Introducing Apache Pivot

Greg Brown
9/14/2010

Bio

- Greg Brown
  - Software Architect
    - 15 years experience developing client and server applications in both services and R&D
    - Founder, Apache Pivot

Project History

- Started by Greg Brown and Todd Volkert of VMware in late 2007
- Released as open source in June 2008; version 1.0 released in October 2008
- Entered Apache Incubator in January 2009 (1.1)
- Graduated from Incubator in December 2009 (1.4)
- Version 1.5 released in June 2010
- Version 2.0 currently in development (Q4 2010/ Q1 2011)

What is Apache Pivot?

- Open-source platform for building rich Internet applications (RIAs) in Java or any other JVM language (JavaScript, Groovy, Scala, Clojure, etc.)
- Similar to Adobe Flex or Microsoft Silverlight, but based on the JVM rather than Flash or Silverlight player
- Pivot applications can be run in a browser via the Java Plugin or as stand-alone desktop application

What is Apache Pivot?

- Like other RIA platforms, includes features that make building modern GUI applications much easier:
  - Declarative XML-based UI markup language ("WTKX" in Pivot 1.x, "BXML" in Pivot 2.0+)
  - Themes (aka "skins")/styling
  - Data binding
  - Effects and transitions (animations)
  - Web services integration (JSON/REST)

Why RIA?

- Web is de facto means of application delivery today
- Still difficult to create a user experience in HTML that is truly on par with that of a desktop application

Why RIA?

- RIA platforms bridge the gap between the web and desktop experiences
- Allow developers to build applications that look and feel more like native desktop applications but are deployable via the web
- Often incorporate visual effects intended to enhance the overall user experience (animations and other dynamic behaviors)

Why RIA?

- Not limited to web browser: AIR, Silverlight 3, and Pivot all allow developers to build cross-platform, internet-enabled applications that can be installed locally
- Accessed via familiar desktop metaphors: Start menu, Dock, etc.
- Support better integration with native OS including local file system access & drag/drop
- Can also operate offline
- Example: iTunes

Why Pivot?

1. Provide a viable option for developers who want to build rich Internet applications in Java or other JVM languages
   - Flex: ActionScript
   - Silverlight: C#/JavaScript
   - JavaFX: JavaFX Script

Why Pivot?

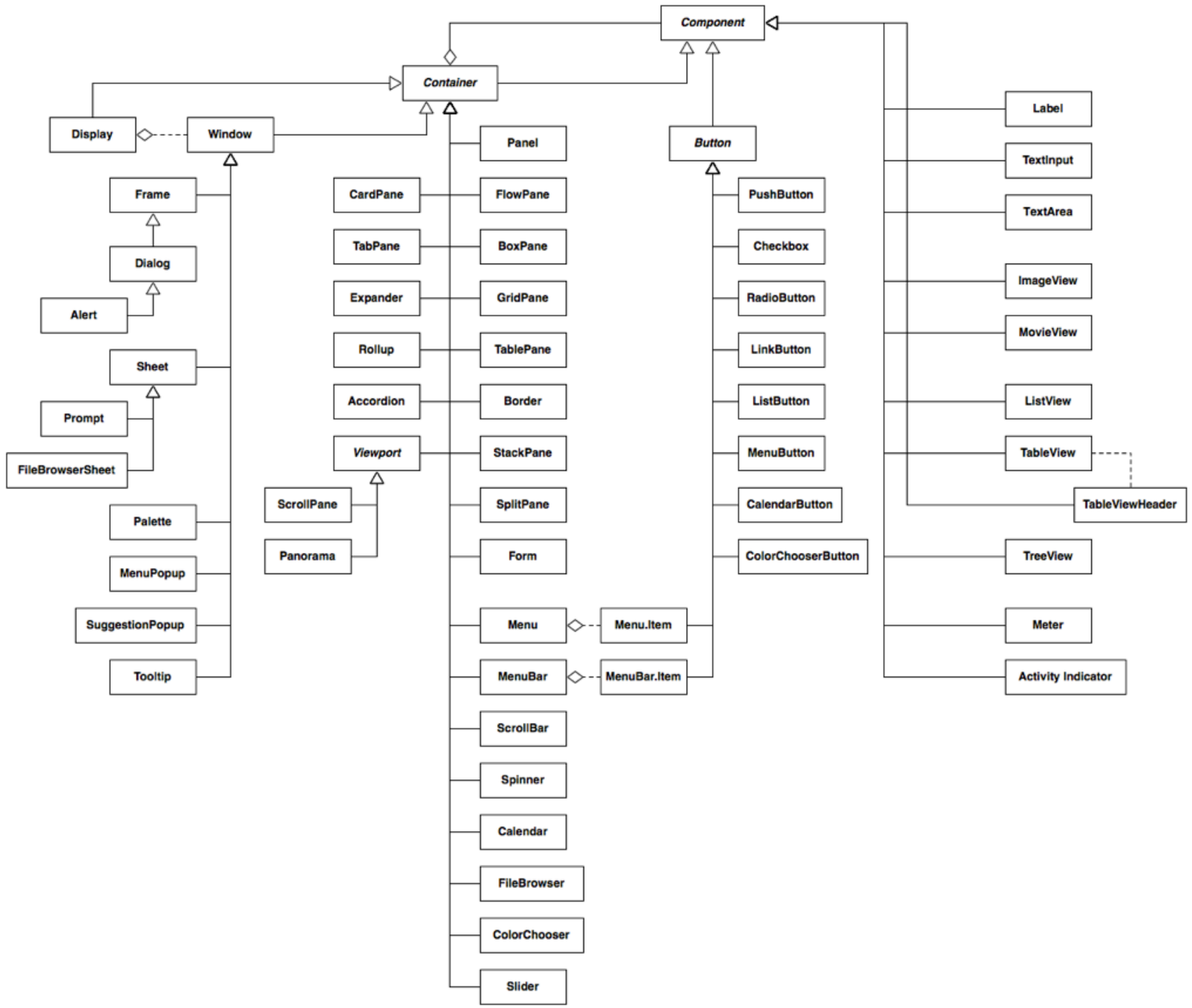
2. Provide a truly open alternative for RIA developers

