THE EXPERT'S VOICE® IN JAVA

# Pro JavaFX 2

## A Definitive Guide to Rich Clients with Java Technology

James L. Weaver, Weiqi Gao, Ph.D., Stephen Chin,
and Dean Iverson, with Johan Vos, Ph.D.

Foreword by Nandini Ramani
Vice President, Fusion Middleware Group, Oracle Corporation

Apress®

# Pro JavaFX 2

## A Definitive Guide to Rich Clients with Java Technology

James L. Weaver
Weiqi Gao, Ph.D.
Stephen Chin
Dean Iverson
with Johan Vos, Ph.D.

Apress®

**Pro JavaFX 2: A Definitive Guide to Rich Clients with Java Technology**



ISBN 978-1-4302-6872-7

ISBN 978-1-4302-6873-4 (eBook)



President and Publisher: Paul Manning
Lead Editor: Richard Carey
Technical Reviewer: Carl Dea
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh
Coordinating Editors: Stephen Moles and Annie Beck
Copy Editor: Valerie Greco and Heather Lang
Production Support: Patrick Cunningham
Indexer: SPi Global
Artist: SPi Global
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/bulk-sales`.

Any source code or other supplementary materials referenced by the author in this text is available to readers at `www.apress.com`. For detailed information about how to locate your book's source code, go to `http://www.apress.com/source-code/`.

# Contents at a Glance

# Contents

# Foreword

I remember it distinctly, like it was yesterday. Standing center stage at Moscone Center when we launched JavaFX at JavaOne 2007. We promised to build a world-class client platform for Java. With the world watching with skeptical eyes and in a crowded client arena, we set out to build the dream. In hindsight, it was a rather ambitious goal.

Fast forward four years, with the release of JavaFX 2, we have taken a huge leap forward in fulfilling that promise. As the vision unfolded, our product plans have shifted to match the evolving RIA market and what developers and the Java community told us they were looking for. As someone who was there at the inception of JavaFX and has watched it mature over the last four years to this current release, my feelings are akin to a parent watching a toddler blossom.

Jim Weaver and Stephen Chin have been traveling through the evolution of JavaFX with me. They have both presented on JavaFX at numerous international conferences and have been developing with and blogging about JavaFX since 2007. Jim is a 30-year software veteran and has authored several books on Java, as well as articles for *Java Magazine* and *Oracle Technology Network*. He has also developed numerous JavaFX applications for a wide variety of customers.

Stephen is passionate about open source technologies and is the the founder of WidgetFX and JFXtras. He also has a deep passion for improving development technologies and processes, as well as agile development methodologies.

Johan Vos is co-founder of LodgOn and holds a PhD in applied physics. His interest lies in the enterprise communication aspects of JavaFX, combining the world of large servers with end-user devices. Johan's analogy to physics: The grand unified theory combines quantum mechanics (small) with relativity theory (large); similarly in software, Java combines JavaFX with Java EE. Dean Iverson is a longtime client developer with a great eye for creating elegant user interfaces. He develops GroovyFX libraries and is a contributor to the JFXtras project. He has been developing and blogging about JavaFX since 2007. Weiqi Gao holds a PhD in mathematics. His expertise is in the language aspects of JavaFX as is reflected in the chapters on Properties and Bindings and Collections and Concurrency.

Today, the core JavaFX team at Oracle still has several of the developers who were part of the early versions of JavaFX and we also have new engineers who have joined us. As we move ahead and open source JavaFX, we are looking forward to having more developers and experts from the extended Java community join us in making JavaFX the number one choice for client development.

I am proud and honored to be part of this key software technology. Given the length of experience and depth of expertise in all aspects of JavaFX and across the Java platform, I cannot think of a better group of authors to bring you JavaFX 2. I hope you will enjoy this book and find JavaFX as satisfying as I have found it over the years. I hope it piques your interest sufficiently to join the JavaFX community in making JavaFX the platform of choice for clients.

