

David Qiao

JIDE Software, Inc.

# THE MERGING POINT OF ANDROID AND SWING

# Myself

- 10-year old Swing developer
- Author of most components offered by JIDE Software
- 1-year old Android developer
- Author of several popular Android apps: Advanced Ruler (free) and Advanced Ruler Pro (paid)

# What is Android?

- Android is a software stack for mobile devices that includes an operating system, middleware and key applications
- Allows any Java developers to make phone apps
  - It's in "Java"!
  - Familiar dev environment – Eclipse, NetBeans or IntelliJ IDEA

# Comparison

Swing	Android
JComponent	View
Container	ViewGroup
AbsoluteLayout	FrameLayout
FlowLayout	LinearLayout
Component.addMouseListener	View.setOnClickListener
JTable.setCellRenderer	GridView.setAdapter
Window.setContentPane	Activity setContentView

# Topics

- Resource Management
- Theme and Style

# Resource Management

# Resources

- What are resources?
  - Images
  - Strings
  - Layouts
  - Values (colors, dimensions, integers etc.)
  - Animations
  - Audios and videos
- Externalize the resources

# Android Project

```
MyProject/  
  src/  
    MainActivity.java  
  res/  
    drawable/  
      icon.png  
    layout/  
      main.xml  
      info.xml  
    values/  
      strings.xml
```

# Different Locales

- `res/values/strings.xml`

Contains English text for all the strings that the application uses

- `res/values-fr/strings.xml`

Contains French text for all the strings

- `res/values-ja/strings.xml`

Contains Japanese text for all the strings

# Different Devices

- `res/drawable/`  
Contains default graphics.
- `res/drawable-small-land-stylus/`  
Contains graphics optimized for use with a device that expects input from a stylus and has a QVGA low-density screen in landscape orientation.

# Resource Management Comparison

	Swing	Android
<b>Strings</b>	Properties file at any folder	XML file under res/values/ strings.xml
<b>Images</b>	Any folder	Image files under res/ drawable/ folder
<b>Layouts</b>	Mostly in Java source code, or form files of some GUI builders	XML files under res/layout/ (res/layout-hori/ or layout- land/)
<b>Colors</b>	In Java code or in UIDefaults	XML files under res/values/ color.xml
<b>Audios and videos</b>	Any folder	Under res/raw/

# What is Missing in Swing?

- Give too many choices to the developers
- The more choices, the more chances of making mistakes, and frustration

