From Swing to DukeScript

From Swing to DukeScript

For a long time, developing applications in Swing was the preferred method for writing cross-platform applications. However, following the decision of Oracle to put Swing in maintenance mode, it is no longer an option for new projects.

Developers of existing projects are also looking for alternatives. This report analyzes the use of DukeScript's Java API[i] for Swing developers and compares it with the alternatives.

Current State of Swing

Oracle has decided to put Swing in „maintenance mode", which means that only serious bugs will be fixed. Swing will still be around for a few years, but there will be no new feature development. Browser vendors discourage the use of the Java Plugin, meaning that you can no longer distribute applications using WebStart and Java Applets. This makes Swing a bad choice for all new application development. Owners of existing applications should consider moving to a different platform.

Alternatives to Swing

There are several alternatives to Swing. For this comparison, we'll look at cross-platform technologies. If you would like to continue using Java, the main options are GWT, JavaFX and DukeScript. If you would like to take the risk of switching to a different language, the primary option for creating cross-platform applications is JavaScript and HTML.

For a Swing developer with an existing code base, these are the criteria for a smooth transition, with maximal reuse and minimal risk:

- Java Syntax: Swing developers already know Java. No additional risk, and no time, and money for training.
- Java APIs: If the framework supports the standard Java SE Libraries, then business code can be reused.
- Embeddable: Some technologies can be combined with Swing. This minimizes risk, and the application can be ported step-by-step.
- Design Tools: Professional Tools for designing and testing are essential for high-quality User Interfaces and development speed.
- Development Tools: Availability of good tools for writing code is essential for bug prevention.
- Tools for Debugging: High-quality debugging tools for code, such as visual UI inspection, can speed up bug fixing.
- Hot Swapping: Being able to deploy code changes into the running application speeds up bug fixing and development.
- External Services: Outsourcing design and UI implementation dramatically boosts developer productivity.
- MVVM: A modern architecture allows for changing the view code and business code independently.
- Testability: Good test coverage reduces maintenance costs.
- Shallow learning curve: Becoming productive with a new technology takes time and effort.

i http://dukescript.com

JavaScript

JavaScript is available on most platforms and has the largest momentum. There are many tools for designing the UI with HTML and CSS. Popular frameworks like AngularJS and Knockoutjs support the MVVM architecture, allowing you to develop the application logic independently of the view. This enables great collaborative opportunities for designers and developers with clear responsibilities. The design and implementation of the view can easily be outsourced.

JavaScript was originally designed for manipulating the DOM of a document. One of the language features that increase productivity is "dynamic typing." This helps with rapid implementation of small tasks, such as creating animations for a website, with less code.

On the other hand, it also prevents the creation of good development tools. Without a type system, the IDE has less information and cannot offer the same quality of assistance to the developer as it does in Java. This is not a major issue for simple web pages being developed by a single web developer, but it quickly becomes a big problem in larger projects, where developers collaborate and rely on code libraries written by others.

```
GraphicsContext2D gc = GraphicsUtils.getOrCreate
gc.fillRect(0, 0, 10, 10);
gc.setLineWidth(30);
    setLineCap(String string)  void
    setLineJoin(String string) void
    setLineWidth(double d)     void
0, 255, 0.5)"
gc.setFont("24px Helvetica");
```

```
var gc = c.getContext("2d");
gc.fillRect(0, 0, 10, 10);
gc.lineW
    No suggestions
```

Figure 1: Code completion in Java (left), JavaScript (right)

With Java code, the editor knows what arguments a method accepts. In JavaScript, a developer must read and understand the foreign code, or the developer of an API needs to provide and maintain additional documentation. This problem is directly proportional to the size of the application. An even bigger problem is that there's no compiler to find errors early on. Many errors can only be caught late in the running application, when they are expensive to fix.

According to the evaluation criteria, the combination of HTML and JavaScript has many of the features we're looking for, but the drawback of having to learn a new language that is inferior to Java.

GWT

GWT tries to solve this problem by compiling Java code to JavaScript to run the application in the browser. This fixes JavaScript's problem with code assistance and allows Swing developers to use their Java knowledge.