Nandini Ramani
Vice President, Fusion Middleware Group
Oracle Corporation

# About the Authors

**James L. (Jim) Weaver** is a Java and JavaFX developer, author, and speaker with a passion for helping rich-client Java and JavaFX become preferred technologies for new application development. Books that Jim has authored include *Inside Java, Beginning J2EE,* and *Pro JavaFX Platform*, with the latter being updated to cover JavaFX 2. His professional background includes 15 years as a systems architect at EDS, and the same number of years as an independent developer. Jim is an international speaker at software technology conferences, including the JavaOne conferences in San Francisco and São Paulo. Jim blogs at http://javafxpert.com, tweets @javafxpert, and may be reached at jim.weaver @javafxpert.com.

**Weiqi Gao** is a principal software engineer with Object Computing, Inc., in St. Louis, MO. He has more than 18 years of software development experience and has been using Java technology since 1998. He is interested in programming languages, object-oriented systems, distributed computing, and graphical user interfaces. He is a presenter and a member of the steering committee of the St. Louis Java Users Group. Weiqi holds a PhD in mathematics.

**Stephen Chin** is chief agile methodologist at GXS and a technical expert in client UI technologies. He is lead author on the Pro Android Flash title and coauthored the Pro JavaFX Platform title, which is the leading technical reference for JavaFX. In addition, Stephen runs the very successful Silicon Valley JavaFX User Group, which has hundreds of members and tens of thousands of online viewers. Finally, he is a Java Champion, chair of the OSCON Java conference, and an internationally recognized speaker featured at Devoxx, Codemash, AnDevCon, Jazoon, and JavaOne, where he received a Rock Star Award. Stephen can be followed on twitter @steveonjava and reached via his blog: http://steveonjava.com.

**Dean Iverson** has been writing software professionally for more than 15 years. He is employed by the Virginia Tech Transportation Institute, where he is a rich client application developer. He also has a small software consultancy called Pleasing Software Solutions, which he cofounded with his wife.

**Johan Vos** started to work with Java in 1995. As part of the Blackdown team, he helped port Java to Linux. With LodgON, the company he cofounded, he has been mainly working on Java-based solutions for social networking software. Because he can't make a choice between embedded development and enterprise development, his main focus is on end-to-end Java, combining the strengths of backend systems and embedded devices. His favorite technologies are currently Java EE/Glassfish at the backend and JavaFX at the frontend. Johan's blog can be followed at http://blogs.lodgon.com/johan, he tweets at http://twitter.com/johanvos, and can be reached at johan@lodgon.com.

# About the Technical Reviewer

**Carl P. Dea** is a currently a senior software engineer working for BCT-LLC on projects with high performance computing (HPC) architectures. He has been developing software for over 15 years with many clients from Fortune 500 companies to nonprofit organizations. He has written software ranging from mission-critical applications to Web applications. Carl has been using Java since the very beginning and he also is a huge JavaFX enthusiast dating back when it used to be called F3. His passion for software development started when his middle-school science teacher showed him the TRS-80 computer. His current software development interests are: rich-client applications, Groovy, game programming, Arduino, mobile phones, and tablet computers. Carl's blog can be found at http://carlfx.wordpress.com, and he tweets at @carldea.

# Acknowledgments

This book is dedicated to my wife, Julie, daughters Lori and Kelli, son, Marty, and grandchildren, Kaleb and Jillian. Thanks to Merrill and Barbara Bishir, Marilyn Prater, and Walter Weaver for being such wonderful examples. A very special thanks to Weiqi Gao, Stephen Chin, Dean Iverson, Johan Vos, and Carl Dea, with whom I had the privilege of writing this book. Thanks also to the amazing JavaFX teams at Oracle and the talented editors at Apress. *"I have told you all this so that you may have peace in me. Here on earth you will have many trials and sorrows. But take heart, because I have overcome the world." (John 16:33)*

Jim Weaver

I would like to thank my wife, Youhong Gong, for her support, understanding, and encouragement during the writing process. My thanks also go to the author and technical review team: Jim Weaver, Stephen Chin, Dean Iverson, Johan Vos, and Carl Dea for making this book a fun project. I share with my coauthors the appreciation to the JavaFX team at Oracle and the editorial team at Apress.

