火龙果•整理
uml.org.cn

JavaFX2.0 基础教程

目录

1 JavaFX 概览 3
1.1 JavaFX 认知 3
1.2 JavaFX 简史 4
1.3 JavaFX2.0 新特性 4
1.4 用 JavaFX 能构建什么 6
1.5 附加资源 7
2 JavaFX 安装 8
3 JavaFX 架构和框架 9
3.1 场景图 9
3.2 JavaFX 特征 API 10
3.3 图形系统 10
3.4 视窗工具 Glass 11
3.5 线程 11
3.6 Pulse（脉冲事件） 11
3.7 媒体和图片 11
3.8 嵌入浏览器 12
3.9 CSS（层叠样式） 12
3.10 UI 控件 13
3.11 布局设计（Layout） 14
3.12 2D 和 3D 转换 15
3.13 可视化效果 15
3.14 部署 15
4 JavaFX 开发入门 16
4.1 建立应用 17
4.2 创建应用基础 17
4.3 增加布景 18
4.4 添加图形 18
4.5 增加可视效果 19
4.6 创建渐变背景 20
4.7 应用混合模式 21
4.8 添加动画 22
4.9 部署应用 23
5 FXML 入门教程 24
5.1 为何使用 FXML 24

火龙果•整理
uml.org.cn

5.1.1 FXML 介绍 24
5.1.2 FXML 简单示例 24
5.1.3 FXML 的好处 25
5.2 创建用户界面 25
5.2.1 准备工作 26
5.2.2 创建工程 27
5.2.3 创建应用基础 27
5.2.4 创建属性文件 28
5.2.5 创建 FXML 文件 28
5.2.6 定义边格布局 29
5.2.7 图片上堆叠文本 29
5.2.8 添加 Grid 布局和控件 30
5.2.9 添加按钮事件 31
5.2.10 使用脚本语言 32
5.2.11 应用式样表 33
5.2.12 教程回顾 34
5.3 接下来… 35
6 JavaFX 开发概要 35
第二篇 JavaFX 内建控件 36
1. 用户见面控件（UI 控件） 37
2. JavaFX2.0 中支持的 UI 控件 37

火龙果•整理
uml.org.cn

前言

关于 JavaFX 的相关基础教程内容，都是翻译自 Oracle 官方网站的相应内容。没有做过多的词句考量，但作为一般技术文档，读者应该都可以理解的。若有不便，还请谅解。

这部分是第一篇，关于 JavaFX2.0 基础教程部分，还有另一部（第二篇）关于内置控件的教程，也会尽快发布到网上。

1 JavaFX 概览

1.1 JavaFX 认知

JavaFX 平台是 java 客户端设计演进，使应用开发者易于创建和部署跨平台且表现一致的 Rich Internet Application（RIAs）。JavafX 是由 Java 技术构建，基于高性能硬件加速的媒体和图形引擎，JavaFX 平台提供了一套丰富的图形和媒体 API，简化了数据驱动的企业客户端部署。

作为 Java 生态体系的一部分，投资于 javafx 平台将使 java 开发者和公司得到如下好处：

1. JavaFX 平台是由 Java 构建的，java 开发者可以继续使用原来的技巧、工具开发 JavaFX 应用；
2. 由于 Java 的广泛使用，更容易找到有经验的开发者，并能很快成为一个高生产性的 JavaFX 应用开发者；
3. 基于 java 同类服务器和客户端平台一套技术，减少了 javafx 平台商务方案的复杂性并降低了投资风险；
4. 基于前述的优点，开发成本也减少了；

火龙果•整理
uml.org.cn

5. JavaFX 平台给开发者提供了一个开发框架和一个运行环境，以便创建支持 java 的跨平台企业和商务应用。

注：查看后文“JavaFX 架构和框架”部分，可学习更多关于 JavaFX 平台的架构和关键概念。

1.2 JavaFX 简史

在 2007 年 JavaOne 大会上，Sun 公司介绍了 Javafx 平台，以便帮助内容开发者和应用开发者去创建基于移动设备、桌面、电视以及其他消费设备的内容丰富的应用。初始版 javafx 由 Mobile 平台和 Javafx 脚本语言构成，其它公开发布的都是基于初始版本的衍生。Javafx1.3 版于 2010 年 4 月 22 发布。

在 Oracle 收购 Sun 后，Oracle 在 2010 年 JavaOne 大会宣布对 Javafx 脚本语言的支持将停止。但同时也宣布作为 JavaFX2.0 产品一部分的 Javafx 脚本 APIs 将被导入到 Java。这也意味着 JavaFX 的功能对于所有 java 开发者将是可用的——不需要他们学习新的脚本语言。同时，Oracle 宣布 Javafx 将是富客户端应用首选的开发环境。

1.3 JavaFX2.0 新特性

Javafx2.0 的发行版主要焦点领域在于包括如下特征中（很多内容也在 JavaFX 架构与框架中也有相应描述）：

- **JavaFX 的 Java APIs：**提供所有 java 开发者常用的、熟悉的语言特性（诸如泛型、标注和多线程等）。这些 APIs 设计友好

火龙果•整理
uml.org.cn

且可选择不同 JVM 语言调用，诸如 JRuby 和 Scala。由于 JavaFX 功能通过 Java APIs 可用，你可以继续使用你钟爱的 java 开发工具（例如 IDEs、代码重构、调试和解析工具等）去开发 JavaFX 应用。

- **新的图形引擎（GPUs）：**新引擎的基础是硬件加速的称之为 Prism 图形管道，并耦合了新的称之为 Glass 视窗工具包。图形引擎为当前和将来创建丰富图形提供了简化、润滑和快速的基础支持。
- **新的声明式标记语言 FXML：**是基于 xml 并用于 JavaFX 应用中定义用户接口。FXML 不需要编译。这样就意味着每次改变布局代码时不需要重新编译代码。
- **新媒体引擎：**支持 web 多媒体内容的重放。JavaFX 该框架提供稳定的、低延迟的媒体框架——是基于 GStreamer 多媒体框架的。
- **Web 组件：**在 JavaFX 应用中使用 WebKit HTML 渲染技术，实现组件的页面嵌入能力。通过 Prism 实现硬件加速渲染有效可用。
- **更新浏览器插件：**javafx 浏览器插件允许基于 Prism 加载 JavaFX applets。
- **丰富的内建 UI 控件：**包括图、表格、菜单、布局面板。另外，通过提供的 API 允许第三方发布 UI 控件供用户社区使用。
- **应用示例：**展示了 JavaFX20 技术的不同特性，并附有大量的

火龙果•整理
uml.org.cn

示例代码和片段。

- **更新的 Doclet:** 使用 javadoc 工具生成 HTML 格式的 JavaFX API 文档（怎么使用这些更新的 doclet，详细信息可以查阅：随 JavaFX 使用 Doclet）。

1.4用 JavaFX 能构建什么

图表-1 展示了 JavaFX 应用的一些示例略图。这些示例包含在 JavaFX2.0 的发布版中。为了运行这些示例和附加应用，可以到这里进一步查看：http://www.oracle.com/technetwork/java/javafx/downloads/index.html.。下载的示例 zip 文件包含 JavaFX 例子以及扩展文件。解压后可双击相应的可执行文件（*.jar），如果 JavaFX2.0 环境正常，将可查看到相应的运行效果。

图表-1 JavaFX 应用示例

应用示例	描述
	JavaFX Ensemble（总效果示例） 总效果演示提供了一个不同 JavaFX 特性应用的展廊，例如动画、图表、控件等。

火龙果•整理
uml.org.cn

应用示例	描述
	Sales Dashboard（销售仪表盘-DataApp） 一个为虚构的全球汽车公司（Henley Automobiles）构建的客户/服务器应用。汽车销售使用 JavaDB 并基于 EJB 模拟。这些数据通过 Derby 和 Restfulweb 服务时可用的。通过使用 FXML 和 JavaFX，实现客户端演示了不同的数据表现
	SwingInterop Sample 这个 Swing 应用展示了 Swing 和 JavaFX 可被如何联合。使用 JavaFX 组件在一个 tab 中实现了图表 在另一个 tab 中实现了一个简单的浏览。

每个示例的源代码在 `javafx-samples-2.0.x\src` 目录下。为了查看源代码，可在 `javafx-samples-2.0.x\src\<sample>` 目录进行查看。每个代码示例目录都是各异 NetBeans 工程。

怎么创建 JavaFX 应用
因为 JavaFX 应用是用 Java 语言编写的，你可以使用你喜欢的编辑器或任何支持 java 语言的 IDE（NetBeans, Eclipse, Oracle JDeveloper, or IntelliJ IDEA ete.）来开发 JavafX 应用。你也可以跟随下面的步骤介绍来开始 JavafX 应用的创建：

1. 到 http://www.oracle.com/technetwork/java/javafx/downloads/index.html/ 下载 JavaFX SDK.到 Release Documentation 页面查看系统需求和安装介绍(包括计划使用的版本)
2. 学习 Getting Started with JavaFX 教程创建简单动画应用。