# James Gosling

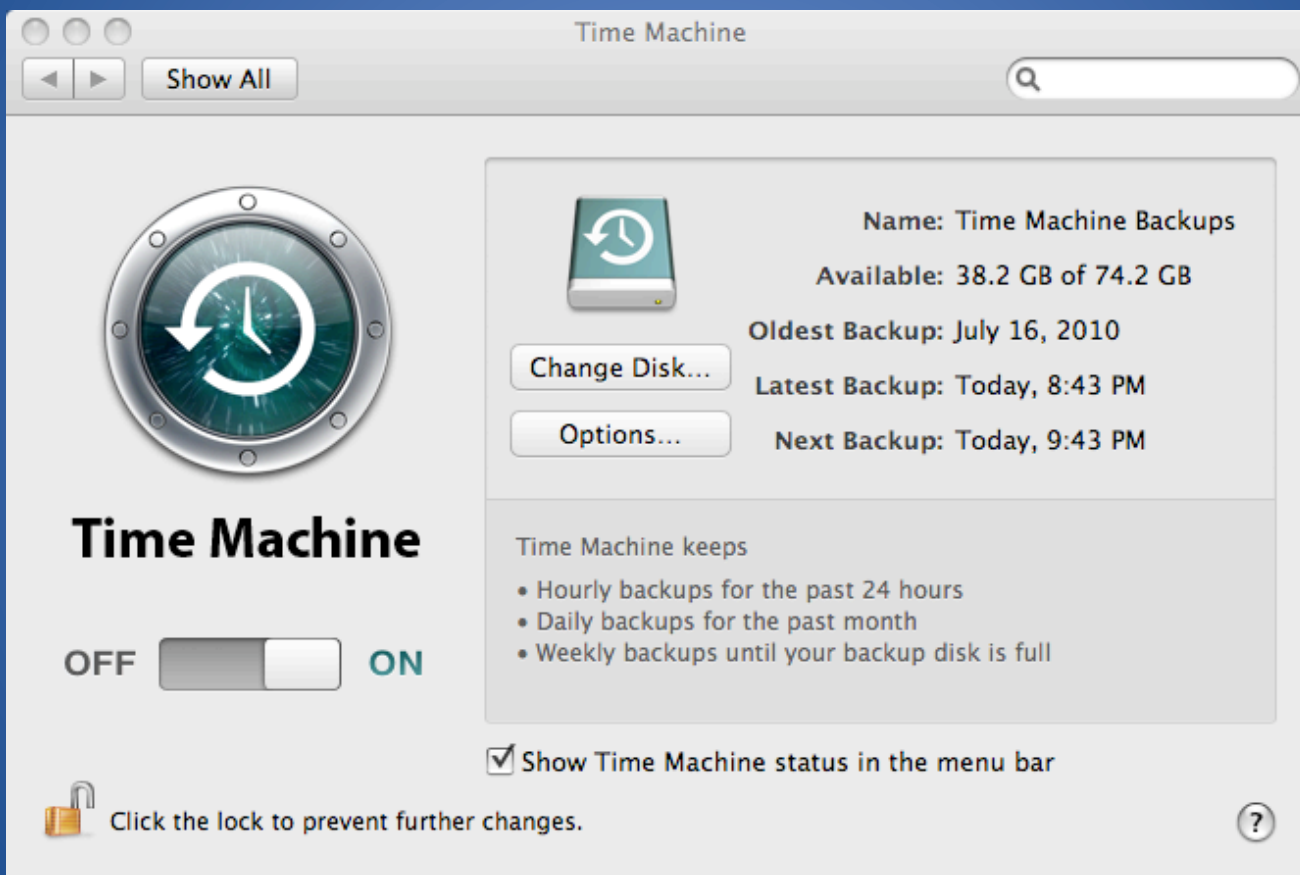
- “The biggest problem with Swing was that it can do so much that it's kind of become like the 747 cockpit if you like APIs. There's a lot of complexity there, and the hard part is figuring out how to use it”



# But

- **Not everyone is a pilot!!!**

# The Simplest is the Best



Genie Timeline Professional Edition

File Tools Help

### Select Data

Genie Timeline will scan your computer to protect the following items:

<p><b>Disaster Recovery</b> ✓ Protects Windows in case of system failure</p>	<p><b>Email</b> ✓ Outlook(2000-2010), Outlook Express, Windows Mail, Windows Live Mail, and contacts</p>	<p><b>Desktop</b> ✓ Files and folders on your Desktop</p>
<p><b>Office Files</b> ✓ Word, Excel, Access, PowerPoint, Visio, Publisher, and other document files</p>	<p><b>Financial Files</b> ✓ Financial files created by MS Money, Quickbooks, Quicken, TurboTax, TaxCut, and Peachtree</p>	<p><b>iPhone/iPad/iPod</b> ✓ Backup your synced iTunes files [?]</p>
<p><b>Pictures</b> ✓ Photos and images from your user account and local drives on your machine</p>	<p><b>Music</b> ✓ Music and audio from your user account and local drives on your machine</p>	<p><b>Videos</b> ✓ Video recordings on your machine</p>

Customize selections

help ?

Genie Timeline Professional Edition

File Tools Help Test Mode

Search Backup

**✓ You are currently 100% protected**

**Status:** All backed up - Monitoring changes

Last Backup: 19 min(s) ago  
Next Backup: 30 min(s)  
Protected Since: 3/25/2010 12:50 PM  
Backup Info: Not compressed or encrypted  
Backed Up: 92 Items

Pause

Genie Timeline constantly monitors your system for changes.  
Backups will run at specified intervals, which can be viewed in [Timeline Explorer](#).

- The recent sections of The Timeline contain more versions.
- Older backups are collected into fewer ticks.
- Home and Pro versions of Genie Timeline will purge/consolidate older backups to save space.

