



# Introducing Apache Pivot

Greg Brown, Todd Volkert  
6/10/2010



# Speaker Bios

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

# Speaker Bios

- Todd Volkert
  - Senior Software Architect
  - 13 years experience developing web and rich client applications
  - Apache Pivot Co-Founder

# What is Apache Pivot?

- Open-source platform for building rich internet applications in Java (or any JVM scripting language: Groovy, JavaScript, Scala, etc.)
- Similar to Adobe Flex or Microsoft Silverlight, but based on the JVM rather than Flash or Silverlight player
- Pivot applications can be run as an applet or as stand-alone desktop application (installed or launched via Web Start)

# 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")
  - Themes (aka "skins")/styling
  - Data binding
  - Effects and transitions (animations)
  - Web services integration (JSON/REST)

# Why RIA?

- Functional requirements for many web applications have begun to scale beyond the capabilities of the browser
- 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 Pivot?

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



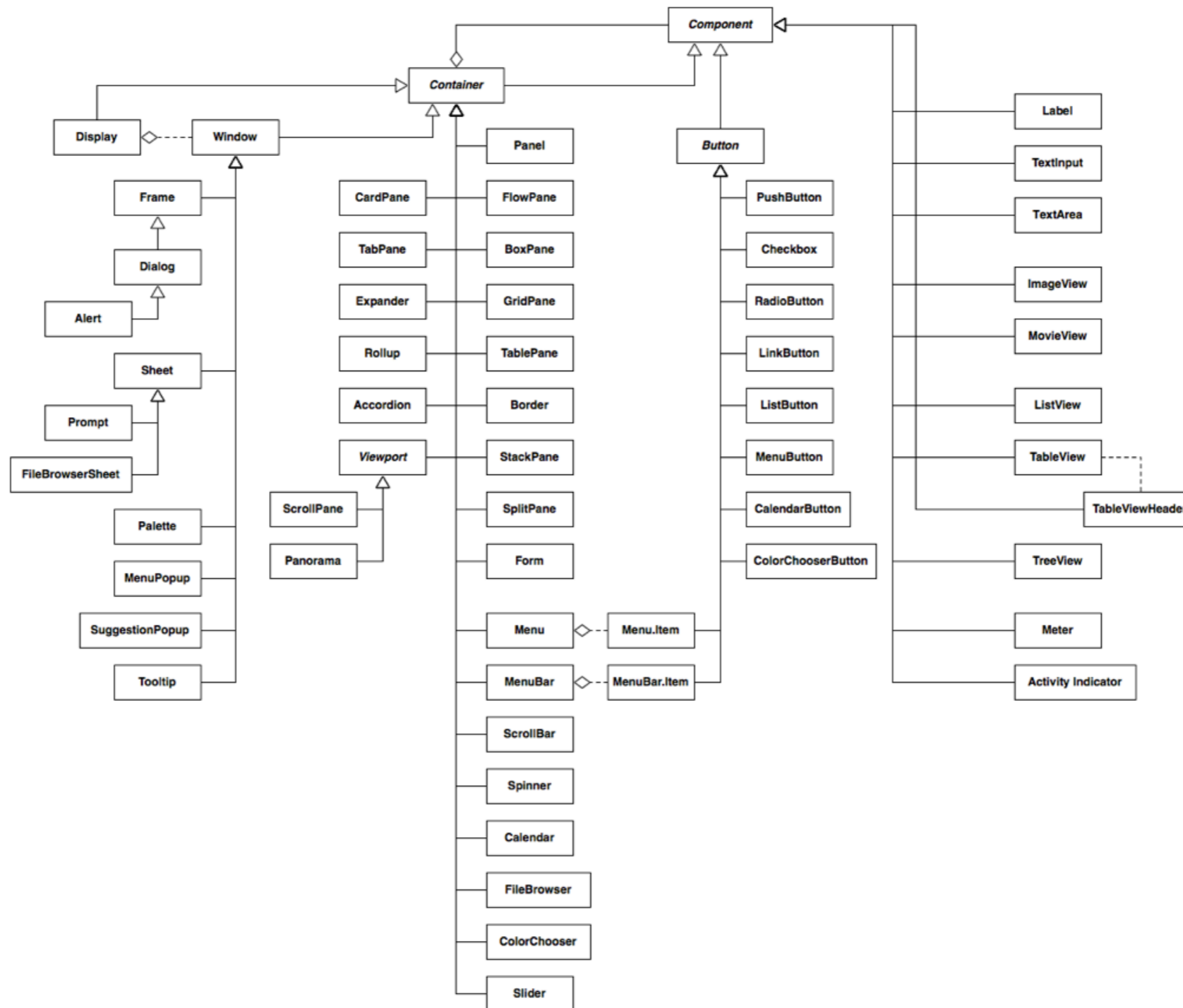
# Why Pivot?