1.5 附加资源

使用下面的资源学习更多关于 JavaFX 技术。

- Download JavaFX 2.0 SDK
- JavaFX Architecture and Framework
- JavaFX API Documentation
- Additional JavaFX tutorials and articles
- JavaFX 2.0 Features
- JavaFX 2.0 FAQ

火龙果•整理
uml.org.cn

- JavaFX 2.0 Roadmap

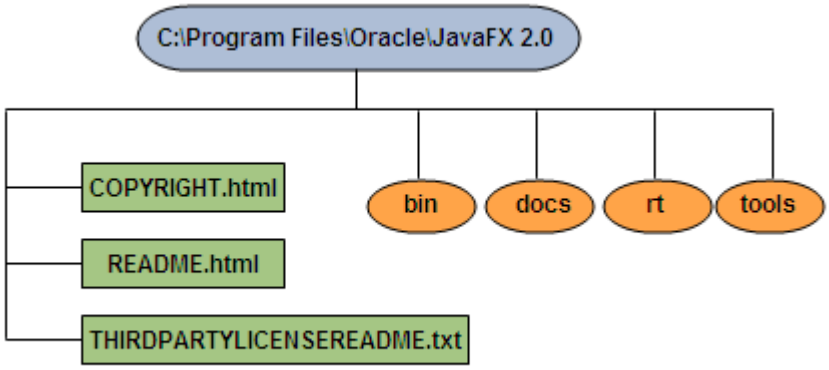
2 JavaFX 安装

众所周知，JavaFX 应用需要相应的环境支持。2.0 版前 JavaFX 不在此讨论范围，感兴趣者可以自行到 Oracle 官网查看。本部分简单介绍安装，详细情况可以查看如下网页内容描述：http://docs.oracle.com/javafx/2.0/installation_2-0-3/jfxpub-installation_2-0-3.htm 。

为了编写和运行 JavaFX 程序，可以通过两种方式来实现：其一，下载 **JDK 7 with JavaFX SDK**。由 Java SE 7 Update 2 开始，这版 JDK 包含 JavaFX SDK。其二，单独下载单机版 JavaFX SDK，推荐 JDK 版本为 JDK 6u26 到 JDK7u1。具体下载哪一版本，则根据需要选择。

如果只是运行 JavaFX 桌面应用和 Applet，可只下载 JavaFX 运行时环境安装即可。

默认 JavaFX SDK 安装目录结构如下所示（可自行选择安装目录）：



bin/：SDK 构建工具。
docs/：API 文档。
rt/：JavaFX 运行时目录。
tools/：基于 Ant 任务的打包和部署工具。

建议使用 NetBeans IDE 7.1 进行 JavaFX 开发。下载 NetBeans 与 JavaFX 绑定版安装后，即可以直接进行开发了，包括进行调试、源码管理等。也可通过插件来使用 Eclipse 进行开发（不做具体介绍）。

为了可以更好的理解和学习 JavaFX 应用开发，建议下载 JavaFX 示例。下载地址：http://www.oracle.com/technetwork/java/javafx/downloads/index.html。为了运行这些示例，必须安装 JavaFX 运行时支持，相应的桌面应用，可以直接双击.jar 文件执行。另外，解压文件后，每个示例源码都是 NetBeans 工程文件，且只能在 NetBeans IDE 7.1 下方可运行调试。

火龙果•整理
uml.org.cn

（下一篇介绍 JavaFX2.0 架构和框架以及关键点）

3 JavaFX 架构和框架

JavaFX2.0 是基于 Java 技术构建的富客户端平台，使应用开发者易于创建和部署跨平台的富互联网应用（Rich Internet Application RIAs）。

图-1 描述了 JavaFX2.0 的架构性组件构成。这部分将描述图表构成的每个组件以及这些组成部分如何交互。JavaFX 公开 APIs 依存于真正运行 JavaFX 代码的应用引擎。它的构成子组件包括新的高性能图形引擎（代号 Prism）、新的体积小且高效的视窗体系（代号 Glass）、媒体引擎、web 引擎。尽管这些组件没有公开暴露，但关于这些的描述能有助于你更好地理解 JavaFX 应用是如何运作的。

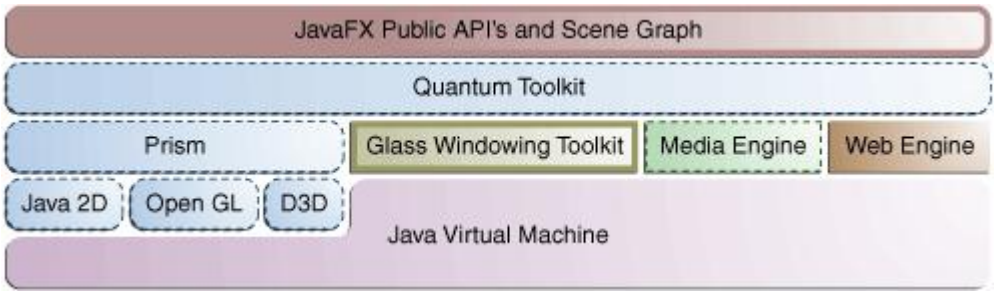


图-1 JavaFX2.0 架构图

3.1场景图

JavaFX 场景图（图-1 顶层部分）是构建 JavaFX 应用的起始点。它是应用图形接口所有可视元素节点的层级节点树，并能处理输入及渲染。

场景图中的单个元素称为节点。每个节点有一个标识 ID、样式类及绑定量。每个节点有单一的父类以及零到多个子节点，节点也可有如下功能：

- 效果化，诸如模糊、阴影化处理；
- 不透明；
- 变换；
- 事件处理，诸如鼠标、键盘以及输入方法等；
- 特定应用状态处理。

不像在 Swing 和 AWT（Abstract Window Toolkit）中，JavaFX 场景图也有基本图形，例如矩形、文本框，还有控件、布局容器、图片以及媒介处理等。对于大多数使用，场景图简化了 UIs 的工作，特别是在使用富 UIs 时。各种各样的动画能更快的完成，并且声明式方法（如基于 XML）也能很好的显效运行。

这个 javafx.scene 的 API 允许创建和规范好几种类型的内容，例如：

节点：图形（2D 和 3D）、图片、媒体、嵌入式页面浏览器、文本框、UI 控件、图表、

火龙果•整理
uml.org.cn

分组以及容器；

状态：变换（节点位置和方向）、可视化效果以及其它可视化状态内容；

动画：随时间变化的场景图对象属性特征；

效果：改变场景图节点面貌的简单效果对象，诸如模糊、阴影、颜色调整等。

关于场景图这部更多信息，可以查看“***基于JavaFX场景图工作***”部分文档描述。

3.2 JavaFX 特征 API

JavaFX2.0 平台包括一套完整的公共 APIs，如图-1 最顶层所示。这些 APIs 为构建富客户端（RIAs）提供了空前的自由和灵活性。JavaFX 结合了 Java 平台最好的功能，以全面的、引人入胜的媒体功能形成直观、一站式开发环境。这些针对 JavaFX 特征的 Java APIs 有：

- 允许使用强大的 java 特性，如泛型、标注和多线程。
- 对于 Web 开发者来讲，使用 JavaFX 进行开发，相比其它流行动态语言，如 JRuby、Groovy 和 JavaScript，更为容易些。
- 允许 Java 开发者使用其它系统语言，如 Groovy，来编写大型的或复杂的 JavaFX 应用。
- 允许使用类似于 JavaFX 脚本语言绑定。这种绑定包括高性能“懒”绑定、绑定表达式、范围序列表达式、局部绑定再赋值等。可选语言（如 Groovy）也能引入使用类似 JavaFX 脚本的绑定语法。
- 扩展 Java 集合库,包括可观察 Lists 和 Maps。这样允许应用连接用户界面和数据模型，观察数据模型的变化，并一致性的更新相应 UI 控件。

JavaFX2.0 API 和编程模型是 JavaFX1.x 产品线的延续，大多数的 JavaFX APIs 已经导入到 Java 中。有些 APIs，如布局和媒体，以及其它许多相关细节，基于发行版 JavaFX1.x 用户反馈，已经进行了改进并简化。JavaFX2.0 依赖于更多的 web 标准，如样式控制 CSS、访问性规范 ARIA。其它附加 web 标准也在审查中。

3.3 图形系统

JavaFX 图形系统（如图-1 蓝色部分）是一个在 JavaFX 场景图层下的细节实现，支持 2D 和 3D 场景图。该图形系统在系统图形硬件加速渲染不足时，可实现软件渲染。JavaFX2.0 平台实现的两个图形加速管道为：

1、Prism 处理渲染工作。它能运行在硬件和软件渲染器上，包括 3-D。负责光栅化和渲染 JavaFX 场景。下列多个渲染途径可基于设备使用：

- Windows XP 和 Windows Vista 上的 DirectX 9；
- Windows 7 上的 DirectX11 ；
- Mac、linux 及嵌入设备上的 OpenGL；
- Java2D（不能硬件加速时）。可能时完全硬件加速可用，但当不可用时，Java2D 渲染途径可用（因为 Java2D 渲染器发行于所有 JREs 中）。当处理 3-D 场景时，这是特别重要的。当然使用硬件渲染器时性能更好些。

