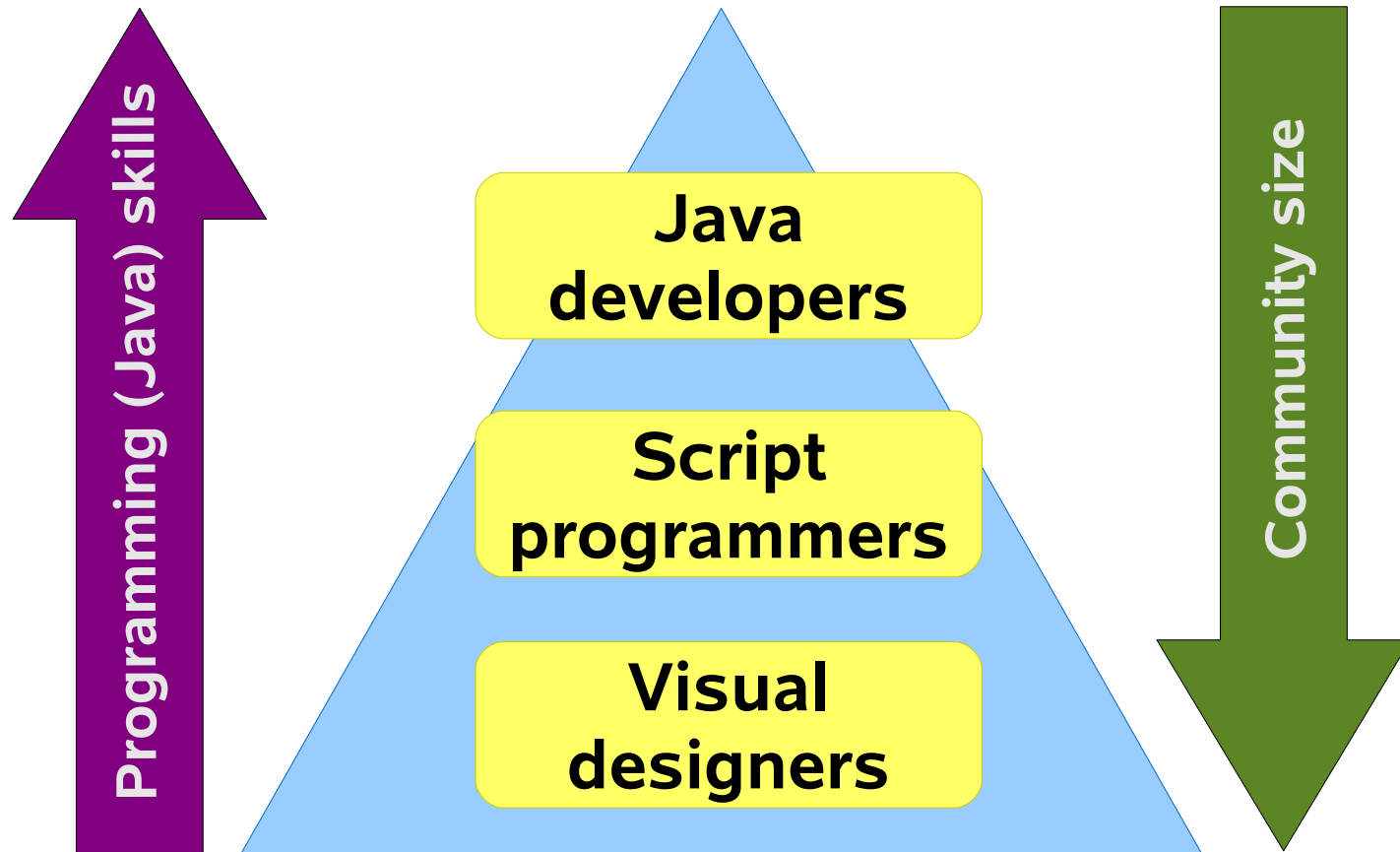
- Java Swing and 2D APIs are very powerful and yet
  - > Why does it take a long time to write GUI programs?
  - > How can we avoid the “Ugly Java technology GUI” stereotype?
  - > Why do Flash programs look different than Java programs?
  - > Why does it seem easier to write web apps than Swing programs?
  - > How can I avoid having an enormous mass of listener patterns?

# Why yet another scripting language for Java, JavaFX Script?

- Language for Spicing up Java GUI Programming easier
- Building and running of media/content- rich Java clients application much easier
- More productive by making Swing and Java 2D™ API accessible to a much the script programmer
- Simple Accessing of Java Objects
- Serve as persistence format for a new class of tools targeted at non programmers like designers and end users