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

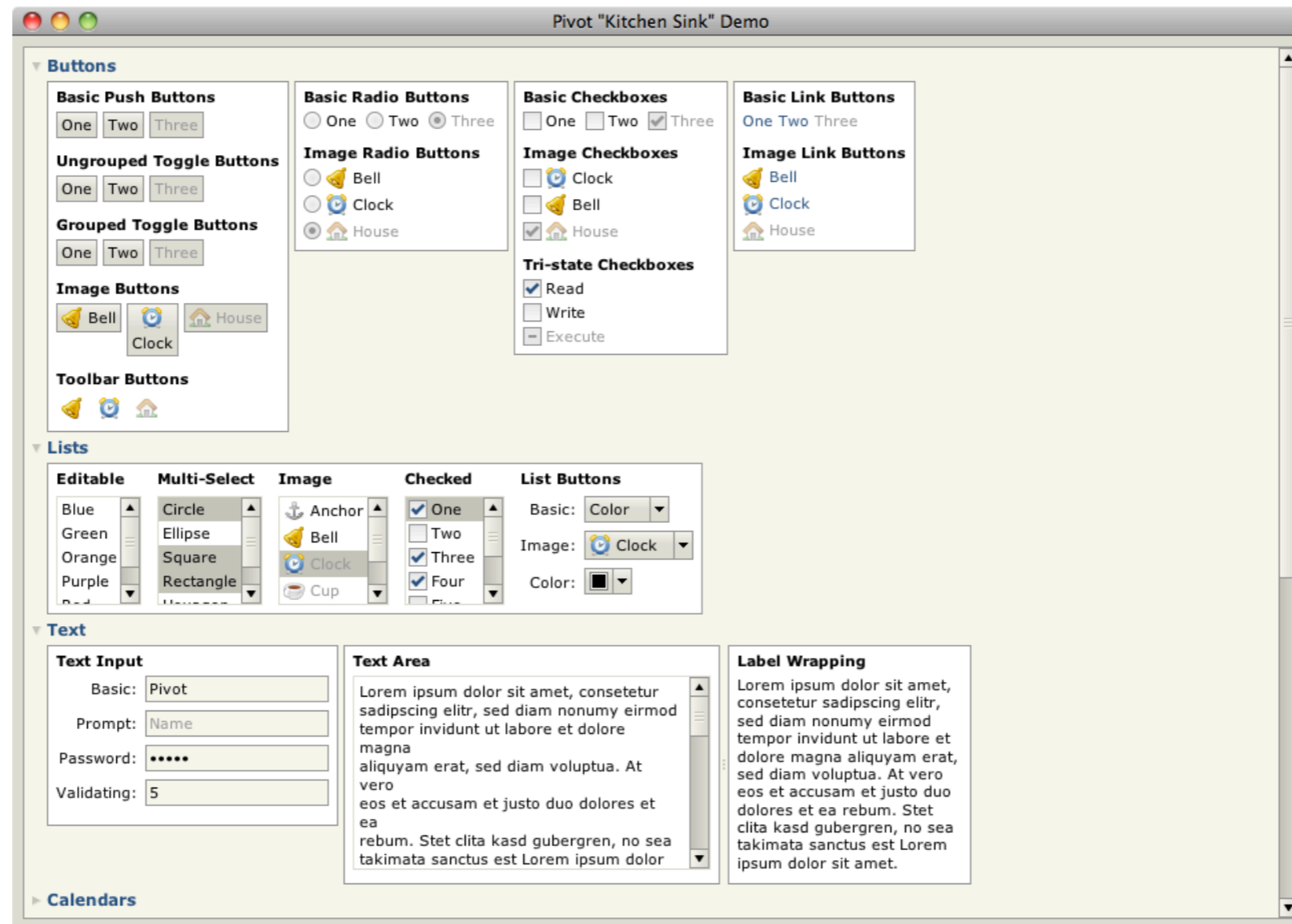
- Pivot classes are grouped into the following libraries:
  - *pivot-core-1.5.jar* - common, non-UI utility classes (collections, event processing, localization, threading, I/O, etc.)
  - *pivot-web-1.5.jar/pivot-web-server-1.5.jar* - REST client/server APIs
  - *pivot-wtk-1.5.jar/pivot-wtk-terra-1.5.jar* - WTK/Terra L&F
  - *pivot-charts-1.5.jar* - charting components (requires charting provider; currently based on JFreeChart)