Restore
 

- Restore from Timeline
- Advanced Restore
- Disaster Recovery

Manage Backup
 

- Modify data selections
- Change backup drive
- Start a new backup

Tools
 

- Configure email notifications
- Disaster Recovery startup disk
- Advanced Settings

Drive (E)  
13% Free Space  
Capacity: 558.00 GB

Outlook (2002-2007)
  Videos
  Music
  Pictures
  Other
  Free Space

Genie Timeline Professional Edition

File Tools Help

Smart Selection

Mark the checkbox

- Libraries
- User
- Computer
  - Floppy Disk Drive (A:)
  - Local Disk (C:)
  - DVD RW Drive (D:) CD\_ROM
  - New Volume (E:)
  - dev\_data (\sgnsa) (E:)
  - public (\sgnsa) (Z:)
  - Timeline Explorer
- Network
- Control Panel
- Recycle Bin
- FAU.x64
- FAU.x86
- New folder
- No-Backup Zone
- test

- Floppy Disk Drive (A:)
- Local Disk (C:)
- DVD RW Drive (D:) C...
- New Volume (E:)
- dev\_data (\sgnsa) (E:)
- public (\sgnsa) (Z:)
- Timeline Explorer

Revert to pre-selected data

Filter my selections

Select All

Clear All

help ?

« previous next »

# Developer Experience Comparison

Tasks	Swing	Android
Missing resources	Runtime error	Compile time error
Resource Lookup	Look up in folder or file	Code completion
Refactoring	Change file name then change references	IDE support
Locations	Multiple locations	Single location (under res folder)
Learning Curve	Steep	Almost none

# Android ID'nize all resources

- Assign a unique int ID for each resource so that all resources can be referred using a unique ID.
- On Android, a tool called *aapt* will do it

# Resource Class on Android

```
package com.example.helloandroid;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int textview=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

# Using ID'nized Resources

- `getWindow().setBackgroundDrawableResource(R.drawable.my_background_image);`
- `getWindow().setTitle(getResources().getText(R.string.main_title));`
- `getWindow().setTitle(R.string.main_title);`
- `// Load a custom layout for the current screen`
- `setContentView(R.layout.main_screen);`

What to Learn from Android?

# Resource to ID

- A utility, just like Android's *aapt* to parse all resources and create resource classes that contains the unique IDs.
  - Under `/com/company/project/res`
- Could be an ant target, a plug-in to IDE so that it will run automatically when needed
  - Generate `com.company.project.res.R` class

# Resource Location?

- A single location as Android does?
  - Probably not.

# API Changes

- AbstractButton only has setText(String)
- There is no setText(int id)

# API Changes

- ResourceUtils res = ResourceUtils.getInstance(new R());
- button.setText(String)
- **New:** res.setText(AbstractButton, int id)
- component.setForeground(Color)
- **New:** res.setForeground(JComponent, int id)

# ResourceUtils

- Placeholder for methods that set resource related properties
- Listen to locale changes and when locale changes, it will automatically update the component text or tooltip or icon based on the ID

# Methods

- setText (JLabel or AbstractButton)
- setToolTipText (JComponent)
- setIcon (JLabel and JButton),
- setDisabledIcon (Rollover, Pressed, Selected etc. for AbstractButton)
- Mnemonic using Window convention (i.e. “&Title” is Title).

# Benefit

- Code completion can be used to find the resource without looking into the properties files or the image folders
- Compile time error if the resource is missing
- Change locale on fly

# Demo Resource Management

# Theme and Style

# A TextView in Android

```
<TextView  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:textColor="#00FF00"  
    android:typeface="monospace"  
    android:text="@string/hello" />
```

# After using Style

```
<TextView  
    style="@style/CodeFont"  
    android:text="@string/hello" />
```

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">  
        <item name="android:layout_width">fill_parent</item>  
        <item name="android:layout_height">wrap_content</item>  
        <item name="android:textColor">#00FF00</item>  
        <item name="android:typeface">monospace</item>  
    </style>  
</resources>
```

# Style

- On Android, “style” is nothing but a bunch of properties for a control
- Support simple inheritance

```
<style name="GreenText" parent="@android:style/TextAppearance">  
  <item name="android:textColor">#00FF00</item>  
</style>
```

# Theme

- When style is used on an application or activity, it becomes theme

```
<style name="CustomDialogTheme" parent="@android:style/Theme.Dialog">  
    <item name="android:windowBackground">@drawable/custom_dialog_background</item>  
</style>
```

```
<activity android:theme="@style/CustomDialogTheme">
```