2、Quantum Toolkit 把 Prism 和 Glass Windowing Toolkit 连接在一起，并使它们在堆栈上对 JavaFX 层可用。它也管理着与事件处理相对与渲染有关的线程规则。

火龙果•整理
uml.org.cn

3.4视窗工具 Glass

如图-1 中部位置所示，Glass 视窗包是 JavaFX2.0 图形堆栈最底层框架。它的主要职责是提供本地化操作服务，诸如管理窗口、定时器以及外观等。作为平台相关层，它负责连接 JavaFX 平台和本地操作系统。

Glass 工具包也负责管理事件队列。它不像 AWT 自我管理事件队列，Glass 使用本地操作系统的事件队列功能来安排线程使用。也不像 AWT，Glass 作为 JavaFX 应用运行在相同的线程上。在 AWT 中，本地一半的 AWT 事件运行在一个线程上，而 Java 层的运行在另一个线程上。这许多问题，在 JavaFX 中通过使用单应用线程方法得到了解决。

3.5线程

在任何给的时间内，系统运行两个或更多个如下线程。

- **JavaF 应用线程**：这是 JavaFX 应用开发者使用的主要线程。在任何“实时”场景中，作为视窗场景一部分，必须通过此线程进入访问。但是，场景也能从后台线程中创建，这使得开发者能够基于后台线程创建复杂场景，以便能在实时性动画场景中保持画质的滑顺和快速。JavaFX 应用线程是与 Swing 和 AWT 的时事件发线程（EDT）不同的，因此必须小心在 Swing 应用中嵌入 JavaFX 代码。
- **Prism 渲染线程**：此线程基于事件分发来分别处理渲染工作。这使得在 N+1 帧被处理时，N 帧能被渲染。这种能力对执行并发处理是非常有利的，特别是现代多处理器系统。Prism 渲染器线程也可有多个栅格化线程，这有助于需要在渲染时完成的过负荷的工作。
- **媒体线程**：这个线程运行在后台，使用 JavaFX 应用线程通过场景图同步最新帧。

3.6Pulse（脉冲事件）

一次 Pulse 是一个事件，它指示 JavaFX 布景用 Prism 适时同步场景中元素状态。一次 Pulse 最大控制在每秒（fps）60 帧内，并可由场景图中运行的动画任意时刻触发。即使动画没有运行，当场景图中有什么变化时，一次 Pulse 也被预定。当一次 Pulse 触发，场景图中元素状态被向下同步到渲染层。一次 Pulse 使开发者有方法处理异步事件。这是个重要的特性，使系统能在 Pulse 上批量处理和执行事件。

布局和 CSS 也被联系到 Pulse 事件。场景图中许多改变能引起多布局或 CSS 更新，这可能会严重降低性能。系统每次 Pulse 自动执行 CSS 和布局传递可以避免性能降级。在需要测量前一个 Pulse 时，应用开发者也能自动触发布局或 CSS 传递。

Glass 视窗包负责执行 Pulse 事件。它使用高解析度本地时间来做执行。

3.7媒体和图片

火龙果•整理
uml.org.cn

JavaFX 媒体处理功能通过包 javafx.scene.media 包的 APIs 调用。JavaFX 支持可视化和音频媒体，支持的媒体格式有 MP3、AIFF、WAV 音频文件以及 FLV 视频文件。JavaFX 媒体功能通过三个独立组件实现：媒体对象代表一个媒体文件，媒体播放器播放媒体文件，媒体视图是一个展示媒介的节点。

为提高媒体引擎组件（图-1 绿色部分）稳定性能，在 2.0 中已经完成重新设计了：既提高性能，也提供了跨平台一致性行为表现。

3.8嵌入浏览器

嵌入浏览器是 JavaFX 新的用户界面组件，基于 APIs 它提供一个 web 查看器和一个完整的浏览功能。如图-1 中橘色部分，web 引擎组件基于 Webkit，它是个一个开源的、支持 HTML5、CSS、Javascript、DOM、以及 SVG 等的 web 浏览器引擎。这使开发者能够在 java 应用中实现如下特性：

- 从本地或远程 URL 渲染 HTML 内容（以 html 方式浏览内容）；
- 支持浏览历史，实现回退和前进导航；
- 重载内容；
- 对 web 组件应用效果化处理；
- 编辑 HTML 内容；
- 执行 Javascript 命令；
- 处理事件。

嵌入浏览器组件有下面相关类构成：

- WebEngine：提供基本页面浏览能力；
- WebView：封装了一个 WebEngine 对象，把 HTML 内容合并到应用场景中，并提供字段和方法应用效果化和转换。这是个 Node 类的扩展。

更多内容可以查看 JavaFX 嵌入浏览器相关介绍。

3.9CSS（层叠样式）

在不需要改变任何源码情况下，CSS 为 JavaFX 应用提供了定制式样风格能力。CSS 可应用到场景图中任何节点及异步节点。CSS 样式很容易实时应用到场景中，允许应用程序动态改变面貌。图-2 演示了同样 UI 控件在两类 CSS 中的不同样式：

火龙果•整理
uml.org.cn

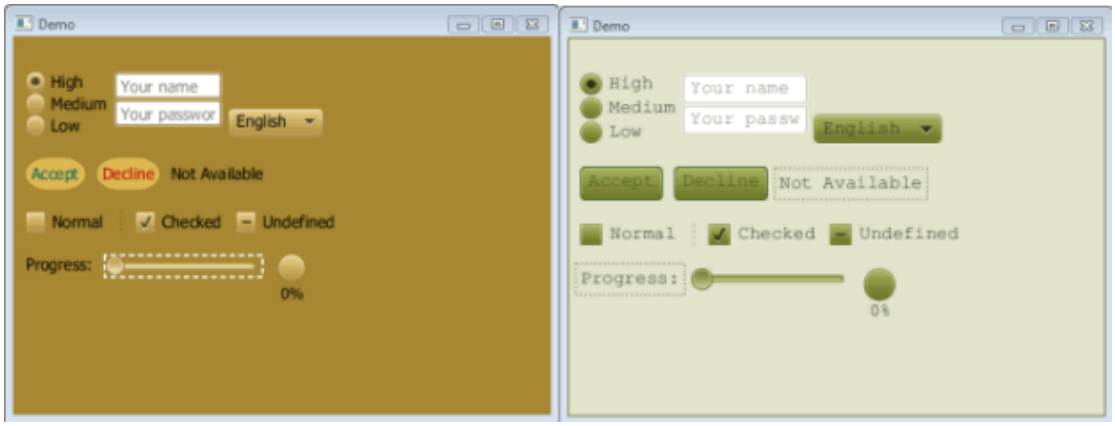


图-2 CSS 样式表示例

JavaFX CSS 是基于 W3C 的 CSS2.1 版规范，并带有一些 3.0 版内容。JavaFX 的 CSS 支持和扩展被设计为可被任何兼容的 CSS 解析器清晰解析，甚至不支持 JavaFX 扩展的解析器。这可以使 JavaFX 得 CSS 和其它目的 CSS（如 HTML 页面）混合为单一式样单。所有 javafx 属性名带有厂商扩展“-fx-”前缀，包括那些可能是兼容标准的 HTML 的 CSS。因为有些 Javafx 值有略微不同的语义。

更多 JavaFX 有关 CSS 请查阅相关专题： Skinning JavaFX Applications with CSS.

3. 10 UI 控件

JavaFX 可用界面控件（UI Controls）经 JavaFX API 在场景图中通过使用节点被构建。这些控件能完全利用可视化丰富 JavaFX 平台特性的的好处，并能轻便夸不同平台。CSS 允许 UI 控件实现不同主题和外观。

图-3 展示了当前支持的图形界面控件。新的 Java UI 控件，像 TitlePane 或 Accordion 已经引入到 JavaFX2.0 的 SDK 中，这些新组建在 javafx.scene.control 包中。

火龙果•整理
uml.org.cn

图-3 JavaFX UI 控件示例

更多界面控件信息，请查阅相关主题介绍：Using JavaFX UI Controls 和 API documentation 。

3.11 布局设计（Layout）

在 JavaFX 应用场景图中，布局容器或面板用于灵活和动态性布置 UI 控件位置。布局 API 包括如下常见的自动化布局模型的容器类：

- BorderPane 类：把内容节点框设在顶部、底部、右部、左部以及中间区域；
- HBox 类：在单行内水平放置内容节点；
- VBox 类：在单列中垂直放置内容节点；
- StackPane 类：在一个从后到前单栈中放置内容节点；
- GridPane 类：使开发者创建灵活的行列网格来安置内容节点；
- FlowPane 类：以水平或垂直“流化”形式放置内容节点，且包装在指定宽（对水平）或高（对垂直）边界内；
- TilePane 类：以一致尺寸的单元格或标题栏放置内容节点；
- AnchorPane 类：使开发者可以在布局的顶部、底部、左边、或中部创建锚节点。