# Expanding the Community

Professionals



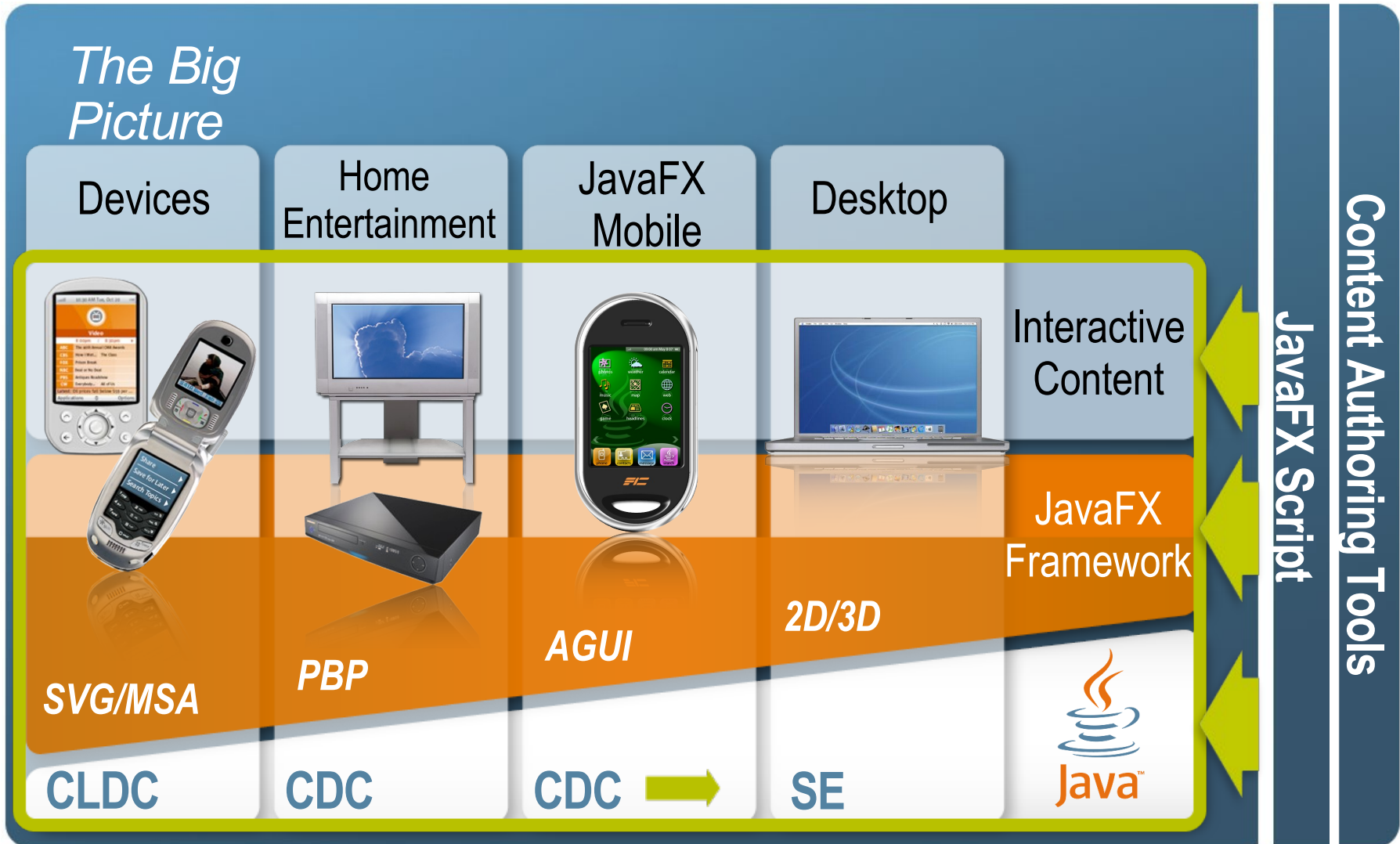
Hobbyist

# JavaFX Product Family & Architecture

# JavaFX Product Family

- Announced at JavaONE 2007
- Leverages the Java platform's write-once-run-anywhere portability, application security model, ubiquitous distribution and enterprise connectivity
- Initially comprised of:
  - JavaFX Script
    - > Highly productive scripting language for content developers to create rich media and interactive content
  - JavaFX Mobile
    - > Mobile software stack available via OEM license to carriers and handset manufacturers seeking a branded relationship with consumers

# JavaFX: An Architecture that Scales



# JavaFX Workflow



Visual Assets

JavaFX Script code

Java FX Source Application

JavaFX Tools

Compilation

Packaging

Packaging

Packaging

Java FX Mobile App Binary

JavaFX Mobile



Java FX TV App Binary

JavaFX TV



Java FX Desktop App Binary

JavaFX Desktop / Consumer JRE



# What is JavaFX Script?



# What is JavaFX Script?

- Formerly known as F3 (Form Follows Function) of Christopher Oliver from Sun Microsystems
  - > <http://blogs.sun.com/chrisoliver/category/JavaFX>
- Java Scripting Language
  - > Object-oriented
  - > Static typing + type inference
  - > Declarative Syntax
  - > Automatic Data Binding
  - > Mixed functional/procedural evaluation model

# What is JavaFX Script?

- Extensive widget library encompassing Swing components and Java2D objects
- Easily create and configure individual components
- Work with all major IDEs
- Statically typed - retains same code structuring, reuse, and encapsulation features of Java
- Capable of supporting GUIs of any size or complexity
- Makes it easier to use Swing
- Rich Internet Applications (RIA)

# JavaFX Script – A Simple Comparison

## *In JavaFX Script*

```
picks.opacity = [0, .01 .. 1 ] dur 1000 linear
```

## *In Java (SwingLabs Timing Framework)*

```
public void class Guitar {
    private GuitarPick pick = ...;
    public Guitar() {
        pick.setOpacity(.5f);
        Animator a =
            PropertySetter.createAnimator(
                300, pick, "opacity", .5f, 1.0f);
        MouseTrigger.addTrigger(pick, a,
            MouseTriggerEvent.ENTER, true);
    }
}
```

## *In Java*

### Main guitar class

```
guitarAnimationThread = new StringOpThread();
.....
.....
if (guitarAnimationThread != null) {
    guitarAnimationThread.run();
}
.....
```

### StringOpThread class

```
.....
public void run() {
    opacityBegin = 0.01;
    opacityEnd = 1.0;
    opacityIncrStep = 0.02;
    opacitySleep = 2;

    for(currOpacity = opacityBegin; \
        currOpacity < opacityEnd; \
        currOpacity+=opacityIncrStep) {
        setPickOpacity();
        repaint();
        try {
            thread.sleep(opacitySleep);
        } catch (InterruptedException e) { }
    }
}
```

# JavaFX Script Examples: Hello World

# Example : Hello world in Java FX

```
import javafx.ui.*;
```

```
Frame {  
    title: "Hello World JavaFX"  
    width: 200  
    height: 50  
    content: Label {  
        text: "Hello World"  
    }  
    visible: true  
}
```



# Example : Hello world in Swing

```
import javax.swing.*;

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

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

# Data Binding

# Data Binding in JavaFX

- Cause and Effect—Responding to change
- The JavaFX *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 runtime takes care of performing updates when things change
- Eliminates listener patterns



# Example: Dynamic Behavior

```
class HelloWorldModel {  
    attribute saying: String;  
}  
  
var model = HelloWorldModel {  
    saying: "Hello World"  
};  
  
