

*Simplifying Desktop  
Development  
with  
Glimmer*

by  
Annas “Andy” Maleh  
andy@obtiva.com  
Obtiva Corp.

# Preview

- Introduction
- Overview of Widgets
- Hello World
- Glimmer Listens
- Data-binding
- Test-driving with MVP
- First Glimmer Game
- Latest Features
- In the lab
- Future

# *Introduction*

- Requirements for a Ruby UI Toolkit:
  - Platform independence
  - Native widget support
  - Industry strength
  - Ease of use

# *Introduction*

- Current Ruby UI Toolkits:
  - TK
  - FXRuby
  - GTK
  - wxWidgets
  - Monkeybars
  - Shoes
  - Limelight
  - ...

# Introduction

- Introducing Glimmer:
  - Leverages SWT library used in Aptana RadRails IDE
  - Works on Windows, Mac, and Linux
  - Offers built in native widget support
  - Provides an ultra-light UI authoring DSL
  - Facilitates clean separation of logic and UI code via Data-Binding

# Overview of Widgets

The screenshot shows a window titled "JRuby on SWT" with a standard OS title bar (minimize, maximize, close). The form contains the following elements:

- Text Fields:** "First Name:" with value "Milo", "Last Name:" with value "Todorovich".
- Spinner:** "Age:" with value "35".
- Tabs:** "Profile" (selected), "Members", "Country".
- Gender:** Radio buttons for "Male" (selected) and "Female".
- Job:** Checkboxes for "Programmer" and "Presenter", both checked.
- Friends:** A list box containing "Owen Mardi", "Heidi Claus", and "Kevin Bowl". "Owen Mardi" is selected.
- Nationality:** A dropdown menu with "U.S.A." selected.
- Progress Bar:** A horizontal progress bar with 10 segments, approximately 70% filled.
- Submit:** A "Submit" button at the bottom.

# Overview of Widgets

- Shell
  - Composite
    - Group
    - TabFolder
    - Combo
    - List
    - Control
      - Text
      - Label
      - Spinner
      - Button

# *Hello World*



# Hello World

- SWT example in Java:

```
Display display = Display.getDefault();
Shell shell = new Shell(display);
shell.setText("SWT");
shell.setLayout(new FillLayout());
Label label = new Label(composite, SWT.NONE);
label.setText("Hello World!");
shell.pack();
shell.open();
while (!display.isDisposed()) {
    if (!display.readAndDispatch()) {
        display.sleep();
    }
}
display.dispose();
```

# *Hello World*

- API Goals
  - Concise and DRY
  - Asks for minimum info needed to accomplish task
  - Convention over configuration
  - As predictable as possible for existing SWT developers

# Hello World

- Glimmer example in JRuby:

```
shell {  
  text "SWT"  
  label {  
    text "Hello World!"  
  }  
}
```

# *Glimmer Listens*

- Listeners allow user interaction with widgets

```
text {
  on_modify_text {
    contact_model.validate
  }
  on_focus_lost {
    contact_model.save
  }
}
```

```
button {
  text "Delete"
  on_widget_selected {
    contact_model.delete
  }
}
```

