



ADOBE® FLEX® 4.5 Fundamentals

Training from the **Source**

Michael Labriola, Jeff Tapper, and Matthew Boles

Foreword by Adam Lehman, Adobe Flash Builder Product Manager



Adobe® Flex® 4.5 Fundamentals

Training from the Source

Michael Labriola

Jeff Tapper

Matthew Boles

Foreword by Adam Lehman, Adobe Flash Builder Product Manager



Adobe® Flex® 4.5 Fundamentals: Training from the Source

Michael Labriola/Jeff Tapper/Matthew Boles



This Adobe Press book is published by Peachpit.

For information on Adobe Press books, contact:

Peachpit

1249 Eighth Street

Berkeley, CA 94710

510/524-2178

510/524-2221 (fax)

For the latest on Adobe Press books, go to www.adobepress.com

To report errors, please send a note to errata@peachpit.com

Copyright © 2012 by Michael Labriola and Jeffrey Tapper

Adobe Press Editor: Victor Gavenda

Project Editor: Nancy Peterson

Development Editor: Robyn G. Thomas

Technical Editor: Steve Lund

Production Coordinator: Becky Winter

Copy Editor: Jessica Grogan

Compositor: Danielle Foster

Indexer: Emily Glossbrenner

Cover Design: Peachpit Press

Notice of Rights

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Notice of Liability

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of the book, neither the authors, Adobe Systems, Inc., nor the publisher shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks

Flash, ColdFusion, and Adobe are registered trademarks of Adobe Systems, Inc. Flex is a trademark of Adobe Systems, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

Printed and bound in the United States of America

ISBN 13: 978-0-321-77712-6

ISBN 10: 0-321-77712-3

9 8 7 6 5 4 3 2 1

*To my wife, Laura, and my daughter, Lilia;
you make life much less quiet, but so much more worthwhile.*

—Michael Labriola

*My efforts on this book are dedicated to my wife, Lisa, and children,
Kaliope and Kagan. Without you to inspire me, this just wouldn't be possible.*

—Jeff Tapper

*To Sandra, my wife, who has made the last 25 years together a joy.
And to Scrappy, my furry fishing buddy.*

—Matthew Boles

Bios

Michael Labriola is a Founding Partner and Senior Consultant at Digital Primates. He has been developing Internet applications since 1995 and has been working with Flex since its 1.0 beta program. Michael is a Flex SDK contributor, architect of both the open source FlexUnit and Spoon Framework projects, and international speaker on Flex and AIR topics who has consulted for many of the world's most recognized brands.

Jeff Tapper is a Founding Partner and Senior Consultant at Digital Primates, a company that provides expert guidance on rich Internet application development and empowers clients through mentoring. He has been developing Internet-based applications since 1995 for a myriad of clients, including Major League Baseball, ESPN, Morgan Stanley, Conde Nast, IBM, Dow Jones, American Express, Verizon, and many others. He has been developing Flex applications since the earliest days of Flex 1. As an instructor, Jeff is certified to teach all of Adobe's courses on Flex, AIR, Flash, and ColdFusion development. He is also a frequent speaker at Adobe Development Conferences and user groups.

Matthew Boles is a Technical Training Specialist for the Adobe Enterprise Training group, and has been developing and teaching courses on Flex since the 1.0 release. Matthew has a diverse background in web development, computer networking, and teaching. He is coauthor of previous versions of this book, as well as a contributing author of the Adobe authorized Flex courseware.

Acknowledgments

Thanks to Robyn, Steve, Jeff, and Matt for their work and dedication to this book. Thanks to my clients and colleagues for the motivation to keep learning new technologies. Thanks to my family for the unwavering support and love. Most importantly, thanks to those who inspire me every day with their words, caring, and wisdom; I promise to always keep trying.

—Michael Labriola

I would like to thank Mike, Matt, Steve, and Robyn for all their hard work, which has helped shape this book. Thanks to Chris Gieger for providing some design love for our application—Chris, sorry we couldn't fully implement your excellent design. Special thanks go to the team at Adobe who has made this all possible. Thanks to the editorial staff at Adobe Press, who was faced with the Herculean task of making our writing intelligible.

—Jeff Tapper

Thanks to Jeff, Mike, and Robyn for making this the easiest book revision I've ever worked on!

—Matthew Boles

Contents

	Forewordx
	Introductionxii
LESSON 1	Understanding Rich Internet Applications	3
	The Evolution of Computer Applications	4
	The Break from Page-Based Architecture	6
	The Advantages of Rich Internet Applications	7
	RIA Technologies	8
	What You Have Learned15
LESSON 2	Getting Started17
	Getting Started with Flex Application Development18
	Creating a Project and an MXML Application18
	Understanding the Flash Builder Workbench24
	Running Your Application28
	Exploring the Flash Builder Debugger34
	Getting Ready for the Next Lessons41
	What You Have Learned43
LESSON 3	Laying Out the Interface45
	Learning About Layouts46
	Laying Out the E-Commerce Application50
	Working with Constraint-Based Layouts58
	Working with View States63
	Refactoring71
	What You Have Learned76
LESSON 4	Using Simple Controls79
	Introducing Simple Controls80
	Displaying Images81
	Building a Detail View85