However, GWT doesn't allow you to reuse existing business code. It only supports a very limited subset of Java's Standard Libraries [ii]. When porting a Swing application to GWT, you cannot reuse your business code.

Developers also need to master a new UI Toolkit with many widgets. GWT creates a pure JavaScript application that runs in a browser. It is not possible to mix Swing and GWT. If you want to port an existing application, you need to start from scratch.

JavaFX

JavaFX was originally designed as a replacement for Swing, so it seems to be the natural choice for Swing developers. JavaFX support a smooth transition from Swing, as it allows mixing with Swing components. JavaFX also allows you to reuse existing Java business code.

Unfortunately, JavaFX doesn't have a clean separation of view and business code. That makes the view logic as hard to test as in Swing applications. Also, the Java developers need to implement the design. Other than in HTML/JavaScript, these tasks cannot be outsourced. The only existing design tool is a very basic form designer (SceneBuilder).

Besides that, Oracle has postponed the decision to make JavaFX its official UI toolkit. The plan to include it via a JSR in Java SE 9 [iii] has been cancelled [iv]. Oracle has cancelled its mobile JavaFX team and discontinued its support for embedded Platforms and the SceneBuilder Tool. This follows an internal trend in Oracle to move from Java to JavaScript-based UI technologies for in-house projects. The "Oracle JET framework," which has been designed for that purpose, has recently been published. Therefore, the future of JavaFX remains unclear.

DukeScript

DukeScript combines all the benefits of HTML 5 with full support for Java. The business logic and view logic of the applications are written in Java, and the view is created in HTML 5. Both are cleanly separated using the MVVM design pattern. This limits the use of HTML 5 to rendering the view, the area where it shines, and leverages the rich and rock-solid Java APIs for the rest.

It also makes the whole view logic unit testable for increased stability and reduced maintenance costs. Developers can use the superior IDE support of Java, which is improved even further by features like hot swapping and a visual inspector.

Existing Java business code can be reused. For a smooth transition, DukeScript can be embedded in existing Swing applications. DukeScript separates view and business logic. View designers can use all the tools available for HTML and CSS, and creating the view can easily be outsourced, freeing up developers to write business code.

ii http://www.gwtproject.org/doc/latest/RefJreEmulation.html

iii http://www.oracle.com/us/corporate/press/1854982

iv http://mail.openjdk.java.net/pipermail/openjfx-dev/2015-July/017529.html

Table 1 summarizes the criteria that make DukeScript the best choice for Swing developers.

Table 1: Feature Comparison for porting Swing Applications to a new technology

Feature	JS & HTML	GWT	JavaFX	DukeScript
Java Syntax	-	+	+	+
Java APIs	-	Small subset	+	+
Embeddable	-	-	+	+
Design tools	+	-	-	+
Development Tools	-	+	+	+
Tools for Debugging	+	-	+	++
Hot Swapping	- (reload)	+/- (special mode)	-	+
External Services	+	-	-	+
MVVM	+	-	-	+
Testability	+	-	-	+
Learning curve	Very high	High	Medium	Low

DukeScript has been designed by Swing developers for Swing developers. DukeScript's core technology is developed by Oracle as part of the NetBeans project. Support for desktop and browser platforms is free and Open Source. Commercial support, training, custom development and additional support for running on iOS and Android is available from Dukehoff GmbH, based in Munich, Germany.

DukeScript Architecture

To run on a platform, DukeScript requires a JVM and an HTML renderer component. This makes DukeScript itself a very lean technology that can easily be adapted to new HTML renderers and JVMs.