Weiqi Gao

To my wife, Justine, and daughter, Cassandra, who supported me in writing this book on top of all my other responsibilities. Also, a huge thanks to the entire author team, including our newest members, Johan Vos and Carl Dea, who both went above and beyond in their contributions to this title. Finally, a great debt of gratitude to the JavaFX team and JVM language designers who have produced technology that will profoundly change the way we design and code UIs going forward.

Stephen Chin

I would like to thank my family, Sondra, Alex, and Matt, for their support and understanding during yet another writing project. You guys make this possible. I would also like to thank the writing and review team of Jim Weaver, Stephen Chin, Weiqi Gao, Johan Vos, and Carl Dea for their dedication and their patience. The editorial team at Apress was, as usual, first rate and utterly professional. And of course none of this would be possible without the hard work of an extremely talented team of engineers on the JavaFX team at Oracle.

Dean Iverson

Writing a book is often done in spare time. I want to thank my wife, Kathleen, and our son, Merlijn, for allowing me to spend evening and weekend time in front of my computer. I'm very pleased to be involved in this JavaFX book, and I want to thank authors Jim Weaver, Weiqi Gao, Stephen Chin, and Dean Iverson, technical reviewer Carl Dea, and the Apress team for their trust in me. A special thanks to my colleagues Joeri Sykora and Erwin Morrhey for helping me with the examples. The JavaFX team at Oracle did a great job releasing JavaFX 2. The combination of their efforts and those of the Java community makes Java an excellent platform for an increasing number of clients.

Johan Vos

I would like to thank my wife, Tracey, and my daughters, Caitlin and Gillian, for their loving support and sacrifices. A big thanks to Jim Weaver for recommending me to this project. I also want to thank the amazing authors Jim Weaver, Weiqi Gao, Stephen Chin, Dean Iverson, and Johan Vos. Thanks to the wonderful people at Apress for their professionalism especially Stephen Moles for keeping the team laser focused. Lastly, I want to give a big kudos and acknowledgment to the people at Oracle involved with moving Java and JavaFX forward by growing the community. *"Iron sharpeneth iron; so a man sharpeneth the countenance of his friend." (Proverbs 27:17)*

Carl Dea

CHAPTER 1

# Getting a Jump Start in JavaFX

*Don't ask what the world needs. Ask what makes you come alive, and go do it. Because what the world needs is people who have come alive.*

—Howard Thurman

At the annual JavaOne conference in May 2007, Sun Microsystems announced a new product family named JavaFX. Its stated purpose includes enabling the development and deployment of content-rich applications on consumer devices such as cell phones, televisions, in-dash car systems, and browsers. Josh Marinacci, a software engineer at Sun, made the following statement very appropriately in a recent Java Posse interview: "JavaFX is sort of a code word for reinventing client Java and fixing the sins of the past." Josh was referring to the fact that Java Swing and Java 2D have lots of capability, but are also very complex. JavaFX allows us to simply and elegantly express user interfaces (UIs) with a declarative programming style. It also leverages the full power of Java, because you can instantiate and use the millions of Java classes that exist today. Add features such as binding the UI to properties in a model and change listeners that reduce the need for setter methods, and you have a combination that will help restore Java to the client side of the RIA equation.

In this chapter, we give you a jump start in developing JavaFX applications. After bringing you up to date on the brief history of JavaFX, we show you how to get the JavaFX software development kit (SDK). We also explore some great JavaFX resources and walk you through the process of compiling and running JavaFX applications. In the process you'll learn a lot about the JavaFX API as we walk through application code together. First, however, we point out a related technology that is enabling the rise of rich-client Java.