var win = Frame {  
    title: bind "{model.saying} JavaFX"  
    width: 200  
    content: TextField {  
        value: bind model.saying  
    }  
    visible: true  
};
```



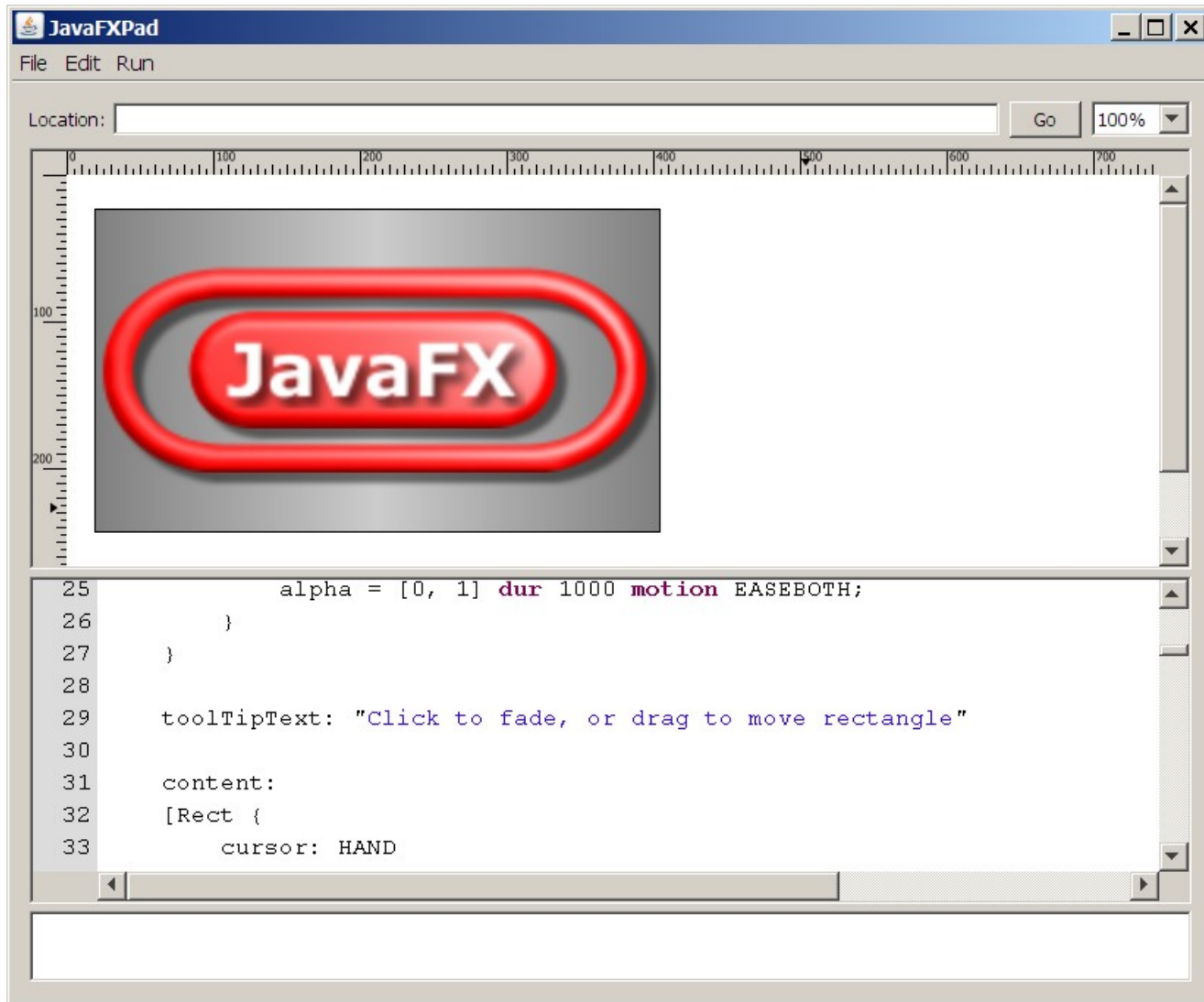
# Java 2D

# Java 2D API

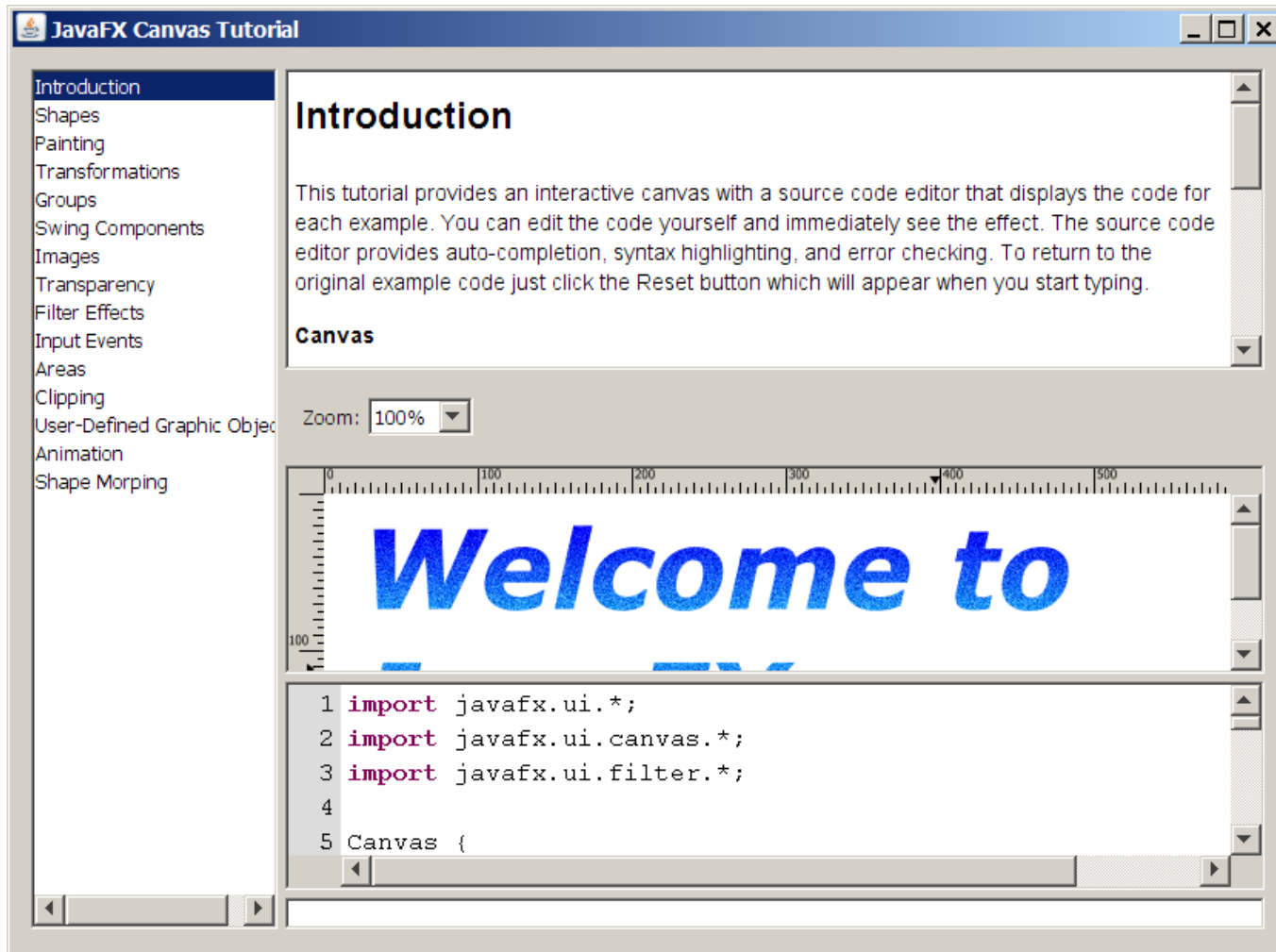
- To match the designs of UI designers requires using Java 2D API
- But Java 2D API doesn't have compositional behavior
  - > The barrier to entry for many Java code programmers is too high (i.e., other than Romain Guy)
- In addition to Swing Components, JavaFX includes SVG-like interfaces to Java 2D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

# **JavaFX Demo: These Are The Exercises You Will Do as Part of the Course**

# Demo 1: JavaFXPad



# Demo 2: JavaFX Canvas



The screenshot shows a window titled "JavaFX Canvas Tutorial". On the left is a navigation pane with a list of topics: Introduction (selected), Shapes, Painting, Transformations, Groups, Swing Components, Images, Transparency, Filter Effects, Input Events, Areas, Clipping, User-Defined Graphic Objects, Animation, and Shape Morphing. The main content area is titled "Introduction" and contains the following text: "This tutorial provides an interactive canvas with a source code editor that displays the code for each example. You can edit the code yourself and immediately see the effect. The source code editor provides auto-completion, syntax highlighting, and error checking. To return to the original example code just click the Reset button which will appear when you start typing." Below the text is a "Canvas" section with a "Zoom: 100%" dropdown menu. The canvas itself displays the text "Welcome to" in a large, blue, italicized font. Below the canvas is a source code editor with the following code:

```
1 import javafx.ui.*;  
2 import javafx.ui.canvas.*;  
3 import javafx.ui.filter.*;  
4  
5 Canvas {
```





# Java ME Programming



# Topics

- What is and Why Java ME?
- Java ME architecture
- NetBeans Mobility Pack Features
- Game development

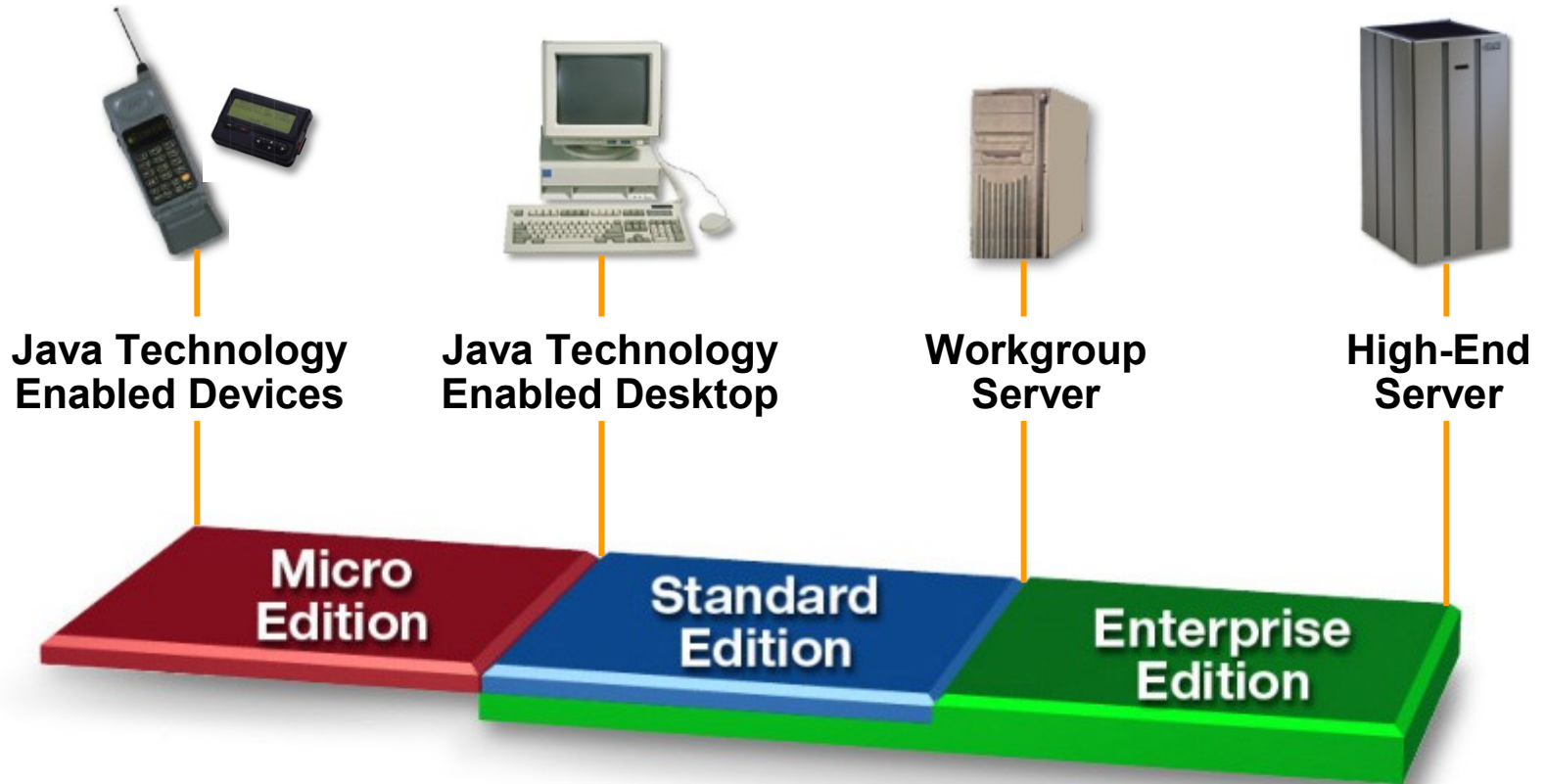


# What is and Why Java ME?

# Why Develop Mobility Applications?

- “Application at your fingertips”
- Personal
  - > One of 4 things you always carry with you (keys, wallet, watch, handphoned)
- Powerful, feature rich, programmable and royalty free
  - > Games in particular
- Anyone commuting will have 30mins to kill
  - > Capture audience
- It's not really that difficult

# The Java™ Platforms



# Java on Mobile Devices



**Mobile Service Architecture (MSA) provides a complete platform for mobile application development**

# Java ME Application Market



\$3B

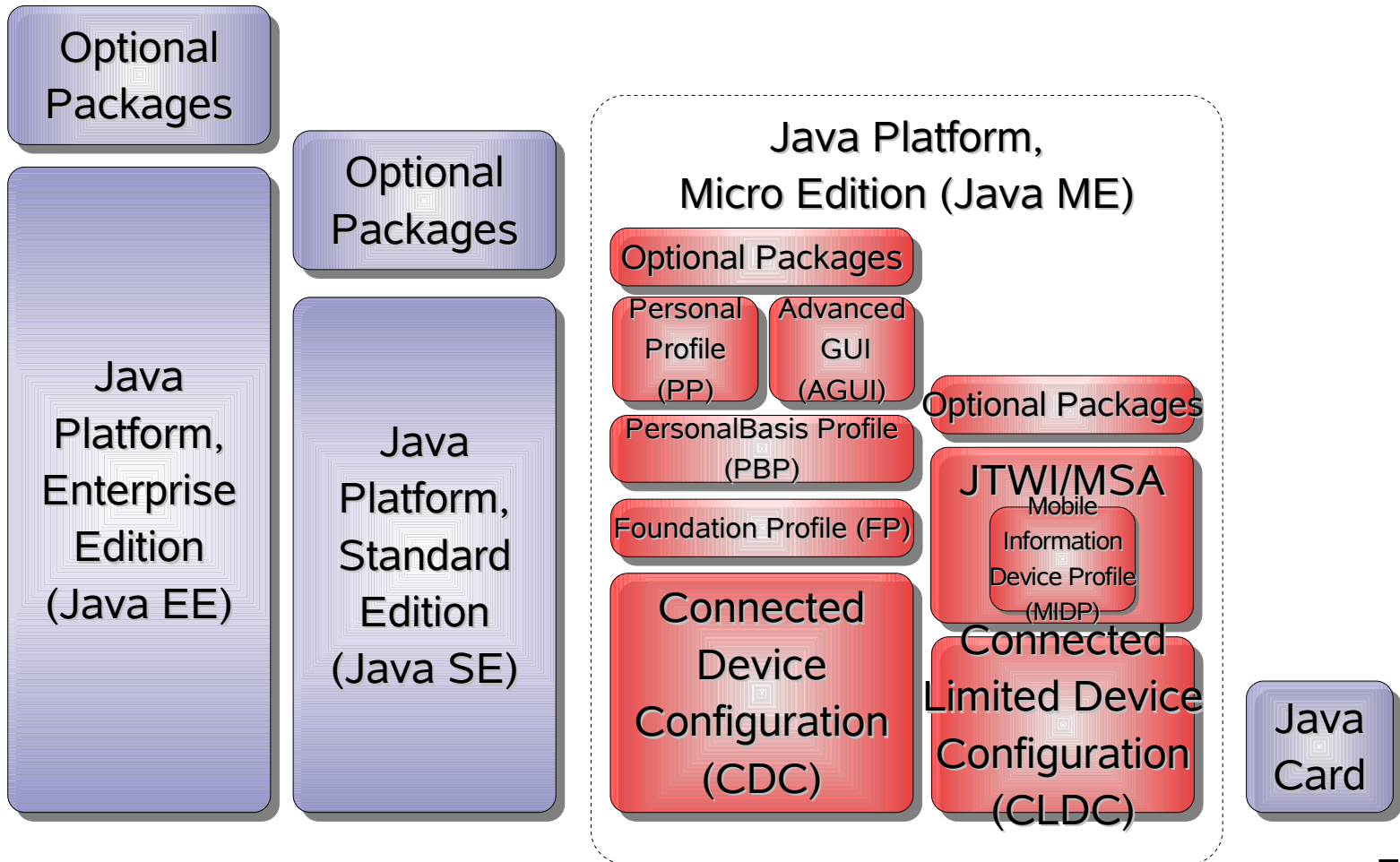
\$3 Billion Java  
Mobile Game Market



# Java ME Architecture



# Java Platforms



# Java ME CLDC Stack

- Connected Limited Device Configuration (CLDC)
  - > For devices with limited memory and processing power
  - > KVM (Kilo VM) Virtual Machine especially designed for devices with limited computing power and resources
- Mobile Information Device Profile (MIDP)
  - > Defines application model, OTA (over-the-air) installation, basic UI model and widgets, media capabilities, basic Game API, RMS (record management system), connectivity, security model

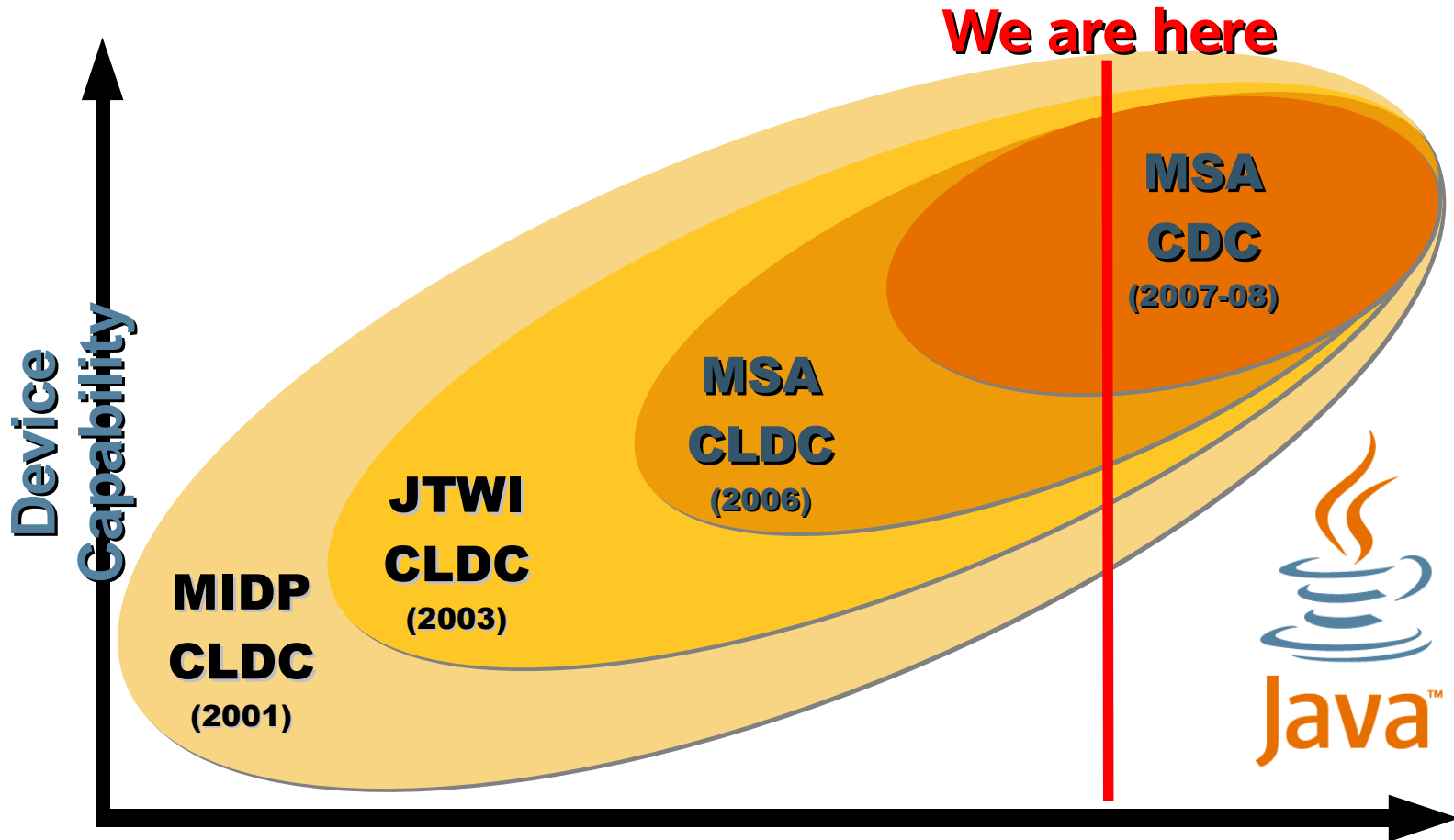


## Java ME CLDC Stack (2)

- Multitude of optional JSRs: Media, Bluetooth, Messaging, Location, SIP, and many more
- More JSRs are in the pipeline
- “Umbrella” JSRs
  - > Define a complete platform from existing JSRs
- Example
  - > JSR 185 (Java Technology for Wireless Industry/JTWI)
  - > JSR 248 (Mobile Services Architecture/MSA)

# Java ME Evolution in Wireless

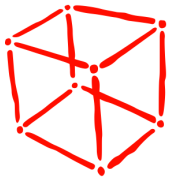
MIDP → JTWI → MSA (Mobile Service Architecture)



# Compelling Feature Phone Platform: MSA (JSR 248)

| Comms  | Graphics   | Security & Commerce   | Application Connectivity   | Personal Information   |
|--|--|---|--|--|
| <div style="border: 1px solid red; border-radius: 10px; background-color: #90EE90; padding: 5px; margin-bottom: 5px;">JSR 82<br/>Bluetooth</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 180<br/>SIP</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 205<br/>MMS<br/>Messaging</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 120<br/>SMS<br/>Messaging</div> | <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 226<br/>2D Scalable<br/>Vector Graphics</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 184<br/>3D Graphics</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 234<br/>Mobile-media<br/>Supplement</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 135<br/>Mobile Media</div> | <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 229<br/>Payment</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 177<br/>Security &amp;<br/>Trust Services</div> | <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 211<br/>Content<br/>Handler</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 172<br/>Web Services</div>                         | <div style="border: 1px solid red; border-radius: 10px; background-color: #90EE90; padding: 5px; margin-bottom: 5px;">JSR 179<br/>Location</div> <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 75<br/>PIM &amp; File</div> |
