Ralph Steyer

Einführung in JavaFX

Moderne GUIs für RIAs und Java-Applikationen

Springer Vieweg

Einführung in JavaFX

Ralph Steyer

Einführung in JavaFX

Moderne GUIs für RIAs und Java-Applikationen

Springer Vieweg

Ralph Steyer
RJS EDV-KnowHow
Bodenheim
Deutschland

ISBN 978-3-658-02835-0 ISBN 978-3-658-02836-7 (eBook)
DOI 10.1007/978-3-658-02836-7

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist eine Marke von Springer DE. Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media
www.springer-vieweg.de

Vorwort

Willkommen in der Welt von JavaFX und Java, in der auf erstaunlich einfache und konsistente Weise unterschiedliche Plattformen für lokale Computer und mobile Endgeräte in Netzwerken bis hin zum Internet verschmelzen. JavaFX bezeichnet dabei ein Framework von Oracle zur Erstellung von modernen, plattform-übergreifenden Java-Applikationen. Mit JavaFX wird der bereits seit Jahren etablierte Zugang zur Java-Welt sowohl fortgeführt als auch vereinfacht. Denn die JavaFX-Technologie tritt nicht zuletzt mit dem Ziel an, das professionelle Erstellen und Verteilen von interaktiven, multimedialen Inhalten und grafischen Oberflächen über sämtliche Java-Plattformen hinweg zu erleichtern. JavaFX stellt dazu eine Reihe an neuen Schnittstellen bereit, über die erfahrene Java-Entwickler erst einmal wie gewohnt die bereits vorhandenen Java-Bibliotheken auf einfache Weise verwenden können. Desgleichen gibt es auch explizite Erweiterungen, die etwa Java Swing bei der Erstellung von Oberflächen ablösen sollen und die Nutzung von Multimediatechniken erleichtern bzw. erweitern. JavaFX bietet aber auch die Möglichkeit, eine grafische Oberfläche ohne Java-Kenntnisse zu erstellen. Das soll aber in nicht bedeuten, dass Java-Kenntnisse nicht von Vorteil wären, wenn man JavaFX effektiv einsetzen und voll funktionale Anwendungen entwickeln möchte.

Nun sollte man gleich zu Beginn festhalten, dass JavaFX kein wirklicher Newcomer ist. Das JavaFX-Framework gibt es schon seit einigen Jahren. Ab der Version 2 hat JavaFX fast nur noch den Namen mit den älteren Varianten gemeinsam. So gesehen ist JavaFX doch **neu** – im positiven Sinn des Wortes. Die neuen Versionen von JavaFX werden als die Zukunft der Entwicklung von grafischen Java-Oberflächen angesehen.

An wen wendet sich das Buch beziehungsweise wer sollte sich für diese Technologie interessieren? Zum einen ist die Zielgruppe neutral als IT-Fachleute mit Neugier auf die neue GUI-Technologie im Java-Umfeld zu benennen. Insbesondere sind dies die Webseitenersteller, Webdesigner beziehungsweise Designer und Webprogrammierer – Sun bzw. Oracle spricht hier von den *creative minds* –, die mithilfe einer Java-Technik multimedial aktive Web-Applikationen erstellen wollen oder an einem anderen Java-Projekt beteiligt sind, ohne direkt mit Java arbeiten zu wollen oder zu können. Diese werden auf Basis klassischer Web-Technologien wie HTML, XML, JavaScript und CSS mit JavaFX schnell und einfach produktiv. Zum anderen fallen (zukünftige) Programmierer für Desktop-Applikationen und mobile Apps in die Zielgruppe für dieses Buch, wenn sie einen Weg

suchen, auf einfache und einheitliche Weise mächtige Java-Techniken zur Erstellung von grafischen Oberflächen zu nutzen, ohne Java direkt anwenden zu müssen und/oder zu tief in Java einsteigen zu wollen. Natürlich dürften auch Java-Programmierer an JavaFX interessiert sein, nicht zuletzt weil JavaFX Swing ablösen soll bzw. Swing selbst um diverse neue Komponenten erweitert. Es ist sehr wichtig, dass Java-Programmierer die Ablösung des etablierten Swing-Konzepts nicht verpassen.

Die Zielgruppe wurde zwar mehrfach mit „Programmierer" bezeichnet, dennoch handelt es sich ausdrücklich um ein Einsteigerbuch – zumindest in Hinsicht auf die Programmierung. Sie müssen für den Umgang mit diesem Buch kein Programmierprofi sein. Kenntnisse in Java werden ebenfalls nicht vorausgesetzt, obgleich sie sehr hilfreich sein können – insbesondere bei den komplexeren Themen im hinteren Teil des Buches. Auf die notwendigen Grundlagen wird im Rahmen des Buchs eingegangen, unter anderem in einem kleinen Java-Crashkurs im Anhang. Grundlegende Erfahrung in einer (beliebigen) Programmier- oder Skriptsprache ist dennoch Voraussetzung, genauso wie Grundlagen im Umgang mit dem Internet. Ebenso werde ich nicht weiter auf elementare Details zu HTML, CSS oder JavaScript eingehen.

Nun habe ich kurz beschrieben, wie ich mir die Leser dieses Buchs vorstelle. Aber was habe ich als Autor mit JavaFX zu tun? Ich stelle mich Ihnen am besten kurz vor. Ich bin Diplom-Mathematiker. Seit 1996 arbeite ich als Fachautor, Fachjournalist, Dozent und Programmierer rund um die Themen Computer, (Java-)Programmierung und Internet. Davor stehen einige Jahre als Programmierer bei einer großen Versicherung im Rhein-Main-Gebiet in meinem Lebenslauf. JavaFX verfolge ich bereits seit den ersten öffentlichen Vorversionen.

Alles in allem finde ich gerade die neue Entwicklung von JavaFX spannend, denn die Technologie öffnet im Bereich der Oberflächenprogrammierung und in Hinsicht auf multimediale Effekte in der Tat neue Welten. Die Arbeit mit JavaFX macht auf jeden Fall viel Spaß. Und solchen Spaß als auch Erfolg wünsche ich Ihnen.

Januar 2014
www.rjs.de

Ralph Steyer

Inhaltsverzeichnis

Einleitung & Grundlagen

1

Ohne Grundlagen geht es nicht

1.1 Was behandeln wir in dem einleitenden Kapitel?

Im einleitenden Kapitel wollen wir die zentralen Grundlagen zum Umfeld behandeln, in dem man mit JavaFX programmiert. Das beginnt bei einem allgemeinen Blick in die Welt von Java und JavaFX mit einer kompakten Vorstellung von Java und insbesondere einer Übersicht zu JavaFX, seiner Historie, der aktuellen Architektur und den Zielen. Ebenso besprechen wir in diesem Paragrafen, was Sie als Voraussetzungen für den Umgang mit JavaFX benötigen. Das umfasst die Vorkenntnisse, die Hardware und das Betriebssystem sowie die Java- und JavaFX-Basisumgebung samt erweiterter Tools wie NetBeans und dem Scene Builder. Desgleichen betrachten wir die Installationen der Tools, denn gerade mit NetBeans und dem Scene Builder sowie selbstverständlich der Java-Basis wollen wir im Buch intensiv arbeiten.