## JavaFX Can't Bring Rich-Client Java Back by Itself

When Java was first introduced in 1995, the hope was that the Java Runtime Environment (JRE) would become the common client platform on which the UI portion of client–server applications could be deployed. Although the JRE became ubiquitous on the server side of the equation, factors such as the browser wars of the late 1990s delayed the prospect of achieving a consistent JRE on client machines. The result has been that web browser technologies such as HTML and JavaScript have stepped in to fill the gap, which we feel has proven suboptimal at best. The software development industry and the users we serve need to have the JRE on all client machines so that we can break free from browser technologies and enable graphically rich, fast-performing applications. Fortunately, the technology known as Java SE 6 Update 10 is solving that problem.

■ **Note** What has come to be known as Java SE 6 Update 10 has actually had several names. It started life as the Consumer JRE, and then Java SE 6 Update N. Then it became known as Java SE 6 Update 10. As of this writing, Java SE 7 has been released, but we just refer to this technology as Java SE 6 Update 10.

Java SE 6 Update 10 consists of several technologies that improve the user experience related to installing the JRE, and to deploying and running rich-client Java (and JavaFX) programs:

- *Java Kernel Online Installer*—The JRE is now divided into small bundles. If the user's machine doesn't have the JRE installed when a Java program is invoked, the online installer will ascertain which of the bundles are needed to run the program. Those bundles will be installed first and the program will begin executing as soon as this takes place.
- *Java Auto-Updater:* This provides a faster and more reliable process for updating the JRE by using a *patch-in-place* mechanism.
- *Java Quick Starter:* After a cold boot of the system, portions of the JRE are prefetched into memory. This enables a Java program to start more quickly.
- *Pack200 Format:* Pack200 is a highly compressed format that enables Java libraries and resources, for example, to download more quickly than traditional JAR files.
- *Java Deployment Toolkit:* This includes a simple JavaScript interface with which to deploy Java applets and applications. The JavaScript library is located at a well-known URL, and is engineered to make the right deployment decisions based on the detected JRE environment on the user's machine.
- *Next Generation Java Plug-In:* This Java plug-in is much more reliable and versatile than its predecessor. For example, you now have the ability to specify large heap sizes, and per-applet command-line arguments. Also, it has built-in Java Network Launching Protocol (JNLP) support as well as improved Java/JavaScript communications.
- *Hardware Acceleration Support:* In a media-rich environment, it is crucial to take advantage of the graphics capabilities on the underlying hardware. For example, Java SE 6 Update 10 currently has a hardware accelerated graphics pipeline based on the Microsoft Direct3D API. This is a predecessor to the new Prism pipeline that JavaFX uses.

The net result is that we are now at a point in software development history when two technologies (JavaFX and Java SE 6 Update 10) are working together to restore rich client Java. We feel that sanity is in the process of being restored to Internet software development, and we want you to join us in this RIA revolution. But first, a brief history lesson about JavaFX.

# A Brief History of JavaFX

JavaFX started life as the brainchild of Chris Oliver when he worked for a company named SeeBeyond. They had the need for richer user interfaces, so Chris created a language that he dubbed F3 (Form

Follows Function) for that purpose. In the article, "Mind-Bendingly Cool Innovation" (cited in the Resources section at the end of this chapter) Chris is quoted as follows. "When it comes to integrating people into business processes, you need graphical user interfaces for them to interact with, so there was a use case for graphics in the enterprise application space, and there was an interest at SeeBeyond in having richer user interfaces."

SeeBeyond was acquired by Sun, who subsequently changed the name of F3 to JavaFX, and announced it at JavaOne 2007. Chris joined Sun during the acquisition and continued to lead the development of JavaFX.