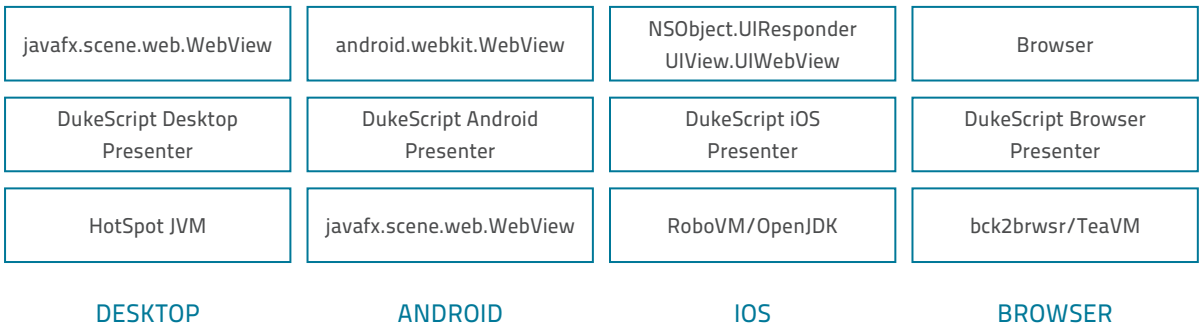

Figure 2: DukeScript Implementations on Different Platforms

The application is based on a clean separation of View and Logic. The View is defined in HTML and binds declaratively to a view model defined in Java. The view model defines the properties for this. When the application updates these properties, the view will automatically update. This architectural pattern is also known as Model View ViewModel (MVVM).

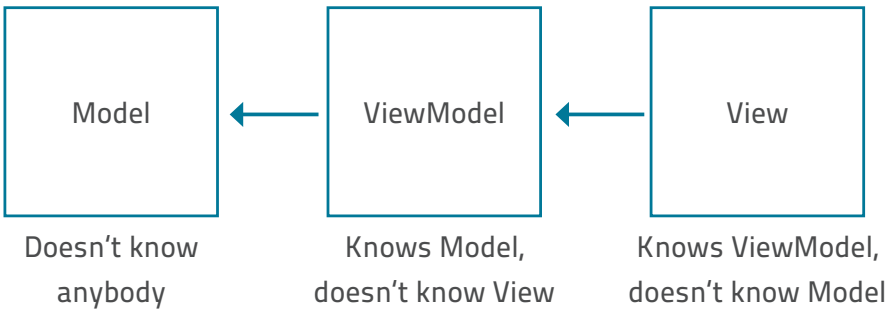

Figure 3: Model View ViewModel Architectural Pattern

The main benefit of this pattern is that the viewmodel doesn't have to know anything about the view. That means the viewmodel does not reference widgets or controls. This allows the view to be easily updated, adjusted or completely replaced without requiring any updates or changes to the viewmodel.

JavaFX has tried to create something similar (FXML), but the work was left incomplete[v]. In JavaFX, you cannot use MVVM and the Java code needs to directly reference the UI elements defined in the view.

Development in Java

DukeScript applications are written in plain Java. Java is a statically typed language, which allows tools to use static code analysis. No other language has comparable tooling, which means that Swing developers can leverage their existing knowledge and keep their great tools. DukeScript is also embeddable in existing Swing applications for a smooth transition without the risk associated with a full reimplementation of an application from scratch.

Separation of Concerns

The biggest design flaw of Swing is seen in the mixing of presentation and logic. More specifically, the developer is required to manually code the view. Creating forms using screen designers can help with that, but it doesn't change the architectural problems.

Developers need to deal with fixing visual inconsistencies and bugs in the layout, instead of writing business code. As this is a complex and difficult task in Swing, there's a large investment in building and maintaining this knowledge in the team. Experience shows that UI problems in Swing are very often responsible for failing to meet release deadlines.

In addition, if the application is supposed to adhere to a company's style guide, this task becomes extremely expensive. Developing a pixel-perfect custom Look and Feel for Swing is a task that requires about 6 person-months for a domain expert. Besides the cost factor, external services for that are hard to find.

v http://fxexperience.com/2011/10/fxml-why-it-rocks-and-the-next-phase/

DukeScript separates the view from the business logic, so developers can focus on writing code and tests. The design can be completely done by a web developer. These can either be found in-house or the design can be completely outsourced. Companies or individuals offering services for creating a design and converting it to assets are fast, cheap, and easy to find.