Ebenso erstellen wir in dem Kapitel bereits die ersten JavaFX-Applikationen, und zwar auf mehrere Weisen: sowohl mit NetBeans auf Basis von Java und reinem JavaFX, aber auch mit dem Scene Builder sowie FXML. In dem Buch möchte ich – wie schon erwähnt – überwiegend mit NetBeans als Referenz-IDE (IDE – Integrated Development Environment) arbeiten. Man kann auch andere Entwicklungsumgebungen verwenden. So wird in diesem Kapitel ebenso erklärt, wie man etwa mit Eclipse eine JavaFX-Applikation erstellt.

Und es gibt zu Beginn noch mehr zu tun. Zwar möchte ich Ihren Einstieg in JavaFX nicht mit zu viel Theorie belasten, aber einige grundlegende Details zu JavaFX-Applikationen benötigen wir unbedingt schon am Anfang. Sonst wird der eigentliche Umgang mit JavaFX in der Folge nicht auf soliden Füßen stehen. Solche ersten grundlegenden Informationen und Hintergründe zu JavaFX-Applikationen erhalten Sie ebenfalls in diesem Kapitel, und zwar gekoppelt mit den ersten Beispielen, damit Sie die Informationen direkt im praktischen Zusammenhang sehen.

Das Deployment beschließt das Kapitel. Das bedeutet, dass das grundsätzliche Verteilen beziehungsweise Aufliefern von JavaFX-Applikationen behandelt wird. Das umfasst

R. Steyer, *Einführung in JavaFX,*
DOI 10.1007/978-3-658-02836-7_1, © Springer Fachmedien Wiesbaden 2014

im Wesentlichen die Technik des **Java Web Starts** und das Erzeugen von ausführbaren **JAR-Dateien** (JAR – Java Archive).

- Im Buch wird an einigen Stellen davon ausgegangen, dass Sie über (geringe) Grundlagen in Java und objektorientierter Programmierung (OOP) verfügen. Diese Grundkenntnisse sind immer dann notwendig, wenn wir mit Java-Quellcode in Berührung kommen. Im letzten Kapitel finden Sie einen Java-Crashkurs, der zwar kein vollständiges Java-Buch beziehungsweise eine allumfassende Einführung in Java sein kann und soll, aber dennoch die wichtigsten Grundlagen zu Java vermittelt.

1.2 Die Welt von Java und JavaFX

Machen wir uns nun auf den Weg in die Welt von JavaFX. JavaFX ist offensichtlich ein Teil der Java-Plattform, die bereits seit vielen Jahren etabliert ist. Wer JavaFX verstehen will oder sich allgemein damit beschäftigt, sollte zumindest Java selbst in Grundzügen kennen. Deshalb werfen wir zunächst einen kleinen Blick auf dessen Struktur und auf die Geschichte von Java im Gesamten.

1.2.1 Was ist Java?

Java ist einerseits eine **Programmiersprache**, bezeichnet andererseits aber auch eine ganze **Plattform** zur Ausführung von stabilen, sicheren und leistungsfähigen Programmen unabhängig vom zugrunde liegenden Betriebssystem. Verallgemeinernd spricht man von der **Java Technology**. Darunter versteht man eine ganze Sammlung von Spezifikationen, die einerseits eine Programmiersprache und andererseits verschiedene Laufzeitumgebungen samt umfangreichen Bibliotheken für Computerprogramme definieren. Die gesamte Spezifikation der Java-Technologie umfasst folgende Bestandteile:

- Die eigentliche Programmiersprache Java,
- die Java-Plattform – eine standardisierte Software-Plattform,
- das grundlegende Entwicklungswerkzeug (Java Development Kit – JDK) samt darauf aufbauender Erweiterungen und
- die Java-Laufzeitumgebung JRE (Java Runtime Environment), um Java-Programme ausführen zu können.

Der Java-Erfinder Sun hat schon zu Beginn Java als Sprache so charakterisiert:

- **Java** eine einfache, objektorientierte, dezentrale, interpretierte, stabil laufende, sichere, architekturneutrale, portierbare und dynamische Sprache, die Hochgeschwindigkeits-Anwendungen und Multithreading unterstützt.

In dieser (technischen) Marketingaussage stecken bereits die wichtigsten Informationen, was genau Java als Sprache, aber auch als Plattform auszeichnet. Etwa die Tatsache, dass Java eine objektorientierte Sprache ist, mit der Anwendungen erstellt werden können, die auf verschiedenen Systemen mit unterschiedlichen Prozessoren und Betriebssystemarchitekturen lauffähig sind. Die fertigen Programme können auf jedem System ausgeführt werden, das die JRE samt der so genannten virtuellen Maschine (Java Virtual Machine beziehungsweise JVM oder kurz VM – ein virtueller Prozessor) von Java implementiert.

Doch wie ist das möglich? Java gilt als interpretiert und kompiliert zur gleichen Zeit, was im Grunde einen Widerspruch darstellt. Beide Vorgänge (Interpretation und Kompilierung) beschreiben den Vorgang der Übersetzung von einem Quelltext in einen lauffähigen Binärcode, der von einem Computer ausgeführt werden kann. Dies kann man auf zwei Arten machen. Entweder, der Quelltext wird auf einen Schlag mit einem geeigneten Programm übersetzt und dann dieser daraus resultierende Binärcode auf den Computer zum Laufen gebracht. Das bedeutet dann, dass der Quelltext kompiliert wurde. Man kann aber auch mit einem anderen Programmtyp den Quelltext laden und Zeile für Zeile lesen und direkt zur Laufzeit des Programms übersetzen lassen. Das ist dann der Vorgang der Interpretation. Java macht beides.

Der eigentliche Quellcode wird in einen binären Zwischencode (so genannten **Bytecode**) kompiliert, der ein architekturneutrales und noch nicht vollständiges Object-Code-Format ist. Er ist noch nicht lauffähig (also noch nicht zu Ende übersetzt) und muss von einer **Laufzeitumgebung** interpretiert[1] werden. Dies ist die oben bereits genannte JRE, deren wesentlicher Bestandteil die JVM ist. Da jede Java-Laufzeitumgebung plattformspezifisch ist, arbeitet das endgültige ausgeführte Programm auf dieser spezifischen Plattform. Dort werden alle Elemente hinzugebunden, die für eine spezielle Plattform notwendig sind. Die Tatsache, dass der letzte Teil der Übersetzung des Bytecodes von einem plattformspezifischen Programm auf der Plattform des Endanwenders ausgeführt wird, nimmt dem Entwickler die Verantwortung, verschiedene Programme für verschiedene Plattformen erstellen zu müssen. Die Interpretation erlaubt zudem, Daten zur Laufzeit zu laden, was eine wichtige Grundlage für das dynamische Verhalten von Java ist.

1.2.2 Etwas zur Historie von Java

Derzeit ist Java in den Versionen 7 beziehungsweise bald 8 verfügbar. Wichtig: Bei beiden Versionen ist JavaFX integraler Bestandteil. Betrachten wir kurz den Weg hin zu diesem Punkt.

Hinter Java stand ursprünglich die amerikanische Firma Sun Microsystems (http://www.oracle.com/us/sun/), wobei diese 2010 von Oracle (http://www.oracle.com/) übernommen wurde. Die Geschichte von Java selbst geht bis ins Jahr 1990 zurück. Zu diesem Zeitpunkt versuchte Sun, im Rahmen eines Projekts mit dem Namen **Green** den zukünftigen Bedarf an EDV zu analysieren, um einen zukunftsträchtigen Markt zu lokalisieren.