为了得到一个理想的布局结构，不同容器可在 JavaFX 应用中嵌套使用。

需要学习更多布局应用，请查看“Working with Layouts in JavaFX”相关文章。相应的

火龙果•整理
uml.org.cn

布局 API 详情，可查看 javafx.scene.layout 包。

3.12 2D 和 3D 转换

场景图中的每个节点都可使用 javafx.scene.tranform 中的类基于 X-Y 坐标进行转变：

- Translate：可以相对初始位置，沿着 XYZ 把节点从一个位置面移到另一个位置面；
- Scale：基于放缩因子，在 XYZ 方向上重设一个节点的大小；
- Shear：旋转一轴，使 X 和 Y 轴不在垂直，使节点的坐标基于指定的乘数被改变；
- Rotate：以指定的的轴点旋转场景中的节点；
- Affine：执行从一个 2D/3D 坐标到另一个 2D/3D 坐标的线性映射，同时保留“直线”和“并行”的直线特性。这个类应该和 `Translate`, `Scale`, `Rotate`, or `Shear` 转换类一起使用，而不是直接使用。

需要学习更多关于 JavaFX 的转换内容，查看“ Applying Transformations in JavaFX”文档。

3.13 可视化效果

基于 JavaFX 场景图的富客户界面的开发，主要包括可视化效果的使用或是应用外观效果的实时性强化。JavaFX 效果主要是基于像素的图像应用，也就是把场景图中的一套节点作为一个图片渲染，并能对它应用指定的效果处理。

JavaFX 中一些有效可视化效果处理类如下：

- Drop Shadow（拖放阴影）：在应用效果的内容后渲染相应内容的阴影；
- Reflection（倒影）：在实际内容对象下面渲染相应的倒影；
- Lighting（照射）：在给定的内容对象上模拟发亮光源，并能形成平面对象，呈现一个真实的、3-D 的面貌。

更多效果示例，请参考“ Creating Visual Effects”文档。相应的 API 可查看 javafx.scene.effect 包。

3.14 部署

JavaFX 应用程序可以三种不同的部署模式部署在桌面或浏览器中：

- 单机部署（Standalone）：应用程序安装在本地驱动，并可双击相应 JAR 文件运行。这种模式在用户不需要或没有联机访问时是理想选择。
- 浏览器部署（Browser）：这种模式中，应用程序被嵌入到页面中，当访问页面时自动启动程序。这通过 Javascript 可实现和页面交互；
- Web Start 方式：这种模式需要从网上中心位置应用程序，然后在桌面上运行之。一旦下载完成，用户可双击 JNLP 来运行相应的应用程序。

更多信息可查看“Deploying JavaFX Applications document”内容。

火龙果•整理
uml.org.cn

4 JavaFX 开发入门

如果想使用 JavaFX 快速开发丰富用户体验的应用，那么这个入门教程适合你。你将通过创建一个如图-1 所示的简单应用，就能学知用少许代码就能获得复杂图形效果。当然，JavaFX 不止乎漂亮的动画图形。在你完成这个学习教程后，看下 JavaFX 系列应用示例，以便带你进入更丰富的开发之旅。

图-1 多彩圆形（ColorfulCircles）应用

这个图形展示了彩色应用效果。图中有 30 个圆跨越黑色背景而活动。这些圆有个模糊的白边，5%的不透明度。这些圆圈显示于黄、蓝、绿、粉及红色的阴影内。

如果熟悉 JavaFX 场景图编程模式，将更容易理解这个应用程序代码。舞台（Stage）是应用的顶层容器，布景供应用内容描绘界面之需。内容是由层次节点树构成的场景图。

图-2 展示了 ColorfulCircles 应用的场景图。应用中分支检点是 Group 类的实例，无分支节点即叶子节点时是 Rectange 和 Circle 类的实例。

火龙果•整理
uml.org.cn

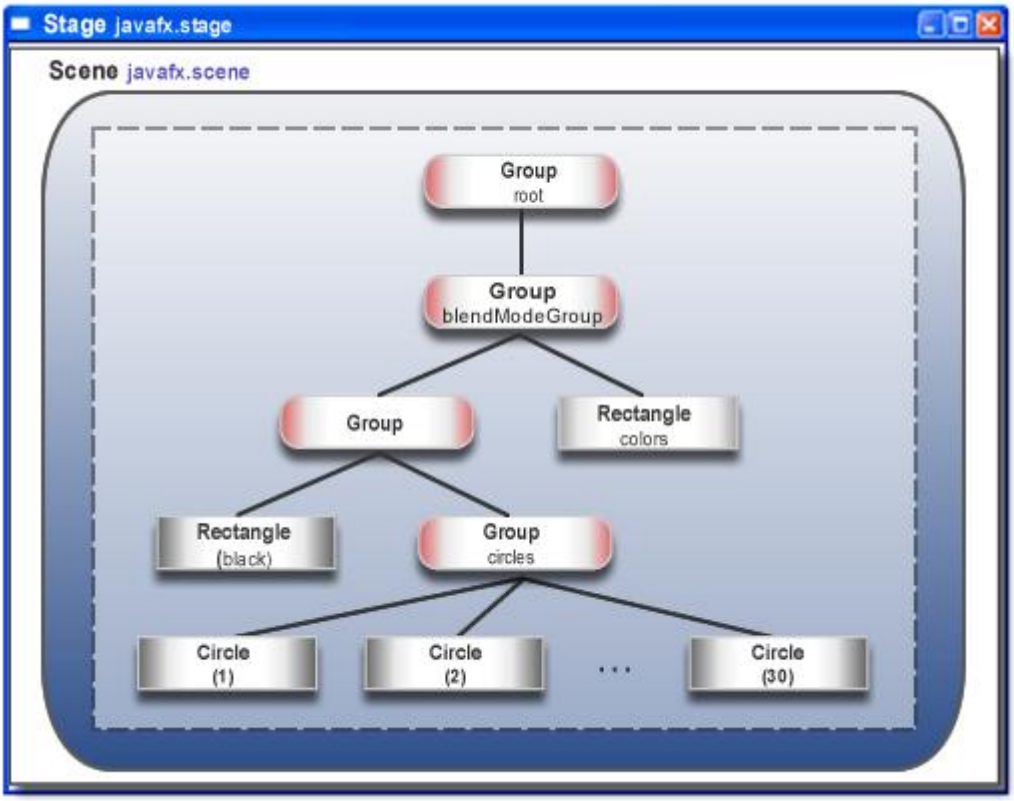


图-2 ColorfulCircles 布景图

这个图形展示了以 Stage 为顶层容器的 ColorfulCirlces 应用，布景为应用内容描绘界面只用。Group 节点时场景的根节点。根节点有一个子节点，即命名为 blendModeGroup 的分组节点。这个 blendModeGroup 有一个分组未命名子节点，包括一个黑色矩形和一个有 30 个圆圈的圆组。称之为 colors 的矩形也是 lendModeGroup 的一个子节点。

4.1建立应用

你可以用任何创建 Java 应用的工具开发 JavaFX 应用。这个教程中开发工具使用的 NetBeans IDE。在开始前，请确定你所使用的 NetBeans IDE 的版本，要求支持 JavaFX 2.0.

1. 在 NB IDE 中建立 JavaFX 工程如下：
2. 在 File 菜单中选择“new Project”（新工程）；
3. 在 JavaFX 应用类别中选择 JavaFX Application。点击 “next”（下一步）；
4. 把工程命名为 ColorCircles，并点击“Finish”（完成）；
5. 打开 ColorCircles.java 文件，复制导入语句，粘贴到工程中，重写 NB IDE 生成的语句。或者按照你自己工作方式来完成这些代码。

4.2创建应用基础

删除 NetBeans 生成的 Colorcircles 类的代码，用示例-1 代码替换。这是运行 JavaFX 应用需求的最少代码。

火龙果•整理
uml.org.cn

```
public class ColorfulCircles extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    @Override
    public void start(Stage primaryStage) {
        primaryStage.show();
    }
}
（示例-1）
```

这个 Colorcircles 类扩展自 Application 类，并包括两个方法。第一个方法是 main()方法，并调用了 launch()方法。JavaFX 的一个最佳实践是：在 main()中只调用 launch()方法。

接下来，Colorcircles 类覆盖了 Application 类中抽象的 start()方法。这个 start()方法需要一个基本的应用级舞台（stage）作为参数。最后一行代码确保舞台效果（stage）可见。

现在可以编译运行这个应用了（在教程的每一步中，为了立即看到结果都可这样做）。如果碰到问题，请查看 Colorcircles.java 文件。

4.3增加布景

现在可以在 primaryStage.show()行前增加布景以及其所有相关内容。如示例-2 所示。

```
@Override
public void start(Stage primarystage) {
    Group root = new Group();
    Scene scene = new Scene(root, 800, 600, Color.BLACK);
    primaryStage.setScene(scene);
（示例-2）
    primaryStage.show()
```