# Platform Overview



WTK Class Hierarchy

# “Kitchen Sink” Demo



“Kitchen Sink” Demo Application

# “Hello WTKX!”

```
25
26 public class HelloWTKX implements Application {
27     private Window window = null;
28
29     @Override
30     public void startup(Display display, Map<String, String> properties)
31         throws Exception {
32         WTKXSerializer wtkxSerializer = new WTKXSerializer();
33         window = (Window)wtkxSerializer.readObject(this, "hello.wtkx");
34         window.open(display);
35     }
36
37     @Override
38     public boolean shutdown(boolean optional) {
39         if (window != null) {
40             window.close();
41         }
42
43         return false;
44     }
45
46     @Override
47     public void suspend() {
48     }
49
50     @Override
51     public void resume() {
52     }
53
54     public static void main(String[] args) {
55         DesktopApplicationContext.main(HelloWTKX.class, args);
56     }
57 }
58
```



```
18
19 <Window title="Hello WTKX!" maximized="true"
20     xmlns:wtkx="http://pivot.apache.org/wtkx"
21     xmlns="org.apache.pivot.wtk">
22     <content>
23         <Label text="Hello WTKX!"
24             styles="{font:'Arial bold 24', color:'#ff0000',
25                 horizontalAlignment:'center', verticalAlignment:'center'}/>
26     </content>
27 </Window>
28
```

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 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

# 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”
- Not mutually exclusive!



# Pivot Compared to GWT

- GWT also allows developers to use Java to write web-based applications
- Runtime environment for a GWT application is the browser, not a JVM:
  - Code executes as interpreted JavaScript, not bytecode
  - Doesn't support full Java API (no I/O, networking, threading, reflection, XML, etc.) - basically, only Java language
  - Presentation performed via CSS and DOM manipulation rather than 2D drawing API

# “Stock Tracker” Tutorial Application

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

**Pivot Stock Tracker**

Symbol	Value	Change
AAPL	\$259.40	+5.41
AMZN	\$132.78	+1.49
EBAY	\$22.64	+0.31
GOOG	\$518.36	-3.29
IBM	\$127.76	+1.49
MSFT	\$29.34	+0.40
ORCL	\$24.43	+0.04

**Apple Inc.**

Value: \$259.40  
Change: \$5.41  
Open: \$252.00  
High: \$259.70  
Low: +250.50  
Volume: 16,640,077

Symbol  + -

Last Update May 11, 2010 1:01:35 PM Data provided by [Yahoo! Finance](#)

# Stock Tracker

## Key Features

- UI markup using WTKX
- Event handling
- Web queries
- Data binding
- Localization

# UI markup using WTKX

- Pivot UI often defined in WTKX
- Hierarchical structure of XML parallels the component hierarchy, makes it easy to visualize the resulting output
- Developers are familiar with markup metaphor
- Can still be defined in code - WTKX is just a “shortcut”
- Not compiled - serialized representation of object graph
  - Generally loaded at runtime from application JARs
  - Can load dynamically (from server, for example)

# UI markup using WTKX

```
01 <stocktracker:StockTrackerWindow title="%stockTracker" maximized="true"
02   xmlns:wtkx="http://pivot.apache.org/wtkx"
03   xmlns:content="org.apache.pivot.wtk.content"
04   xmlns:stocktracker="org.apache.pivot.tutorials.stocktracker"
05   xmlns="org.apache.pivot.wtk">
06   <content>
07     <TablePane styles="{padding:8, horizontalSpacing:6, verticalSpacing:6}">
08       <columns>
09         <TablePane.Column width="1*" />
10       </columns>
11
12       <rows>
13         ...
14
15         <TablePane.Row height="1*">
16           <SplitPane splitRatio="0.4">
17             <left>
18               ...
19             </left>
20             <right>
21               <Border styles="{padding:6, color:10}">
22                 <content>
23                   <wtkx:include wtkx:id="detailPane" src="detail_pane.wtkx"/>
24                 </content>
25               </Border>
26             </right>
27           </SplitPane>
28         </TablePane.Row>
29
30         <TablePane.Row height="-1">
31           <BoxPane styles="{horizontalAlignment:'left', verticalAlignment:'center'}">
32             <Label text="%symbol" styles="{font:{bold:true}}"/>
33             <TextInput wtkx:id="symbolTextInput" textSize="10"
34               maxLength="8" />
35           </BoxPane>
36         </TablePane.Row>
37       </rows>
38     </TablePane>
39   </content>
40 </stocktracker:StockTrackerWindow>
```