The first version of JavaFX Script was an interpreted language, and was considered a prototype of the compiled JavaFX Script language that was to come later. Interpreted JavaFX Script was very robust, and there were two JavaFX books published in the latter part of 2007 based on that version. One was written in Japanese, and the other was written in English and published by Apress (*JavaFX Script: Dynamic Java Scripting for Rich Internet/Client-Side Applications*, Apress, 2007).

While developers were experimenting with JavaFX and providing feedback for improvement, the JavaFX Script compiler team at Sun was busy creating a compiled version of the language. This included a new set of runtime API libraries. The JavaFX Script compiler project reached a tipping point in early December 2007, which was commemorated in a blog post entitled "Congratulations to the JavaFX Script Compiler Team—The Elephant Is Through the Door." That phrase came from the JavaFX Script compiler project leader Tom Ball in a blog post, which contained the following excerpt.

> *An elephant analogy came to me when I was recently grilled about exactly when the JavaFX Script compiler team will deliver our first milestone release. "I can't give you an accurate date," I said. "It's like pushing an elephant through a door; until a critical mass makes it past the threshold you just don't know when you'll be finished. Once you pass that threshold, though, the rest happens quickly and in a manner that can be more accurately predicted."*

A screenshot of the silly, compiled JavaFX application written by one of the authors, Jim Weaver, for that post is shown in Figure 1-1, demonstrating that the project had in fact reached the critical mass to which Tom Ball referred.

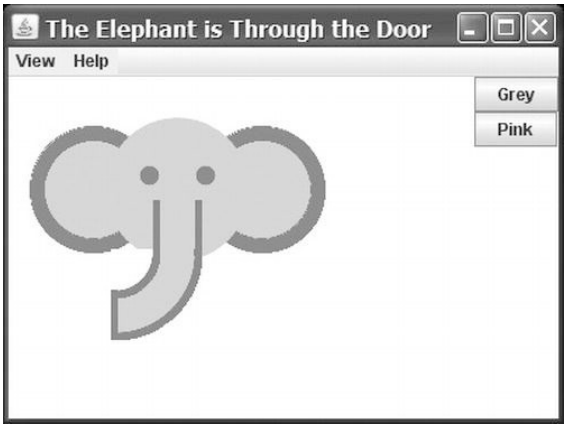


***Figure 1-1.*** *Screenshot for the "Elephant Is Through the Door" program*

Much progress continued to be made on JavaFX in 2008:

- The NetBeans JavaFX plug-in became available for the compiled version in March 2008.
- Many of the JavaFX runtime libraries (mostly focusing on the UI aspects of JavaFX) were rewritten by a team that included some very talented developers from the Java Swing team.
- In July 2008, the JavaFX Preview SDK was released, and at JavaOne 2008 Sun announced that the JavaFX 1.0 SDK would be released in fall 2008.
- On December 4, 2008, the JavaFX 1.0 SDK was released. This event increased the adoption rate of JavaFX by developers and IT managers because it represented a stable codebase.
- In April 2009, Oracle and Sun announced that Oracle would be acquiring Sun. The JavaFX 1.2 SDK was released at JavaOne 2009.
- In January 2010, Oracle completed its acquisition of Sun. JavaFX 1.3 SDK was released in April 2010, with JavaFX 1.3.1 being the last of the 1.3 releases.

At JavaOne 2010, JavaFX 2.0 was announced. The JavaFX 2.0 roadmap was published by Oracle on the Web page noted in the Resources section below, and includes items such as the following.

- Deprecate the JavaFX Script language in favor of using Java and the JavaFX 2.0 API. This brings JavaFX into the mainstream by making it available to any language (such as Java, Groovy, and JRuby) that runs on the JVM.
- Make the compelling features of JavaFX Script, including binding to expressions, available in the JavaFX 2.0 API.
- Offer an increasingly rich set of UI components, building on the components already available in JavaFX 1.3.
- Provide a Web component for embedding HTML and JavaScript content into JavaFX applications.
- Enable JavaFX interoperability with Swing.
- Rewrite the media stack from the ground up.