Shallow Learning Curve

« A steep learning curve is the price to pay for JavaFX's more modern UI options. »

Rob Terpilowski [vi]

When switching to other technologies, such as JavaFX, developers need to learn new concepts, like how to work with a SceneGraph. Besides understanding these concepts, developers also need to learn a completely new UI Toolkit, primarily the layout mechanisms, widgets and controls. The current JavaFX API consists of 691 classes. Getting familiar with these APIs is a huge task. Becoming highly productive and being able to solve problems requires an even greater effort. Without the proper experience, the process involves a lot of testing and detours. Architectural decisions made in early phases of the project often need to be corrected in later phases.

A typical example of that in JavaFX projects is the decision for or against the use of FXML. It requires a lot of experience to find the right level of detail where FXML can help without damaging performance. Changes made later in the project are expensive, can cause delays, and can result in poor code quality.

This is different with DukeScript, as developers don't have to learn a UI Toolkit. Writing the view logic in DukeScript is a skill that any developer can learn in a single day. The complete API currently consists of 45 classes. A team that is newly introduced to DukeScript can be productive within a week.

The only thing developers need to know about the view is a very basic understanding of HTML. This provides a huge advantage of being productive with your team almost from day one.

Improved Testability

Swing applications are very difficult to unit test, which is why many projects have poor test coverage or meaningless unit tests. The inventors of Swing can't be blamed for that, because when Swing was designed, unit testing barely existed. However, the same problems apply for JavaFX. Since the view code isn't separated from the business code, the Toolkit needs to be initialized for any test, and special care is required for managing UI Threads. In a more complex project, this quickly becomes a huge problem. Test coverage has become the core metric for predicting cost of ownership. Poor test coverage is synonymous with high maintenance costs.

With DukeScript, the view is separate from the view logic, so the logic is completely unit testable. Developers can follow the Test Driven Development (TDD) approach, a process where for every feature, a Unit test is created, even before starting to implement it.

vi http://www.theserverside.com/feature/Pros-cons-of-moving-from-Swing-to-JavaFX-UI-tools-a-plus

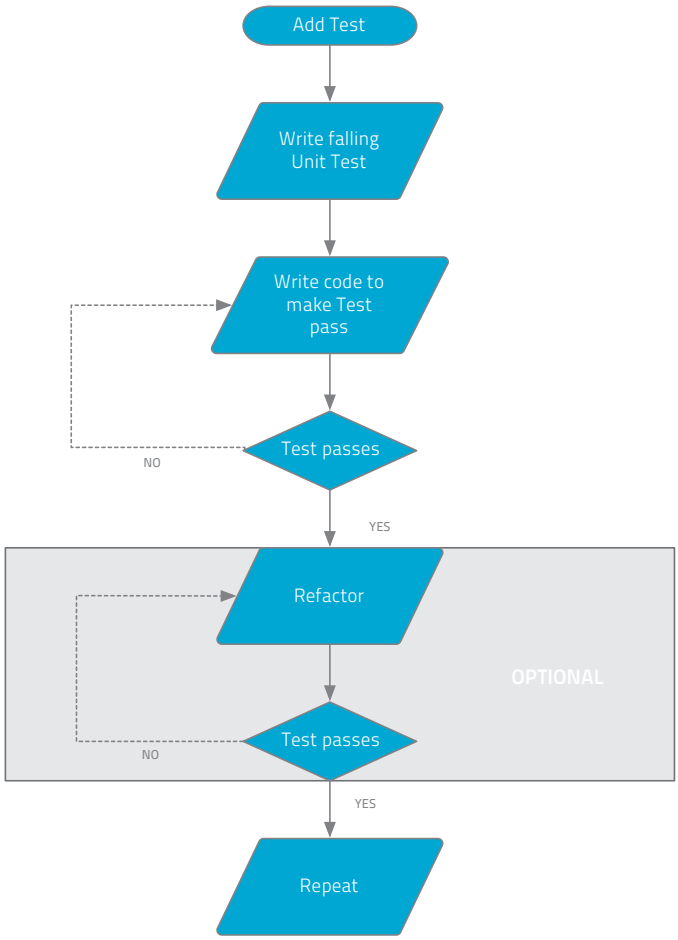

Figure 4: Test-Driven Development

With good test coverage, code can be refactored easily without any fear of side effects. In addition, unit tests are a good way to document code. To learn how a piece of code works, a developer can simply look at the associated unit tests. Unlike written documentation, a unit test cannot be outdated or incomplete, because it would fail during the testing or compilation. This is important if new developers need to take over responsibility for existing code.