[1] Dann erst entsteht der plattformspezifische Maschinencode.

Haupterkenntnis des Green-Projektes war, dass die Computerzukunft weder im Bereich der Großrechner (einem Geschäftsschwerpunkt von Sun) noch bei PCs oder Kleincomputern in der damals aktuellen Form zu sehen war. Der Konsumentenbereich der allgemeinen Elektronik (Telefone, Videorekorder, Waschmaschinen, Kaffeemaschinen und eigentlich alle elektrischen Maschinen, die Daten benötigen) wurde als der (!) Zukunftsmarkt der EDV prognostiziert – ein extrem heterogenes Umfeld mit den unterschiedlichsten Prozessoren beziehungsweise grundverschiedenen Kombinationen von Hard- und Software-Komponenten. Dafür eine gemeinsame Plattform zu schaffen und damit frühzeitig einen Standard festzulegen, wurde von Sun als die Zukunftschance der EDV schlechthin vorausgesagt.

Wichtigste Forderungen an eine solche auf allen denkbaren Systemen lauffähige Plattform waren eine erhebliche Fehlertoleranz, eine leichte Bedienbarkeit und eine bedeutend bessere Stabilität als diejenige, die bei allen bis dahin vorhandenen Plattformen existierte. Die Plattform musste deshalb ein neues Betriebssystem, oder zumindest eine neue Betriebssystemergänzung, für alle populären Betriebssysteme bereitstellen. Ebenso sollte möglichst eine neue Programmiersprache entwickelt werden, denn alle bis dahin vorhandenen Programmiersprachen wiesen zu große Schwächen hinsichtlich der Stabilität auf. Gerade bei Bedienungsfehlern waren damalige Techniken und Programme einfach zu intolerant.

Ab dem Frühjahr 1991 gingen die Planungen in die Generierung eines Prototyps für eine solche universale Plattform über, der **Oak** (Eiche) genannt wurde. Dem Gerücht nach geht der Name auf eine imposante Eiche vor den Sun-Büros zurück. Eine alternative Erklärung für Oak ist, dass es die Abkürzung für *Object Application Kernel* war. 1992 präsentierte das Green-Team mit **Duke**, einer kleinen Trickfigur in einem virtuellen Haus, das erste Ergebnis. Duke gilt auch heute noch als das Java-Maskottchen.

Und obwohl Duke überzeugte und in der Folge im Rahmen eines Sun-Tochterunternehmen namens **First Person** vorangetrieben wurde, konnte aus dem Projekt keine konkrete Anwendung etabliert werden. Einige Zeit später wurde jedoch das Internet und insbesondere das WWW als Zielplattform für ein weiterentwickeltes Oak erkannt, das zudem in ***Java*** umbenannt wurde. Der Name Java steht in Amerika (eigentlich altenglisch) für eine bestimmte Sorte Kaffee. Deshalb ist auch das Logo von Java eine dampfende Kaffeetasse.

1995 präsentierte Sun das für das Internet aufbereitete und optimierte Java auf Basis der Java-Applets. Dazu wurde das zugehörige kostenlose und frei zu verwendende Paket von Entwicklungs-Tools (**JDK** – Java Development Kit) in der Version 1.0 vorgestellt. Im Laufe der Zeit gab es diverse Entwicklungstools für Java mit integrierten Programmiertechniken, die aber meist Aufsätze auf dem JDK darstellten.

Als Plattform der Java-Applets lieferte Sun mit **HotJava** gleich auch eine erste komplexe und vollständig in Java geschriebene Anwendung, die auf allen Betriebssystemen lauffähig war, für die eine virtuelle Maschine verfügbar war. Mit anderen Worten: HotJava war einerseits der erste javafähige Browser und demonstrierte auf der anderen Seite, dass auch außerhalb von Webseiten Programme mit Java laufen konnten.

Natürlich wurde Java über die Jahre erheblich weiterentwickelt. Im Jahr 2011 kam Java 7 heraus und Java 8 soll nach der derzeitigen Roadmap 2014 erscheinen. Insbesondere

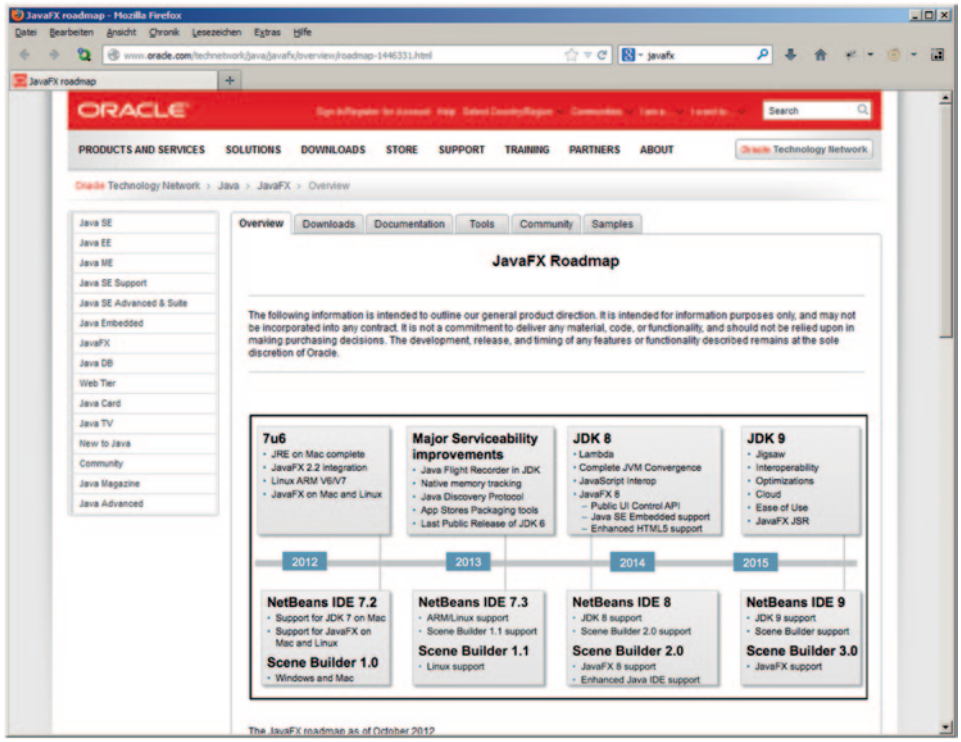

Abb. 1.1 Die Architektur, wie sie Oracle offiziell darstellt

ist Java mittlerweile eng mit JavaFX verzahnt, weshalb es eine einheitliche Roadmap gibt (Abb. 1.1).

Die JavaFX 2.x-Laufzeitumgebung wird ab der Version Java SE Runtime 7 Update 6 mitinstalliert, kann aber auch separat heruntergeladen und installiert werden.