此时运行了本应用，你将可看到全黑的布景。

4.4添加图形

接下来，创建 30 个圆形。在 primaryStage.show()行前添加示例-3 代码

```
Group circles = new Group();
for (int i = 0; i < 30; i++) {
   Circle circle = new Circle(150, Color.web("white", 0.05));
   circle.setStrokeType(StrokeType.OUTSIDE);
   circle.setStroke(Color.web("white", 0.16));
   circle.setStrokeWidth(4);
   circles.getChildren().add(circle);
```

火龙果•整理
uml.org.cn

```
}
root.getChildren().add(circles);
```

（示例-3）

这儿的代码创建了一个 circles 组，然后通过 for 循环向组中添加了 30 个圆。每个圆半径为 150，白色填充，不透明等级为 5%（意味着大部分透明）。

为了创建圆边框，代码中包含了 StrokeType 类。OUTSIDE 描边类型意味着圆边界由内向外扩张了 StrokeWidth 值的大小。描边演示是白色，不透明度为 16%，以便比圆的色彩更亮。

最后一行是把圆组天剑到根节点。这是临时性结构。后面将可改变布景图以匹配图-2 展示。

图-3 展示的应用程序。由于代码并没指定每个圆的位置，这些圆都以左上角为中心点绘制在另外圆的上方。覆盖圆的不透明度与黑色背景交互，产生了灰色圆效果。

图-3

这幅图展示了一个黑色背景，并在左上角带四分之一圆的情形。圆是白边灰色。

4.5增加可视效果

把模糊效果应用到应用示例的圆上，使其些许的模糊不清。把示例-4 的代码增加到 primaryStage.show()代码行前，来看看效果。

```
circles.setEffect(new BoxBlur(10, 10, 3));
```

示例-4 模糊效果

上述代码设置模糊半径为 10 像素宽 10 像素高以及模糊迭代为 3，使其近似高斯模糊。这个模糊效果产生了一个圆滑的边缘效果。如图-4 所示。

火龙果•整理
uml.org.cn

图-4 圆的盒模糊

这幅图展示了一个黑色背景，并在左上角带四分之一圆的情形。圆的边是圆滑、模糊效果。

4.6创建渐变背景

创建一个矩形，并以线性渐变填充。如示例-5 代码所示。为了使渐变矩形在圆后，把示例代码增加到 root.getChildren().add(circles)代码行前。

```
Rectangle colors = new Rectangle(scene.getWidth(), scene.getHeight(),
    new LinearGradient(0f, 1f, 1f, 0f, true, CycleMethod.NO_CYCLE, new
       Stop[]{
          new Stop(0, Color.web("#f8bd55")),
          new Stop(0.14, Color.web("#c0fe56")),
          new Stop(0.28, Color.web("#5dfbc1")),
          new Stop(0.43, Color.web("#64c2f8")),
          new Stop(0.57, Color.web("#be4af7")),
          new Stop(0.71, Color.web("#ed5fc2")),
          new Stop(0.85, Color.web("#ef504c")),
          new Stop(1, Color.web("#f2660f")),}));
         root.getChildren().add(colors);
```

示例-5 线性渐变

这部分代码创建了一个命名为 colors 的矩形。这个矩形宽和高与布景一样，并以线性渐变填充，同时线性渐变开始于左手边(0,1)，结束于右手边(1,0)。值 true 意味着渐变与矩形是成比例的，NO_CYCLE 说明颜色循环不再进行。Stop[]序列说明在特定点渐变色应该是什么。代码最后一行把 colors 矩形加入根节点。

模糊边线的灰色圆呈现在彩虹色上方，如图-5 所示。

火龙果•整理
uml.org.cn

图-5 线性渐变色

图形说明：这幅图在渐变色上展示了左上角模糊圆的四分之一。渐变从左下方黄色到右上角红色结束。

图-6 展示了内部场景结构。在这点，Circles 组和 colors 矩形是根节点的子节点。

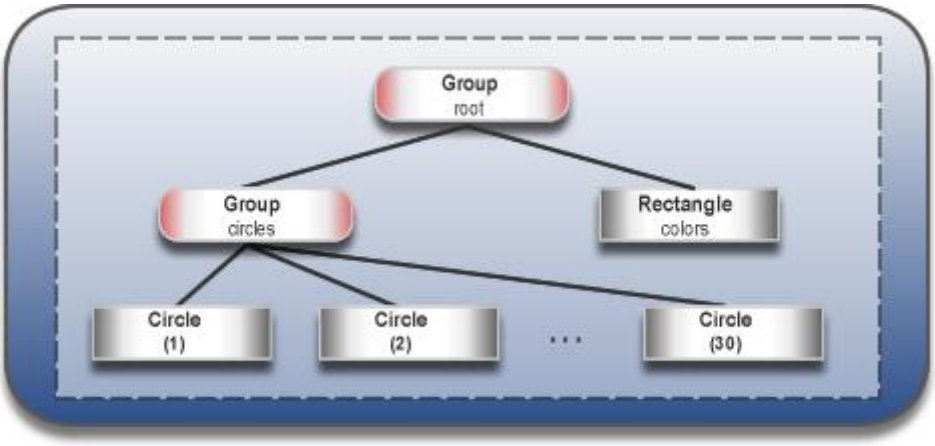


图-6 内部场景结构

图-6 说明：内部场景结构展示了带有两个子节点的根节点：一个 circles 组和一个 colors 矩形。Circles 组包含 30 个圆形。

4.7应用混合模式

现在向圆形添加色彩，并通过增加叠加混合效果暗化布景。这项任务需要些内务处理，及需要从场景图上移走 circles 组和渐变矩形，并把它们添加到新的层叠混合组。

1. 移走下面两行代码：root.getChildren().add(colors);root.getChildren().add(circles);
2. 把示例-6 的代码添加到你移走代码的地方；

```
Group blendModeGroup =
   new Group(new Group(new Rectangle(scene.getWidth(), scene.getHeight(),
        Color.BLACK), circles), colors);
colors.setBlendMode(BlendMode.OVERLAY);
3. root.getChildren().add(blendModeGroup);
```

火龙果•整理
uml.org.cn

示例-6 混合模式

blendModeGroup 组设定了层叠混合结构。这组包含两个子节点。第一个子节点是个新的未命名组，并容纳了一个未命名的黑色矩形和前边创建的 circles 组。第二个子节点是前面创建的 colors 矩形。

setBlendMode()方法把层叠混合模式应用到 colors 矩形。最后代码行把 blendModeGroup 作为根节点的孩子增加到布局图内，如图-2 描述。

层叠混合是图形设计应用中普通的效果它们可以暗化或量化一副图片，或者兼有，这依赖于混合模式中的色彩。在这个案例中，线性渐变矩形拥有层叠。黑色矩形用于保持背景黑暗，即使当几乎透明的圆从这渐变色中捡取色彩，但也是暗化的。如图-7 所示结果。当下一步中是这些圆动态化时，你将可看到完整的层叠混合效果。

图-7 层叠混合

4.8 添加动画

教程最后一步是使用 JavaFX 动画移动这些圆：

1. 如果还没导入“import static java.lang.Math.randow;”到引入列表，请增加；
2. 把代码示例-7 中的动画代码增加到 primaryStage.show()行前；

```
Timeline timeline = new Timeline();
for (Node circle: circles.getChildren()) {
   timeline.getKeyFrames().addAll(
      new KeyFrame(Duration.ZERO, // set start position at 0
         new KeyValue(circle.translateXProperty(), random() * 800),
         new KeyValue(circle.translateYProperty(), random() * 600)
      ),
      new KeyFrame(new Duration(40000), // set end position at 40s
         new KeyValue(circle.translateXProperty(), random() * 800),
         new KeyValue(circle.translateYProperty(), random() * 600)
      )
   );
```

火龙果•整理
uml.org.cn

```
}
// play 40s of animation
timeline.play();
```

动画有时间线驱动，因此这些代码创建时间线，并运用 for 循环增加两个关键帧到每个圆中（共 30 个圆）。第一个关键帧在 0 秒时使用 translateXproperty 和 translateYProperty 两属性设置窗口内随机位置。第二个关键帧在 40 秒时，做和第一帧同样的事情。这样，当时间线播放时，它将所有圆从一个随机位置，40 秒后变化到另一个随机位置。

图-8 展示了 30 个彩色圆的动画状态，至此，完成了整个应用程序的工作。

图-8 动画圆形

4.9部署应用

当在 Netbeans IDE 中运行此应用或使用 Clean&Build 命令，应用可打包为各种 JavaFX 部署类型（可以通过工程属性设定选项）。然后进入 ColorfulCircles 工程的 dist 目录下，可双击下 3 个文件的每一个：

- ColorfulCircles.jar 以单机模式运行；
- ColorfulCircles.jnlp 以 Web Start 模式运行；
- ColorfulCircles.html 打开一页面，含有一个 Web Start 连接，并且也能在运行在浏览器中。