Being able to refactor code with confidence and deal with changing teams is crucial for maintainability and total cost of ownership.

Single Responsibility

A typical Swing or JavaFX developer needs to implement the domain logic of an application while simultaneously developing the layout and the view, managing widgets, and converting a design into code. Besides the investment in acquiring these skills and learning the UI Toolkit, these are very different tasks that require constant switching of contexts.

Problems with the UI typically cannot be solved by logic reasoning alone, but instead require experience, and involve spending time on the Internet looking for solutions. In

the worst case scenario, solving these problems requires inspecting the source code of the UI Toolkit to find a hook for implementing the required feature, or working around a bug.

DukeScript developers only need to implement domain logic and solve algorithmic problems. This is their single responsibility. Essentially, DukeScript allows developers to focus on their core skills. Combined with a Test-Driven Approach, this allows for a straightforward and predictable development process that simplifies planning and helps in meeting project deadlines.

Tooling

When programming DukeScript applications, developers can continue to use their Java IDE and don't need to learn new tools. Through the use of Maven, DukeScript can be developed with all major IDEs.

Since the UI is developed in HTML, there are many professional commercial and free design and development tools. Web developers and designers can also continue to use their existing tools, rather than using inferior proprietary tools.

On Device Debugging

It is important to test and debug applications directly on the target device. With DukeScript, you can debug applications running on Android, and iOS devices directly from your IDE.

Visual Inspection

DukeScript allows you to visually inspect the running application. This is great for integrating the view with the view logic. The NetBeans Plugin for DukeScript integrates this directly into the IDE; with Eclipse and IDEA, similar support is built into the application browser. Visual inspection allows you to browse the live DOM tree, which highlights the corresponding element in the UI, allowing you to inspect and edit all CSS styles and properties. All changes take immediate effect. In inspection mode, you can select elements in the running application with the mouse to find and inspect them in the DOM Tree.