	Using Data Binding to Link a Data Structure to a Simple Control88
	Using a Form Layout Container to Lay Out Simple Controls.89
	What You Have Learned.92
LESSON 5	Handling Events.95
	Understanding Event Handling.96
	Handling System Events107
	What You Have Learned.111
LESSON 6	Using Remote XML Data113
	Using Embedded XML.114
	Using XML Loaded at Runtime119
	Retrieving XML Data via HTTPService.124
	Searching XML with E4X127
	Using Dynamic XML Data133
	Using the XMLListCollection in a Flex Control.137
	What You Have Learned.139
LESSON 7	Creating Classes.141
	Building a Custom ActionScript Class142
	Building a Value Object143
	Building a Method to Create an Object.150
	Building Shopping Cart Classes.154
	Manipulating Shopping Cart Data159
	What You Have Learned.167
LESSON 8	Using Data Binding and Collections169
	Examining Data Binding170
	Being the Compiler176
	Understanding Bindable Implications.183
	Using ArrayCollection184
	Refactoring ShoppingCartItem204
	What You Have Learned.205
LESSON 9	Breaking the Application into Components207
	Introducing MXML Components208
	Splitting Off the ShoppingView Component213
	Breaking Out a ProductItem Component221
	Creating Components to Manage Loading the Data230
	What You Have Learned.238

LESSON 10	Using DataGroups and Lists	241
	Using Lists	242
	Using DataGroups	245
	Virtualization with Lists	255
	Displaying Grocery Products Based on Category Selection	257
	What You Have Learned.	259
LESSON 11	Creating and Dispatching Events.	261
	Understanding the Benefits of Loose Coupling.	262
	Dispatching Events.	263
	Declaring Events for a Component.	267
	Identifying the Need for Custom Event Classes.	269
	Building and Using the UserAcknowledgeEvent	270
	Understanding Event Flow and Event Bubbling	274
	Creating and Using the ProductEvent Class	280
	What You Have Learned.	289
LESSON 12	Using the Flex DataGrid	291
	Introducing DataGrids and Item Renderers	292
	Displaying the ShoppingCart with a DataGrid	292
	What You Have Learned.	309
LESSON 13	Using Drag and Drop	311
	Introducing the Drag and Drop Manager	312
	Enhanced Dragging and Dropping Between Two Lists	313
	Standard Dragging and Dropping Between a DataGrid and a List	315
	Using a Non-Drag-Enabled Component in a Drag-and-Drop Operation	321
	Dragging a Grocery Item to the Shopping Cart.	326
	What You Have Learned.	331
LESSON 14	Implementing the Checkout Process	333
	Introducing Navigation with States	334
	Introducing Two-Way Bindings	334
	Creating the OrderInfo valueObject	335
	Creating CheckoutView.	337
	Creating CreditCardInfo.	345
	Creating Review	350
	Completing the Order.	355
	What You Have Learned.	358

LESSON 15	Using Formatters and Validators	361
	Introducing Formatters and Validators	362
	Using Formatter Classes	364
	Examining a Second Locale	368
	Using Validator Classes	369
	What You Have Learned.	372
LESSON 16	Customizing a Flex Application with Styles	375
	Applying a Design with Styles and Skins	376
	Cleaning Up the Appearance	376
	Applying Styles	377
	Changing CSS at Runtime	395
	What You Have Learned.	397
LESSON 17	Customizing a Flex Application with Skins	399
	Understanding the Role of Skins in a Spark Component	400
	Understanding the Relationship between Skins and States	404
	Creating a Skin for the Application.	413
	What You Have Learned.	417
LESSON 18	Creating Custom ActionScript Components	419
	Introducing Components with ActionScript 3.0	420
	Building Components Can Be Complex.	420
	Understanding Flex Components	421
	Why Make Components?.	422
	Defining a Component	424
	Creating the Visuals	432
	Adding Functionality to the Component.	439
	Creating a Renderer for the Skin	450
	What You Have Learned.	452
APPENDIX	Setup Instructions	455
	Software Installation.	455
	Importing Projects	458
	Index	462

Foreword

Over a decade ago, Adobe (then Macromedia) coined the term rich Internet application, or RIA, to describe the future of browser-based applications. This new breed of application supplemented existing server-based applications with an enhanced client-side user experience. As Internet users became increasingly sophisticated, demand for improved user experiences grew. At the center of this paradigm shift was Adobe Flex, a simple and light-weight framework for developing applications.

Once a novelty, Internet usage on phones and tablets has exploded. Users can now access the Internet more from mobile devices than from personal computers. As such, user demand for browser-based applications is shifting to applications installed on devices. Yet again, the Flex framework can be found leading the charge. With the release of the Flex 4.5 SDK, Flex applications can now be deployed as native applications to Android, Apple iOS, and Blackberry devices. With this book, you hold in your hands all the knowledge and best practices necessary to deliver killer applications for not just one of the leading mobile platforms...but all of them!

Adobe Flex is composed of a number of elements. It uses a declarative markup language called MXML to help structure your application and ActionScript, a highly productive scripting language, to glue all the pieces together. The framework also has built-in support for CSS and a simple but comprehensive skinning model. These complimentary languages will probably look familiar to those with HTML and JavaScript experience. In addition to the languages that power Flex, the framework provides layout containers, form controls, validators, effects, state management frameworks, a multipurpose animation library, and much more to help you rapidly build the next generation of web applications.

Of course, what good is a slick interface if you can't connect it to live data and services? Fortunately, Flex offers a multitude of ways to connect to nearly any backend service, whether it is raw XML over HTTP, SOAP web services, or the blazingly fast remoting protocol called Action Message Format (AMF). If you're looking for an enterprise-grade data management solution to share data with multiple users simultaneously, Flex offers tight integration with the Adobe Digital Enterprise Platform and Adobe LiveCycle DataServices.