| <b>Application Environment</b>   | <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 185<br/>JTWI</div>   | <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 118<br/>MIDP 2.0</div>  | <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 238<br/>I18N</div>   |  |
| <b>Virtual Machine</b>   |  | <div style="border: 1px solid red; border-radius: 10px; background-color: #FFD700; padding: 5px; margin-bottom: 5px;">JSR 139<br/>CLDC 1.1</div>  | <div style="display: inline-block; width: 20px; height: 15px; background-color: #90EE90; border: 1px solid black; margin-right: 5px;"></div> Conditional APIs<br><div style="display: inline-block; width: 20px; height: 15px; background-color: #ADD8E6; border: 1px solid black; margin-right: 5px; margin-top: 5px;"></div> JTWI APIs |  |

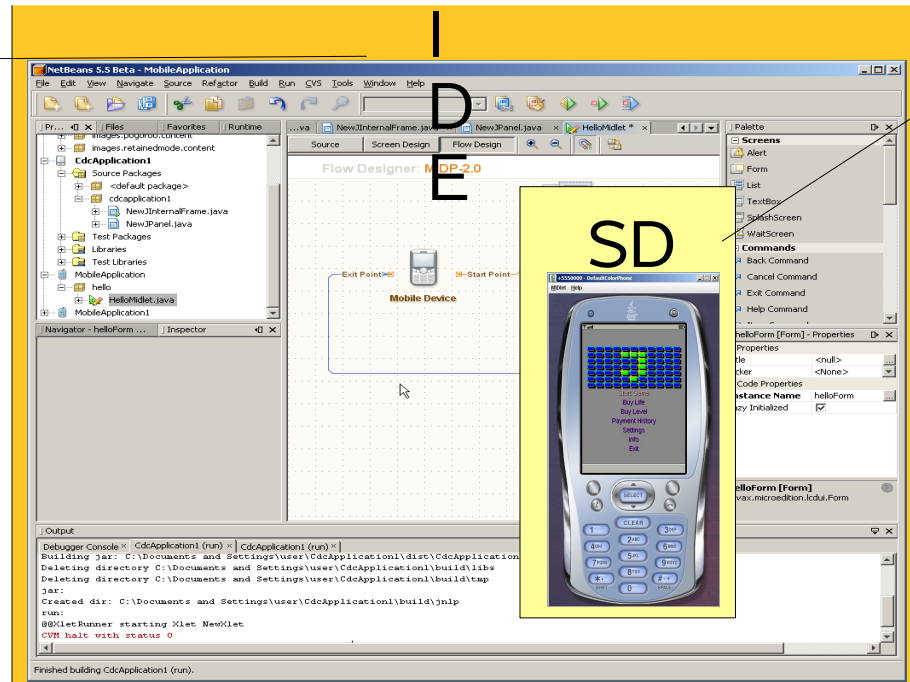
# NetBeans Mobility Pack



# NetBeans Mobility Pack

A full IDE built around Sun Java™  
Wireless Toolkit

- Open, integrated development environment
- Build, test, deploy rich applications
- Accelerate application time-to-market
- Visual, drag-and-drop authoring



- Java ME Platform emulation
- Device agnostic
- Provides additional utilities for application support

- Over 2,000,000 combined downloads
- Industry voted awards



# Visual Designer

The image displays the Sun Visual Designer interface for developing mobile applications. On the left is a mobile device mockup showing a screen with the Sun logo, a 'form' containing a 'stringItem' with the text 'Hello World', a 'choiceGroup' with two radio buttons labeled 'Choice Element 1' and 'Choice Element 2', and a 'Back' button at the bottom. The central area is a flowchart showing the application's logic flow between components like 'Mobile Device', 'helloForm', 'list', 'waitScreen', and 'smsComposer'. On the right, a 'Palette' lists various UI components such as Alert, Form, List, Text Box, File Browser, Login Screen, PIM Browser, SMS Composer, Splash Screen, Wait Screen, and Commands. Below the palette is a 'Device Screen' preview showing the rendered UI elements. At the bottom right, an 'Assigned Resources' panel lists 'Assigned Commands' (including 'backCommand'), 'Assigned Item Commands', and 'Resources'.

# SVG

- SVG = Scalable Vector Graphics
- Automatically adjusts to screen size
- Two roles
  - > Graphical designers create images
  - > Developers add business logic
- Visual designer for SVG
  - > SVG Menu, SVG Splash Screen, SVG Wait Screen, SVG Image, ...

# NetBeans 6.0 Mobility New Feature

- New game builder
  - > Easier to create mobile games with the Mobility Pack's visual editing support for the MIDP 2.0 Game API
  - > API supports animated sprites and the ability to arrange tiled layers into scenes
- New Integrated UI for CLDC/MIDP and CDC development
- New Visual Mobile Designer
- Design analysis
- New components for Flow Control



# Game Development

# Development of Java Technology-Based Games

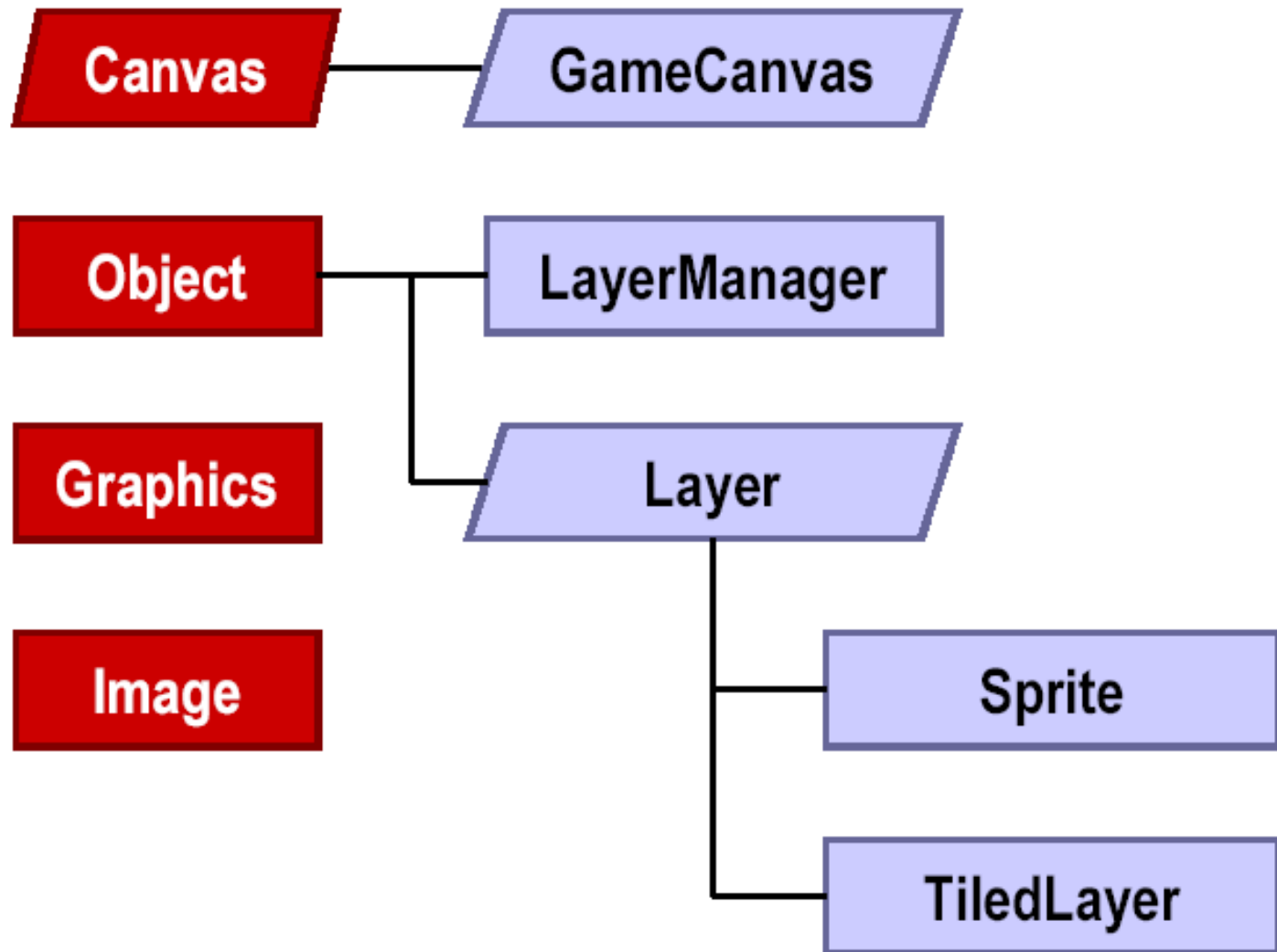


- JSR 184 (3D Graphics)
  - 3D world creation and manipulation
- JSR 135 (Mobile Media)
  - Sounds
- JSR 82 (Bluetooth)
  - P2P gaming
- JSR 180 (SIP)
  - P2P over the network
- JSR 229 (Payment)
  - Payment of new levels

# Examples of 2D Games



# Core Game API Class Hierarchy



# Game Canvas

- GameCanvas specifically for developing game display
  - > Class extends Canvas to use it
- Subclass of Canvas but provides the following improvements
  - > Off-screen buffering
  - > Keypad polling
  - > Access to Graphics object
  - > Do not need to override paint(Graphics) method
- Behavior can be set to be backward compatible with Canvas

# GameCanvas Example – Main Game Loop

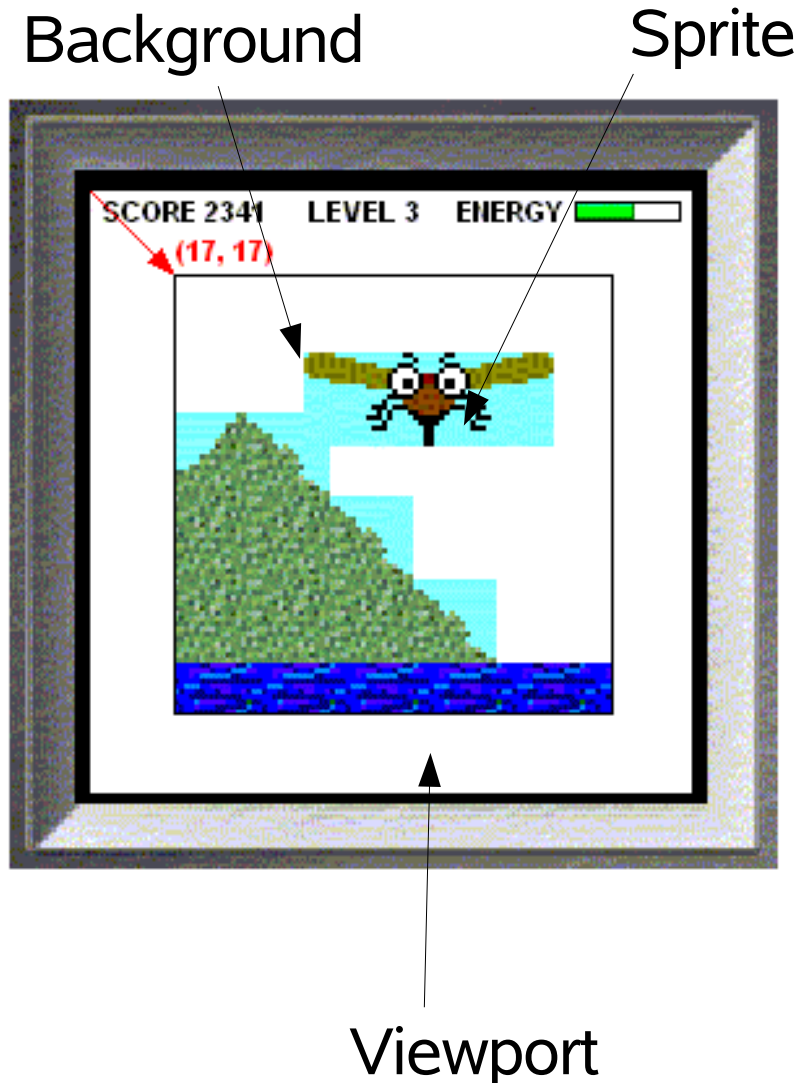
```

