

# Swing Application Framework and Beans Binding

Roman Strobl  
[roman.strobl@sun.com](mailto:roman.strobl@sun.com)

# Agenda

- **Swing app. framework**
- Beans binding
- Putting it all together
- Conclusion

# Swing App. Framework

- **Swing v 0.1 debuts in 1997**
- **It's widely adopted now**
- **But there was no standard framework**

# Motivation

```
public static void main(String args[]) {  
    // good luck!  
}
```

# Framework Goals

- **Goals:**

- As small as possible
- Explain it all in an hour
- Work well for small/medium apps

- **Out of scope:**

- Modular system, docking, filesystems, data models, help system, updates, etc.
- Not a replacement for NetBeans platform

# Framework Features

- **Lifecycle support**
- **Resources**
- **Actions**
- **Tasks**
- **Session state**

# Lifecycle Support

## Plain Swing

```
import javax.swing.*;

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

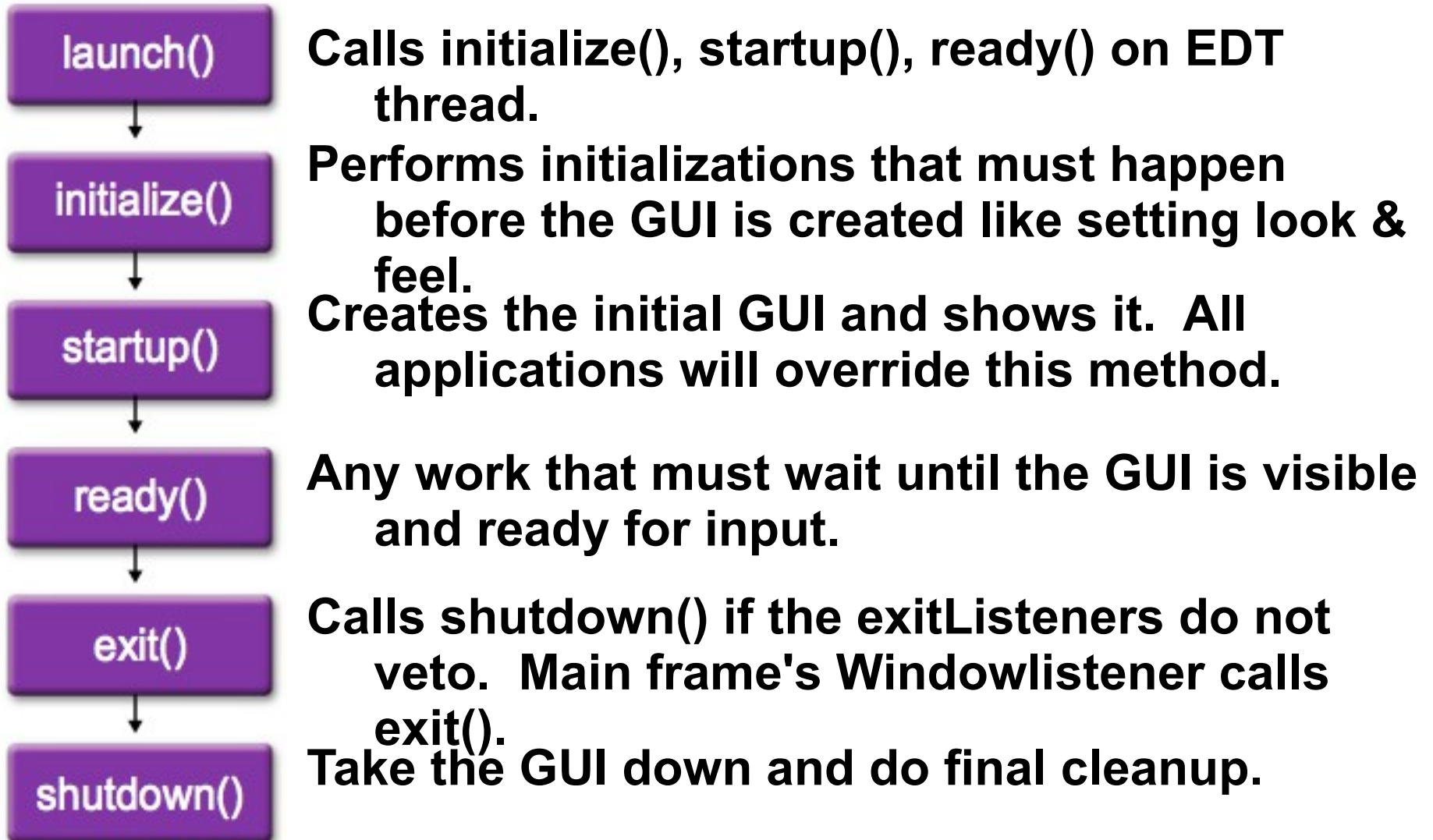
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

# Lifecycle Support

## Swing application framework