1.2.3 Wo findet Java heutzutage Anwendung?

Wie erwähnt, waren Applets am Anfang der (Erfolgs-)Geschichte von Java die populärsten Java-Anwendungen. Aber heutzutage findet man im Web nur noch sehr wenige Seiten, die Java-Applets einsetzen. Dafür gibt es diverse Gründe:

- Schlechte Performance und Ressourcenhunger der ersten Applets beziehungsweise von Java selbst.
- Relativ schwache Hardware bei Anwendern in den prägenden Jahren von Java im Web (90er-Jahre).
- Zu langsame Internet-Verbindungen bei Privatanwendern – ebenfalls in den entscheidenden Jahren.
- Permanente Konflikte zwischen Sun und Microsoft samt der mangelhaften Unterstützung von Java im Internet Explorer in den 90iger-Jahren.

- Allgemeiner Rückgang der clientseitigen Web-Programmierung um das Jahr 2000.
- Zwingend notwendige Installation einer passenden JVM bei einem Anwender.

Doch Java hat neue Einsatzgebiete gefunden und sich dort etabliert. Java kommt heutzutage zum Beispiel in folgenden Bereichen zum Einsatz:

- Erstellung von plattformunabhängigen Desktopanwendungen.
- Serverseitige Dienste im Internet beziehungsweise Intranet, etwa in Form eines Java Application Servers (z. B. Apache Tomcat, JBoss, GlassFish, usw.) oder rein von der Programmierseite mit Hilfe von Java-Servlets, Java Server Pages (JSP), Java Server Faces (JSF) und spezieller Web-Frameworks.
- Anwendungen auf Chipkarten – sogenannten Java Cards – und eingebetteten Systemen, für die Java ja auch ganz am Anfang entwickelt wurde.
- Gerade unter Android werden Apps für Smartphones und Tablets überwiegend in Java geschrieben.
- Die Clientplattform im Web ist immer noch ein Einsatzgebiet von Java. Auch wenn Applets wie erwähnt mittlerweile selten geworden sind. Gerade mit JavaFX und Java Web Start soll Java die Basis von RIAs (Rich Internet Applications) werden. JavaFX integriert auch ein Browser Plug-in, um JavaFX-Applets schneller laden zu können sowie einen Preloader für JavaFX-Applets. Und auch Java-Frameworks wie das Google Web Toolkit (GWT), mit dem zur Designzeit in Java entwickelt und erst das endgültige Resultat für die Auslieferung in HTML, JavaScript und CSS übersetzt wird, zielen explizit auf das Web.

1.2.4 Die Java-Plattform

Betrachten wir Java noch einmal unter dem Gesichtspunkt der Plattform. Unter der Java-Plattform versteht man die Java-Laufzeitumgebung (JRE) und Java-Programmierschnittstellen (das Java-API – Application Programming Interface – Schnittstelle zur Anwendungsprogrammierung). Natürlich werden Programme für diese Java-Plattform in der Regel in der Programmiersprache Java erstellt. Allerdings kann man auch andere Programmiersprachen verwenden, wenn diese die Spezifikationen der Java-Plattform erfüllen. Beispiele sind Nice oder Groovy.

Es gibt nun verschiedene Versionen der Java-Plattform, die unterschiedliche Ausrichtungen gestatten:

- Die **Java Platform Java Card** ist darauf optimiert, Java-Card-Applets auf Chipkarten auszuführen. Die Java-Card-Applets basieren auf Java-Applets für das Internet, wurden aber im Leistungsumfang reduziert, schlanker und vor allen Dingen für dieses spezielle Umfeld angepasst.
- Die **Java Platform Micro Edition** (kurz Java ME) ist eine Plattform für die eingebetteten Systeme. Das umfasst auch Smartphones.

- Die **Java Platform Standard Edition** (kurz Java SE) ist die Kerntechnologie der Java-Plattform. Sie umfasst grundlegende APIs für den Einsatz auf Computern. Auf der Java SE bauen unmittelbar die Java-EE- und Java-ME-Technologien auf. Sie wird auch die Basis von unserem Buch sein.
- Die **Java Platform Enterprise Edition** (kurz Java EE) ist eine Erweiterung der Java SE. Die Erweiterungen umfassen die Unterstützung von transaktionsorientierten, komplexen Business-Anwendungen sowie Web-Anwendungen.

1.3 Was ist JavaFX?

JavaFX ist eine Erweiterung von Java, die erstmals auf der JavaOne-Konferenz von Sun im Jahr 2007 vorgestellt wurde, und zwar als neue Skripting-Plattform für Web- und Desktop-Applikationen sowie mobile Anwendungen. Ende 2007 beziehungsweise Anfang 2008 erschienen erste offizielle Vorversionen und um den Jahreswechsel 2008 auf 2009 wurde die erste Finalversion festgeschrieben. Eine erste große Referenzapplikation war eine RIA (Rich Internet Application – eine Webseite mit erweiterten interaktiven Möglichkeiten) zu den Olympischen Winterspielen in Vancouver, die es aber auch parallel auf Basis von reinen Webtechnologien gab, und die auch mittlerweile im Web nicht mehr gehostet wird.

Dabei stellte in den ersten Versionen eine integrierte Skriptsprache mit Namen **JavaFX Script** als zentraler Kern der gesamten Technologie die Mittel zur Verfügung, um allgemeine visuelle, hochleistungsfähige Anwendungen auf Basis von Java zu erzeugen. Obwohl auch ganz einfache Applikationen mit Konsolenausgabe erstellt werden können, wurde JavaFX Script von Anfang an speziell dafür designed, um den kreativen Prozess der Erstellung von Benutzeroberflächen zu optimieren. Wie Sun ausdrücklich betont hatte, wollte man mit JavaFX Script neben den klassischen Java-Programmierern vor allen Dingen die so genannten »creative minds« erreichen, die nicht unbedingt Java-Kenntnisse und möglicherweise sogar insgesamt wenig Programmiererfahrung haben. Dieser Ansatz mit JavaFX Script kann jedoch als gescheitert angesehen werden und die Skriptsprache wird in JavaFX ab der Version 2.0 nicht mehr zur Verfügung gestellt.

► Quellcode, der mit JavaFX Script erzeugt wurde, kann in neuen Versionen von JavaFX nicht mehr verwendet werden! Insgesamt sind Quellcodes für JavaFX vor der Version 2.0 gar nicht oder nur mit sehr viel Mühe auf die neuen Versionen zu aktualisieren. Grundsätzlich ist es meines Erachtens meist leichter, die Programme vollkommen neu zu erstellen und dabei gegebenfalls nur einzelne Elemente wie reine Java-Klassen zu übernehmen, sofern dies möglich ist.

Insgesamt muss man zugeben, dass die ersten Versionen von JavaFX nicht den Erfolg hatten, den Sun und Oracle wohl erwartet hatten. Nicht zuletzt deshalb wurde JavaFX für die Version 2.0 technisch vollkommen verändert. JavaFX bezeichnet ab der Version 2.0 ein Framework von Oracle zur Erstellung von modernen, plattformübergreifenden Java-Applikationen beziehungsweise Rich Internet Applications, das eben mit den ersten Versio-

nen fast nur noch den Namen gemeinsam hat. Mit der Version 2 hat JavaFX jedoch einen festen, modernen und ausgereiften Stand erreicht und die neuen Versionen von JavaFX werden als die Zukunft der Entwicklung von grafischen Java-Oberflächen angesehen. JavaFX soll sogar Swing ablösen, denn es bietet mehr Möglichkeiten (neue Komponenten und Widgets) und ist gleichzeitig einfacher in der Anwendung. Oracle selbst spricht bei dem neuen JavaFX von einer Evolution der Java Rich Client Platform mit einer leichtgewichtigen, hardwarebeschleunigten Benutzerschnittstelle, die ohne eine Anwendung unterschiedlicher Technologien auskommt. JavaFX kann auf diversen Endgeräten wie Mobilfunkgeräten, Set-Top-Boxen, Desktop-Computern und Multimediageräten zum Einsatz kommen.