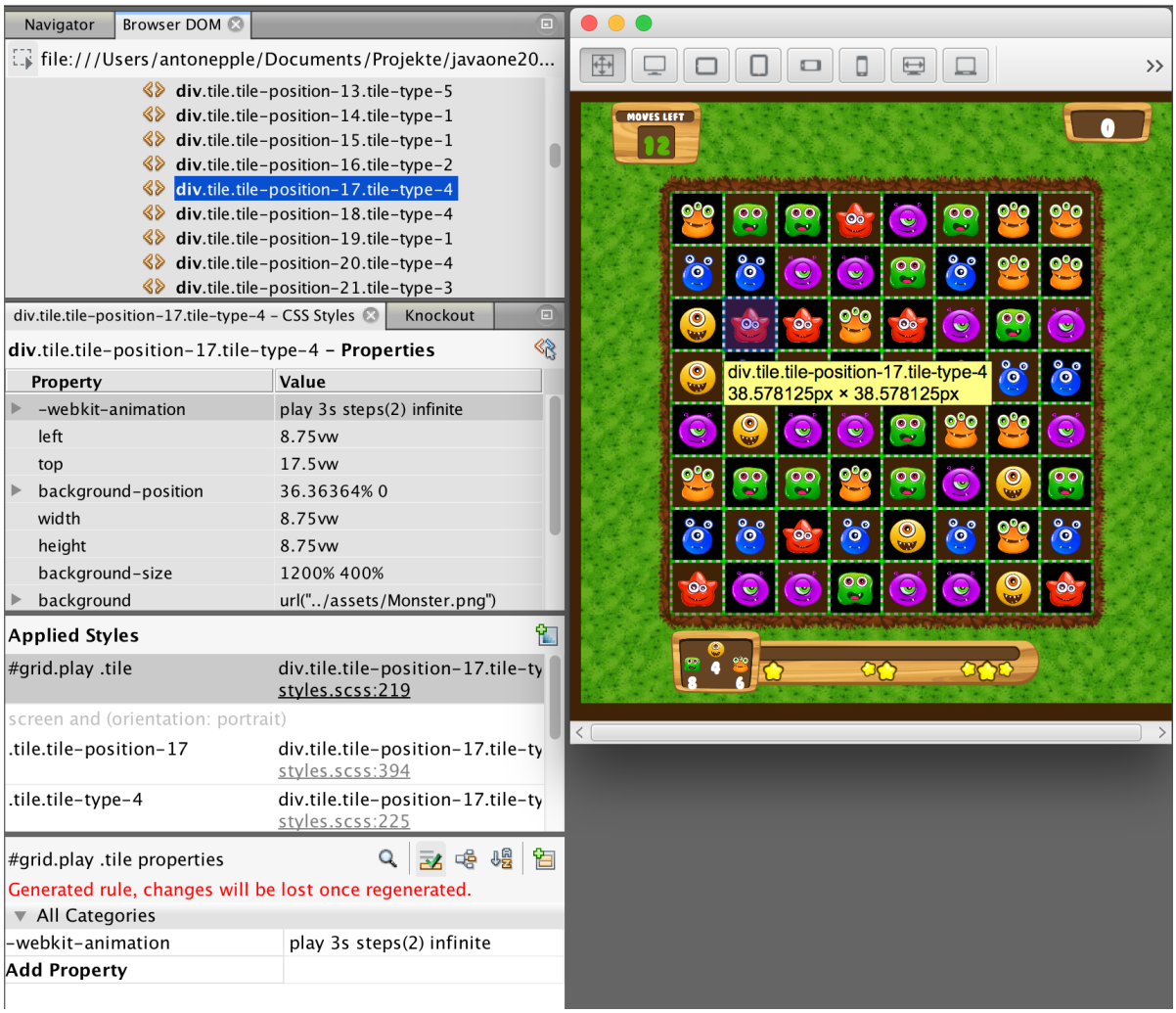

Figure 5: Visual inspection of the running application

Hot Swapping

DukeScript has excellent support for developing applications. Most importantly, it allows developers to deploy code changes into the running application. There's no need to recompile the application after a code change. Even more important than saving time, this allows developers to stay focused. This process is not simply "on par" with modern web development; it's even better than that, as the application retains its state during an update. There's no need to repeat the same steps over and over after each update to get the application into the same state as before the update. This enables a more continuous workflow without forced pauses.

Rapid Application Development With Controlsjs

The approach shown so far assumes that web developers complete the design of the application either as part of a team, or as an external resource. For small projects, where this is not an option, developers need to create the view themselves. That's fine if they're familiar with HTML and CSS, but there's also another option that can speed up

development. Position s.r.o. [vii] offers a UI Designer for creating UIs without the need to use HTML or CSS. The UIs are fully skinnable.