   - Flex, Silverlight, and JavaFX are all proprietary platforms
   - Pivot is completely open source and driven entirely by the software development community

Platform Overview

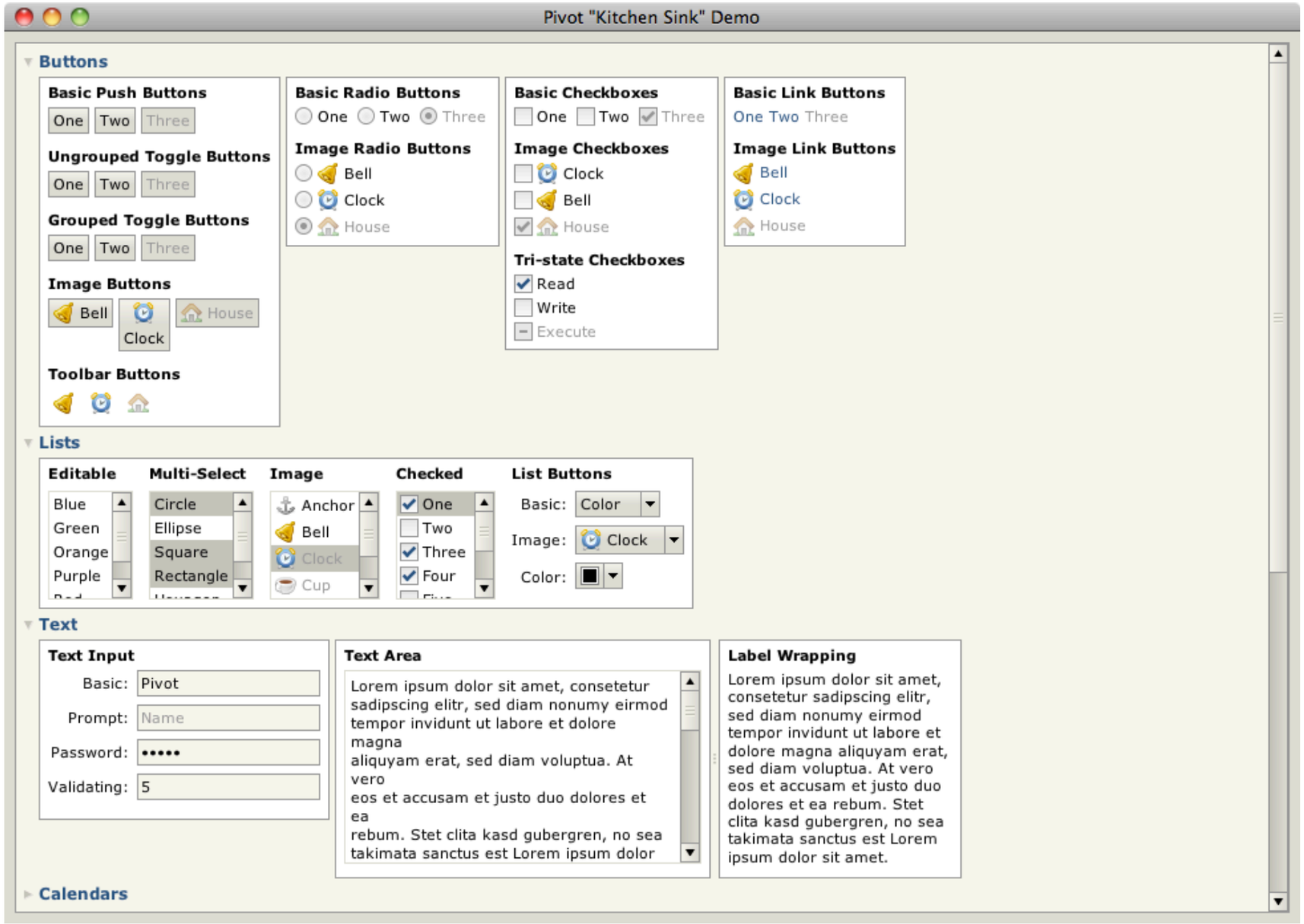
- Pivot libraries:
  - "Core" *pivot-core-x.x.jar*
    Common, non-UI utility classes (collections, serialization, event processing, localization, threading, I/O, etc.)
  - "Web" *pivot-web-x.x.jar, pivot-web-server-x.x.jar*
    REST client/server APIs
  - "WTK" *pivot-wtk-x.x.jar, pivot-wtk-terra-x.x.jar*
    Windowing Toolkit/"Terra" L&F
  - "Charts" *pivot-charts-x.x.jar, pivot-jfree-x.x.jar*
    Charting components

