Eclipse + Html: A Journey

Kris De Volder <kdevolder@gopivotal.com>, Pivotal Software
Martin Lippert <mlippert@gopivotal.com>, Pivotal Software
Outline

- Goal
- Motivation
- Case Studies
- The Journey
- API Comparison
- Lessons Learned
- What's next?
- Questions?
Goals

- Embed a **Rich HTML5 UI** into Eclipse.

**Seamless Integration**
- Eclipse Code Drives the Web UI
- Web UI Triggers Eclipse Functionality
- Web UI Looks like it Belongs

**Cross Platform**
- **OS**: Windows / Mac / Linux
- **Eclipse**: 3.8 … 4.x
- **Java**: 6, 7, (8)
Why?

- It is Cool
  - HTML 5 is all the rage
  - More 'Sexy' UI
  - Stuff that is hard with standard SWT widget

- It is Economical
  - Build once
  - Reuse for Web & Desktop
Why? An Example
Case Study 1: STS Dashboard

Before:
Case Study 1: STS Dashboard

After:
Case Study 2: Spring Live Beans

Graph

Before:
Case Study 2: Spring Live Beans Graph

After:
Outline

- Goal
- Motivation
- The Case Studies
- The Journey
- API Comparison
- Lessons Learned
- What's next?
- Questions?
The Journey Begins

SWT Browser Widget as 'Advertised'

- Since Eclipse 3.0
- Uses Platform's native Browser.
- Embedded into a SWT Widget.
- Can be placed inside UI amongst other widgets.
- Linux / Mac and Windows Supported
- Good API to Integrate With Eclipse
  - listeners
  - inject code and functions
The Journey Begins

SWT Browser Widget as 'Advertised'

- Since Eclipse 3.0
- Uses Platform's native Browser.
- Embedded into a SWT Widget.
- Can be placed inside UI amongst other widgets.
- Linux / Mac and Windows Supported
- Good API to Integrate With Eclipse listeners

The Ideal Solution! But is it?
SWT Browser on Windows / Mac

- Uses Native Browser Support (IE / WebKit)
  - Works out of the box.
  - Minor issues experienced on Windows:
    - IE Browser Quirks
    - SWT Browser forced lower IE <9 mode. Even if user has IE 10 installed.
    - Can be dealt with by setting extra system property: -Dorg.eclipse.swt.browser.IEVersion=10001
SWT Browser Linux Support

- Linux is Supported But...
  - Requires Specific Native Libraries
    - Webkit for GTK2 or ...
    - XULRunner (ancient version)
  - Depending on Distro / Version and installed Third Party Components
    - Libraries may be missing
    - Libraries may cause frequent JVM crashes
SWT Browser Linux Support

- Personal Experience on Ubuntu (*)
  - Ubuntu 10.04 (now obsolete)
    - Great: Works out of the box
  - Ubuntu 12.04 (current LTS)
    - Install non-default libWebKit for GTK2.
      - causes JVM crashes in combination with Google Talk plugin (**)
    - Downloading ancient XULRunner and set some system props => works.
  - ...

(*) = https://bugs.eclipse.org/bugs/show_bug.cgi?id=420030
(**) = https://bugs.eclipse.org/bugs/show_bug.cgi?id=334466
SWT Browser Linux Support

- Personal Experience on Ubuntu (cont)
  - Ubuntu 13.04
    - Ancient XULRunner no longer works (crashes immediately)
    - WebKitGTK solution works but also affected by JVM Crash bug (*)

(*) = https://bugs.eclipse.org/bugs/show_bug.cgi?id=334466
SWT Browser: Linux

(*): https://bugs.eclipse.org/bugs/show_bug.cgi?id=334466
The Journey Continues

PLAN B: JavaFX “WebView”

- JavaFX ships with JDK 7
- Embedded Copy of WebKit Included
- e(fx)clipse: tooling and runtime for Eclipse / OSGI.
JavaFX Experience

APIs / Integration

- Comparable to SWT widget but no 1-1 mapping.
  - Inject Code / Functions
  - Some listener support

=> Viable substitute for SWT Widget.