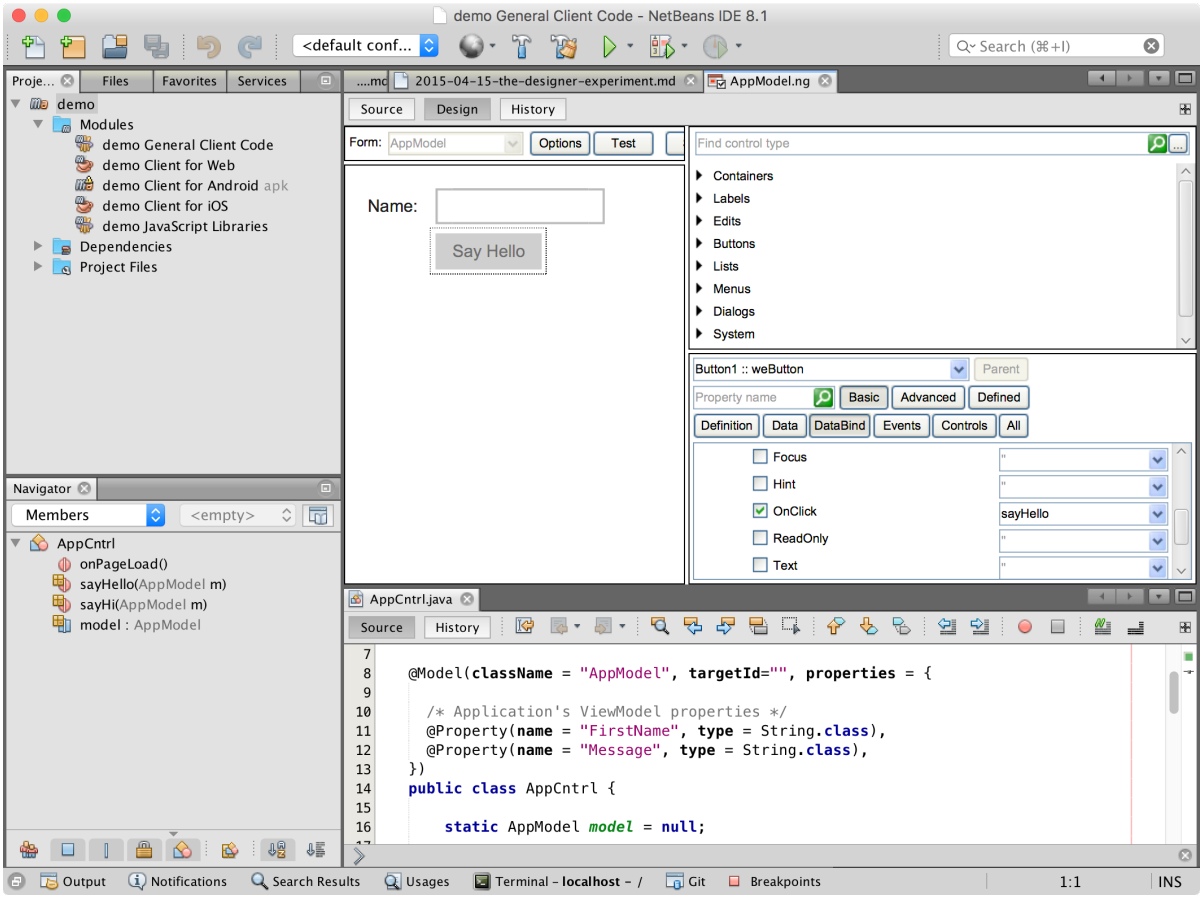

Figure 6: The contolsjs visual designer

Porting Existing Applications

For very large applications, a complete reimplementation is risky. In that situation, it makes more sense to port the application step-by-step. DukeScript can be seamlessly embedded into existing Swing applications. In the first phase, all new feature development happens in DukeScript. That way, the team has time to get familiar with the new concepts.

In the second phase, individual screens are ported to the new technology. Through improved testability and ease of development, code quality and test coverage will continuously improve as more parts of the application begin using DukeScript. In the third phase, a new project is created with no dependencies on Swing. The individual screens are moved to the new application to complete the porting.

vii http://controlsjs.com/java/

Conclusion

DukeScript provides a painless way for Java developers to embrace the latest changes in UI technology. DukeScript combines all the benefits of HTML 5, the most powerful view technology, with proven and rock-solid Java. DukeScript builds on the existing knowledge of your team to enable instant productivity. Development will benefit from better testability, a clean separation of view and logic, and the best tool support available in the market. Existing projects can be ported step-by-step, without the risks of a complete reimplementation. Business code can be reused, and tedious design tasks can easily be outsourced.

Munich-based Dukehoff GmbH offers commercial training and support. To learn more about DukeScript, visit http://dukescript.com. To get started, we recommend our book:

Get it for free here: http://dukescript.com/free_ebook.html