更多的部署信息，可以查看“部署 JavaFX 应用”相关描述。

至此，本开发入门教程完毕。

下一篇，将介绍 FXML 入门，敬请期待。谢谢。

火龙果•整理
uml.org.cn

5 FXML 入门教程

本部分教程包括两部分内容：1、为什么使用 FXML（基本介绍以及用 FXML 创建用户界面的好处）；2、使用 FXML 创建用户界面（通过创建简单登录应用来完成本教程部分）。

5.1 为何使用 FXML

FXML 是基于 XML 的语言，它提供与应用逻辑代码分离的构建用户界面的结构。展现层和应用逻辑的分离对 web 开发者更有吸引力：因为他们可以利用 java 组件组装用户界面而不需要掌握获取并填充数据的代码。

接下的部分提供更多关于 FXML 的信息，之后你会选择 FXML 创建用户界面而不是其它方法。下面分三部分介绍 FXML

- FXML 介绍；
- FXML 简单示例；
- FXML 好处。

5.1.1 FXML 介绍

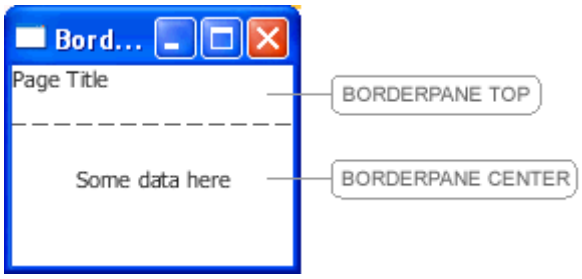
FXML 没有相应的模式，但有一个基本的预定义结构。用 FXML 表达什么，如何用它构建场景图，依赖于你构造对象的 API。因为 FXML 直接映射到 Java，参照 API 文档以更好地什么理解元素和属性允许应用。通常，大多数 JavaFX 类能作为元素使用，大多数 Bean 的 Properties 特性可以用作属性。

来源于模型-视图-控制器（MVC）观点，FXML 文件包含用户界面描述的部分称之为视图。控制器是可选性实现 Initializable 类的 Java 类，这被声明为 FXML 文件的控制器。模型由定义 java 端的域对象构成，并通过控制器连接到视图。在“使用 FXML 创建用户界面”中有一个结构性示例。

使用 FXML 创建用户界面时，对大而复杂的布景、表单、数据实体或是复杂动画，使用 FXML 是特别有用的。FXML 也是适合定义静态布局结构，诸如表单、控件和表等。另外，通过包含脚本还能使用 FXML 构建动态布局。

5.1.2 FXML 简单示例

最简单展示 FXML 好处的方法是实例。如图 1-1 所示，它展示了一个包含 Pane 布局的用户界面，Pane 布局有一个顶部和中部区域，每个区域中包含一个 label 标签。



火龙果•整理
uml.org.cn

图 1-1 边格简单示例

说明：布局类 BorderPane 定义两个区域：顶部区域和中部区域。顶部区域包含一个“Page Title”标签，中部区域包含一个“some data here”标签。

首先，看下用户界面结构如何，在源代码中如何直接创建，如下示例 1-1 所示。

```
BorderPane border = new BorderPane();
Label toppanetext = new Label("Page Title");
border.setTop(toppanetext);
Label centerpanetext = new Label ("Some data here");
border.setCenter(centerpanetext);
```

示例 1-1 用户界面的 java 代码

现在看下示例 1-2，展示相同的用户界面，但是用 FXML 标记的。这儿能看到用户界面的层次结构，这更容易依次添加控件构建用户界面。

```
<BorderPane>
   <top>
      <Label text="Page Title"/>
   </top>
   <center>
      <Label text="Some data here"/>
   </center>
</BorderPane>
```

示例 1-2 FXML 标记用户界面

5.1.3 FXML 的好处

除了提供给 web 开发者一个熟悉的用户界面设计方法万，FXML 还提供如下好处：

- 由于场景图在 FXML 中更透明，这使开发团队更容易创建和维护一个可测试的用户界面；
- FXML 是非编译语言，无需重编译就可查看变化；
- FXML 文件内容可以本地化处理。例如，如果一个 FXML 文件使用 en_US 本地化加载，它基于下面资源字符串产生“First Name”字符串标签：`<Label text="%firstName"/>`。如果本地化变为 fr_FR，这个 FXML 文件重新加载，这标签将展示“Prénom”。相同的事情对 java 代码则不行。因为你需要获得相应引用后，必须手动更新每个用户界面元素内容，并正确调用 setter。
- 可以和任何 JVM 语言一起使用 FXML，诸如 Java、Scale 或者 Clojure。
- FXML 不局限与 MVC 模型的视图部分。你可以构造服务、任务或域对象，并且你可以在 FXML 中使用 javascript 或其它脚本语言。关于使用脚本示例，请参考本教程“使用脚本语言”部分。

5.2 创建用户界面

通过前述介绍，现在已经对怎么使用 FXML 有了一定认知，接下来将在教程实战中更好理解。这个实战介绍怎么创建如图 2-1 所示的登录界面。

火龙果•整理
uml.org.cn

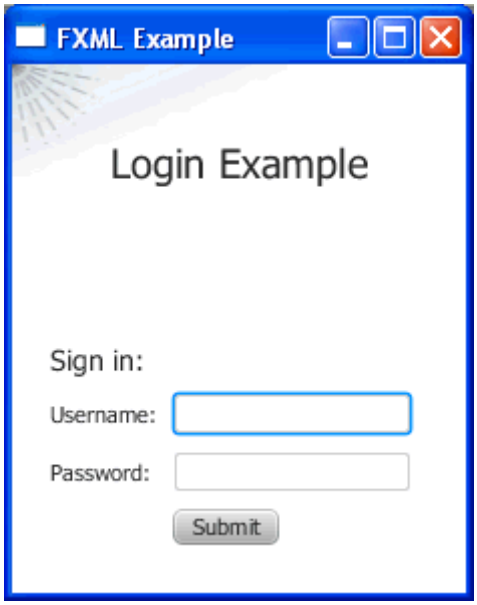


图 2-1 登录界面

图例说明：应用标题为“Login Example”。标签重叠了一个背景图片。从顶到底，应用程序中的控件是：带文本“Sign In”的标签、带“UserName”的文本框、带“Password”的密码框以及一个提交按钮。

开始前，熟悉下登录界面布局，如图 2-2 所示。用户界面使用了一个包含两个区域的边格布局（Border Pane）。顶部区域包含一个堆栈布局，内有文字“Label Example”并层叠在一个图片上。中间区域包含一个网格布局，并含有标签、文本框、密码框以及按钮。

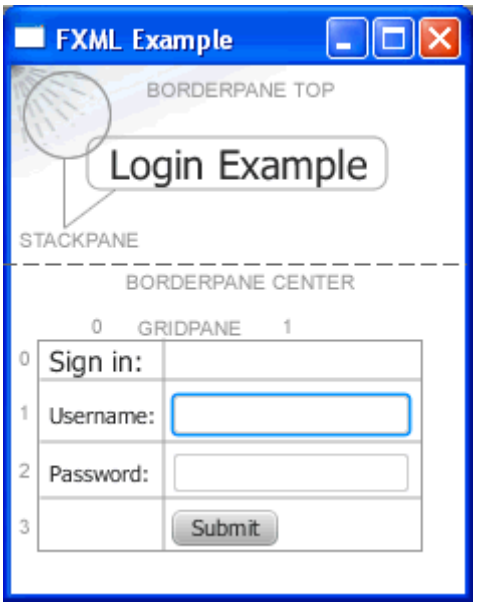


图 2-2 登陆界面布局

5.2.1 准备工作

本部分教程使用到 NetBeans IDE。确定你在使用的 NetBeans IDE 版本支持 JavaFX2.0。更多相关细节参考“系统要求”部分。

火龙果•整理
uml.org.cn

为了完成这个教程，应该熟悉如何用 JavaFX 程序性构建用户界面。了解如何与布景图一起工作时特别有益处的。因为 FXML 的层次结构和 JavaFX 布景图结构是密切相关的。

5.2.2 创建工程

首先在 NetBeans 的 IDE 中创建一个 JavaFX 工程。

1. 从“文件”（File）菜单，选择“新工程”（New Project）；
2. 在 JavaFX 应用分类中选择 JavaFX FXML 应用。点击下一步；
3. 命名工程为 FXMLExample 并点击“完成”（Finish）。IDE 打开带有 3 个文件的工程：FXMLExample.java，Sample.fxml 以及 Sample.java；
4. 下载一个淡蓝渐变的背景图片，并保持到桌面，然后拖动到在源代码包目录下的 fxmlexample 目录。

5.2.3 创建应用基础

每个 FXML 应用必须包括 java 代码。最少包括创建 stage 和 scene 以及加载 launch 应用的代码。

打开 FXMLExample.java，删除 IDE 生成的代码，并用示例 2-1 代码替换。

```
package fxmlexample;

import java.util.ResourceBundle;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class FXMLExample extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        stage.setTitle("FXML Example");

        Parent root =
FXMLLoader.load(getClass().getResource("fxml_example.fxml"),
            ResourceBundle.getBundle("fxmlexample.fxml_example"));
        Scene scene = new Scene(root, 226, 264);
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
```