# UI markup using WTKX

- Quick WTKX primer:
  - Elements
    - Uppercase = class instance
    - Lowercase = property
  - Attributes = properties
  - Namespaces = Java packages
  - “wtkx” prefix (IDs, includes, etc.)
  - Script code (logic)

# UI markup using WTKX

- Resolution operators:
  - Used in WTKX attribute values
  - ‘%’ = resource resolution (localization)
  - ‘@’ = location resolution (relative URL)
  - ‘\$’ = variable resolution

# UI markup using WTKX

- WTKX binding:
  - Maps objects defined in WTKX to Java member variables (“dependency injection”)
  - `wtkx:id` maps to `@WTKX` annotation

```
1 @WTKX private TextInput symbolTextInput = null;  
2 @WTKX private Button addSymbolButton = null;  
3 @WTKX private Button removeSymbolsButton = null;  
4 @WTKX private BoxPane detailPane = null;  
5 @WTKX private Label lastUpdateLabel = null;  
6 @WTKX private Button yahooFinanceButton = null;
```



# UI markup using WTKX

- Implementing Bindable interface ensures that bindings are processed
- Resources argument allows bound instance to retain reference to the resource bundle used to process the WTKX file

```
20
21 /**
22  * Allows WTKX serializer to automatically bind to an instance of a
23  * deserialized class.
24  */
25 public interface Bindable {
26     /**
27      * Called to initialize the class after it has been completely
28      * processed and bound by the serializer.
29      */
30     public void initialize(Resources resources);
31 }
32
```

# Event Handling

- WTKX = *structure*, code = *behavior*
- Generally executed in response to an “event” (button pressed, selection changed, etc.)
- Event listeners often wired up in Bindable#initialize()
- Can also be registered in inline script, similar to HTML

```
01 @Override
02 public void initialize(Resources resources) {
03     stocksTableView.getTableSelectionListeners().add(new TableViewSelectionListener.Adapter() {
04         @Override
05         public void selectedRangesChanged(TableView tableView, Sequence<Span> previousSelectedRanges) {
06             ...
07         }
08     });
09
10     ...
11
12     addSymbolButton.setAction(addSymbolAction);
13     removeSymbolsButton.setAction(removeSymbolsAction);
14
15     ...
16
17     yahooFinanceButton.getButtonPressListeners().add(new ButtonPressListener() {
18         @Override
19         public void buttonPressed(Button button) {
20             ...
21         }
22     });
23 }
```

# Event Handling

- Actions:
  - Extend abstract `org.apache.pivot.wtk.Action` class
  - Defines abstract `perform()` method
  - Used to attach application behaviors to multiple UI elements (e.g. toolbar button, menu item, etc.)
  - Can be enabled/disabled; attached components reflect state

# Web Queries

- Pivot's native means of server communication
- Part of "Web" class library
- Similar to XMLHttpRequest in web browser
- Facilitate communication with and implementation of REST services
- Use JSON by default, but can use any data format (XML, CSV, Java serialization, etc.)

# Web Queries

- Quote data returned by HTTP GET request to <http://download.finance.yahoo.com/d/quotes.csv/>
- Query string arguments specify symbols and fields to retrieve, returns CSV file:

```
"AAPL", "APPLE INC", 171.06, 169.59, 172.17, 166.00, +2.88, 12995693  
"AMZN", "AMAZON.COM INC", 72.54, 72.35, 73.83, 70.52, +1.10, 2748930  
"EBAY", "EBAY INC", 27.09, 27.35, 27.44, 27.04, -0.02, 3426369
```

# Web Queries

- Stock Tracker uses an instance of `org.apache.pivot.web.GetQuery` to retrieve the data
- POST, PUT, and DELETE also supported
- Uses an instance of `org.apache.pivot.serialization.CSVSerializer` to deserialize the data
- Returns the quotes as an instance of `org.apache.pivot.collections.List` which is used as the model data for the table view