01 //main game loop
02 while (true) {
03     //Poll key press
04     int key = getKeyStates();
05     if ((key & GameCanvas.LEFT_PRESSED) != 0) {
06         //update game objects
07     }
08     if ((key & GameCanvas.UP_PRESSED) != 0) {
09         //update game objects
10     }
11     //Get off-screen graphics object
12     Graphics g = getGraphics();
13     //update game screen
14     ...
15     //update the 'dirty' portion to the screen
16     flushGraphics(x, y, dirtyWidth, dirtyHeight);
17 }

```

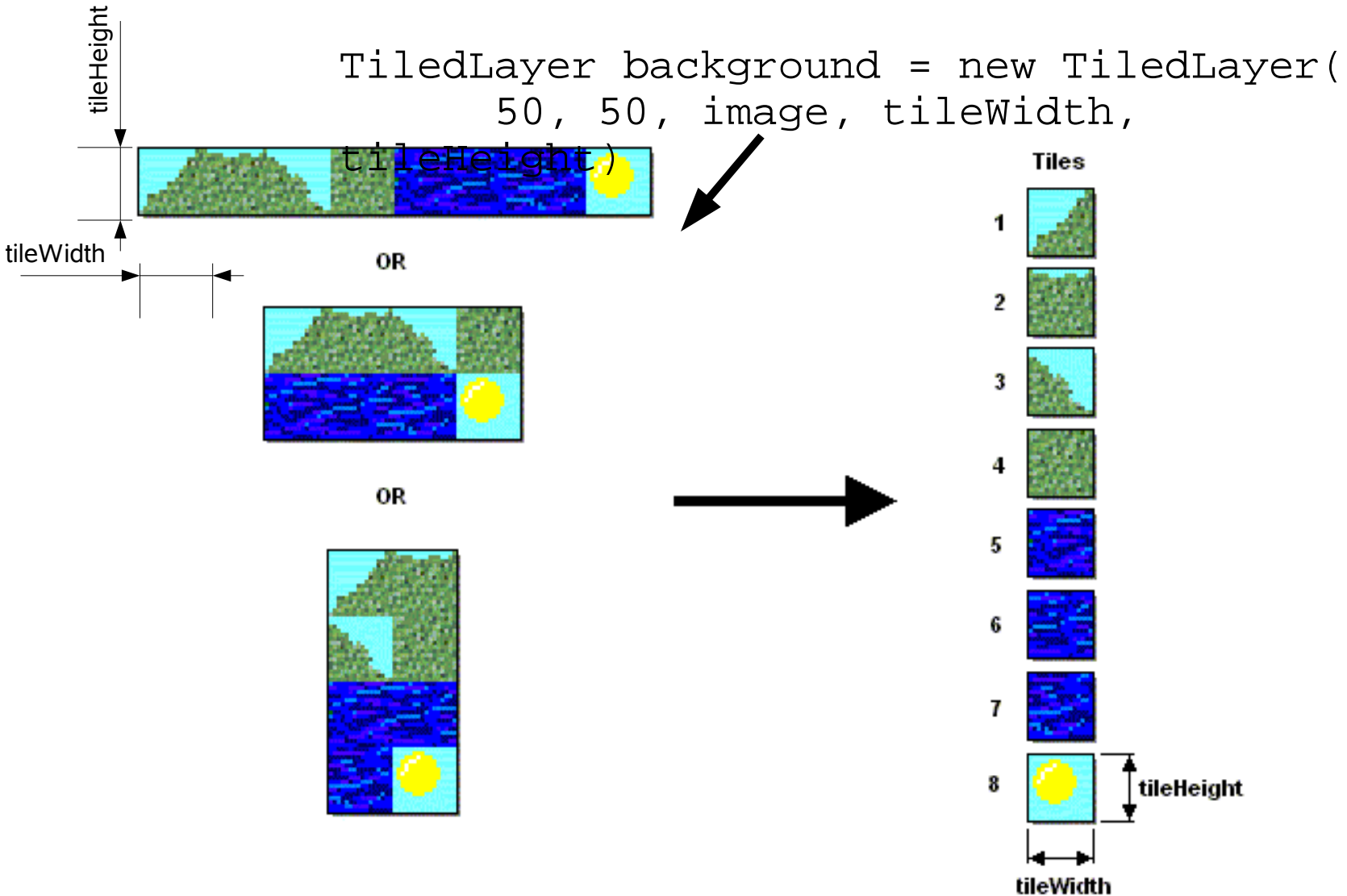


# Elements in a Game



- Background
  - > Composable from basic elements
  - > Static and animated objects
- Background layers
  - > Different layers
  - > Layer management
- Sprites
  - > Transformation
  - > Collision detection

# Creating a TiledLayer





# Creating the Background

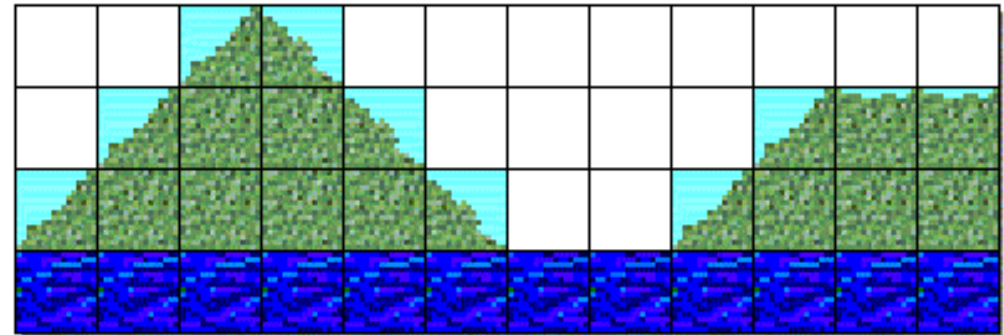
**Tiles**

**Cells**

|   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 4 | 4 | 3 | 0 | 0 | 0 | 0 | 1 | 2 | 2 |
| 1 | 4 | 4 | 4 | 4 | 3 | 0 | 0 | 1 | 4 | 4 | 4 |
| £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ |

tileHeight

tileWidth