Most of the improvements in Flex 4.5 are focused around mobile and device development. Rather than introducing a separate mobile version of Flex, we upgraded the existing framework for mobile development. You can now use the same tools and languages to build a Flex mobile application that you do to build a Flex application for the browser of the desktop. Built on the foundation of Spark, the next generation component model introduced in Flex 4, Flex 4.5 continues to add new components and capabilities. The Flex compiler has also undergone numerous improvements to ensure applications run faster with even less memory.

Flex is open source and free. Outside this book, you don't have to purchase anything else to develop rich Internet applications for the browser, desktop, or mobile devices. You can just open your favorite text editor, write some code, and compile your application at the command line. But if you're like me, you'll probably want some better tooling support. This book uses Adobe Flash Builder 4.5, the premiere IDE for Flex and ActionScript development. Flash Builder 4.5's rock-solid code editor and intuitive features, like Quick Assist, will make you fall in love with ActionScript coding. If that isn't enough, Flash Builder 4.5 supports the new mobile workflow, from the creation of a new mobile project to debugging your application live on a connected device. Additionally, there is a large and vast ecosystem of third-party tools, libraries, and extensions (some written by your authors!) to enhance productivity and aid in the development of your applications.

There is a wealth of reference information on Flex freely available on the Internet, but to build the next killer app, you need to know how to put all the pieces together. *Adobe Flex 4.5: Training from the Source* draws from the expertise of its authors to present lessons that not only introduce you to the Flex framework but also teach you the best practices you need to be successful.

Times are changing. Whether its browser, desktop, or mobile devices, the Flex SDK and Adobe Flash Builder provides the tools you need to build a better Internet. The next fabulous app is just a few clicks away.

Adam Lehman
Senior Product Manager
Adobe Systems, Inc.

Introduction

Macromedia introduced Flex in 2004 so that developers could write web applications for the nearly ubiquitous Flash platform. These applications benefited from the improved design, usability, and portability that Flex made possible, dramatically changing the user experience. These features are a cornerstone of Web 2.0, a new generation of Internet applications focused on creativity and collaboration.

Since the introduction of Flex, Macromedia—and now Adobe—has released versions 1.5, 2, 3, 4, and 4.5 of Flex. With each subsequent version, creating rich, compelling, intuitive applications has gotten easier, and the bar has been raised on users' expectations of web applications. Countless organizations have discovered the benefits of Flex and have built and deployed applications that run on the Flash platform.

But Flex 1 and 1.5 were most definitely not mass-market products. The pricing, lack of IDE, limited deployment options, and other factors meant that those early versions of Flex were targeted specifically for large and complex applications as well as for sophisticated developers and development. However, with the new releases of the Flex product line, all this has changed.

Flex 2 was released in 2006 and made Flex development a possibility for many more people, as it included a free software development kit (SDK). With the open sourcing of Flex 3, and the announcement of free versions of Flash Builder for students, Flex development is within the grasp of any developer with enough foresight to reach for it. The release of Flex 4 made it even easier to build rich, efficient, cutting-edge applications, and streamlined the workflow between designer and developer, greatly easing the process of bringing intuitive, compelling designs to even more Flex applications. In this latest release, Flex 4.5, Adobe has further extended the reach of Flex, making it possible to deploy applications not only to browsers and desktops, but to phones, tablets, televisions, and other connected devices.

Getting started with Flex is easy. Flex itself is composed of two languages: MXML, an XML-based markup language, and ActionScript, the language of Flash Player. MXML tags are easy to learn (especially when Flash Builder writes them for you). ActionScript has a steeper learning curve, but developers with prior programming and scripting experience will pick it up easily. Still, there's more to Flex development than MXML and ActionScript.

To be a successful Flex developer, you'll need to understand a number of concepts, including the following:

- How Flex applications should be built (and how they should not)
- What the relationships between MXML and ActionScript are, and when to use each
- How to load data into a Flex application
- How to use the Flex components, and how to write your own
- What the performance implications are of the code you write
- Which practices you should employ to write code that is scalable, manageable, and reusable

Developing these skills is where this book comes in. As the authors, we have distilled our hard-earned Flex expertise into a series of lessons that will jump-start your own Flex development. Starting with the basics, and then incrementally introducing additional functionality and know-how, the author team guides your journey into the exciting world of RIAs, ensuring success every step of the way.

Flex is powerful, highly capable, fun, and incredibly addictive. And *Adobe Flex 4.5: Training from the Source* is the ideal tour guide on your journey to the next generation of application development.

Adobe Flex 4.5: Training from the Source is an update to the popular *Adobe Flex 4: Training from the Source*. It is our sincere intention that readers of the earlier book, as well those who are first exploring Flex with this book, will find this content compelling. Since the release of our previous book, the Flex SDK has been improved, with features that include:

- Support for internationalization of Flex applications
- Additional components, such as the DataGrid, added to the Spark component set
- Support for deploying applications to desktops, browsers, phones, tablets, and other connected devices
- And much more

It's an incredible time to be an RIA developer, and we hope that this book provides you with all the tools you need to get started with Flex.