```
public class MyApp extends SingleFrameApplication {
    @Override protected void startup() {
        JLabel label = new JLabel("Hello World");
        show(label);
    }
    public static void main(String[] args) {
        Application.launch(MyApp.class, args);
    }
}
// SingleFrameApplication is a subclass of Application
```

# Lifecycle Support



# Resources

- **Defined with regular ResourceBundles**
- **ResourceManager class is used**
- **Three options:**
  - Manually load, use ResourceMap objects
  - Automatically inject resources into UI components
  - Automatically inject resources into object fields

# Manual Loading

```
# resources/MyForm.properties
```

```
aString = Just a string
```

```
aMessage = Hello {0}
```

```
anInteger = 123
```

```
aBoolean = True
```

```
anIcon = myIcon.png
```

```
aFont = Arial-PLAIN-12
```

```
colorRGBA = 5, 6, 7, 8
```

```
color0xRGB = #556677
```

```
ApplicationContext c =
```

```
Application.getContext(MyForm.class).getInstance();
```

```
ResourceMap r = c.getResourceMap(MyForm.class);
```

```
r.getString("aMessage", "World") => "Hello World"
```

```
r.getColor("colorRGBA") => new Color(5, 6, 7, 8)
```

```
r.getFont("aFont") => new Font("Arial", Font.PLAIN, 12)
```

# Resource Injection

## Resource file:

```
btnShowTime.text = Show current time!  
btnShowTime.icon = refresh.png
```

## Java class:

```
btnShowTime = new JButton();  
txtShowTime = new JTextField();  
  
btnShowTime.setName("btnShowTime");  
txtShowTime.setName("txtShowTime");
```

- **Resources are injected automatically!**

# Object Field Injection

Resource file:

```
MyPanel.greetingMsg = Hello, %s, a string was injected!
```

Java class:

```
@Resource
```

```
String greetingMsg;
```

```
ResourceMap resource =
```

```
    ctxt.getResourceMap(MyPanel.class);
```

```
resource.injectFields(this);
```

```
String personalMsg = String.format(greetingMsg,  
    txtName.getText());
```

```
JOptionPane.showMessageDialog(this, personalMsg);
```

# Let's Look at Some Code

Resources

# Actions

- **Instead of ActionListeners**
- **Asynchronous actions are easy**
- **@Action annotation**
  - Possible simple logic

```
@Action(enabledProperty = "changesPending")  
public void save() { ... }
```

```
public boolean getChangesPending() { ... }
```

# Actions

```
// define sayHello Action - pops up message Dialog
@Action public void sayHello() {
    String s = textField.getText();
    JOptionPane.showMessageDialog(s);
}
```