Platform Overview



WTK Class Hierarchy

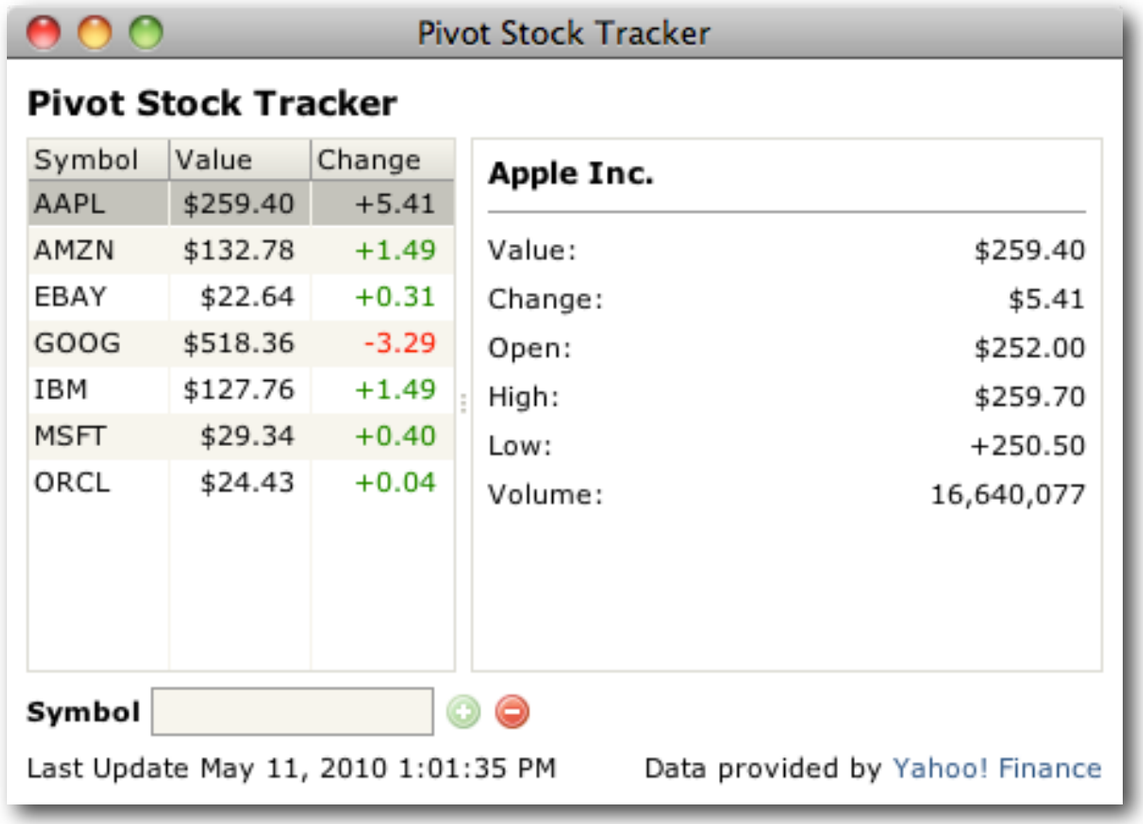
“Kitchen Sink”