Prerequisites

To make the most of this book, you should at the very least understand web terminology. This book isn't designed to teach you anything more than Flex, so the better your understanding of the World Wide Web, the better off you'll be. This book is written assuming that you're comfortable working with programming languages and that you're working with a server-side language such as Java, .NET, PHP, or ColdFusion. Although knowledge of server-side technologies is not required to succeed with this book, we invoke many comparisons and analogies to server-side web programming. This book is not intended as an introduction to programming or as an introduction to object-oriented programming (OOP). Experience with OOP is not required, although if you have no programming experience at all, you might find the materials too advanced.

Outline

As you'll soon discover, this book mirrors real-world practices as much as possible. Where certain sections of the book depart from what would be considered a real-world practice, every attempt has been made to inform you. The exercises are designed to get you using the tools and the interface quickly so that you can begin to work on projects of your own with as smooth a transition as possible.

This curriculum should take approximately 28–35 hours to complete and includes the following lessons:

Lesson 1: Understanding Rich Internet Applications

Lesson 2: Getting Started

Lesson 3: Laying Out the Interface

Lesson 4: Using Simple Controls

Lesson 5: Handling Events

Lesson 6: Using Remote XML Data

Lesson 7: Creating Classes

Lesson 8: Using Data Binding and Collections

Lesson 9: Breaking the Application into Components

Lesson 10: Using DataGroups and Lists

Lesson 11: Creating and Dispatching Events

Lesson 12: Using the Flex DataGrid

Lesson 13: Using Drag and Drop

Lesson 14: Implementing the Checkout Process

Lesson 15: Using Formatters and Validators

Lesson 16: Customizing a Flex Application with Styles

Lesson 17: Customizing a Flex Application with Skins

Lesson 18: Creating Custom ActionScript Components

Who Is This Book For?

All the content of this book should work well for users of Flash Builder on any of its supported platforms. The earlier “Prerequisites” section details what a reader should know prior to reading this, in order to get the most out of this book.

The Project Application

Adobe Flex 4.5: Training from the Source includes many comprehensive tutorials designed to show you how to create a complete application using Flex. The application that you’ll create is an online grocery store that displays data and images and takes a user through the checkout process, ending just before the data would be submitted to a server.

By the end of the book, you’ll have built the entire application using Flex. You’ll begin by learning the fundamentals of Flex and understanding how you can use Flash Builder in developing the application. In the early lessons, you’ll use Design mode to begin laying out the application, but as you progress through the book and become more comfortable with the languages used by Flex, you’ll spend more and more time working in Source mode, which gives you the full freedom and flexibility of directly working with code. By the end of the book, you should be fully comfortable working with the Flex languages and may even be able to work without Flash Builder by using the open source Flex SDK and its command-line compiler.

Errata

Although we have made every effort to create a flawless application and book, occasionally we or our readers find problems. The errata for the book will be posted at www.flexgrocer.com.

Standard Elements in the Book

Each lesson in this book begins by outlining the major focus of the lesson at hand and introducing new features. Learning objectives and the approximate time needed to complete all the exercises are also listed at the beginning of each lesson. The projects are divided into exercises that demonstrate the importance of each skill. Every lesson builds on the concepts and techniques learned in the previous lessons.

The following are some consistent elements and styles you'll encounter throughout the book:

▶ **TIP:** An alternative way to perform a task or a suggestion to consider when applying the skills you are learning.

* **NOTE:** Additional background information to expand your knowledge, or advanced techniques you can explore to further develop your skills.

▼ **CAUTION!** Information warning you of a situation you might encounter that could cause errors, problems, or unexpected results.

Boldface text: Words that appear in **boldface** are terms that you must type while working through the steps in the lessons.

Boldface code: Lines of code that appear in **boldface** within code blocks help you easily identify changes in the block to be made in a specific exercise step.

```
<mx:HorizontalList dataProvider="{dp}"  
  labelFunction="multiDisplay"  
  columnWidth="130"  
  width="850"/>
```

Code in text: Code or keywords appear slightly different from the rest of the text so you can identify them easily.

Code block: To help you easily identify ActionScript, XML, and HTML code within the book, the code has been styled in a special font that's different from the rest of the text. Single lines of ActionScript code that are longer than the margins of the page are wrapped to the next line. They are designated by an arrow at the beginning of the continuation of a broken line and are indented under the line from which they continue. For example:

```
public function Product (_catID:Number, _prodName:String,  
    ➤ _unitID:Number, _cost:Number, _listPrice:Number,  
    ➤ _description:String, _isOrganic:Boolean, _isLowFat:Boolean,  
    ➤ _imageName:String)
```

Italicized text: *Italics* are used to show *emphasis* or to introduce *new vocabulary*.

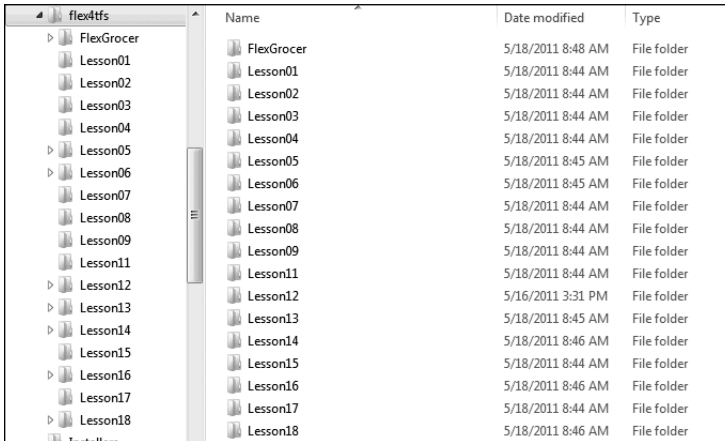
Italics are also used for placeholders, which indicate that a name or entry may change depending on your situation. For example, in the path *driveroot:/flex4tfs/flexgrocer*, you would substitute the actual name of your root drive for the placeholder.