JavaFX 2.0 was released at JavaOne 2011, and has enjoyed a greatly increased adoption rate due to the innovative features articulated previously. Now that you've had the obligatory history lesson in JavaFX, let's get one step closer to writing code by showing you where some examples, tools, and other resources are.

## Going to the Source: Oracle's JavaFX Web Site

Oracle's JavaFX.com site is a great resource for seeing example JavaFX programs, downloading the JavaFX SDK and tools, taking tutorials on JavaFX, and linking to other resources. See Figure 1-2 for a screenshot of this web site.

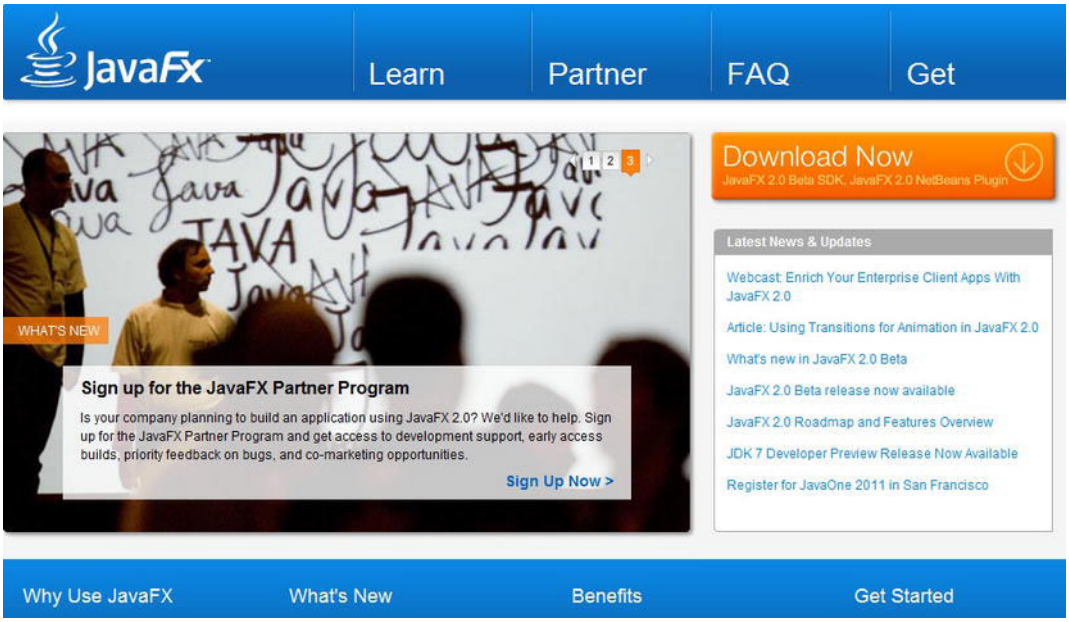


***Figure 1-2.*** *Oracle's official JavaFX web site*

In addition, blogs maintained by JavaFX engineers and developers are great resources for up-to-the-minute technical information on JavaFX. For example, Oracle JavaFX Engineers Richard Bair, Jasper Potts, and Jonathan Giles keep the developer community apprised of the latest JavaFX innovations at `http://fxexperience.com`. In addition, the Resources section at the end of this chapter contains the URLs of the blogs that the authors of this book use to engage the JavaFX developer community.

Take a few minutes to explore these sites. Next we point out some more valuable resources that are helpful.

# Accessing the JavaFX SDK API

A useful resource available from the JavaFX sites is the SDK API JavaDoc documentation, shown in Figure 1-3.