“Kitchen Sink” Demo Application

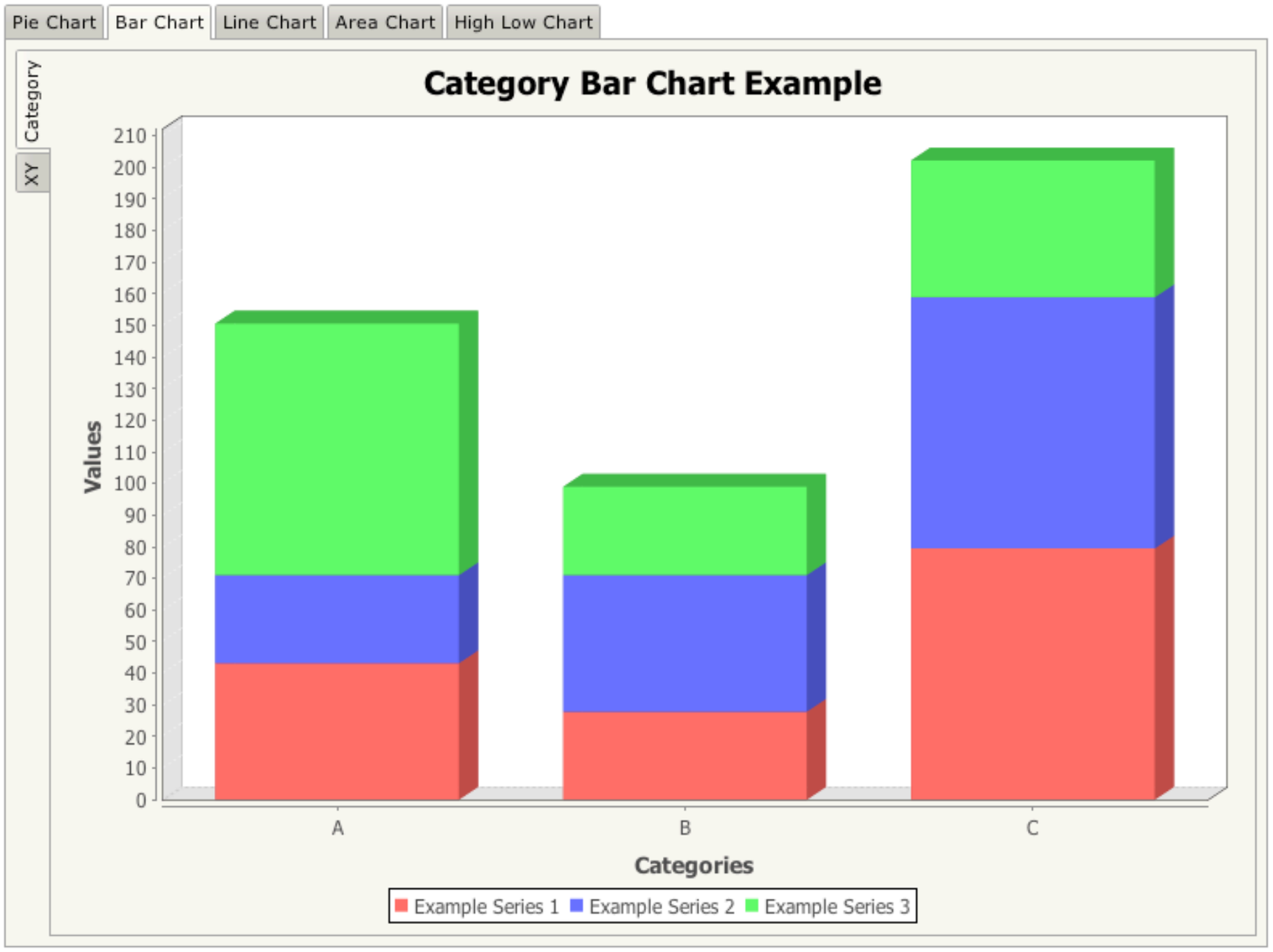
"Stock Tracker"