Menu commands and keyboard shortcuts: There are often multiple ways to perform the same task in Flash Builder. The different options will be pointed out in each lesson. Menu commands are shown with angle brackets between the menu names and commands: Menu > Command > Subcommand. Keyboard shortcuts are shown with a plus sign between the names of keys to indicate that you should press the keys simultaneously; for example, Shift+Tab means that you should press the Shift and Tab keys at the same time.

CD-ROM: The CD-ROM included with this book includes all the media files, starting files, and completed projects for each lesson in the book. These files are located in the start and complete directories. Lesson 1, “Understanding Rich Internet Applications,” does not include exercises. If you need to return to the original source material at any point, you can restore the FlexGrocer project. Some lessons include an intermediate directory that contains files in various stages of development in the lesson. Other lessons may include an independent directory that is used for small projects intended to illustrate a specific point or exercise without impacting the FlexGrocer project directly.

Anytime you want to reference one of the files being built in a lesson to verify that you are correctly executing the steps in the exercises, you will find the files organized on the CD-ROM under the corresponding lesson. For example, the files for Lesson 4 are located on the CD-ROM in the Lesson04 folder, in a project named FlexGrocer.fxp.

The directory structure of the lessons you'll be working with is as follows:



Name	Date modified	Type
FlexGrocer	5/18/2011 8:48 AM	File folder
Lesson01	5/18/2011 8:44 AM	File folder
Lesson02	5/18/2011 8:44 AM	File folder
Lesson03	5/18/2011 8:44 AM	File folder
Lesson04	5/18/2011 8:44 AM	File folder
Lesson05	5/18/2011 8:45 AM	File folder
Lesson06	5/18/2011 8:45 AM	File folder
Lesson07	5/18/2011 8:44 AM	File folder
Lesson08	5/18/2011 8:44 AM	File folder
Lesson09	5/18/2011 8:44 AM	File folder
Lesson11	5/18/2011 8:44 AM	File folder
Lesson12	5/16/2011 3:31 PM	File folder
Lesson13	5/18/2011 8:45 AM	File folder
Lesson14	5/18/2011 8:46 AM	File folder
Lesson15	5/18/2011 8:44 AM	File folder
Lesson16	5/18/2011 8:46 AM	File folder
Lesson17	5/18/2011 8:44 AM	File folder
Lesson18	5/18/2011 8:46 AM	File folder

Directory structure

Adobe Training from the Source

The *Adobe Training from the Source* and *Adobe Advanced Training from the Source* series are developed in association with Adobe and reviewed by the product support teams. Ideal for active learners, the books in the *Training from the Source* series offer hands-on instruction designed to provide you with a solid grounding in the program's fundamentals. If you learn best by doing, this is the series for you. Each *Training from the Source* title contains hours of instruction on Adobe software products. They are designed to teach the techniques that you need to create sophisticated professional-level projects. Each book includes a CD-ROM that contains all the files used in the lessons, completed projects for comparison, and more.

What You Will Learn

You will develop the skills you need to create and maintain your own Flex applications as you work through these lessons.

By the end of the book, you will be able to:

- Use Flash Builder to build Flex applications.
- Understand MXML, ActionScript 3.0, and the interactions of the two.
- Work with complex sets of data.

- Load data using XML.
- Handle events to allow interactivity in an application.
- Create your own event classes.
- Create your own components, either in MXML or ActionScript 3.0.
- Apply styles and skins to customize the look and feel of an application.
- And much more.

Minimum System Requirements

Windows

- 2 GHz or faster processor
- 1 GB of RAM (2 GB recommended)
- Microsoft Windows XP with Service Pack 3, Windows Vista Ultimate or Enterprise (32 or 64 bit running in 32-bit mode), Windows Server 2008 (32 bit), or Windows 7 (32 or 64 bit running in 32-bit mode)
- 1 GB of available hard-disk space
- Java Virtual Machine (32 bit): IBM JRE 1.6, or Sun JRE 1.6
- 1024x768 display (1280x800 recommended) with 16-bit video card
- Flash Player 10.2 or later

Macintosh

- Intel processor based Mac
- OS X 10.6 (Snow Leopard)
- 1 GB of RAM (2 GB recommended)
- 1.5 GB of available hard-disk space
- Java Virtual Machine (32 bit): JRE 1.6
- 1024x768 display (1280x800 recommended) with 16-bit video card
- Flash Player 10.2 or later

The Flex line of products is extremely exciting, and we're waiting to be amazed by what you will do with it. With a strong foundation in Flex, you can expand your set of skills quickly.

Flex is not difficult to use for anyone with programming experience. With a little bit of initiative and effort, you can fly through the following lessons and be building your own custom applications and sites in no time.

Additional Resources

Flex Community Help

Flex Community Help brings together active Flex users, Adobe product team members, authors, and experts to give you the most useful, relevant, and up-to-date information about Flex. Whether you're looking for a code sample, an answer to a problem or question about the software, or want to share a useful tip or recipe, you'll benefit from Community Help. Search results will show you not only content from Adobe, but also from the community.

