

JavaFX

Datasources:

Getting Real-World Data into JavaFX Controls

Johan Vos - LodgON
Jonathan Giles - Oracle

JavaOne 2011

Disclaimer

Disclaimer #1:

This is **not** an Oracle talk – this is a talk by Johan and Jonathan on a project that we do in our own time. This may or may not have any relationship to future Oracle products. Jonathan is an employee of Oracle, but is **not** speaking for Oracle in this session.

Disclaimer #2:

THE FOLLOWING IS INTENDED TO OUTLINE OUR GENERAL PRODUCT DIRECTION. IT IS INTENDED FOR INFORMATION PURPOSES ONLY, AND MAY NOT BE INCORPORATED INTO ANY CONTRACT. IT IS NOT A COMMITMENT TO DELIVER ANY MATERIAL, CODE, OR FUNCTIONALITY, AND SHOULD NOT BE RELIED UPON IN MAKING PURCHASING DECISION. THE DEVELOPMENT, RELEASE, AND TIMING OF ANY FEATURES OR FUNCTIONALITY DESCRIBED FOR ORACLE'S PRODUCTS REMAINS AT THE SOLE DISCRETION OF ORACLE.

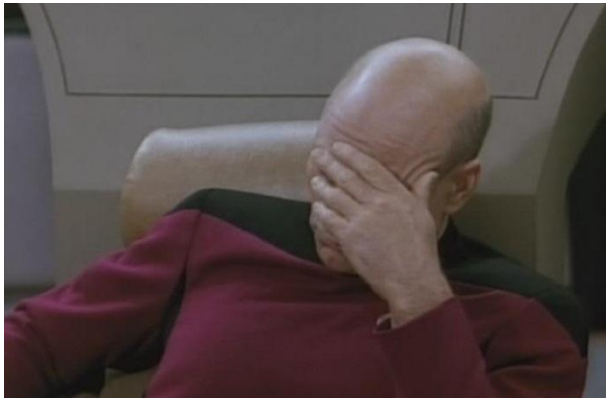
The Problem

Manager: “Bob, our users are complaining we aren’t showing them enough information”

Bob: *internalises anguished scream, already knowing what the next sentence is going to be*

Manager: “We need you to add a table into our app that shows all the information that is missing”

Bob:



JavaOne 2011

Problem Statement

- Getting data into a Table can often be difficult:
 - Retrieving data from various data sources
 - Massaging data
 - Loading data into lists / tables efficiently (maybe on-demand)
 - Rendering the data can be fidgety (when not text-only)
 - Allowing editing of data (with proper bindings to the data source) can be complex
- Hint: We think we have some code in this talk that may help...

Outline

- Introduction to JavaFX 2.0 controls
 - ListView
 - TableView
- Introduction to JavaFX 2.0 cell factories
- Introduction to data sources
 - Retrieving/parsing/rendering data
 - Mapping data with controls
 - Static and dynamic data
- 'EnergyCo' Example

JavaFX 2.0 Controls

- Primary focus of this talk: ListView and TableView
- Common API:
 - Both have a generic type
 - 'items' ObservableList: the raw data.
 - 'cell factory' support: the renderer of the raw data
- JavaFX Controls leverage generics, e.g.
 - ListView<T>
 - TableView<T>
 - TreeView<T>
 - With <T> the type of the items contained within the control.

Creating a ListView Instance

```
// Get the data that we want to show in the ListView
ObservableList<Person> people = ...

// Create a ListView control
ListView<Person> peopleList = new ListView<Person>();

// Set the data in the items list (or just pass in to the constructor)
peopleList.setItems(people);
```

Creating a TableView Instance

```
// Get the data that we want to show in the TableView
ObservableList<Person> people = ...

// Create a TableView control
TableView<Person> peopleTable = new TableView<Person>();

// Set the data in the items list (or just pass in to the constructor)
peopleTable.setItems(people);

// Create TableColumn instances for each column we want
TableColumn<Person, String> firstNameCol = new TableColumn<>("First");

// Tell the TableColumn how to extract the value from the row object
firstNameCol.setCellValueFactory(...);

// Add the TableColumn to the columns list of the TableView
peopleTable.getColumns().addAll(firstNameCol);
```

How to Populate a TableColumn

- Each TableColumn needs a Cell Value Factory
 - Otherwise the column will be blank
- It tells the TableColumn how to extract a value for a cell from a single item from the TableView.items list
- It is a `Callback<CellDataFeatures<S,T>, ObservableValue<T>>`
 - S is the type of the TableView
 - T is the type of the TableColumn