- Simple but practical sample application
- Highlights key platform features and development best practices



"Stock Tracker" Tutorial Application

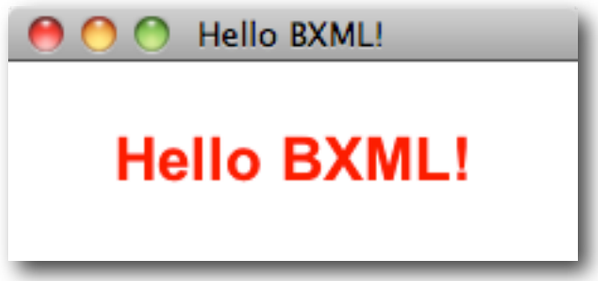
Charting



Pivot/JFreeChart Demo Application

“Hello BXML!”

```
public class HelloBXML implements Application {
    private Window window = null;

    @Override
    public void startup(Display display, Map<String, String> properties)
        throws Exception {
        BXMLSerializer bxmlSerializer = new BXMLSerializer();
        window = (Window)bxmlSerializer.readObject(HelloBXML.class, "hello.bxml");
        window.open(display);
    }

    @Override
    public boolean shutdown(boolean optional) {
        if (window != null) {
            window.close();
        }

        return false;
    }

    @Override
    public void suspend() {
    }

    @Override
    public void resume() {
    }
}
```




```
<Window title="Hello BXML!" maximized="true"
    xmlns:bxml="http://pivot.apache.org/bxml"
    xmlns="org.apache.pivot.wtk">
    <Label text="Hello BXML!"
        styles="{font:'Arial bold 24', color:'#ff0000',
            horizontalAlignment:'center', verticalAlignment:'center'}"/>
</Window>
```

Source code for
“Hello World”
in Pivot

Pivot Compared to Swing

- Swing can also be used to build RIAs
- Both Pivot and Swing use Java2D under the hood
- Pivot offers numerous advantages that make it a more compelling, modern alternative

Pivot Compared to Swing

- Pivot advantages:
  - Provides XML-based markup language for simplifying user interface construction
  - Built-in support for JSON and REST-based data services
  - Built-in data binding support
  - Platform-level support for visual effects and transitions
  - Takes advantage of newer Java language features: generics, enums, for..each loops, varargs, and annotations
  - Designed from ground up for resolution independence

Pivot Compared to JavaFX

- Pivot allows developers to build applications in Java, vs. JavaFX scripting language
- Slightly different emphasis: "Application" vs. "Rich" (media delivery) in "RIA"

Pivot Compared to GWT

- GWT also allows developers to use Java to write web-based applications
- Runtime enviroment for a GWT application is the browser, not a JVM:

- Doesn't support complete Java API (no I/O, networking, threading, reflection, XML, etc.) - Java language only
- Limited to GWT RPC for server communication (i.e. no REST, SOAP, AMF, etc.)
- No support for 3rd-party Java libraries (IM, OSGi, dependency injection, etc.)
- Doesn't support other JVM languages
- No desktop integration (file browsing, clipboard, drag/drop)
- Presentation performed via CSS and DOM manipulation rather than true 2D API
  - Cumbersome to customize
  - Performance/scalability issues with large data sets

Pivot Compared to GWT

- Pivot is easier to work with:
  - Cleaner, more intuitive API (not tied to DOM)
  - Easier to develop/debug/deploy (no server required)

Sample Application

- Simple example
- Demonstrates some key features:
  - UI Markup
  - “Code Behind”
  - Event handling
  - Scripting

UI Markup

- Pivot UI often defined in markup (BXML)
- Hierarchical structure of XML parallels the component hierarchy, makes it easy to visualize the resulting output
- Developers are familiar with markup metaphor