1.3.1 Die Architektur

Betrachten wir die Architektur von JavaFX etwas genauer.

- Bei der JavaFX Architektur (Abb. 1.2) befindet sich ganz unten – wie bei jeder Java-Applikation – die JVM. Darauf setzen diverse Erweiterungen auf, die man teils schon seit Jahren bei Java nutzen kann, und die auch JavaFX zur Verfügung stehen, wie etwa **Java 2D** zum Zeichnen von zweidimensionalen Formen.
- Mit dem **Glass Windowing Toolkit** bekommen Sie Zugriff auf native Betriebsleistungen wie die Fensterverwaltung, Timer oder Ereignisverwaltung. Dabei handelt es sich explizit um eine **plattformabhängige** Schicht.
- Bei **Prism** handelt es sich um eine Grafik-Pipeline[2], die auf Hardware- und Software-Renderern ausgeführt werden kann. Unter anderem werden damit auch Features von Java 2D und diverse grafische Effekte wie Schatten, Spiegelungen, Transformationen, Animationen, etc. auf einer hohen Ebene verfügbar gemacht.
- Das **Media**-Framework beziehungsweise die **Media Engine** basiert auf GStreamer und bietet – vereinfacht gesagt – sehr umfangreiche Unterstützung für Audio und Video.
- Die **Web Engine** beziehungsweise **WebView**-Komponente erlaubt die Einbettung von Web-Inhalten in JavaFX-Applikationen. Das umfasst das Rendern von HTML auf Basis der Webkit-Engine[3] sowie das hardwareabhängige Rendern über Prism. Besonders interessant ist die Möglichkeit des DOM-Zugriffs (DOM – Document Object Model) und der Manipulation des DOM.
- Das **Quantum Toolkit** fügt nun Prism und das Glass Windowing Toolkit sowie die Engine für das Web und für Multimediaverarbeitung zusammen und macht diese einheitlich den JavaFX APIs zugänglich.

[2] Gewisse andere Verwendungen in Hinsicht auf Datenspionage sind eine etwas unglückliche Mehrdeutigkeit.

[3] Diese ist die Basis von Chrome oder Safari.

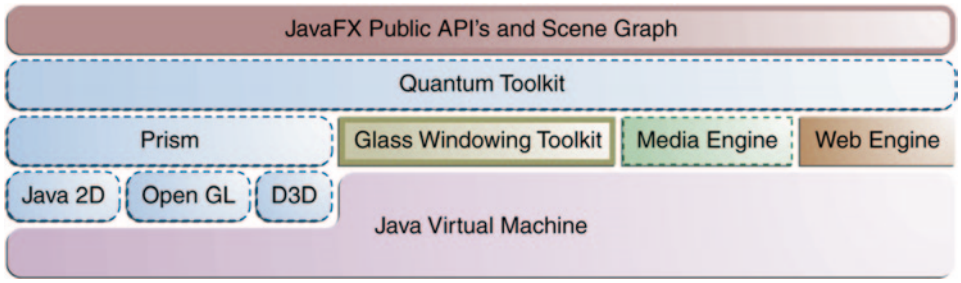

Abb. 1.2 Die Architektur, wie Sie Oracle offiziell darstellt

Insbesondere braucht man aber als Entwickler bei JavaFX im Grunde gar nicht wissen, wie die tiefliegenden Ebenen arbeiten. Das JavaFX API und das Quantum Toolkit kapseln diese Details und verbergen diese damit vor dem Anwender. Dabei gestatten die Java APIs für JavaFX eine End-to-end-Java-Entwicklung mit allen bekannten Java-Möglichkeiten bis hin zu Generics oder Multithreading. Neue UI-Komponenten lassen sich mit JavaFX schnell und einfach erstellen und alle UI-Komponenten lassen sich per CSS gestalten. Auch ist bei JavaFX explizit eine Zusammenarbeit zwischen Swing und JavaFX vorgesehen. So können JavaFX-Bestandteile in vorhandene Swing-Applikationen eingefügt werden. Ebenso gibt es diverse Enterprise-Möglichkeiten bei JavaFX wie die Bindung an Datensevices, Multitouch-Operationen, den Aufbau von Serverarchitekturen und die Kommunikation zwischen einem Webserver und dem Client (mit HTTP-GET, REST oder Webservices) oder der HTML5-Integration (inklusive einem Canvas-API).

1.3.2 JavaFX ohne Java programmieren zu müssen

Obwohl JavaFX Script in neuen Versionen von JavaFX verschwunden ist, wurde die grundsätzliche Intention bei JavaFX, dass die Erstellung von grafischen Oberflächen für Java-Applikationen vereinfacht werden sollte[4] und insbesondere Java-Kenntnisse keine zwingende Voraussetzung mehr darstellen sollten, nicht aufgegeben. Ganz im Gegenteil: Gerade für Entwickler ohne vertiefende Java-Grundlagen stellt JavaFX nun statt JavaFX Script mit **FXML** eine einfach zu lernende, deklarative Sprache zur Verfügung, die eine alternative Definition von grafischen Oberflächen rein über XML erlaubt[5]. Zudem können Sie dort auch Web-Technologien wie CSS (Cascading Style Sheets) oder JavaScript (aber auch andere Sprachen wie Groovy) einsetzen[6], die in den XML-Code[7] eingebettet oder damit ver-

[4] Das offizielle Stichwort lautet: Fokus auf die Fähigkeiten anstelle der Technologie.

[5] So etwas kennt man beispielsweise aus der Programmierung unter dem.NET-Framework von Microsoft in Form von XAML (Extensible Application Markup Language) oder auch der Android-Programmierung.

[6] Was aber auch auf der Java-Ebene von JavaFX geht.

[7] Aber auch Java-Code.

knüpft werden können. Das schafft ganz neuen Gruppen Zugang zur Java-Welt, die bisher ohne entsprechenden Java- und OO-Background ausgegrenzt waren.

Und dieser Ansatz geht mit einem GUI-Designer mit visuellen Controls zur Erstellung von Oberflächen einher – dem **Scene Builder**. Dieser erlaubt sogar die Erstellung einer GUI per Drag & Drop beziehungsweise rein visuell.

Und wie bei den Architekturbetrachtungen oben gesehen, arbeitet JavaFX nahtlos mit reinem Java zusammen und fügt sich harmonisch in die gesamte Java-Architektur ein, was eine große Flexibilität zur Folge hat. Oder anders ausgedrückt: Egal welche Art von JavaFX-Applikationen Sie letztendlich erzeugen, JavaFX-Applikationen laufen wie alle Java-Applikationen auf sämtlichen Plattformen, die eine passende virtuelle Java-Maschine bereitstellen. Auch für JavaFX-Applikationen gelten die Java-typischen Kriterien „write-once-run-anywhere", das Sicherheitsmodel für Applikationen, die einheitliche Distribution und die Enterprise-Connectivity.