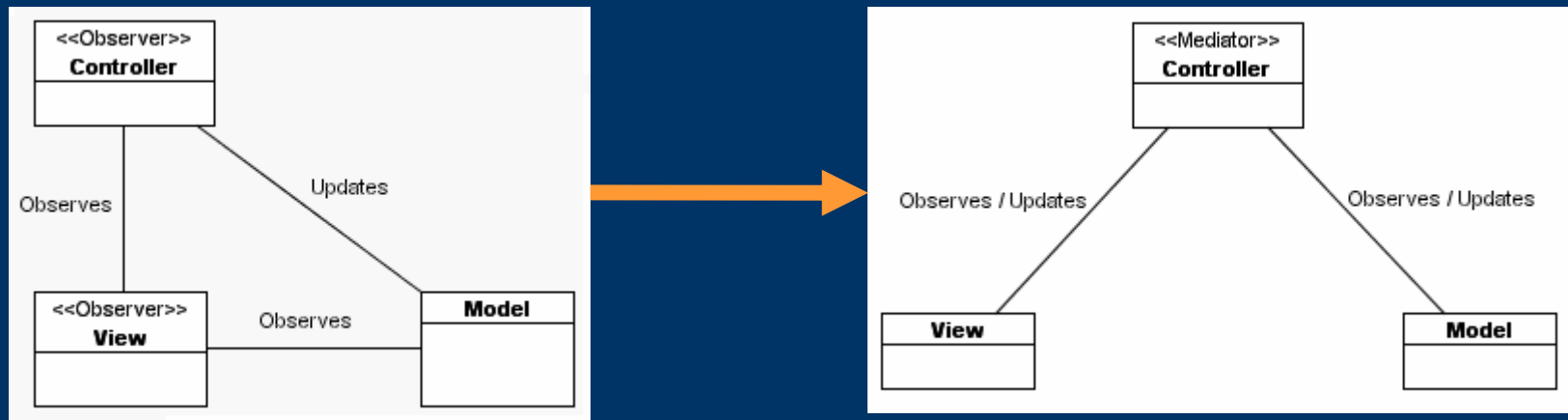
*Glimmer Listens*

*Demo*

(<http://glimmer.rubyforge.org/svn/samples/login.rb/>)

# Data Binding

- Enables syncing of UI state with model state bi-directionally
- Facilitates clean separation between application logic and UI code
- Requires much less code than listeners
- Relies on modified version of MVC pattern