Creating Cell Value Factories

```
Callback<CellDataFeatures<Person, String>, ObservableValue<String>> callback =  
    new Callback<>() {  
        public ObservableValue<String> call(CellDataFeatures<Person, String> p) {  
            // Note: We are returning an ObservableValue, not the value itself!  
            return p.getValue().firstNameProperty();  
        }  
    };
```

```
firstNameCol.setCellValueFactory(callback);
```

Alternatively...

```
firstNameCol.setCellValueFactory(new PropertyValueFactory("firstName"));

// PropertyValueFactory uses reflection to attempt to find either a
// firstNameProperty() method that returns the correct type, and if that
// fails, it will attempt to return getFirstName() / isFirstName().

// Of course, getFirstName() returns a String, not an
// ObservableValue<String>...

// So, we wrap using a ReadOnlyObjectWrapper
// You can do this too if your objects aren't JavaFX classes
// The downside is that the UI won't update dynamically
```

Alternatively...

```
ObservableList<Map> personsMapList = ...
TableView<Map> tableView = new TableView<Map>(personsMapList);

firstNameCol.setCellValueFactory(new MapValueFactory("firstName"));

// "firstName" is assumed to be a key in the Map.
// There is a Map for each row in the TableView
```

Cell Factories

- Common API across ListView, TreeView, TableView
- Responsible for displaying a single row/cell in a control
 - ListView / TreeView: One cell per row
 - TableView: One cell per row + one cell for each column
- Default cells call toString() on the cell.item object
- Custom cells can do anything...

Creating a Cell Factory

```
// The default TableColumn cell factory
new Callback<TableColumn, TableCell>() {
    @Override public TableCell call(TableColumn param) {
        return new TableCell() {
            @Override protected void updateItem(Object item, boolean empty) {
                if (item == getItem()) return;

                super.updateItem(item, empty);

                if (item == null) {
                    super.setText(null);
                    super.setGraphic(null);
                } else if (item instanceof Node) {
                    super.setText(null);
                    super.setGraphic((Node)item);
                } else {
                    super.setText(item.toString());
                    super.setGraphic(null);
                }
            }
        };
    }
};
```

Supporting Cell Editing

```
public class TextFieldCell extends ListCell<String> {
    private TextField textField;

    @Override public void startEdit() {
        super.startEdit();

        if (! isEditable() || ! getListView().isEditable()) return;

        if (textField == null) {
            createTextField();
            textField.setText(getItem());

            setText(null);
            setGraphic(textField);
            textField.selectAll();
        }

        @Override public void cancelEdit() {
            super.cancelEdit();

            setText(getItem());
            setGraphic(null);
        }
    }

    private void createTextField() {
        textField = new TextField();
        textField.setOnKeyReleased(new EventHandler<KeyEvent>() {
            @Override public void handle(KeyEvent t) {
                if (t.getCode() == KeyCode.ENTER) {
                    commitEdit(textBox.getText());
                } else if (t.getCode() == KeyCode.ESCAPE) {
                    cancelEdit();
                }
            }
        });
    }
}
```

Supporting Cell Editing

```
@Override public void updateItem(String item, boolean empty) {
    super.updateItem(item, empty);

    if (empty) {
        setText(null);
        setGraphic(null);
    } else {
        if (isEditing()) {
            textField.setText(item);
            setText(null);
            setGraphic(textField);
        } else {
            setText(item);
            setGraphic(null);
        }
    }
}
```

Data Sources

- Controls can visualize
 - Different kinds of data (ListView<T>)
 - From different sources (file, database, network)
 - In different formats (XML,CSV,JSON, Java Objects)
 - With different dynamic capabilities (static versus dynamic)
- JavaFX Data abstracts the source and the format of the data, and allows for dynamic updates
- Visualization is not dependent on the source and the format of the data

Examples of Data Sources

CSV:

```
Year,Rank,Company,Revenue (in millions),Profit (in millions)
1955, 1,General Motors,9823.5,806
1955,2,Exxon Mobil,5661.4,584.8
1955,3,U.S. Steel,3250.4,195.4
1955,4,General Electric,2959.1,212.6
1955,5,Esmark,2510.8,19.1
1955,6,Chrysler,2071.6,18.5
```

XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<people>
  <person>
    <firstName>Jonathan</firstName>
    <lastName>Giles</lastName>
    <country>Australia</country>
  </person>
  <person>
    <firstName>Johan</firstName>
    <lastName>Vos</lastName>
    <country>Belgium</country>
  </person>
</people>
```

JavaObjects

Data Sources: Division Of Labor

- Retrieving, parsing and rendering data are three different, domain-agnostic tasks.
 - Retrieving: Pulling in data from local or remote data sources (File, local/remote database, network resources)
 - Parsing: Converting the data into a form that can be passed into UI controls (XML, JSON, CSV, Java Objects)
 - Rendering: Providing a means of viewing information in visually rich ways.
- By using Cell Factories, JavaFX Controls provide a customizable way to render data...
- ...But at present JavaFX lacks support for simple data retrieval, parsing, and rendering of common data formats.

Introducing the DataFX Project

Goal: To fill gaps in both the data source and cell factory areas of JavaFX 2.0.

Limit amount of boilerplate code in typical JavaFX applications by providing an extendable set of convenient CellFactories and DataSources

Retrieving Data

- org.javafxdata.datasources.io package provide a number of implementations of DataSourceReader
 - FileReader
 - NetworkReader
 - JDBCReader (To be implemented...)
- DataSource classes do not know what parts of the data are needed
 - A JDBC based resource contains many columns, but only 2 of those should be rendered in a TableView
 - An XML resource contains structured information, but not all elements/attributes should be rendered

Parsing data and mapping with Controls

- `org.javafxdata.datasources.protocol` package contains wrappers that:
 - Gather the application-specific portions of the data and provide those to the JavaFX Controls
 - Transform a specific protocol (e.g. CSV, XML) into selectable items for the JavaFX Controls.
- **Examples:**
 - Column headers match the CSV headers
 - Column headers can be mapped with XML elements or attributes (XPath)
 - Column headers can be mapped with some specific fields of a (remote) Java Object

Putting it together

```
DataSourceReader dsr1 = new FileReader("foo.xml");  
DataSourceReader dsr2 = new NetworkReader (http://foo.bar/foo.xml);  
  
XmlDataSource ds1 = new XmlDataSource(dsr1);  
  
TableView tableView = new TableView();  
tableView.setItems(ds1);  
tableView.getColumns().addAll(ds1.getColumns());
```

Static versus Dynamic

- Some data elements are static
 - Read-only file system
 - Single call to a webservice
- Some data elements are dynamic
 - Update requested by control (pull)
 - Poll for new mail every 5 minutes
 - Update requested by external component (push)
 - Requires synchronization protocol
- DataSources hide the complexity
 - Use static data in local development, and dynamic server-updated data in staging environment

RedFX

- Library that allows remote synchronization of data.
 - Remote webservices
 - Messaging
 - Remote Objects
- Remote Objects:
 - If the value of a data element changes on one client, or in a backend component, the same element is updated on all other clients.
 - RedFX Remote Objects can be processed via an ObjectDataSource wrapper.
 - The visualized data is updated immediately.
- <http://www.redfx.org>

Common Cell Factories

- Ranked in order of importance (imho):
 - Text editing
 - Check box
 - Drop-down menu
 - Context-specific rendering (red for negative, green for positive)
 - Images
 - Progress bar
 - Expand-on-click
 - Everything else...
- JavaFX 2.0 does not ship with any cell factories

Common Cell Factories

- Ranked in order of importance (imho):
 - ✓ Text editing
 - ✓ Check box
 - ✓ Drop-down menu
 - ✓ Context-specific rendering (red for negative, green for positive)
 - ✓ Images
 - ✓ Progress bar
 - ✓ Expand-on-click
 - ✓ Everything else...
- We've pre-built all of the above...

EnergyCo Demo

JavaOne 2011

What's Next?

- Current implementation is read-only.
- Write support can be imagined simply as list write operations.
- Support for TreeView API
- Support for Charts API
- More data sources
 - JDBC, common web services, etc
- More cell factories
- More convenience
- On-Demand support

Where Can I Get This?

- This is a free, open source project built on top of JavaFX 2.0
- It is **not** an Oracle-sponsored project
- BSD-ish license (we are not lawyers) – contact us if you're concerned
- It can be downloaded right now:

<http://www.javafxdata.org>

Thanks

Johan Vos
Jonathan Giles

johan@lodgon.com
jonathan.giles@oracle.com

JavaOne 2011