# A Sample Theme

```
<style name="Theme">

    <item name="colorForeground">@android:color/bright_foreground_dark</item>
    <item name="colorForegroundInverse">@android:color/bright_foreground_dark_inverse</item>
    <item name="colorBackground">@android:color/background_dark</item>
    <item name="disabledAlpha">0.5</item>
    <item name="backgroundDimAmount">0.6</item>

    <!-- Text styles -->
    <item name="textAppearance">@android:style/TextAppearance</item>
    <item name="textAppearanceInverse">@android:style/TextAppearance.Inverse</item>

    <item name="textColorPrimary">@android:color/primary_text_dark</item>
    <item name="textColorSecondary">@android:color/secondary_text_dark</item>
    <item name="textColorTertiary">@android:color/tertiary_text_dark</item>
    <item name="textColorPrimaryInverse">@android:color/primary_text_light</item>
    <item name="textColorSecondaryInverse">@android:color/secondary_text_light</item>
    <item name="textColorTertiaryInverse">@android:color/tertiary_text_light</item>
    <item name="textColorPrimaryDisableOnly">@android:color/primary_text_dark_disable_only</item>
    <item name="textColorPrimaryNoDisable">@android:color/primary_text_dark_nodisable</item>
    <item name="textColorSecondaryNoDisable">@android:color/secondary_text_dark_nodisable</item>
    <item name="textColorPrimaryInverseNoDisable">@android:color/primary_text_light_nodisable</item>
    <item name="textColorSecondaryInverseNoDisable">@android:color/secondary_text_light_nodisable</item>
    <item name="textColorHint">@android:color/hint_foreground_dark</item>
    <item name="textColorHintInverse">@android:color/hint_foreground_light</item>
```

# Java Look And Feel

- Theme is equivalent to Java L&F.
- UIDefaults table in Swing is the same as the style XML file in Android
- Synth LookAndFeel already uses XML file to define L&F

# Synth XML

```
<synth>
  <style id="defaultStyle">
    <font name="Verdana" size="16"/>
    <state>
      <color value="WHITE" type="BACKGROUND"/>
      <color value="BLACK" type="FOREGROUND"/>
    </state>
  </style>
  <bind style="defaultStyle" type="region" key=".*"/>
  <style id="textfield">
    <state>
      <color value="yellow" type="BACKGROUND"/>
    </state>
    <imagePainter method="textFieldBorder" path="textfieldborder.png"
      sourceInsets="5 6 6 7" paintCenter="false"/>
    <insets top="5" left="6" bottom="6" right="7"/>
  </style>
  <bind style="textfield" type="region" key="TextField"/>
</synth>
```

# Problems with Synth

- Too complex
  - Unnecessary hierarchy
- Limited binding
  - Static binding only
    - i.e. can't do thing like `tabbedPane.loadStyle("newStyle.xml");`
  - By region or name only
    - i.e `tabbedPane.setName("newStyle")`. What if the name is used for other purpose?
- Poor extensibility for 3<sup>rd</sup> party components

# Nimbus LookAndFeel

- Built on top of Synth
- The first pure vector based L&F
- Dynamic binding (can define new state)
- Flat

# Problems with Nimbus

- Very complex
  - You will be surprised if you try to read nimbus source code
  - No XML
- Too ambitious
  - Pure vector based L&F
  - Instead of using vector, I can create three images at different resolutions which will cover for at least next five years

# What is REALLY a LookAndFeel

- A bunch of colors
  - control, controlText, Label.foreground, ...
- A bunch of fonts
  - Label.font, TextField.font, ...
- A bunch of painters
  - To paint components in different states.
- More flags for each component
  - To fine tune the L&F, such as iconTextGap, insets.

# XML? Or just plain text?

- I don't really care
- I choose properties file in favor of simplicity

# Creating a new L&F

- `load(packageName + "colors");`
- `load(packageName + "button"); // painters and flags for button`
- `load(packageName + "comboBox"); // painters and flags for combobox`
- ...

# Extending a L&F

- Customize a LookAndFeel
  - Just call `load(packageName + "myApp")`, overwrite existing UIDefaults
- Extend a LookAndFeel
  - Just call `load(packageName + "newComponent")`, a file with new properties and add to UIDefaults
- Customize a particular component or panel
  - `apply(component, packageName + "myComp")`
  - `applyRecursively(panel, packageName + "myPanel")`

# Painter

- Painter is the way to go
- Painter could be image based or Java code (vector)
  - Concrete Implementations: ColorPainter, GradientPainter, ImagePainter
  - For different resolutions, load a different set of properties files

# Demo

## Look and Feel

Q & A