With Adobe Community Help you can:

- Fine-tune your search results with filters that let you narrow your results to just Adobe content, community content, just the ActionScript Language Reference, or even code samples.
- Download core Adobe Help and ActionScript Language Reference content for offline viewing via the new Community Help AIR application.
- See what the community thinks is the best, most valuable content via ratings and comments.
- Share your expertise with others and find out what experts have to say about using your favorite Adobe products.

If you have installed Flash Builder 4.5 or any Adobe CS5 product, then you already have the Community Help application. This companion application lets you search and browse Adobe and community content, plus you can comment and rate any article just like you would in the browser. However, you can also download Adobe Help and reference content for use offline. You can also subscribe to new content updates (which can be downloaded automatically) so that you'll always have the most up-to-date content for your Adobe product at all times. You can download the application from <http://www.adobe.com/support/chc/index.html>.

Community Participation

Adobe content is updated based on community feedback and contributions: You can contribute content to Community Help in several ways: add comments to content or forums, including links to web content; publish your own content via the Community Publishing System; or contribute Cookbook Recipes. Find out how to contribute at www.adobe.com/community/publishing/download.html.

Community Moderation and Rewards

More than 150 community experts moderate comments and reward other users for helpful contributions. Contributors get points: 5 points for small stuff like finding typos or awkward wording, up to 200 points for more significant contributions like long tutorials, examples, cookbook recipes, or Developer Center articles. A user's cumulative points are posted to their Adobe profile page and top contributors are called out on leader boards on the Help and Support pages, Cookbooks, and Forums. Find out more at www.adobe.com/community/publishing/community_help.html.

Frequently Asked Questions

You might find the following resources helpful for providing additional instruction:

For answers to frequently asked questions about Community Help see <http://community.adobe.com/help/profile/faq.html>.

Adobe Flex and Flash Builder Help and Support www.adobe.com/support/flex/ is where you can find and browse Help and Support content on adobe.com.

Adobe TV <http://tv.adobe.com> is an online video resource for expert instruction and inspiration about Adobe products, including a How To channel to get you started with your product.

Adobe Developer Connection www.adobe.com/devnet is your source for technical articles, code samples, and how-to videos that cover Adobe developer products and technologies.

Cookbooks <http://cookbooks.adobe.com/home> is where you can find and share code recipes for Flex, ActionScript, AIR, and other developer products.

Resources for educators www.adobe.com/education includes three free curriculums that use an integrated approach to teaching Adobe software and can be used to prepare for the Adobe Certified Associate exams.

Also check out these useful links:

Adobe Forums <http://forums.adobe.com> lets you tap into peer-to-peer discussions, questions, and answers on Adobe products.

Adobe Marketplace & Exchange www.adobe.com/cfusion/exchange is a central resource for finding tools, services, extensions, code samples, and more to supplement and extend your Adobe products.

Adobe Flex product home page www.adobe.com/products/flex is the official home page from Adobe for Flex related products.

Adobe Labs <http://labs.adobe.com> gives you access to early builds of cutting-edge technology, as well as forums where you can interact with both the Adobe development teams building that technology and other like-minded members of the community.

Adobe Certification

The Adobe Certified program is designed to help Adobe customers and trainers improve and promote their product-proficiency skills. There are four levels of certification:

- Adobe Certified Associate (ACA)
- Adobe Certified Expert (ACE)
- Adobe Certified Instructor (ACI)
- Adobe Authorized Training Center (AATC)

The Adobe Certified Associate (ACA) credential certifies that individuals have the entry-level skills to plan, design, build, and maintain effective communications using different forms of digital media.

The Adobe Certified Expert (ACE) program is a way for expert users to upgrade their credentials. You can use Adobe certification as a catalyst for getting a raise, finding a job, or promoting your expertise.

If you are an ACE-level instructor, the Adobe Certified Instructor (ACI) program takes your skills to the next level and gives you access to a wide range of Adobe resources.

Adobe Authorized Training Centers offer instructor-led courses and training on Adobe products, employing only Adobe Certified Instructors. A directory of AATCs is available at <http://partners.adobe.com>.

For information on the Adobe Certified program, visit www.adobe.com/support/certification/main.html.

This page intentionally left blank

LESSON 4

What You Will Learn

In this lesson, you will:

- Define the user interface (UI) for the e-commerce FlexGrocer application
- Use simple controls such as the Image control, text controls, and CheckBox control
- Define the UI for the checkout screens
- Use the Form container to lay out simple controls
- Use data binding to connect controls to a data model

Approximate Time

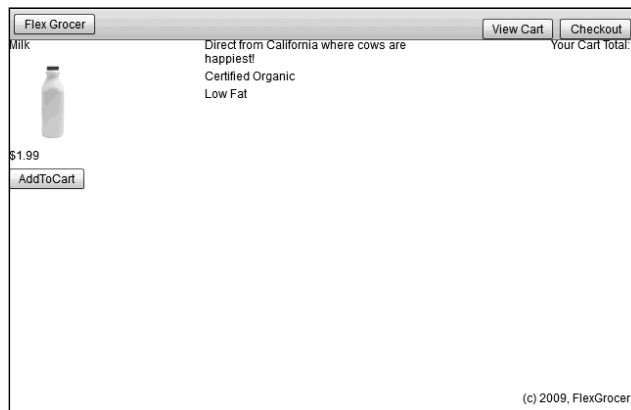
This lesson takes approximately 45 minutes to complete.