火龙果•整理
uml.org.cn

```
        launch(args);
    }
}
```

示例 2-1 FXMLExampel.java

作为 JavaFX 程序员，创建布景和舞台以及加载应用的代码应该是熟悉的。下面这行指向于相应的 FXML 文件：

```
Parent root = FXMLLoader.load(getClass().getResource("fxml_example.fxml"),
ResourceBundle.getBundle("fxmlexample.fxml_example"));
```

这个 FXMLLoader.load()方法从资源文件 fxml_example.fxml 中加载层级对象，并赋值到名为 root 的变量。getRsources()实现资源绑定，以便于具体化界面中的字符串，也更容易资源本地化处理。接下来的两部分，将回到资源绑定部分——属性文件和创建 FXML 源文件。

总体来说，创立布景对象，并设置 root 变量未场景图的根。FXML 中这个根元素将作为布景图的根节点。

5.2.4 创建属性文件

一个最佳实践是具体化用户界面的文本字符串到一个属性文件。可采取如下步骤创建登录界面的属性文件。

1. 在工程窗口，右键点击 fxmlexample 文件夹，并选择“新建”（new），然后点击“其它”（Other）；
2. 在新文件对话框中点击“其它”（Other），选择 Properties 文件，点击“下一步”（next）；
3. 键入 fxml_example 作为文件名，点击“完成”（Finish）。然后 IDE 打开以.properties 为扩展名的文件；
4. 键入资源名称和值，如下示例 2-2 所示：

```
loginExample=Login Example
signIn=Sign in:
userName=Username:
password=Password:
submit=Submit
```

5.2.5 创建 FXML 文件

现在创建一个名称为 fxml_example.fxml 的 FXML 文件，并插入 xml 声明，导入语句。

1. 在项目窗口，在 fxmlexample 文件夹下右键点击 Sample.fxml，选择“重命名”（Rename）；
2. 键入 fxml_example，点击“OK”；
3. 打开 fxml_example 文件，删除 IDE 生成的代码，并以示例 2-3 的的代码替换。

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.layout.*?>
```

火龙果•整理
uml.org.cn

```
<?import javafx.scene.control.*?>
<?import javafx.scene.*?>
<?import javafx.scene.image.*?>
```

示例 2-3 声明和导入语句

所有 FXML 文件必须以 XML 声明开始。它定义了 XML 的版本(1.0)和编码类型(UTF-8)。与在 JavaFX 里一样，类名必须合乎规范（包括包名），或者也可以使用导入语句导入，如示例 2-3 所示。如果你喜欢，也可使用具体导入语句指向你的类。

5.2.6 定义边格布局

接下来开始构建用户界面。再导入语句之后插入边格布局内容，如示例 2-4 所示：

```
<BorderPane fx:controller="fxmlexample.FXMLExampleController"
    xmlns:fx="http://javafx.com/fxml">
     <top>
     </top>
     <center>
     </center>
</BorderPane>
```

示例 2-4 边格布局

在这个示例中，场景图的根是边格布局类。它定义了顶部和中部两个区域。fx:controller 属性指定控制文件，且必须在根 FXML 元素内声明。在后续的讲学习更多关于控制器内容。

xmlns:fx=" http://javafx.com/fxml" 属性把命名空间映射到 URL 为 http://javafx.com/fxml 的位置。必须在 FXML 根元素中声明命名空间。这个属性可以确保相应 JavaFXAPI 元素能通过 FXML 标签来使用。

5.2.7 图片上堆叠文本

接下来在顶部区域嵌套一个堆栈窗格。这个窗格上包含了一个叠加在图片的标签。如图 2-3 所示。

图 2-3 边格布局顶部区域，包含栈窗格

堆窗格代码如示例 2-5 所示。把下列代码插入到<top>和</top>标签之间。

```
<StackPane>
          <children>
              <ImageView>
                 <image>
```

火龙果•整理
uml.org.cn

```
                <Image url="@fx_boxback.jpg"/>
            </image>
        </ImageView>
        <Label text="%loginExample" style="-fx-font: NORMAL 20 Tahoma;"/>
    </children>
</StackPane>
```

StackPane 布局类把它的子节点放在单一堆栈内，每新增一个节点都会防止在前一个节点之上。这种布局模式提供了一种简单的图片上叠加文字的方式。

标签<children>实现把子节点增加到布局图上，这很类似于 JavaFX 中使用 getChildren().add()方法。

图片 Image 类加载“fx_boxback.jpg”，此文件假设放置在当前 FXML 文件位置。ImageView 类把图片展示在屏幕上。

标签 Label 类有个文本属性设置资源名称 loginExample。在 FXML 中，一个资源名称由%号操作符唯一指定。加载时，FXML 加载器用它的值（Login Example）替换 loginExaple 资源名称。

注意：FXML 用 style 属性定义风格。在示例 2-5 中，Label 类使用了 style 属性设置了字体风格。

另一种定义风格的方法是定义 CSS，和在 java 中使用方法一样。使用 CSS 可以更容易在以后重定制应用风格。

5.2.8 添加 Grid 布局和控件

接下来，在边格布局的中部区域嵌入 Grid 布局。以水平和垂直的方式在 Grid 布局上放置控件。如图 2-4 所示。

图 2-4 边格布局中部的 Grid 面板

示例 2-6 包含了这个 grid 布设的代码。把它插入到 和 标签中间。

```
<GridPane alignment="top_center" hgap="8" vgap="8"
          style="-fx-padding: 40 0 0 0">
          <children>
              <Label text="%signIn"
                  style="-fx-font: NORMAL 14 Tahoma;"
                  GridPane.columnIndex="0" GridPane.rowIndex="0"/>
```

火龙果•整理
uml.org.cn

```
            <Label text="%userName"
                GridPane.columnIndex="0" GridPane.rowIndex="1"
                labelFor="$usernameField"/>
            <TextField fx:id="usernameField" prefColumnCount="10"
                GridPane.columnIndex="1" GridPane.rowIndex="1"/>

            <Label text="%password"
                GridPane.columnIndex="0" GridPane.rowIndex="2"
                labelFor="$passwordField"/>
            <PasswordField fx:id="passwordField" prefColumnCount="10"
                GridPane.columnIndex="1" GridPane.rowIndex="2"
                onAction="#handlePasswordFieldAction"/>

            <Button fx:id="submitButton" text="%submit"
                GridPane.columnIndex="1" GridPane.rowIndex="3"
                onAction="#handleSubmitButtonAction"/>

            <Label fx:id="buttonStatusText"
                GridPane.columnIndex="1" GridPane.rowIndex="4"
                style="-fx-text-fill: #ff0000;"/>
        </children>
    </GridPane>
```

示例 2-6 带控件的 Grid 布设

这儿的代码现在看起来很熟悉，但也有些新属性需要解释。GridPane.columnIndex 和 GridPane.rowIndex 属性对应 GridPane 类的 setColumnIndex()和 setRowIndex()方法。行列的数量从 0 开始。例如，第一个 Label 控件的位置是(0,0)，这意味着他在第一行第一列。或是左上角。PrefColumnCount 属性被用于 TextField 和 PasswordField 控件，预设置 10 列。

把值 fx:id 赋予一个文档命名空间里创建变量的元素，以便后续代码中引用。变量引用通过$操作符来标记。在示例 2-6 中，TextField 的 ID 是 usernameField。这个 ID 被赋予控件 Label 的 labelFor 属性，以便为 TextField 设置标签。

注意 PasswordField 和 Button 控件有个 onAction 属性，这个数字签名引用一个定制方法，这个方法你可以在下一节创建 FXMLExampleController 类时定义。

尽管 FXML 是定义应用用户接口结构的便捷方式，但并不提供应用行为的实现方式。这个行为必须在 java 代码或脚步语言中实现（下一节将描述）。

5.2.9 添加按钮事件

现在，创建一个控制器来处理按钮事件。这个例子演示了 java 写的事件处理器如何与 FXML 标记关联。

1. 在工程窗口，在 fxmlexample 文件夹下右击 Sample.java，并选择“Refactor”，然后选择“Rename”（重命名）；

火龙果•整理
uml.org.cn

2. 键入 FXLExampelController 并点击“Refactor”；
3. 打开 FXMLExampleController.java 文件，删除 IDE 生成代码，并用示例 2-7 代码替换之。

```
package fxmlexample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class FXMLExampleController {
  @FXML private Label buttonStatusText;

   @FXML protected void handleSubmitButtonAction(ActionEvent event) {
       buttonStatusText.setText("Submit button pressed");
    }