```
// use sayHello - set the action property
Action sayHello = Application.getContext().
getActionMap(MyForm.class, this).get("sayHello");
textField.setAction(sayHello);
button.setAction(sayHello);
```

# Let's Look at Some Code

Actions

# Tasks

- **Inherit the SwingWorker API**
- **Support for monitoring**
  - Title, Start/Done, etc.
- **Asynchronous @Actions return Tasks**
- **Easy to handle non-blocking actions**

# Session State

- **Easily save properties:**
  - Window size
  - Application position
  - Sizes of columns in jTable, etc.
- **ApplicationContext.getSessionStorage()**
- **SessionStorage.save(RootComponent, filename)**
  - Default for SingleFrameApplication is windowname + “.session.xml”
- **Uses XMLEncoder & XMLDecoder**

# JSR 296 vs. NB Platform

- **JSR 296 vs. NetBeans Platform**
- **Platform is better for:**
  - Applications that can reuse platform APIs
  - Editor-based applications
  - If you need update centers, window system, file systems, data systems, help system, visual designers, explorers, etc.
- **JSR 296 is much easier to learn**

# When and Where?

- **When will JSR-296 be in a JDK?**
  - Planned - Java Platform 7
- **I can't wait. Where can I get it?**
  - Bundled with NetBeans 6 (version 1.0)
  - Source available as a NetBeans project at <https://appframework.dev.java.net/>

# Demo

## JSR 296 - Flickr Demo

<http://wiki.netbeans.org/wiki/view/NBDemoFlickr>

# Agenda

- Swing app. framework
- **Beans binding**
- Putting it all together
- Conclusion

# Beans Binding

- **Keeps two properties of two objects in sync**
- **Source properties can specified using Expression Language:**
  - ex: “`${customer}`” or “`${employee.salary}`”