1.4 Was benötigen Sie?

Schaffen wir uns jetzt einen Überblick über die Voraussetzungen, die bei Ihnen zur Arbeit mit JavaFX erfüllt sein müssen.

1.4.1 Die Hardware und das Betriebssystem

Sie benötigen selbstverständlich zu einer erfolgreichen Arbeit mit dem Buch und natürlich zur realen Programmierung erst einmal einen Computer (in der Regel wird das ein PC oder Apple sein), der eine ausreichende Leistungstärke haben sollte. Wir arbeiten mit JavaFX auf Basis einer Java-Umgebung und die fordert einigermaßen Power von dem ausführenden Rechner. Zusätzlich werden Sie unter Umständen Entwicklungsprogramme verwenden, die generell einige Ressourcen erfordern. Moderne Rechner sollten allerdings solche Voraussetzungen grundsätzlich erfüllen. Und Sie sollten auf jeden Fall einen (möglichst schnellen) Zugang zum Internet haben. Das betrachte ich als selbstverständlich. Auch ein lokaler Webserver oder Zugang zu einem Webserver im Internet ist sinnvoll, wenn wir an das Verteilen von JavaFX-Applikationen kommen.

Die Frage, welches Betriebssystem Sie haben sollten, ist nicht unwesentlich, aber sehr einfach zu beantworten. Sie benötigen ein Betriebssystem, für das es das JDK mit JavaFX-Unterstützung gibt. Unter dem Link http://www.oracle.com/technetwork/java/javafx/downloads/index.html (Abb. 1.3) wird ein Link *Certified System Configurations* angezeigt[8].

Wenn Sie den Link auswählen, erhalten Sie auf der folgenden Webseite eine Auflistung aller Betriebssysteme, mit denen Sie arbeiten können (Abb. 1.4).

[8] Beachten Sie, dass sich der Aufbau der Webseite jederzeit ändern und diese Struktur nicht garantiert werden kann. Aber Oracle wird ziemlich sicher irgendwo auf der Webseite diese Information bereitstellen.

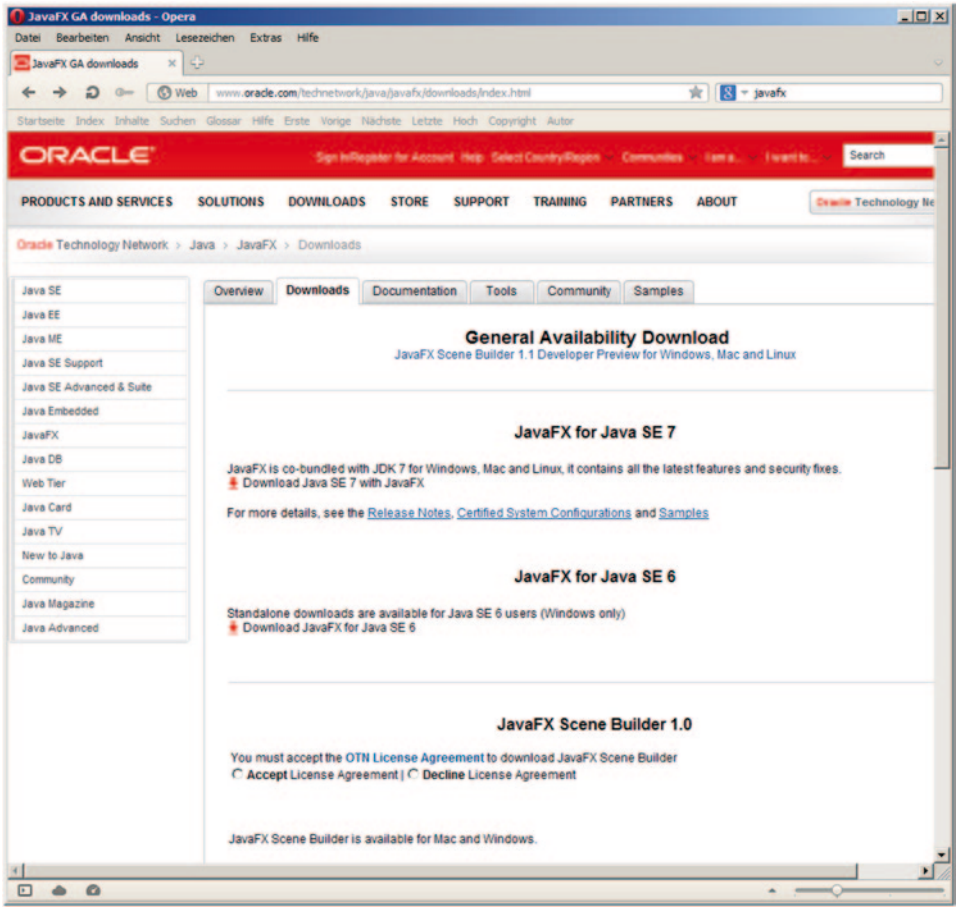

Abb. 1.3 Hinweise zum Download der Tools für die JavaFX-Entwicklung

Im Wesentlichen sind das verschiedene Versionen von Windows (32 und 64 Bit) und Linux (32 und 64 Bit) sowie MacOS X mit 64 Bit, wobei wir uns im Buch auf Windows als Referenzsystem konzentrieren werden[9]. Solaris[10] wird dort zwar nicht erwähnt, ist aber als Plattform durchaus verwendbar.

1.4.2 Die Java-Basisumgebung

Nun wird es spannend. Wir bewegen uns mit JavaFX natürlich im Java-Umfeld. Java-Applikationen benötigen eine spezielle Laufzeitumgebung, damit sie ausgeführt werden

[9] Das spielt aber keine wichtige Rolle – das ist ja gerade das Wesen von Java.

[10] Das Unix-Betriebssystem von Sun beziehungsweise Oracle im Wesentlichen für Mainframes.