```
// Draw the water
for (int x = 0; x < 12; x++)
    background.setCell(x, y, 5);
```

Tile index

# Creating Animated Tile

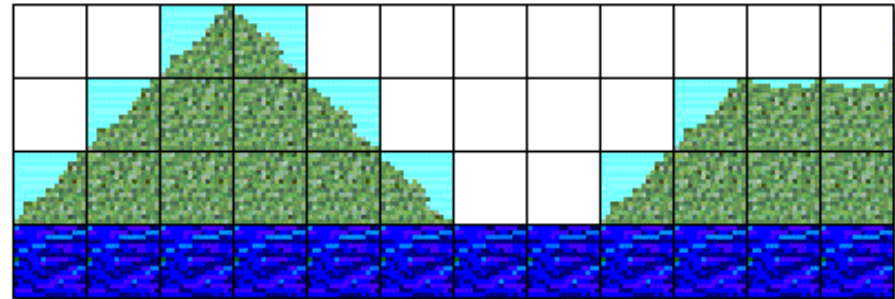
**Tiles**

**Cells**

|    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 0  | 0  | 1  | 3  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0  | 1  | 4  | 4  | 3  | 0  | 0  | 0  | 0  | 1  | 2  | 2  |
| 1  | 4  | 4  | 4  | 4  | 3  | 0  | 0  | 1  | 4  | 4  | 4  |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

**Animated Tiles**

`-1` = `5`