- **Does not require special object types:**
  - `Bindings.createAutoBinding(READ_WRITE,source,source Prop, target, target Prop);`

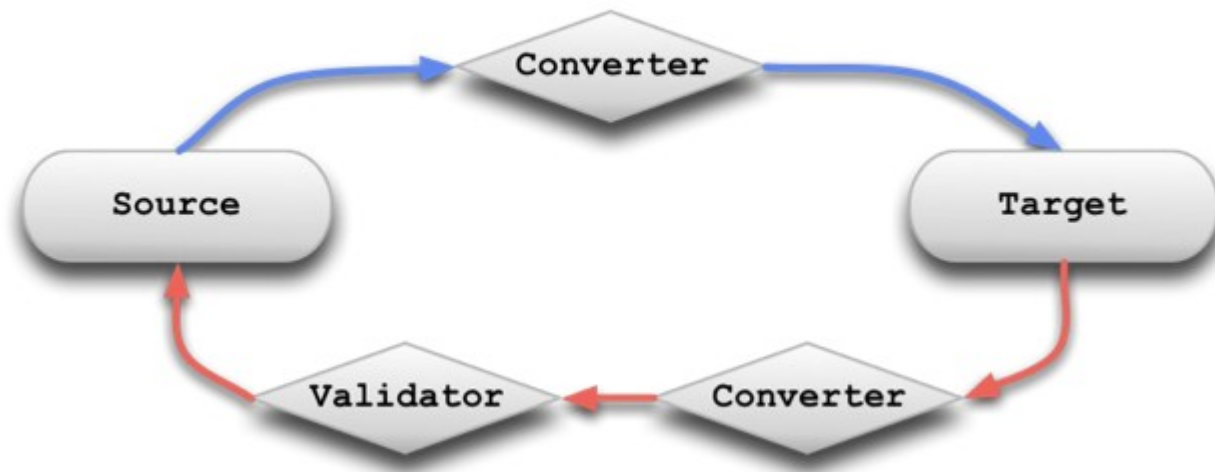
# Builds Upon...

- **JavaBeans**
  - PropertyChangeListeners
  - Doesn't require strict following of spec
- **Collection classes**
  - Standard way to encapsulate common data types

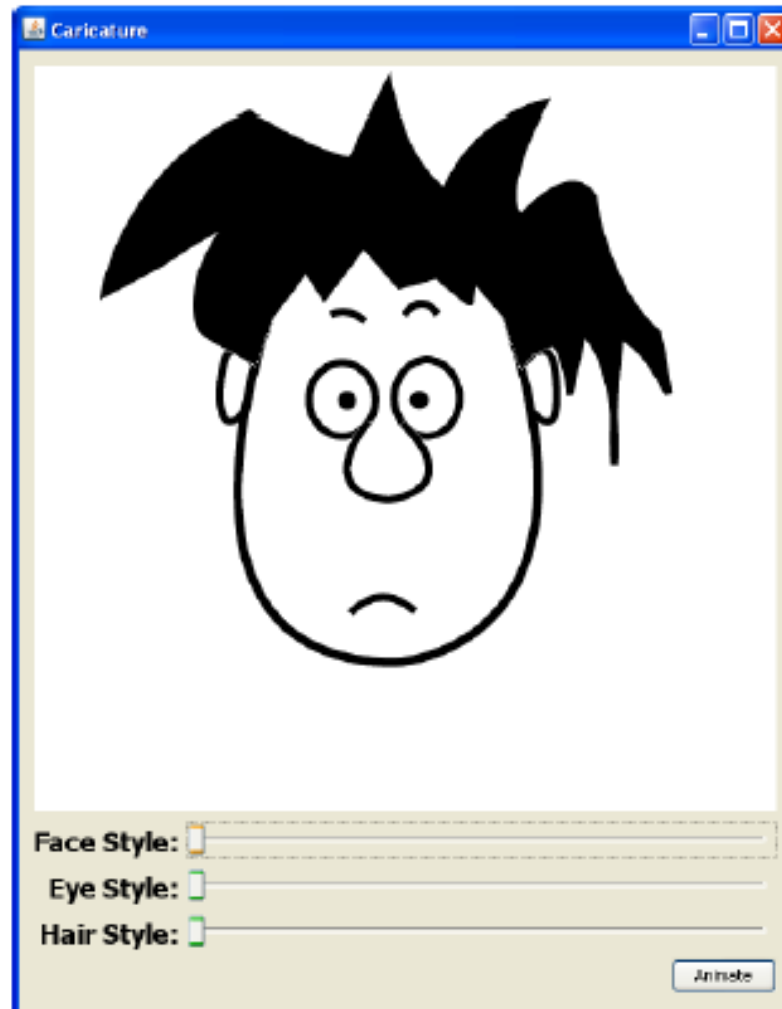
# Features

- **Different update strategies**
  - Read once, read only from source, keep source and target in sync
- **Ability to do validation**
- **Ability to transform value**
  - String to Color, Date to String

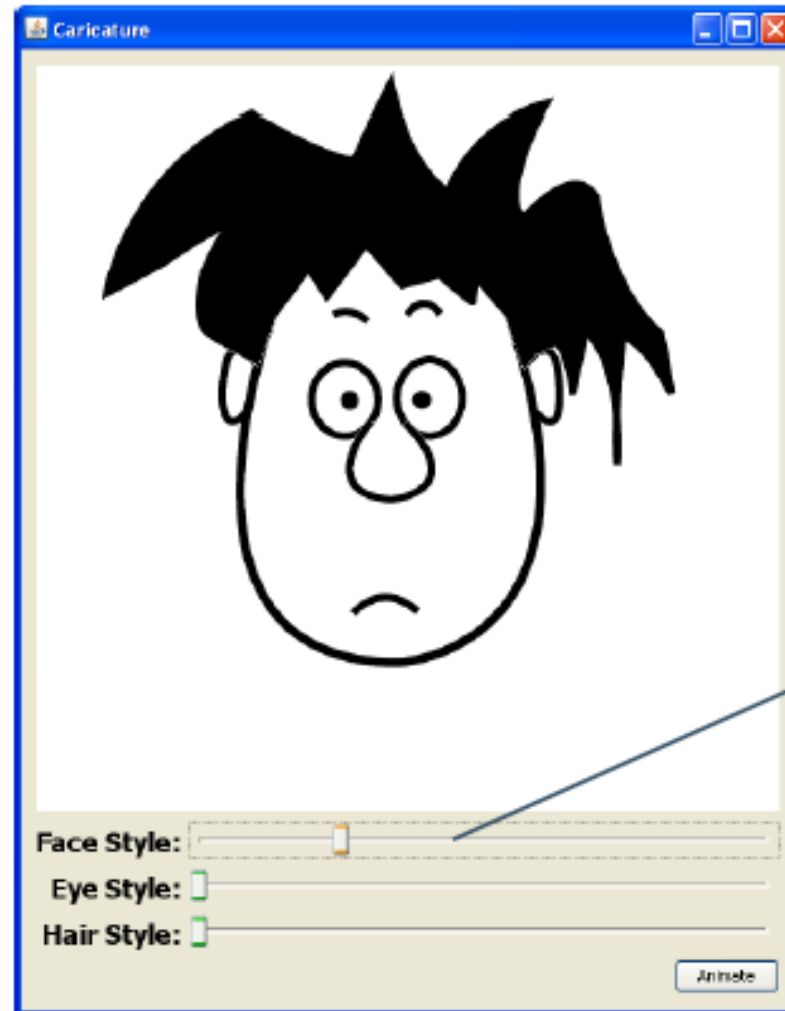
# Binding



# Example

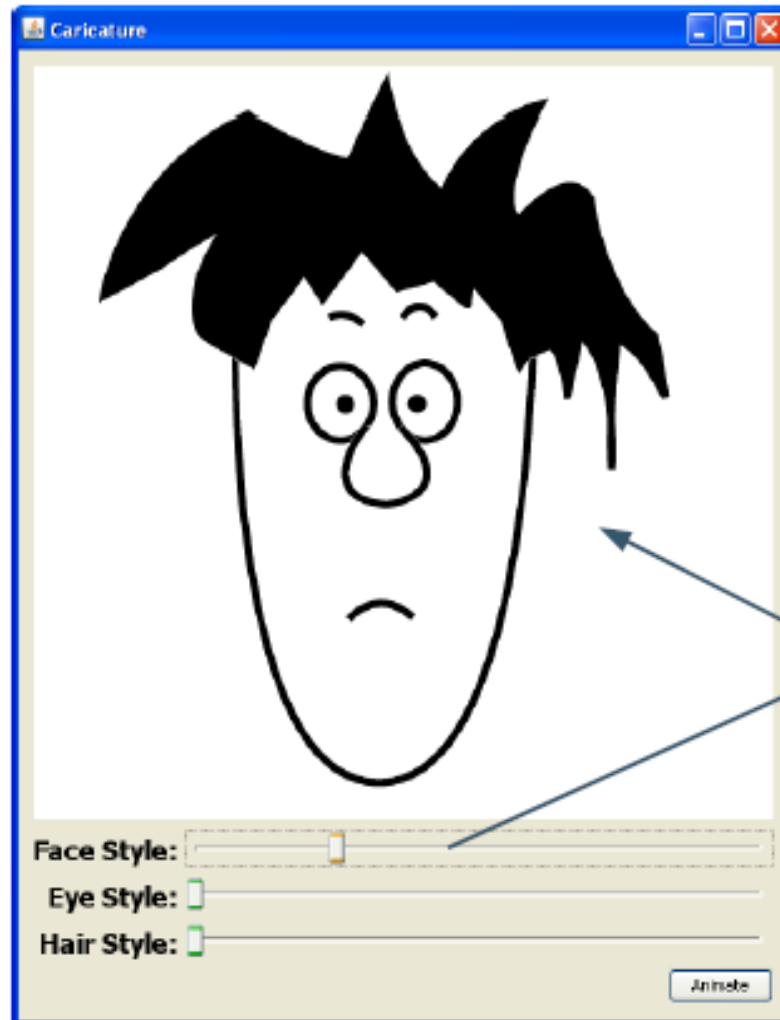


# Example



ChangeListener

# Example



`ChangeListener`  
`setFaceStyle(x);`

# Example

PropertyChangeListener  
slider.setValue();

Face Style:

Eye Style:

Hair Style:

Animate

The image shows a Java Swing window titled "Caricature" with a blue title bar and standard window controls. Inside the window is a simple line drawing of a cartoon face with spiky black hair and a sad expression. Below the drawing are three sliders, each with a small colored knob (orange for Face Style, green for Eye Style, and green for Hair Style). A button labeled "Animate" is located in the bottom right corner of the window. To the left of the window, there is a code snippet: "PropertyChangeListener" on the first line and "slider.setValue();" on the second line. Two blue arrows originate from the text: one points from "PropertyChangeListener" to the top of the Face Style slider, and the other points from "slider.setValue();" to the top of the Eye Style slider.

# Example - Before 295