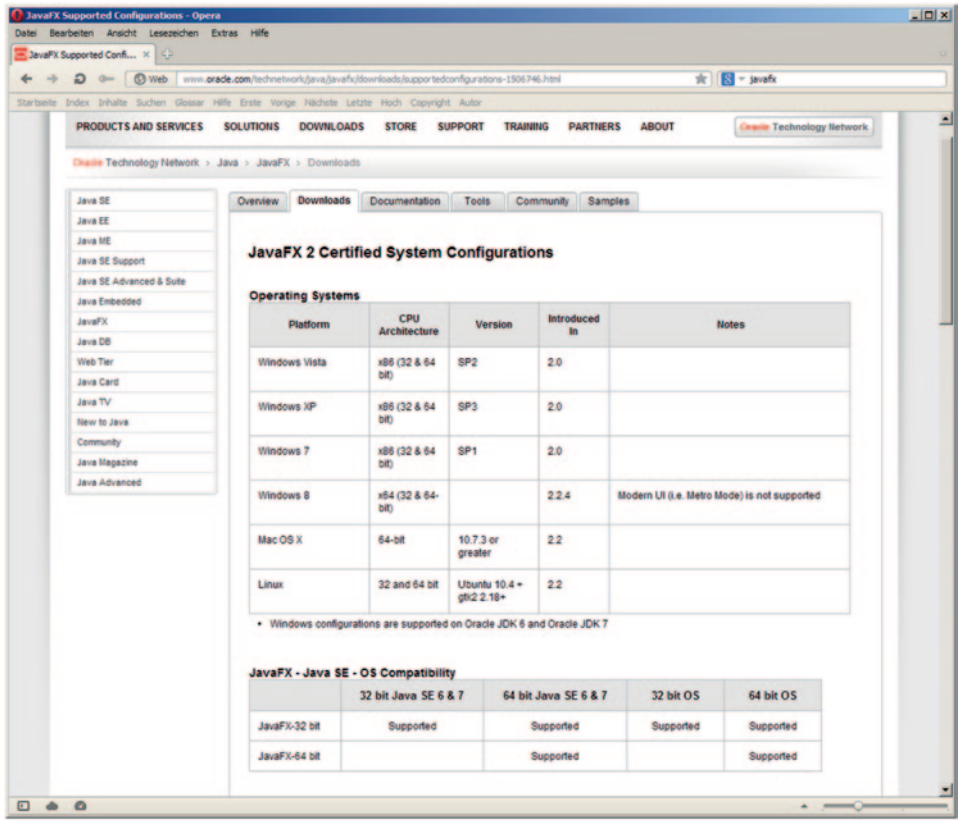

Abb. 1.4 Die unterstützten Betriebssysteme

können. Und das bedeutet, dass Sie auf Ihrem Computer auf jeden Fall eine Java Laufzeitumgebung (JRE) benötigen. Genauso wie jeder Anwender, der eine Java- beziehungsweise JavaFX-Applikation ausführen möchte.

Eine solche Java-Laufzeitumgebung gibt es mittlerweile für fast jedes moderne PC-Betriebssystem und auch für die meisten mobile Geräte wie Handys oder andere technische Geräte des täglichen Lebens. Bei vielen Betriebssystemen wird sie sogar bereits in der Grundausstattung zur Verfügung gestellt.

Möglicherweise haben Sie aber auf Ihrem Entwicklerrechner nun nicht die passende Version installiert. Und auch aus anderen Gründen müssen wir auf die JRE und die genaue Version noch etwas detaillierter eingehen.

Wir wollen ja nicht nur Java- beziehungsweise JavaFX-Applikationen ausführen, sondern selbstverständlich ebenso erstellen. Und dazu benötigen wir mehr als nur eine Laufzeitumgebung.

Wir brauchen entsprechende Entwicklungstools für Java respektive JavaFX. Im Kern bedeutet das, dass wir erst einmal minimal ein aktuelles Java Entwicklungspaket (das Java SE Development Kit oder nur Java Development Kit – abgekürzt JDK) benötigen. Und das müssen Sie sich erst einmal besorgen. Oben wurde bereits der entsprechende URL http://

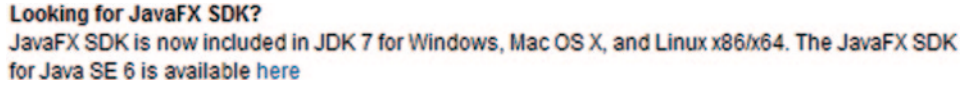

Abb. 1.5 JavaFX ist integraler Bestand des JDK 7

www.oracle.com/technetwork/java/javafx/downloads/index.html zum Download eines passenden JDK erwähnt (Abb. 1.4). In der derzeit aktuellen Version JDK ist JavaFX-Unterstützung integraler Bestandteil und Sie benötigen keinen separaten Download des JavaFX SDK mehr, wie es etwa bis zum JDK 6 noch notwendig war (Abb. 1.5).

Klicken Sie die entsprechenden Links auf den Java- beziehungsweise JavaFX-Seiten von Oracle an und dann können Sie das passende JDK für Ihr Betriebssystem auswählen (Abb. 1.6) und auf Ihren Rechner speichern.

Sie müssen lediglich vor einem Download der meisten Features die Lizenzbedingungen für die Verwendung akzeptieren – eine Registrierung oder etwas Ähnliches ist meistens nicht notwendig.

Die Installation der Java-Umgebung respektive des JDK ist absolut unproblematisch. Sie werden bei allen neueren Java-Versionen von einem typischen Installationsassistenten für Ihr Betriebssystem geführt. Dieser richtet alle notwendigen Einstellungen in Ihrem Betriebssystem ein. Anschließend sind auf Ihrem Rechner die passenden Verzeichnisse vorhanden und die Systemeinstellungen zum Ausführen von Java-Applikationen und den JDK-Tools angepasst. Mehr muss man zur Installation definitiv nicht mehr sagen. Wenn Sie das JDK installiert haben, haben Sie damit mehrere Fliegen mit einer Klappe geschlagen, denn das JDK enthält neben den eigentlichen Entwicklungstools eine vollständige Java-Laufzeitumgebung.

Das JDK selbst besteht aus einer ganzen Reihe von Programmen, die sich nach der Installation im Unterverzeichnis *bin* des JDK-Installationsverzeichnisses befinden. Dort können Sie sie über die Befehlszeilenebene mit eventuell benötigten Parametern aufrufen. Wenn Sie ein Programm des JDK ohne Parameter aufrufen, erhalten Sie in der Konsole Hinweise zu den optionalen und ebenso zu den zwingenden Parametern. Zu den Basisprogrammen des JDK im engeren Sinn zählen der Compiler (*javac*), der Interpreter (*java* beziehungsweise *javaw*), der Appletviewer, ein Debugger (*jdb*), das Dokumentations-Tool *javadoc* und das Java Archive Tool mit Namen *jar*.

1.4.3 Integrierte Entwicklungsumgebungen für JavaFX

Prinzipiell ist es möglich, dass Sie mithilfe eines reinen Texteditors und den Tools des JDK beziehungsweise JavaFX SDK JavaFX-Applikationen erstellen und ausführen.[11] Doch ist dieser Weg nicht sonderlich bequem und in der Praxis viel zu aufwendig, langsam und fehlerträchtig. Wenn man wirklich effektiv mit JavaFX in der Praxis programmieren möchte, führt kein Weg um eine vollständig integrierte Entwicklungsumgebung herum.

[11] Das werden wir im Buch auch einmal durchspielen.

Abb. 1.6 Auswahl des passenden JDK

- Selbstverständlich können Sie die gleichen JavaFX-Dateien mit verschiedenen Editoren beziehungsweise IDEs verarbeiten. Das werden wir in diesem Kapitel auch demonstrieren.

1.4.3.1 NetBeans

Eine IDE, mit der Sie JavaFX-Applikationen erstellen und ausführen können, ist **NetBeans**. Oder man kann es etwas schärfer formulieren: Derzeit ist NetBeans meines Erachtens die ideale IDE für die Erstellung von JavaFX-Applikationen. Warum das? Es handelt sich hierbei um die in Java geschriebene eigene IDE des Erfinders von Java und JavaFX, nämlich Oracle beziehungsweise das ehemalige Sun. Damit ist NetBeans nicht nur eine hervorragende Java-IDE, sondern die allgemeine Unterstützung und nahtlose Integration von JavaFX ist in diesem Tool sehr gut.