# Web Queries

- By default, CSVSerializer returns an ArrayList of HashMaps
  - Untyped - all data are strings
- Can be configured to return instances of any Java Bean type
- Stock Tracker uses a StockQuote bean class to convert strings to numbers (for sorting)

# Web Queries

- `org.apache.pivot.web.Query` extends `org.apache.pivot.util.concurrent.Task`
- Abstract (generic) base class for executing background operations
- Defines a single abstract `execute()` method that returns the result of the operation
- `getQuery` returns `Object` (in this case, the result data)



# Web Queries

- `execute()` is synchronous - blocks UI
- Task provides an overload that takes an instance of `org.apache.pivot.util.concurrent.TaskListener`
- Caller is notified asynchronously via callback when task has succeeded or failed
- UI remains responsive

```
18
19 /**
20  * Task listener interface.
21  *
22  * @param <V>
23  * The return type of the task.
24  */
25 public interface TaskListener<V> {
26     /**
27      * Called when the task has completed successfully.
28      *
29      * @param task
30      * The source of the task event.
31      */
32     public void taskExecuted(Task<V> task);
33
34     /**
35      * Called when task execution has failed.
36      *
37      * @param task
38      * The source of the task event.
39      */
40     public void executeFailed(Task<V> task);
41 }
42
```

# Data Binding

- Maps values between a set of user interface elements and a data structure, called the “bind context”
- Eliminates tedious boilerplate code for manually populating field data

# Data Binding

- Uses a load/store model:
  - load() populates UI with values from context
  - store() populates context with values from UI
- Maps well to REST-based applications:
  - GET - load()
  - POST/PUT - store()

# Data Binding

- Bind context is either an instance of `org.apache.pivot.collections.Dictionary` or a Java Bean that can be wrapped in `org.apache.beans.BeanAdapter` (which implements `Dictionary`)
- Easy to bind to JSON data returned by web query
  - JSON Objects are returned as instances of `HashMap`, which implements `Dictionary`

# Data Binding

- Stock Tracker uses binding to populate quote detail form:

```
01 <Form styles="{padding:0, fill:true, showFlagIcons:false, showFlagHighlight:false,  
02   leftAlignLabels:true}">  
03   <sections>  
04     <Form.Section>  
05       <wtkx:define>  
06         <stocktracker:ValueMapping wtkx:id="valueMapping"/>  
07         <stocktracker:ChangeMapping wtkx:id="changeMapping"/>  
08         <stocktracker:VolumeMapping wtkx:id="volumeMapping"/>  
09       </wtkx:define>  
10  
11       <Label wtkx:id="valueLabel" Form.label="%value"  
12         textKey="value" textBindMapping="$valueMapping"  
13         styles="{horizontalAlignment:'right'}/>  
14       <Label wtkx:id="changeLabel" Form.label="%change"  
15         textKey="change" textBindMapping="$valueMapping"  
16         styles="{horizontalAlignment:'right'}/>  
17       <Label wtkx:id="openingValueLabel" Form.label="%openingValue"  
18         textKey="openingValue" textBindMapping="$valueMapping"  
19         styles="{horizontalAlignment:'right'}/>  
20       <Label wtkx:id="highValueLabel" Form.label="%highValue"  
21         textKey="highValue" textBindMapping="$valueMapping"  
22         styles="{horizontalAlignment:'right'}/>  
23       <Label wtkx:id="lowValueLabel" Form.label="%lowValue"  
24         textKey="lowValue" textBindMapping="$changeMapping"  
25         styles="{horizontalAlignment:'right'}/>  
26       <Label wtkx:id="volumeLabel" Form.label="%volume"  
27         textKey="volume" textBindMapping="$volumeMapping"  
28         styles="{horizontalAlignment:'right'}/>  
29     </Form.Section>  
30   </sections>  
31 </Form>
```

# Data Binding

- “textKey” property associates Label text with bind key
- Bind context is an instance of the StockQuote bean returned by GetQuery/CSVSerializer
- Uses “bind mapping” to transform data during binding:

```
01 public class ValueMapping implements Label.TextBindMapping {
02     private static final DecimalFormat FORMAT = new DecimalFormat("$0.00");
03
04     @Override
05     public String toString(Object value) {
06         return Float.isNaN((Float)value) ? null : FORMAT.format(value);
07     }
08
09     @Override
10     public Object valueOf(String text) {
11         throw new UnsupportedOperationException();
12     }
13 }
```