LESSON 4

Using Simple Controls

In this lesson, you will add user interface elements to enable the customer to find more details about the grocery items and begin the checkout process. An important part of any application is the user interface, and Adobe Flex contains elements such as buttons, text fields, and radio buttons that make building interfaces easier. Simple controls can display text and images and also gather information from users. You can tie simple controls to an underlying data structure, and they will reflect changes in that data structure in real time through data binding. You're ready to start learning about the APIs (application programming interfaces) of specific controls, which are available in both MXML and ActionScript. The APIs are fully documented in the ActionScript Language Reference, often referred to as ASDoc, which is available at http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/index.html.

The Flex framework has many tools that make laying out simple controls easier. All controls are placed within containers (see Lesson 3, "Laying Out the Interface"). In this lesson, you'll become familiar with simple controls by building the basic user interface of the application that you will develop throughout this book. You'll also learn about timesaving functionality built into the framework, such as data binding and capabilities of the Form layout container.



FlexGrocer with Image and text controls bound to a data structure

Introducing Simple Controls

Simple controls are provided as part of the Flex framework and help make rich Internet application development easy. Using controls, you can define the look and feel of your buttons, text, combo boxes, and much more. Later in this book, you'll learn how to customize controls to create your own unique look and feel. Controls provide a standards-based methodology that makes learning how to use them easy. Controls are the foundation of any RIA.

The Flex SDK includes an extensive class library for both simple and complex controls. All these classes can be instantiated via an MXML tag or as a standard ActionScript class, and their APIs are accessible in both MXML and ActionScript. The class hierarchy comprises nonvisual classes as well, such as those that define the event model, and it includes the display attributes that all simple controls share.

You place the visual components of your Flex application inside containers, which establish the size and positioning of text, controls, images, and other media elements (you learned about containers in the previous lesson). All simple controls have events that can be used to respond to user actions, such as clicking a button, or system events, such as another component being drawn (events will be covered in detail in the next lesson). You will learn in later lessons how to build your own events. Fundamentally, events are used to build easily maintainable applications that reduce the risk that a change to one portion of the application will force a change in another. This is often referred to as building a “loosely coupled” application.

Most applications need to display some sort of text, whether it be static or dynamically driven from an outside source like an XML file or a database. Flex has a number of text controls that can be used to display editable or noneditable text:

- **Label:** You have already used the Label control to display text. The Label control cannot be edited by an end user; if you need that functionality, you can use a TextInput control.
- **TextInput:** The TextInput control is used for data input. It is limited to a single line of text.
- **RichText:** The RichText control is used to display multiple lines of text, but is not editable and does not display scroll bars if the text exceeds the available screen space.
- **TextArea:** The TextArea component is useful for displaying multiple lines of text, either editable or noneditable, with scroll bars if the available text exceeds the available screen space.

All text controls support HTML 1.0 and a variety of text and font styles.

*** NOTE:** All four text controls mentioned here support Adobe's Flash Text Engine and some of the controls (RichText and RichEditableText) support even more advanced layout using the Text Layout Framework (TLF). While you will not be using TLF as part of the application in this book, many new and interesting features are available with TLF. You can learn about TLF on Adobe's open source site: <http://opensource.adobe.com/wiki/display/tlf/Text+Layout+Framework>.

To populate text fields at runtime, you must assign an ID to the control. Once you have done that, you can access the control's properties; for example, all the text controls previously mentioned have a `text` property. This property enables you to populate the control with plain text using either an ActionScript function or inline data binding. The following code demonstrates assigning an ID to the label, which enables you to reference the Label control in ActionScript:

```
<s:Label id="myLabel"/>
```

You can populate any text control at runtime using data binding, which is denoted by curly bracket syntax in MXML. The following code will cause the `yourLabel` control to display the same text as the `myLabel` control in the previous example:

```
<s:Label id="yourLabel" text="{myLabel.text}"/>
```

Also, you can use data binding to bind a simple control to underlying data structures. For example, if you have XML data, which might come from a server-side dataset, you can use data binding to connect a simple control to the data structure. When the underlying data changes, the controls are automatically updated to reflect the new data. This provides a powerful tool for the application developer.

The Flex framework also provides a powerful container for building the forms that we will cover in this lesson. The Form container allows developers to create efficient, good-looking forms with minimal effort. Flex handles the heading, spacing, and arrangement of form items automatically.

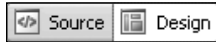
Displaying Images

In this exercise, you will display images of grocery products. To do this, you must use the Image control to load images dynamically. The Image control can load JPG, GIF, SWF, and PNG files at runtime. If you are developing an offline application that will not access the Internet, you can use the `@Embed` directive to include the Image control in the completed SWF file.

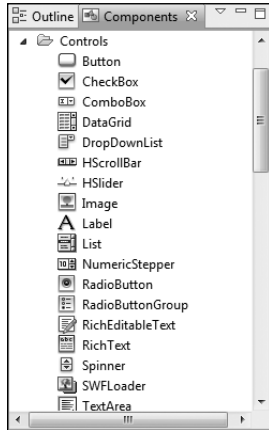
- 1 Open the `FlexGrocer.mxml` file that you created in the previous lesson.

If you didn't complete the previous lesson, you can import the `Lesson04/start` files. Please refer to the appendix for complete instructions on importing a project should you skip a lesson or if you have a code issue you cannot resolve.

- 2** Switch Flash Builder to Design view by clicking the Design View button.



- 3** Be sure that the Components view is open. If it's not, choose Window > Components.