- Similar Capabilities
- Refactoring existing code requires effort.
JavaFX Experience

APIs

- Comparable to SWT widget but no 1-1 mapping.
  - Inject Code / Functions
  - Some listener support

Viable substitute for SWT Widget. Refactoring existing code requires some effort.
JavaFX Experience

Availability / Cross Platform

- Sun JDK 7 or better required
- Eclipse 4.x

Viable substitute for SWT Widget.
Cross Platform not Ideal
OpenJDK on Linux?
Expected to improve over time (newer JVM = better)
Outline

- Goal
- Motivation
- The Case Studies
- The Journey
- API Comparison
- Lessons Learned
- What's next?
- Questions?
API comparison

- How to
  - Embed in SWT
  - Call Java function from Browser
SWT: Embed Browser in SWT

- How? => Relatively Trivial

The SWTBrowserWidget is an SWT widget. Just instantiate and use like any other widget.

```java
import org.eclipse.swt.browser.Browser;

Composite parent = ... 
...
Browser browser = new Browser(parent, SWT.NONE);
//Note: style constant can also specify Mozilla vs Webkit
```
JavaFX: Embed Browser in SWT

- How? Slightly harder
  - Use EFX JavaFX runtime
  - Wrap JavaFX component in a FXCanvas

From the JavaDocs:

```java
public class FXCanvas
extends org.eclipse.swt.widgets.Canvas

FXCanvas is a component to embed JavaFX content into SWT applications.
The content to be displayed is specified with the
setScene(javafx.scene.Scene) method that accepts an instance of JavaFX
Scene. After the scene is assigned, it gets repainted automatically. All
the input and focus events are forwarded to the scene transparently to
the developer.
```
JavaFX: Embed Browser in SWT

```java
public class JavaFxBrowserViewer extends Composite {

    public JavaFxBrowserViewer(Composite parent, int style) {
        ...
        final FXCanvas fxCanvas = new FXCanvas(this, SWT.NONE);
        fxCanvas.setLayoutData(...);
        fxCanvas.setLayout(GridLayoutFactory.fillDefaults().create());
        browser = new WebView();
        BorderPane border = new BorderPane();
        Scene scene = new Scene(border);
        border.setCenter(browser);
        fxCanvas.setScene(scene);
        ...
    }

```
API comparison

- How to
  - Embed in SWT
  - Call Java function from Browser
Calling Java from The Browser

Html/JavaScript:

```html
<a href=""
   onclick="ide.call('openWizard', 'org.spring.ide.newSpringProject')">
   Spring Project
</a>
```
Adding a Browser Function

- Both JavaFX and SWT Browser each provide a mechanism to define a function in Java and inject it into your page(s).
- The main mechanism to integrate between Eclipse & Browser functionality
- Capabilities similar, APIs slightly different.
import org.eclipse.swt.browser.Browser;
import org.eclipse.swt.browser.BrowserFunction;

public class MyJSFunction extends BrowserFunction {

    public ImportJSFunction(Browser browser) {
        super(browser, "ide");
    }

    @Override
    public Object function(Object[] arguments) {
        ...do your thing...
    }
}

SWT Browser: Injecting A Function
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;

/**
 * @author Miles Parker
 */

public class JavaFxBrowserManager {

    private WebEngine engine;
    private WebView view;

    ...

    public void setClient(WebView view) {
        ...
        this.engine = view.getEngine();
    }

}
JavaFX: Injecting a Function

```java
public class JavaFxBrowserManager {
    ...

    public void setClient(WebView view) {
        ...
        this.engine = view.getEngine();
        JSObject window = (JSObject)engine.executeScript("window");
        window.setMember("ide", this);
        ...
    }

    /**
     * Called via reflection by JavaFx.
     */
    public void call(String functionId, String argument) {
        ...insert your code here...
    }
```
Outline

- Goal
- Motivation
- The Case Studies
- The Journey
- API Comparison
- Lessons Learned
- What's next?
- Questions?
## Lessons Learned

<table>
<thead>
<tr>
<th></th>
<th>SWT</th>
<th>JavaFX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API/Integration</strong></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>X-Platform</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mac</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Windows</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Linux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenJDK</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>SunJDK</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Java 6</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Eclipse 3.x</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Recent (J7 E4)</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Other Lessons Learned

- Debugging Complications
  - Embedded widget may behave differently
  - No 'Chrome Dev Tools' available
    => Complicates debugging embedded app

- Solutions
  - console.log (but doesn't work with JavaFX webkit)
  - create your own console.log by adding messages to the dom.
Outline

- Goal
- Motivation
- The Case Studies
- The Journey
- API Comparison
- Lessons Learned
- What's next?
- Questions?
What's Next?

- Neither JXF or SWT cover all cases we care about.

- More Linux Trouble Ahead?
  - SWT Switch on GTK3 by Default
    => Incompatible with JFX at this time.
  - Ironically... SWT Widget will probably work better in GTK3 on most Linux Systems!

  => Refactor to use either SWT or JavaFX WebView interchangeably
Thanks

Kris De Volder <kdevolder @ gopivotal.com>
Martin Lippert <mlippert @ gopivotal.com>

Credit to:
- Alex Boyko <aboyko @ gopivotal.com>
- Miles Parker <miles.parker @ tasktop.com>