    @FXML protected void handlePasswordFieldAction(ActionEvent event)
{
       buttonStatusText.setText("Enter key pressed");
    }
}
```

示例 2-7 FXMLExampleController.java

@FXML 标注用于标签非公共控制器成员字段域，并且由标签使用相应的处理器方法。另外，对于 java 编程语言，你也可使用其它编译语言，如 Scala 来实现控制器。

现在可以运行这个应用程序。在两个输入域键入文本并点击“Submit”（提交）测试下这个应用，看看如何。

完整的应用代码可以下载这个压缩包“FXMLExample.zip”。

5.2.10 使用脚本语言

相对于使用 java 常见事件处理器，你也能使用提供了 JSR223 兼容的脚本引擎的语言来创建处理器。诸如 JavaScript、Groovy、Jython 和 Clojure。在登录示例中采用 javascript 按如下步骤编辑 FXML 文件。

1. 在 fxml_example.fxml 文件中，在 XML 声明后增加 Javascript 声明。
   <?language javascript?>
2. 在按钮标记中，改变下功能名称，调用如下所示：
   onAction=”handleSubmitButtonAction(event);”
3. 更新 PasswordField 标记，类似如下：
   onAction=”handlePasswordFieldAction(event);”
4. 从 BorderPane 标记中删除 fx:controller 属性，并在<script>标签中直接增加

火龙果•整理
uml.org.cn

JavaScript 功能，如示例 2-8 所示：

```
<BorderPane xmlns:fx="http://javafx.com/fxml">
        <fx:script>
            function handleSubmitButtonAction() {
                buttonStatusText.setText("Calling the
JavaScript");
            }
            function handlePasswordFieldAction(event) {
                buttonStatusText.text = "More JavaScript";
            }
        </fx:script>
```

示例 2-8FXML 中 JavaScript

另外一种选择是你可以把 Javascript 功能放在一个外部文件（如 fxml_example.js）中，然后通过下面脚本包含近来：

```
<fx:script source="fxml_example.js"/>
```

如果你考虑和 FXML 一起使用脚本语言，注意，在调试期间可能 IDE 不支持进入相应脚本代码。

5.2.11 应用式样表

作为内嵌样式替代物，你可以对布景使用式样表，然后设置节点的样式类。按照如下步骤创建式样表，并定义了 Grid 面板和 Label 控件的样式。

1. 创建式样表
   a. 在工程窗口，右击源码包目录下的 fxmlexample 文件夹，选择“New”然后选择“Other”；
   b. 在新的文件对话框，选择 Other，然后选择“Cascading Style Sheet”,再点击“Next”；
   c. 键入 fxmlstylesheet,点击“Finish”；
   d. 删除 IDE 生成代码，用示例 2-9 的代码替换

```
@charset "utf-8";
/*
    Document   : FXMLstylesheet.css
*/
.grid-pane {
  -fx-padding: 80 0 0 0;
}
.label {
    -fx-font: normal 36px Tahoma;
}
```

2. 打开 FXMLExample.java 文件在代码行 stage.show()前包含如下代码增加样式表到布景中

火龙果•整理
uml.org.cn

```
scene.getStylesheets().add("fxmlexample/fxmlstylesheet.css");
```

3. 到 fxml_example。Fxml 文件并增加式样类。
   a. 为<String>元素增加导入语句。`<?import java.lang.*?>`
   b. 用示例 2-10 代码替换 GridPane 标签内容

```
<GridPane alignment="top_center" hgap="8" vgap="8">
         <styleClass>
             <String fx:value="grid-pane"/>
          </styleClass>
```

   示例 2-10

   c. 用示例 2-11 代码替换“Sign In”标签

```
<Label text="%signIn"
   GridPane.columnIndex="0" GridPane.rowIndex="0">
       <styleClass>
            <String fx:value="label"/>
       </styleClass>
</Label>
```

   在一个对象内使用<styleClass>标签时，式样被应用到所有同类实例（除了类有自己的内联式样）。因此，当你运行 FXMLExample 应用是，示例 2-11 式样被应用到不仅仅是 Sign 标签，也应用到了 Username 和 Password 标签。式样没有应用到登录示例的标签，因为他有一个内有式样覆盖了这个样式表。

5.2.12 教程回顾

下面简单回顾一下教程信息点：

- FXML 是 JavaFX 的组件。它是个创建 JavaFX 应用用户界面的可选项；
- FXML 是基于 XML 的；
- FXML 文件作为规则 JavaFX 工程的一部分，不需要编译；
- FXML 层次架构和 JavaFX 布景图结构对应；
- 开始 FXML 文件的最佳编码实践是带有 XML 声明：`<?xml version="1.0" encoding="utf-8"?>`；
- FXML 命名空间必须定义在 XML 代码中组件顶层，以便 FXML 标签整体使用，例如：`<rootElement xmlns:fx="http://javafx.com/fxml">`；
- 通过资源文件的字符串提取，FXML 能本地化处理。资源替代通过%标识；
- FXML 定义样式和 java 代码中 setStyle()定义 CSS 类似。既可以使用内联式样，也可使用样式表；
- 给元素 fx:id 赋值，以便后续引用。引用时通过$标识变量；

火龙果•整理
uml.org.cn

- 除了 java，FXML 支持其他编译语言实现的控制器，如 Scala；
- FXML 支持任何 JSR223 兼容的脚本引擎，这样的语言包括 JavaScript、Groovy、Jython 以及 Clojure 等。

5.3接下来…

现在已熟悉了 FXML，可以进一步查看 FXML 介绍文档，文档提供了关于 FXML 语言元素的更多信息。文档包含在 javafx.fxml 包里，详见这里。

另外，也可自行扩展上面的 FXML 应用，比如校验用户名不能为空，增加取消按钮等。

6 JavaFX 开发概要

通过前文介绍，对 JavaFX 基本知识及开发技术已经基本熟悉，下面简要的总结一下开发的关键步骤流程，以便加深 JavaFX 应用开发的理解。

1、 第一步，是下载并安装 JavaFX SDK，或者完整版 SDK，或者安装 JavaFX 运行时；
2、 第二步，创建应用框架。每个 JavaFX 应用有相似的框架结构：扩展自 Application 类的应用主类；main()方法调用 launch()方法；start()方法设置并展示 stage 场景（包含应用 UI 控件）；基本程序结构如下所示：

```
public class XXXMainClass extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    @Override
    public void start(Stage primaryStage) {
        //其它内容……
        primaryStage.show();
    }
}
```

3、 第三步，定义用户界面。应用的 stage 场景界面包含一个或以上 scense 布景，布景中依次包含控件、图形和图片，有这些内容装饰用户界面；
4、 第四步，布设 UI 控件。决定在 UI 中需要什么控件后，使用内置布局面板来统一管理控件的位置和大小属性等内容；
5、 第五步，部署。JavaFX 应用可以运行在桌面、浏览器以及 Web 上。当应用程序就绪时，用户可以创建.jar、.html 以及.jnlp 文件，之后可以按照相应的方式运行起来。

附注：JavaFX 应用的开发，如果通过文本编辑器开发的，则可像一般的 java 应用应通过命令行编译，然后再通过 javafxpackage 打包部署。具体怎么编译和打包部署，这里不再赘述。

火龙果•整理
uml.org.cn

第二篇　JavaFX 内建控件

本篇教程内容覆盖 JavaFX 内建的有效图形界面控件，包括如下章节内容：

- JavaFX UI Controls
- Label
- Button
- Radio Button
- Toggle Button
- Checkbox
- Choice Box
- Text Field
- Password Field
- Scroll Bar
- Scroll Pane
- List View
- Table View
- Tree View
- Separator
- Slider
- Progress Bar and Progress Indicator
- Hyperlink
- Tooltip
- HTML Editor
- Titled Pane and Accordion
- Menu

每部分内容提供相应示例和应用代码，以描述控件的功能作用。并可在内容列表中找到应用的代码文件以及相应 netbeans 工程文件。

火龙果•整理
uml.org.cn

1. 用户见面控件（UI 控件）

JavaFX UI 的可用控件由使用中的节点经由 API 创建于场景图中。因此，这些控件可以使用可视化地丰富 javaFX 平台特性。由于 JavaFX APIs 完全由 java 实现，你可以很容易把 JavaFX UI 控件集成到已存在的 Java 应用中。

图 1-1 展示了一些典型的 UI 控件的应用情况

2. JavaFX2.0 中支持的 UI 控件

构建 UI 控件的类所在的 API 包是 javafx.scene.control。UI 控件包含的典型界面组件可能与你以前的 java 应用客户端开发所见大差不离。但是 JavaFX2.0 SDK 介绍了些新的 Java 界面控件，如 TitlePane 和 TableView。

图 1-2 展示了 3 个带有设置清单的 TitlePane 元素。这些列表清单可以滑进和滑出（收缩扩展）。

火龙果•整理
uml.org.cn

详情可以查看 UI 控件的完整 API 描述文档。

图形界面控件（UI control，下文简称为 UI 控件）类基于 Control 类之上，以直观的方式提供了附加的变量和方法来支持典型的用户交互。你能通过应用 CSS 为你的 UI 组件指定样式。对于一些不常用的任务，可以自行扩展 Control 类创建定制的 UI 组件，或者使用 Skin 接口为存在的控件定义新的皮肤样式。