NetBeans wird Ihnen wie das JDK von Oracle in verschiedenen Versionen und für verschiedene Betriebssysteme kostenlos zum Download zur Verfügung gestellt. Diese finden

Sie an der gleichen Stelle, von wo Sie auch das JDK beziehungsweise JavaFX SDK laden können. Genau genommen bündelt Oracle NetBeans mit dem JDK (Abb. 1.7), so dass Sie eigentlich nur das NetBeans-Bundle laden und installieren müssen. Sollte ein JDK schon vorhanden sein, schadet eine nachträgliche Installation des Bundles aber nicht.

Die Installation von NetBeans ist vollkommen unproblematisch – es führt ein klassischer Installationsassistent zum Erfolg.

1.4.3.2 Eine NetBeans-Alternative – Eclipse

Statt NetBeans bietet sich auch **Eclipse** zur Entwicklung von JavaFX-Anwendungen an. Bei Eclipse handelt es sich um eine Open-Source-IDE zur Programmierung, das ursprünglich jedoch von einer Firma (IBM) entwickelt und erst später freigegeben wurde. Eclipse gibt es für verschiedene Betriebssysteme und ist extrem mächtig und leistungsfähig. Obwohl Eclipse selbst in Java geschrieben und im Schwerpunkt auf die Entwicklung mit Java ausgerichtet ist, können in Eclipse mithilfe unzähliger PlugIns nahezu alle Programmiertechniken abgedeckt werden. Eclipse finden Sie beispielsweise unter http://www.eclipse.org kostenlos zum Download. Wählen Sie bei Bedarf dort die passende Version für Ihr Betriebssystem aus. Die Installation von Eclipse ist vollkommen unproblematisch, wenn auf einem Rechner bereits ein neueres JDK installiert ist. Eclipse wird als komprimierte Datei ausgeliefert, die Sie einfach extrahieren müssen. Beim ersten Start sucht sich die IDE die passende Java-Umgebung und richtet das System weitgehend automatisch ein.

- Früher benötigten Sie in Eclipse für JavaFX ein zusätzliches Plugin, aber da JavaFX mittlerweile in das JDK integriert ist, ist das nicht mehr notwendig. Es kann nur notwendig sein, dass Sie die Datei *jfxrt.jar* aus dem *jre/lib*-Verzeichnis explizit in Ihrem Projekt bekanntmachen.

1.4.4 Der Scene Builder

Oracle stellt mit dem **Scene Builder** ein grafisches Tool zum Aufbau einer Oberfläche zur Verfügung, das einem Anwender visuell die XML-Strukturen von FXML generieren lässt. Wenn es gewünscht wird, kann man damit gänzlich ohne Java-Kenntnisse und sogar ohne XML-Kenntnisse eine vollständige GUI einer JavaFX-Applikation erstellen, die sogar mit echter Funktionalität (bis zu einem gewissen Grad) versehen werden kann, ohne die Java- oder XML-Ebene direkt zu berühren.

Auch den Scene Builder stellt Oracle auf der mittlerweile bekannten Download-Seite unter http://www.oracle.com/technetwork/java/javafx/downloads/index.html zur Verfügung. Zum Zeitpunkt der Bucherstellung ist die Version 1.1, mit der wir im Buch arbeiten werden, aktuell.

Sie sollten den Scene Builder auf jeden Fall laden und installieren. Er kann sowohl als Standalone-Anwendung, aber auch aus NetBeans heraus verwendet werden.

- Bei allen spezifischen Ausführungen zu konkreten Programmen kann es über die Zeit sein, dass sich verschiedene Schritte in neuen Versionen ändern. In

Abb. 1.7 Download von NetBeans, aber auch dem JDK und JavaFX

einem Buch ist es leider nicht ganz zu vermeiden, dass solche Situationen auftreten können. Im Fall des Falles müssten Sie in den aktuellen Dokumentationen zu den Programmen, aber gegebenenfalls auch APIs etc. nachsehen.

1.5 Ein erstes JavaFX-Beispiel mit dem Scene Builder

Lassen Sie uns ein erstes JavaFX-Beispiel erstellen. Dabei wollen wir ganz einfach beginnen und die Quellcodeebene gar nicht direkt beachten. Dazu bietet sich der Scene Builder an, denn mit ihm können Sie rein visuell ein JavaFX-Programm erstellen[12].

[12] Unsere Referenzversion ist 1.1.

► Beachten Sie, dass wir ohne Codeberührung natürlich nur ganz einfache Beispiele erstellen können. Genau genommen sind wir dabei auch weitgehend auf die Erstellung einer grafischen Oberfläche beschränkt. Aber das ist ja auch schon beachtlich, wenn man dies im Vergleich zu dem Aufwand bei einer codebasierten Erstellung einer grafischen Benutzerschnittstelle betrachtet.

1.5.1 Ein erster Blick auf den Scene Builder

Starten Sie nun den Scene Builder. Sie sollten nach einer Installation einen Eintrag im Startmenü, auf dem Desktop oder der auf Ihrem Betriebssystem üblichen Ebene finden (Abb. 1.8).

Speichern Sie das Beispiel unter dem Namen *Erstes Beispiel* in einem Projektverzeichnis, das Sie auch für die anderen Beispiele in dem Kapitel verwenden sollten. Wenn Sie wie üblich den Speichernsdialog aufrufen, sehen Sie, dass als Vorgabe für die Datei vom Scene Builder das Dateiformat ***FXML*** gewählt wird (Abb. 1.9)[13]. Das ist das schon erwähnte XML-Format von JavaFX, um damit eine Oberfläche losgelöst von der eigentlichen Java-Ebene deklarativ zu beschreiben. Diese wollen wir uns aber wie gesagt noch gar nicht näher ansehen, sondern erst einmal die visuellen Möglichkeiten der Scene Builders verwenden.

1.5.1.1 Ein Rundblick im Scene Builder

Schauen wir uns nun ein bisschen im Scene Builder um. Sie werden – falls Sie entsprechende Erfahrungen haben – große Ähnlichkeiten zu visuellen Entwicklungsumgebungen wie dem Visual Studio, Delphi oder ähnlichen Tools erkennen. Im Grunde bieten alle diese Tools ähnliche Möglichkeiten an. Hier ist eine kurze Beschreibung des Scene Builder, wenn er in der Grundeinstellung gestartet wurde:

- Ganz oben sehen Sie wie üblich ein klassisches Menü. Daneben finden Sie einen Nachrichtenbereich für Fehlermeldungen und (interaktive) Statusanzeigen.
- Auf der linken Seite sollten Sie in der Grundeinstellung einen Bereich ***Library*** sehen (Abb. 1.9) – eine **Komponentenbibliothek**. Hier können Sie sich Komponenten auswählen, aus denen Sie die Oberfläche zusammensetzen können. Das sind etwa Schaltflächen und allgemeine Formularfelder, Menüs, grafische Formen, Diagramme, aber auch Anordnungscontainer.
- Unter dem Bereich *Library* sollten Sie in der Grundeinstellung einen Bereich ***Hierarchy*** sehen (Abb. 1.9). Dort sehen Sie in einer Art Baumstruktur ineinander verschachtelte Komponenten[14]. Wenn Sie etwa eine Schaltfläche in ein Fenster ziehen, wird die Schalt-

[13] Die Datei heißt also *ErstesBeispiel.fxml*

[14] Eine grafische Oberfläche als Baum zu verstehen ist mittlerweile etabliert, beispielweise im DOM-Konzept einer Webseite.