# Localization

- Translatable text and other resources stored in “resource bundles”
- In Pivot, resource bundles are JSON files rather than .properties files
- Use UTF-8 natively, vs. ISO-8859
- May be hierarchical, vs. flat

# Localization

- Stock Tracker resource bundles (default and 'fr'):

```
01 {   stockTracker: "Pivot Stock Tracker",
02     symbol: "Symbol",
03     companyName: "Company",
04     value: "Value",
05     openingValue: "Open",
06     highValue: "High",
07     lowValue: "Low",
08     change: "Change",
09     volume: "Volume",
10     addSymbol: "Add symbol",
11     removeSymbol: "Remove selected symbols",
12     lastUpdate: "Last Update",
13     dataProvidedBy: "Data provided by",
14     yahooFinance: "Yahoo! Finance"
15 }
```

*StockTracker.json*

```
01 {   stockTracker: "La Bourse Pivot",
02     symbol: "Code",
03     companyName: "Société",
04     value: "Cours",
05     openingValue: "Ouverture",
06     highValue: "+ Haut",
07     lowValue: "+ Bas",
08     change: "Variation",
09     volume: "Volume",
10     addSymbol: "Ajouter un code",
11     removeSymbol: "Enlever codes sélectionnés",
12     lastUpdate: "Dernier échange",
13     dataProvidedBy: "Données fournies par",
14     yahooFinance: "Yahoo! Finance"
15 }
```

*StockTracker\_fr.json*



# Localization

- Quote detail form uses localized form labels:

```
01 <Form styles="{padding:0, fill:true, showFlagIcons:false, showFlagHighlight:false,  
02   leftAlignLabels:true}">  
03   <sections>  
04     <Form.Section>  
05       <wtkx:define>  
06         <stocktracker:ValueMapping wtkx:id="valueMapping"/>  
07         <stocktracker:ChangeMapping wtkx:id="changeMapping"/>  
08         <stocktracker:VolumeMapping wtkx:id="volumeMapping"/>  
09       </wtkx:define>  
10  
11       <Label wtkx:id="valueLabel" Form.label="%value"  
12         textKey="value" textBindMapping="$valueMapping"  
13         styles="{horizontalAlignment:'right'}/>  
14       <Label wtkx:id="changeLabel" Form.label="%change"  
15         textKey="change" textBindMapping="$valueMapping"  
16         styles="{horizontalAlignment:'right'}/>  
17       <Label wtkx:id="openingValueLabel" Form.label="%openingValue"  
18         textKey="openingValue" textBindMapping="$valueMapping"  
19         styles="{horizontalAlignment:'right'}/>  
20       <Label wtkx:id="highValueLabel" Form.label="%highValue"  
21         textKey="highValue" textBindMapping="$valueMapping"  
22         styles="{horizontalAlignment:'right'}/>  
23       <Label wtkx:id="lowValueLabel" Form.label="%lowValue"  
24         textKey="lowValue" textBindMapping="$changeMapping"  
25         styles="{horizontalAlignment:'right'}/>  
26       <Label wtkx:id="volumeLabel" Form.label="%volume"  
27         textKey="volume" textBindMapping="$volumeMapping"  
28         styles="{horizontalAlignment:'right'}/>  
29     </Form.Section>  
30   </sections>  
31 </Form>
```

# Localization



- Et voilà!

## La Bourse Pivot

Code	Cours	Variation
AAPL	\$259,85	-3,27
AMZN	\$126,71	-2,05
EBAY	\$22,57	+0,39
GOOG	\$505,70	+0,10
IBM	\$126,56	-1,40
MSFT	\$26,40	-0,46
ORCL	\$22,73	-0,11

## Apple Inc.

Cours:	\$259,85
Variation:	-\$3,27
Ouverture:	\$258,47
+ Haut:	\$261,90
+ Bas:	+257,10
Volume:	10 813 601

Code   

Dernier échange 4 juin 2010 12:02:20

Données fournies par [Yahoo! Finance](#)

# Summary

- Pivot is a platform for building modern GUI applications in Java that can be deployed via the web or to the desktop
- Stock Tracker tutorial demonstrates some key features and is a great quick-start example

# Further Information

- <http://pivot.apache.org>
- <http://pivot.apache.org/demos/>
- <http://pivot.apache.org/tutorials/>
- <http://pivot.apache.org/1.5/docs/api/>

# Q & A