Projects and Components

OpenJFX is made up of three essential elements: source code for the runtime itself, sample applications, and tests. The runtime is further divided up into eight major functional components or modules: Base, Graphics, Controls, Swing, SWT, FXML, Web, and Media. All source code for OpenJFX is contained in a single Mercurial repository named "rt" (which stands for "runtime". It may be simply renamed "openjfx" when we create the repository for the post-Java 8 releases). Within this repository all of the apps, tests, and source code can be found and built. All of these components, or modules, can be found within the rt directory under "modules".

Base

The Base component contains no external dependencies, and does not rely on any graphics APIs. It is possible to use the Base component in any application, including pure headless components. It is comprised of APIs including FX beans and Observable collections.

Graphics

Everything needed to put pixels on the screen is contained in the Graphics component. This includes windowing (Glass), drawing (Prism), rasterizing (Pisces), effects (Decora), the scene graph, images, text, and so forth. Every GUI application requires at the minimal the Graphics and Base components.

Controls

This module contains all of the basic user interface charts and controls, including such things as Label, Button, and AreaChart. It is possible (using the Text node and basic primitives in the Graphics component) to build an application without the Controls component, but it is unlikely. Most applications will make use of this component.

Swing

Contains support for embedding Swing components inside of an OpenJFX scene graph, and also for embedding OpenJFX scenes within a Swing component. Also contains APIs for converting to / from BufferedImages. This component is only available on the desktop (Mac, Windows, Linux) and not on embedded or iOS or Android.

SWT

Similar in nature to the Swing component, but for SWT. This component allows you to embed OpenJFX scenes within an SWT application. This is also only available for desktop systems, not embedded, iOS, or Android.

Web

Contains all of the source code (both Java and native) for WebView. This includes the current implementation based on WebKit. This is a large and complicated component, and is not presently available for embedded systems (although we plan to bring it there in the future). Building web takes over half the time of building all of OpenJFX!

Media

The Media component is comprised of both Java and native sources for the javafx.scene.media package and associated implementation. It includes the sources for building GStreamer. The default implementation is based on h.264. ON2 support, shipped with JavaFX, is not part of OpenJFX for licensing reasons.

FXML

The FXML component contains the "FX Markup Language" parser and tooling. FXML is designed to be used either by hand (by editing XML directly) or through tooling such as with SceneBuilder.

In addition to these major functional components are additional projects which may be used for tooling, such as for SceneBuilder or command line tools for packaging JavaFX applications. These components are called DesignTime, FXPackager, and JMX. Finally, there is a Builders component which contains the deprecated builders API (which will be removed in Van Ness).

DesignTime

Contains the beginnings of support for a design-time API for third parties. This is presently used by SceneBuilder. A 3rd party developer should be able to use this API to tell tools such as SceneBuilder about their 3rd party control or layout, such that it will plug seamlessly into SceneBuilder. This component is not yet a public API, but is in development.

FXPackager

Shipped as a tool in the JDK, the FXPackager is a set of ant tasks and command line tools for producing application bundles and JNLP files (and applets) for a JavaFX application. This component is a work in progress.

JMX

Experimental support for using JMX to observe the runtime behavior of an OpenJFX application.

Apps and Tests

In addition to these modules, the rt workspace contains apps and tests. The apps are all located beneath the "apps" directory. The list of apps available will grow over time, however two are of particular mention: Ensemble and Modena. The Ensemble application is our sampler and contains many different individual samples, including source code and documentation. It is often run as a "test of last resort" to see if anything unexpected broke that wasn't caught by the other tests, and is one of the samples we publish.

Modena is another test app which is specifically used to verify that the Modena UI theme is both correct and pleasing. There are automated tests for running this app and verifying that the UI hasn't changed unexpectedly. It is also a useful application when working on modena.css, to see what effect your changes have.

There are many different kinds of tests used by OpenJFX, including:

- White box unit tests
- Black box functional tests
- Manual tests
- Performance tests

The white box unit tests are co-located with the project or component that the test belongs to. For example, the FXCollectionsTest is located in modules/base/src/test/java, and the sources for FXCollections live in modules/base/src/main/java. The other kinds of tests are all located beneath the tests directory in rt. These tests are further subdivided into "functional", "manual", and "performance" directories[1].

Functional tests are black box tests, which means that they never test implementation classes and are expected to work solely against the public API. These tests should be written without leveraging deep knowledge of the class to be tested, but rather, should use the API in all the ways that a normal user would do. These tests should make use of JemmyFX where appropriate, such that the tests can be run in any of a number of environments including WebStart, Applet, desktop, and headless deployments[2]. The existing SQE tests will be folded into this location.

Finally, performance tests are based on the jmh tool from OpenJDK, used for performance benchmarking.

This portion of the repository has not been finished, and there are few tests in this location today

Headless testing with JemmyFX is not yet possible and requires a headless version of Glass