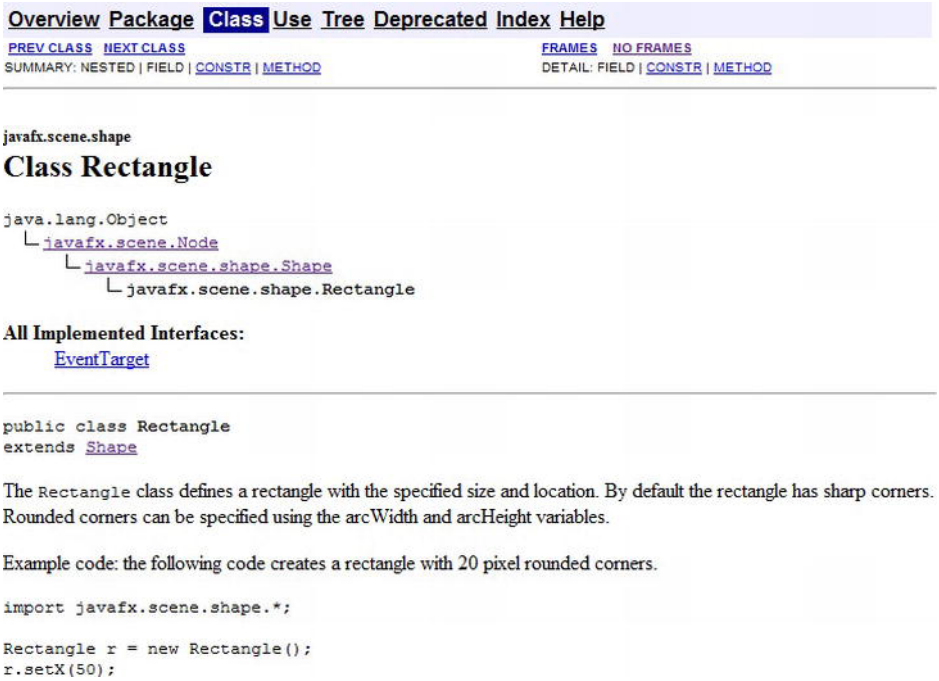


***Figure 1-3.*** *JavaFX SDK API Javadoc*

The API documentation in Figure 1-3, for example, shows how to use the Rectangle class, located in the javafx.scene.shape package. Scrolling down this web page shows the properties, constructors, methods, and other helpful information about the Rectangle class. By the way, this API documentation is available in the JavaFX SDK that you'll download shortly, but we wanted you to know how to find it online as well.

Now that you've explored Oracle's JavaFX web site and available resources, it's time to obtain the JavaFX SDK and related tools so that you can begin developing JavaFX programs.

# Obtaining the JavaFX SDK

You can get the JavaFX SDK from Oracles's JavaFX web site mentioned earlier. Currently you have the choice of downloading the JavaFX SDK, the JavaFX Runtime, and the JavaFX Plugin for NetBeans IDE. To develop JavaFX applications you'll need the JavaFX SDK. In addition, we recommend that you download the JavaFX Plugin for NetBeans IDE as it contains modules that will help you develop and package JavaFX 2.0 applications. The instructions for the examples in this first chapter assume that you have the NetBeans Plugin installed.

Go ahead and download the JavaFX SDK, and the JavaFX Plugin for NetBeans, following the installation instructions. In addition, so that you can compile and run the JavaFX application from the command-line, the JAR file that contains the JavaFX runtime must be on the classpath. The name of this file is jfxrt.jar, and it is located in the rt/lib directory subordinate to the directory in which the JavaFX SDK is installed.

# Other Available Tools

There are other tools available for developing JavaFX applications. For example, there is a JavaFX plug-in being developed by Tom Schindl for the Eclipse IDE, but at the time of this writing it isn't as mature as the NetBeans plugin. The URL for the Eclipse plugin is listed in the Resources section.

Now that you have the tools installed, we show you how to create a simple JavaFX program, and then we walk through it in detail. The first program that we've chosen for you is called "Hello Earthrise," which demonstrates more features than the typical beginning "Hello World" program.