```
faceSlider.addChangeListener(new ChangeListener() {
    public void stateChanged(ChangeEvent e) {
        caricature.setFaceStyle(faceSlider.getValue());
    }
});

caricature.addPropertyChangeListener(new
PropertyChangeListener() {
    public void propertyChange(PropertyChangeEvent e) {
        if (e.getPropertyName() == "faceStyle") {
            faceSlider.setValue(caricature.getFaceStyle());
        }
    }
});
```

# Example - JSR 295

```
Property faceStyleP =  
    ELProperty.create("${faceStyle}");
```

```
BeanProperty valueP =  
    BeanProperty.create("value");
```

```
Binding binding = new Binding(READ_WRITE,  
    caricature, faceStyleP, // source  
    faceSlider, valueP); // target
```

```
binding.bind();
```

# Let's Look at Some Code

Binding

# Demo

## JSR 295 - Ticker Demo

<http://wiki.netbeans.org/wiki/view/NBDemoTicker>

# When and Where?

- **When will JSR-295 be in a JDK?**
  - Planned - Java Platform 7
- **I can't wait. Where can I get it?**
  - Version 1.0
    - Released Sept. 5, 2007
    - Bundled with NetBeans 6.0
    - Available at  
<https://beansbinding.dev.java.net/>

# Agenda

- **Swing app. framework**
- Beans binding
- Putting it all together
- Conclusion

# Demo

JSR 295 + JSR 296

<http://wiki.netbeans.org/wiki/view/NBDemoMatisseInNB6>

# Conclusion

- **JSR 295 and 296 are making Swing development fun again**
- **Both are very well supported in NetBeans GUI builder (project Matisse)**
- **Desktop Java becoming interesting for “Delphi” and “VB” developers**
- **Desktop Java is well and alive!**

# Questions?



# Swing Application Framework and Beans Binding

Roman Strobl  
[roman.strobl@sun.com](mailto:roman.strobl@sun.com)