```
//Indicate the tile to be animated
int animatedTile = background.createAnimatedTile(5);
```

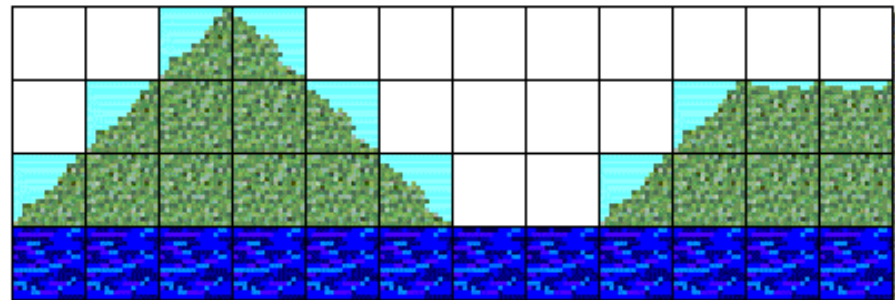
**Tiles**

**Cells**

|    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|
| 0  | 0  | 1  | 3  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0  | 1  | 4  | 4  | 3  | 0  | 0  | 0  | 0  | 1  | 2  | 2  |
| 1  | 4  | 4  | 4  | 4  | 3  | 0  | 0  | 1  | 4  | 4  | 4  |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

**Animated Tiles**

`-1` = `7`



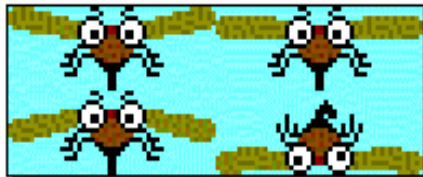
```
//Switch the animated tile
background.setAnimateTile(animatedTile, 6);
```

# Creating Sprites

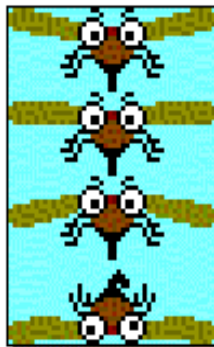
Loading is similar to TiledLayer



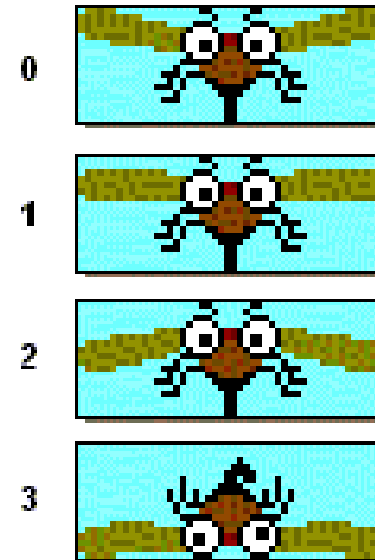
OR



OR



Default Frame Sequence

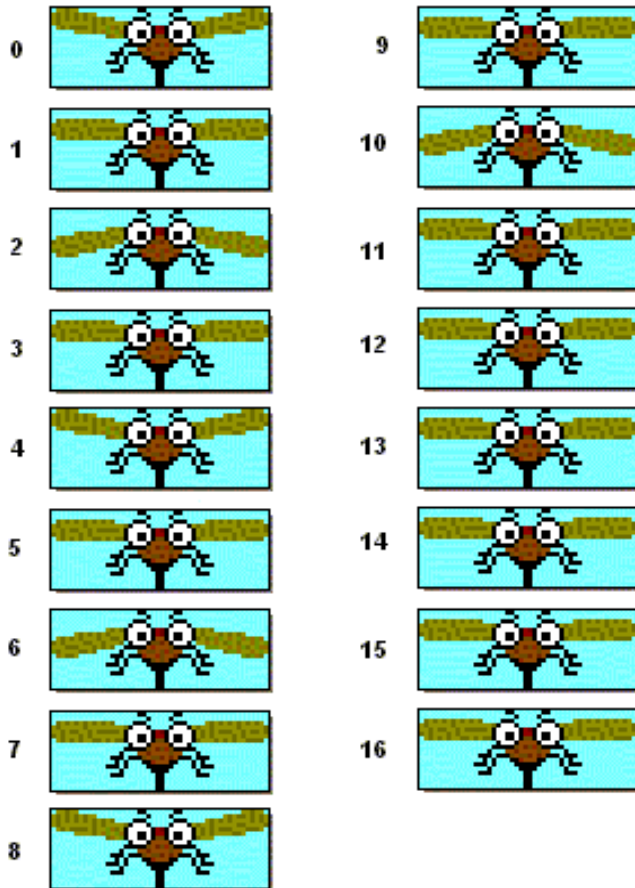


Calling `nextFrame()` will display the next frame

# Changing the Frame Sequence

Special Frame Sequence

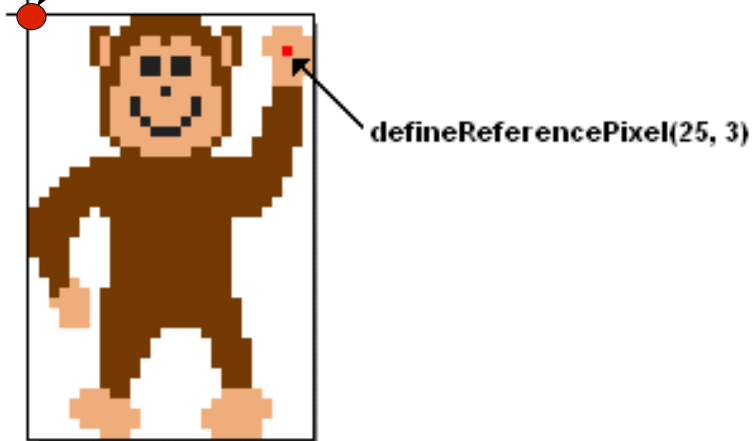
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 0 | 1 | 2 | 1 | 0 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



```
sprite.setFrameSequence
(new int[]
    {0, 1, 2, ... 1, 1});
sprite.nextFrame();
```

# Reference Pixel on Sprites

Default  
reference pixel  
at (0, 0)





# Real World Technologies: NetBeans GUI Builder, JRuby, JavaFX, and Java ME