# Developing Your First JavaFX Program: "Hello Earthrise"

On Christmas Eve in 1968 the crew of Apollo 8 entered lunar orbit for the first time in history. They were the first humans to witness an "Earthrise," taking the magnificent picture shown in Figure 1-4. This image is dynamically loaded from this book's web site when the program starts, so you'll need to be connected to the Internet to view it.


***Figure 1-4.*** *The Hello Earthrise program*

In addition to demonstrating how to dynamically load images over the Internet, this example shows you how to use animation in JavaFX. Now it's time for you to compile and run the program. We show you two ways to do this: from the command-line, and using NetBeans with the JavaFX plug-in.

# Compiling and Running from the Command-Line

We usually use an IDE to build and run JavaFX programs, but to take all of the mystery out of the process we use the command-line tools first.

■ **Note** For this exercise, as with most others in the book, you need the source code. If you prefer not to type the source code into a text editor, you can obtain the source code for all of the examples in this book from the code download site. See the Resources section at the end of this chapter for the location of this site.

Assuming that you've downloaded and extracted the source code for this book into a directory, follow the directions in this exercise, performing all of the steps as instructed. We dissect the source code after the exercise.

### COMPILING AND RUNNING THE HELLO EARTHRISE PROGRAM FROM THE COMMAND

You'll use the javafxc and javafx command-line tools to compile and run the program in this exercise. From the command-line prompt on your machine:

1. Navigate to the Chapter01/Hello directory.
2. Execute the following command to compile the HelloEarthRiseMain.java file.

   ```
   javac -d . HelloEarthRiseMain.java
   ```

3. Because the `-d` option was used in this command, the class files generated are placed in directories matching the `package` statements in the source files. The roots of those directories are specified by the argument given for the –d option, in this case the current directory.
4. To run the program, execute the following command. Note that we use the fully qualified name of the class that will be executed, which entails specifying the nodes of the path name and the name of the class, all separated by periods.

   ```
   java projavafx.helloearthrise.ui.HelloEarthRiseMain
   ```

The program should appear as shown in Figure 1-4 earlier, with the text scrolling slowly upward, reminiscent of the Star Wars opening crawls.

Congratulations on completing your first exercise as you explore JavaFX!

## Understanding the Hello Earthrise Program

Now that you've run the application, let's walk through the program listing together. The code for the Hello Earthrise application is shown in Listing 1-1.

***Listing 1-1.*** *The HelloEarthRiseMain.java Program*

```
/*
*  HelloEarthRiseMain.java - A JavaFX "Hello World" style example
 *
 *  Developed 2011 by James L. Weaver jim.weaver [at] javafxpert.com
 *  as a JavaFX SDK 2.0 example for the Pro JavaFX book.
 */
package projavafx.helloearthrise.ui;

import javafx.animation.Interpolator;
import javafx.animation.Timeline;
import javafx.animation.TranslateTransition;
import javafx.application.Application;
import javafx.builders.GroupBuilder;
import javafx.builders.ImageViewBuilder;
import javafx.builders.RectangleBuilder;
import javafx.builders.SceneBuilder;
import javafx.builders.TextBuilder;
import javafx.builders.TranslateTransitionBuilder;
import javafx.geometry.VPos;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.scene.text.TextAlignment;
import javafx.stage.Stage;
import javafx.util.Duration;

/**
 *  Main class for the "Hello World" style example
 */
public class HelloEarthRiseMain extends Application {

  /**
   * @param args the command-line arguments
   */
  public static void main(String[] args) {
    Application.launch(args);
  }

  @Override
  public void start(Stage stage) {

    String message =
      "Earthrise at Christmas: " +
      "[Forty] years ago this Christmas, a turbulent world " +
      "looked to the heavens for a unique view of our home "  +
      "planet. This photo of Earthrise over the lunar horizon " +
      "was taken by the Apollo 8 crew in December 1968, showing " +
```