UI Markup

- BXML source code for sample application:

```
<scripting:JavaWindow title="Java Window" maximized="true"
    xmlns:bxml="http://pivot.apache.org/bxml"
    xmlns:scripting="org.apache.pivot.examples.scripting"
    xmlns="org.apache.pivot.wtk">
    <BoxPane orientation="vertical"
        styles="{horizontalAlignment:'center', verticalAlignment:'center'}">
        <Label text="This is a Java window."/>
        <PushButton bxml:id="sayHelloButton" buttonData="Say Hello"/>
    </BoxPane>
</scripting:JavaWindow>
```

UI Markup

- UI can still be defined in code - BXML is just a "shortcut":

```
<Window title="Hello BXML!" maximized="true"
    xmlns:bxml="http://pivot.apache.org/bxml"
    xmlns="org.apache.pivot.wtk">
    <Label text="Hello BXML!"
        styles="{font:'Arial bold 24', color:'#ff0000',
            horizontalAlignment:'center', verticalAlignment:'center'}"/>
</Window>
```

```
@Override
public void startup(Display display, Map<String, String> properties) {
    window = new Window();

    Label label = new Label();
    label.setText("Hello World!");
    label.getStyles().put("font", new Font("Arial", Font.BOLD, 24));
    label.getStyles().put("color", Color.RED);
    label.getStyles().put("horizontalAlignment",
        HorizontalAlignment.CENTER);
    label.getStyles().put("verticalAlignment",
        VerticalAlignment.CENTER);

    window.setContent(label);
    window.setTitle("Hello World!");
    window.setMaximized(true);

    window.open(display);
}
```

UI Markup

- Other key BXML features:
  - Resource injection (localization)
  - Includes (modularization)
  - URL resolution (image resources)

“Code Behind”

- Root BXML element is <scripting:JavaWindow>
- When BXML file is loaded, creates an instance of org.apache.pivot.examples.scripting.JavaWindow (subclass of org.apache.pivot.wtk.Window)
- Implements *Bindable* interface (asociates a Java class with markup)
- Defines the “code behind” logic for structure defined in BXML file

```
public class JavaWindow extends Window implements Bindable {
    @BXML private PushButton sayHelloButton = null;

    @Override
    public void initialize(Map<String, Object> namespace, URL location,
        Resources resources) {
        sayHelloButton.getButtonPressListeners().add(new ButtonPressListener() {
            @Override
            public void buttonPressed(Button button) {
                sayHello();
            }
        });
    }

    @Override
    public void open(Display display, Window owner) {
        super.open(display, owner);

        sayHelloButton.requestFocus();
    }

    private void sayHello() {
        Prompt.prompt("Hello from Java!", this);
    }
}
```

“Code Behind”

- BXML binding:
  - Maps objects defined in BXML to Java member variables (“dependency injection”)
  - bxml:id maps to @BXML annotation
- Implementing Bindable ensures that bindings are processed, notifies root element that bindings are available
- Also provides access to page resources and origin

```
public class JavaWindow extends Window implements Bindable {
    @BXML private PushButton sayHelloButton = null;

    @Override
    public void initialize(Map<String, Object> namespace, URL location,
        Resources resources) {
        sayHelloButton.getButtonPressListeners().add(new ButtonPressListener() {
            @Override
            public void buttonPressed(Button button) {
                sayHello();
            }
        });
    }

    @Override
    public void open(Display display, Window owner) {
        super.open(display, owner);

        sayHelloButton.requestFocus();
    }

    private void sayHello() {
        Prompt.prompt("Hello from Java!", this);
    }
}
```

Event Handling

- Application logic generally executed in response to an "event" (button pressed, selection changed, etc.)

- Event listeners often wired up in Bindable#initialize()

- Can also be implemented in inline script, similar to HTML (demonstrated next)

```
public class JavaWindow extends Window implements Bindable {
    @BXML private PushButton sayHelloButton = null;

    @Override
    public void initialize(Map<String, Object> namespace, URL location,
        Resources resources) {
        sayHelloButton.getButtonPressListeners().add(new ButtonPressListener() {
            @Override
            public void buttonPressed(Button button) {
                sayHello();
            }
        });
    }

    @Override
    public void open(Display display, Window owner) {
        super.open(display, owner);

        sayHelloButton.requestFocus();
    }

    private void sayHello() {
        Prompt.prompt("Hello from Java!", this);
    }
}
```

Event Handling

- Clicking the “Say Hello” button produces this friendly greeting:

Scripting

- Application logic can also be implemented in script (either inline or defined in an external file)
- Java 6+ includes a JavaScript engine;, so JavaScript is default BXML scripting language
- Other JVM languages also supported...

```
<Window bxml:id="window" title="JavaScript Window" maximized="true"
    xmlns:bxml="http://pivot.apache.org/bxml"
    xmlns="org.apache.pivot.wtk">
    <bxml:script>
    <![CDATA[
    importPackage(org.apache.pivot.wtk);

    function sayHello() {
        Prompt.prompt("Hello from JavaScript!", window);
    }
    ]]>
    </bxml:script>

    <windowStateListeners>
    <![CDATA[
    function windowOpened() {
        sayHelloButton.requestFocus();
    }
    ]]>
    </windowStateListeners>

    <BoxPane orientation="vertical"
        styles="{horizontalAlignment:'center', verticalAlignment:'center'}">
        <Label text="This is a JavaScript window."/>
        <PushButton bxml:id="sayHelloButton" buttonData="Say Hello">
            <buttonPressListeners>
            <![CDATA[
            function buttonPressed() {
                sayHello();
            }
            ]]>
            </buttonPressListeners>
        </PushButton>
    </BoxPane>
</Window>
```

Scripting

- ...for example, Groovy (since Groovy compiles to bytecode, it can also be used to create a "code behind" class, as in the Java example)...

```
<?language groovy?>

<Window bxml:id="window" title="Groovy Window" maximized="true"
    xmlns:bxml="http://pivot.apache.org/bxml"
    xmlns="org.apache.pivot.wtk">
    <bxml:script>
    <![CDATA[
    import org.apache.pivot.wtk.*

    sayHello = {
        Prompt.prompt("Hello from Groovy!", window)
    }
    ]]>
    </bxml:script>

    <windowStateListeners>
    <![CDATA[
    windowOpened = {
        sayHelloButton.requestFocus()
    }
    ]]>
    </windowStateListeners>

    <BoxPane orientation="vertical"
        styles="{horizontalAlignment:'center', verticalAlignment:'center'}">
        <Label text="This is a Groovy window."/>
        <PushButton bxml:id="sayHelloButton" buttonData="Say Hello">
            <buttonPressListeners>
            <![CDATA[
            buttonPressed = {
                sayHello()
            }
            ]]>
            </buttonPressListeners>
        </PushButton>
    </BoxPane>
</Window>
```