# *Data Binding*

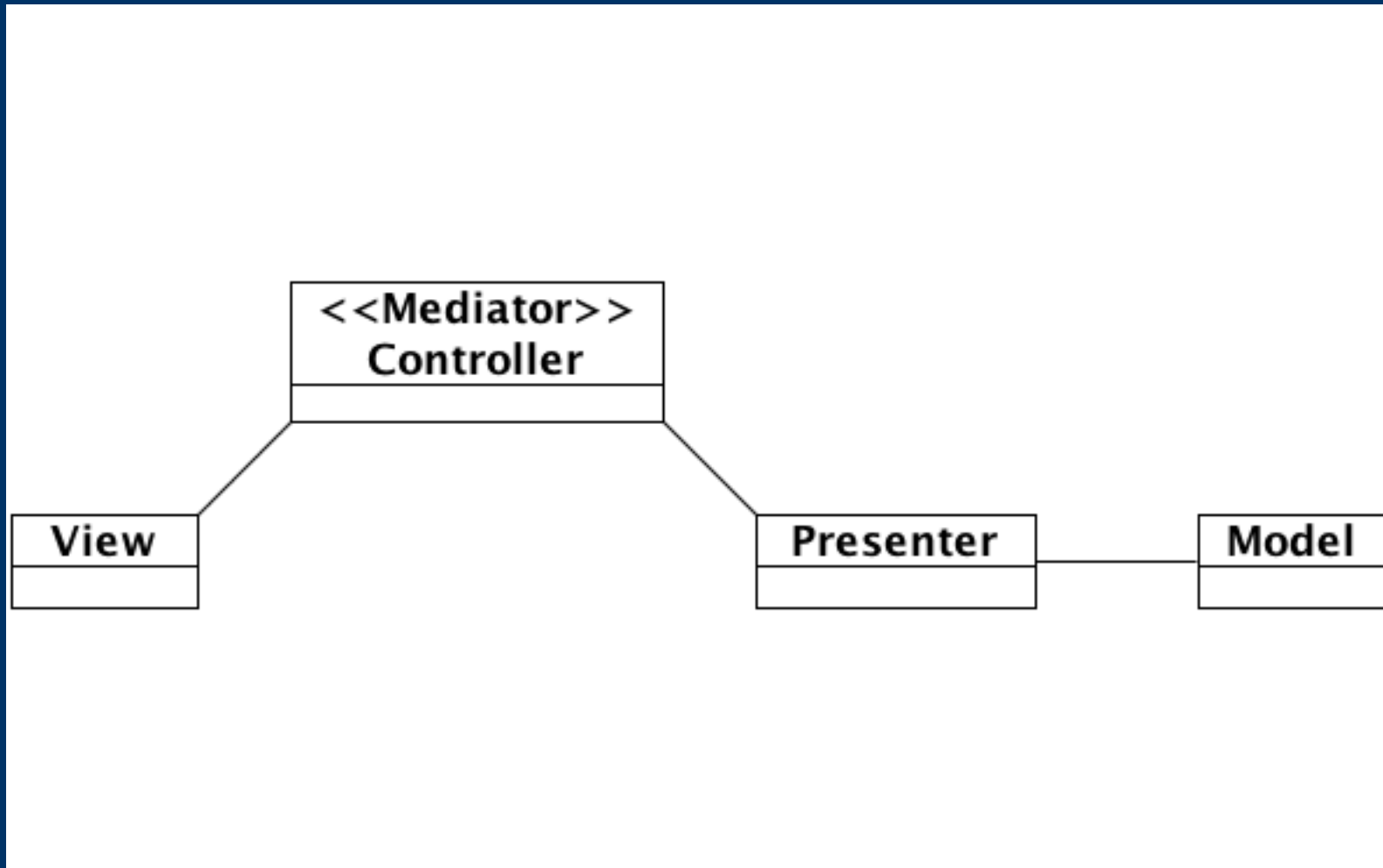
# *Demo*

([http://glimmer.rubyforge.org/svn/samples/contact\\_manager/](http://glimmer.rubyforge.org/svn/samples/contact_manager/))

# *Test-driving with MVP*

- A presenter is a special kind of model that handles presentation logic
  - Examples:
    - Enablement/disablement of fields based on certain events
    - Action logic triggered by clicking a button
    - Aggregation/formatting of data to display
- Abstracts the UI
  - Properties represent different fields on the UI
  - Methods represent different action widgets (e.g. buttons) on the UI

# Test-driving with MVP



# *Test-driving with MVP*

- Write UI code using Glimmer (without data-binding and event listeners)
- Test-drive presenter that abstracts UI
  - Every UI field is represented by a property on the presenter
  - Every UI button is represented by a method on the presenter
- Update UI code to data-bind

# *First Glimmer Game*

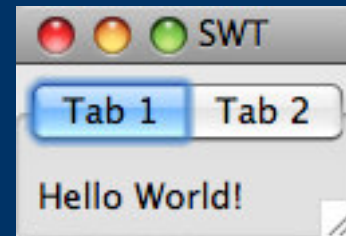
# *Demo*

(<http://glimmer.rubyforge.org/svn/samples/tictactoe/>)

# Latest Features

- Simpler Tab Syntax

```
shell {  
  tab_folder {  
    tab_item {  
      text "Tab 1"  
      label {  
        text "Hello World!"  
      }  
    }  
    tab_item {  
      text "Tab 2"  
      label {  
        text "Bonjour Univers!"  
      }  
    }  
  }  
}
```

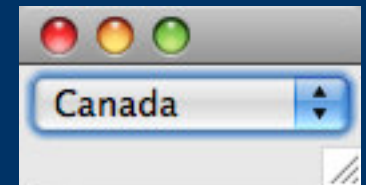


# Latest Features

- Combo Box Data-Binding by Convention

```
class Person
  attr_accessor :country, :country_options
  def initialize
    self.country_options=["", "Canada", "US", "Mexico"]
    self.country = "Canada"
  end
end

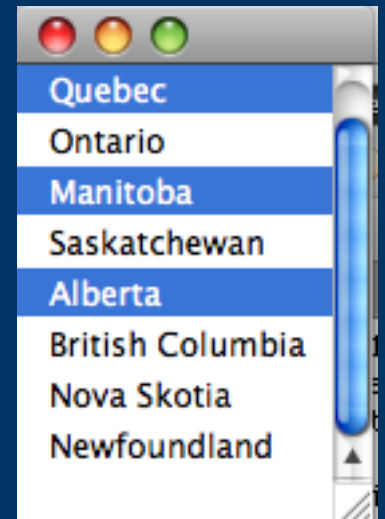
class HelloCombo
  include Glimmer
  def launch
    shell {
      combo(read_only) {
        selection bind(Person.new, :country)
      }
    }.open
  end
end
```