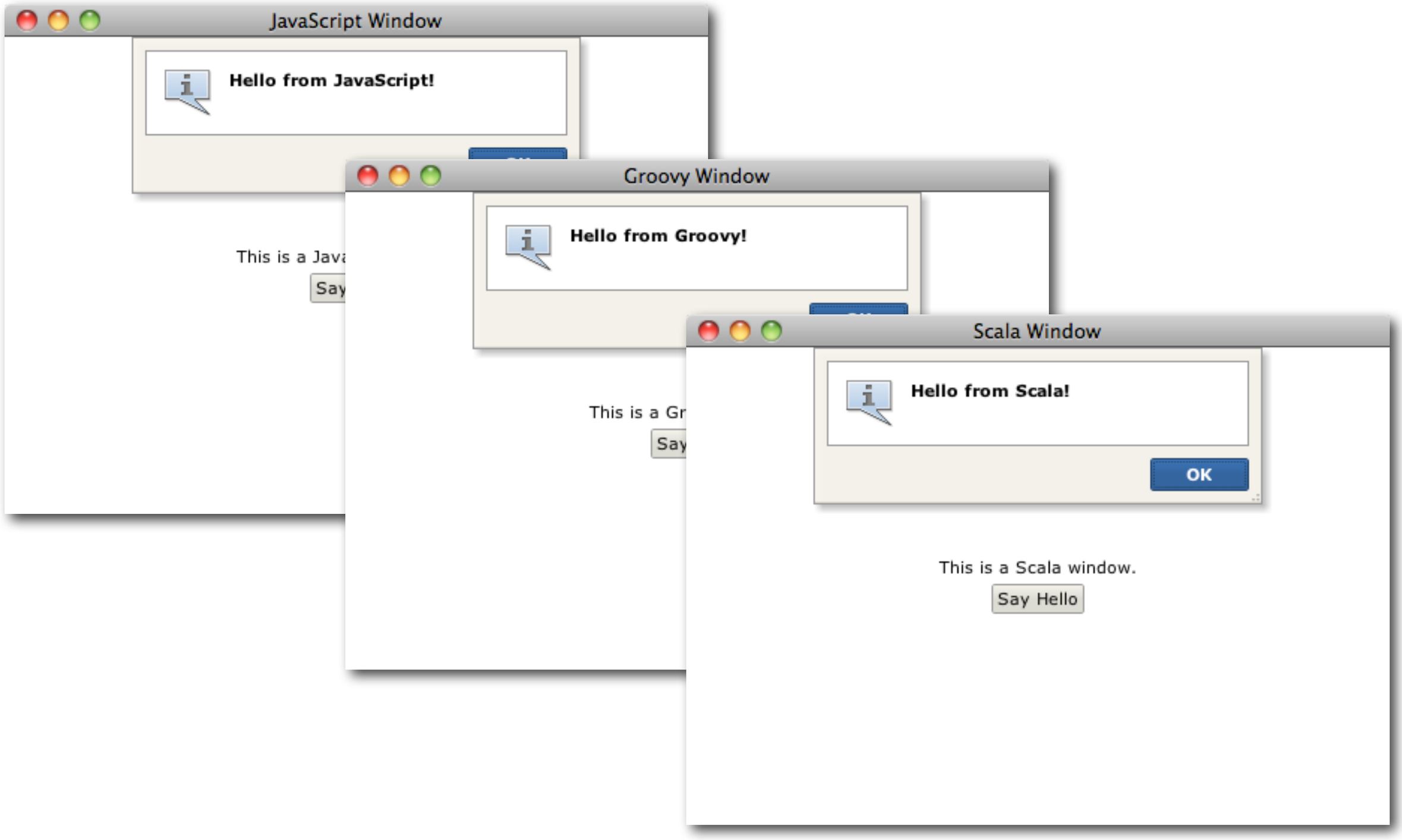
Scripting

- ...and Scala (used to implement code-behind)...

```
<scripting:ScalaWindow title="Scala Window" maximized="true"
    xmlns:bxml="http://pivot.apache.org/bxml"
    xmlns:scripting="org.apache.pivot.examples.scripting"
    xmlns="org.apache.pivot.wtk">
    <BoxPane orientation="vertical"
        styles="{horizontalAlignment:'center', verticalAlignment:'center'}">
        <Label text="This is a Scala window."/>
        <PushButton bxml:id="sayHelloButton" buttonData="Say Hello"/>
    </BoxPane>
</scripting:ScalaWindow>
```

```
class ScalaWindow extends Window with Bindable {
    private var sayHelloButton:PushButton = null;

    override def initialize(namespace: Map[String, Object], location:URL,
        resources:Resources) {
        sayHelloButton =
            namespace.get("sayHelloButton").asInstanceOf[PushButton];

        sayHelloButton.getButtonPressListeners().add(new ButtonPressListener {
            override def buttonPressed(button:Button) {
                sayHello();
            }
        });
    }

    override def open(display:Display, owner:Window) {
        super.open(display, owner);

        sayHelloButton.requestFocus();
    }

    private def sayHello() {
        Prompt.prompt("Hello from Scala!", this);
    }
}
```

Scripting


JavaScript Window
Hello from JavaScript!
This is a Java
Say
Groovy Window
Hello from Groovy!
This is a Gr
Say
Scala Window
Hello from Scala!
OK
This is a Scala window.
Say Hello

Summary

- Pivot is a great way to build and deploy engaging, cross-platform applications for the enterprise
- It is the only RIA framework that allows developers to build rich Internet applications using any JVM language (Java, Groovy, etc.)
- It is also the only truly open RIA framework: completely open source and driven entirely by the software development community

Summary

- Pivot allows developers to take advantage of tools and technologies they already know (and love!) to build rich Internet applications

Further Information

- Apache Pivot:
  - http://pivot.apache.org
  - http://pivot.apache.org/demos/
  - http://pivot.apache.org/tutorials/
- Apache Software Foundation
  - http://www.apache.org/

Q & A

Examples

- Geiger (order entry and management system)
- Satellite Consulting (expense entry and reporting system)
- University of Maryland Institute for Advanced Computer Study (storage management system)
- Murata Electronics Trading (web quotation workflow system)

Examples

- Synacom Communication & Software (n-tier business app)
- Calvino Coffee (POS system)
- Tagetik (internal management app)