# Latest Features

- List Data-Binding by Convention

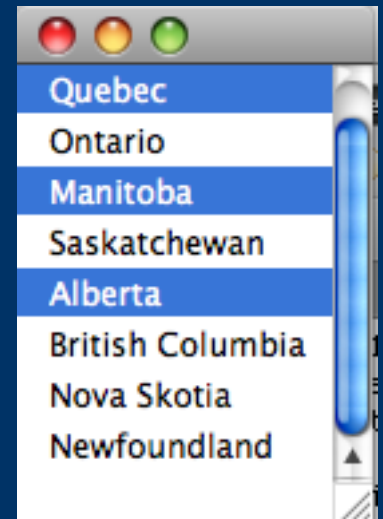
```
class Person
  attr_accessor :provinces, :provinces_options
  def initialize
    self.provinces_options=[
      ""
      ,
      "Quebec",
      "Ontario",
      "Manitoba",
      "Saskatchewan",
      "Alberta",
      "British Columbia",
      "Nova Skotia",
      "Newfoundland"
    ]
    self.provinces = ["Quebec", "Manitoba", "Alberta"]
  end
end
```



# Latest Features

- List Data-Binding by Convention

```
class HelloMultiSelectList
  include Glimmer
  def launch
    shell {
      list(multi) {
        selection bind(Person.new, :provinces)
      }
    }.open
  end
end
```



# In The Lab

- Acceptance Testing via Cucumber

Feature: Login feature

In order to keep user data secure

As a user

I would like to be authenticated before I can access my data

Scenario: Login

Given I fill in "User" for "Username"

And I fill in "Pass" for "Password"

When I "Login"

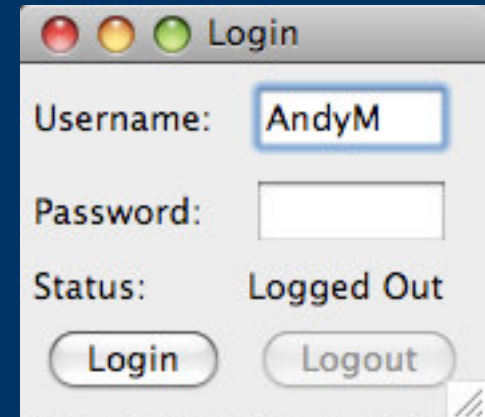
Then my status should be "Logged In"

Scenario: Logout

GivenScenario Login

When I "Logout"

Then my status should be "Logged Out"



# Future

- Glimmer editor plugin for Eclipse
- Table selection data-binding support
- Tree data-binding support
- Context menu support
- Drag and drop support
- CSS-like styling capabilities
- Animation
- ... more at  
<http://www.eclipse.org/glimmer/upcoming.php>

# *More Information*

Glimmer Eclipse Technology Project Page:

<http://www.eclipse.org/glimmer>

Glimmer Newsgroup:

<http://www.eclipse.org/newsportal/thread.php?group=eclipse.technology.glimmer>

Sneak Peak at Glimmer:

<http://andymaleh.blogspot.com/2007/11/sneak-peak-at-glimmer.html>

Glimmer Data-Binding:

<http://andymaleh.blogspot.com/2007/12/glimmers-built-in-data-binding-syntax.html>

Glimmer Listens:

<http://andymaleh.blogspot.com/2007/12/listeners-in-glimmer.html>

# *Contact*

**Andy Maleh**  
**andy@obtiva.com**  
**andymaleh.blogspot.com**

**Obtiva Corp.**  
**www.obtiva.com**