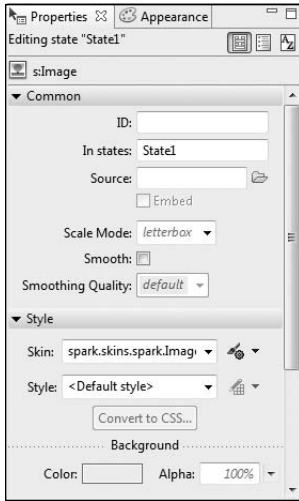


- 4** Select the Image control from the Controls folder and drag the control between the Milk and 1.99 Label controls you already added.



When you drag the Image control from the Components view to the container, Flash Builder automatically adds the MXML to place the Image control on the screen and positions it where you drop it.

- 5** Be sure that the Flex Properties view is open. If it's not, choose Window > Properties.



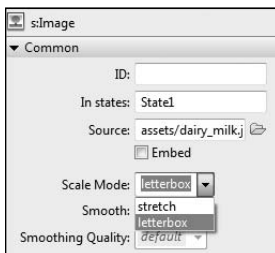
The Flex Properties view shows important attributes of the selected component—in this case, the Image control. You can see the Source property, which specifies the path to the Image file. The ID of the Image control references the instance created from the `<s:Image>` tag or Image class in ActionScript.

- 6 Click the Source folder icon and navigate to the assets directory. Select the `dairy_milk.jpg` image and click Open.

The image you selected is displayed in Design view. The source property is also added to the MXML tag.

- 7 Click the Scale Mode drop-down menu and change the value to `letterbox`.

In an ideal world, all the images that you use in the application would be a perfect size, but this is not always the case. Flex can scale the images in two ways. You can choose `letterbox` to keep the aspect ratio of the original images correct even as their size is adjusted, or you can choose `stretch` to distort the images to make them fit into any given width and height.



- 8 Switch back to Source view and notice that Flash Builder has added an `<s:Image>` tag as well as the attributes you specified in the Flex Properties window.

*** NOTE:** Letterbox is the default selection if you don't choose a Scale Mode. So, if you didn't explicitly choose it from the drop-down list and instead left it as the default, you may not see it in your code. Feel free to add it or just understand that difference going forward.

As you can see, it is easy to switch between Source view and Design view, and each one has its advantages. Notice as you switch back to Source view that the Image tag you were working on is now highlighted.

```

25< <s:VGroup id="products" width="100%" height="150"
26     visible.cartView="false" width.cartView="0" height.cartView="0">
27     <s:Label id="prodName" text="Milk"/>
28     <s:Image includeIn="State1" scaleMode="letterbox" source="assets/dairy_milk.jpg"/>
29     <s:Label id="price" text="$1.99"/>
30     <s:Button id="add" label="Add To Cart"/>
31 </s:VGroup>

```

- 9 In the `<s:Image>` tag that you added, insert an `@Embed` directive to the Image control.

```

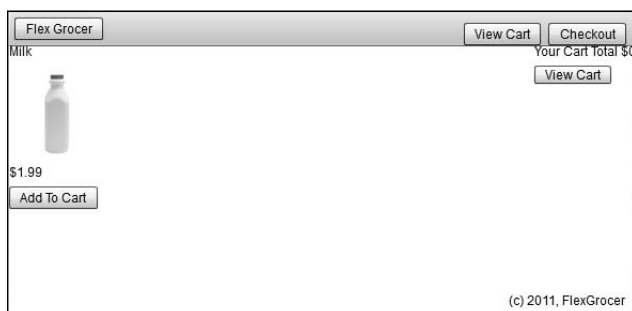
<s:Image includeIn="State1" scaleMode="letterbox"
  ➤ source="@Embed('assets/dairy_milk.jpg')"/>

```

The `@Embed` directive causes the compiler to transcode and include the JPG in the SWF file at compile time. This technique has a couple advantages over the default of loading the image at runtime. First, the image is loaded at the start of the application, so the user doesn't have to wait for the image to load before displaying when it is needed. Also, this technique can be useful if you are building offline applications that do not need to access the Internet because the appropriate images are included in the SWF file and will be correctly displayed when needed. Remember, though, that using this technique greatly increases the size of your SWF file.

- 10 Save, compile, and run the application.

You should see that the Image and Label controls and button fit neatly into the layout container.



Building a Detail View

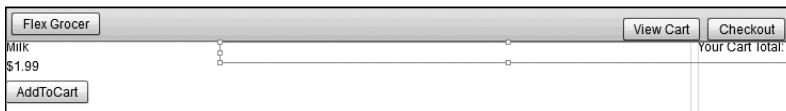
In this exercise, you will use a rollover event to display a detailed state of the application. You will explore different simple controls to display text and review how application states work.

- 1 Be sure that you are still in Source view in Flash Builder. Near the top of the file, locate the `<s:states>` block, which contains definitions for the `State1` and `cartView` states. Add a new state definition named `expanded`.

```
<s:State name="expanded"/>
```

You will define this third state for the application to show details of a product.

- 2 Switch to Design view, set the state selector to **expanded**, and drag a `VGroup` from the Layout folder of the Components view into the application. (To position this correctly, you should drag the `VGroup` into the gray area below the existing white background.) In the Properties view, verify that the `In` state's value is `expanded`, the `X` value is 200, and the `Width` value is 100 percent. Remove the `Y` and `Height` values so that the fields are blank.



This new `VGroup` needs to be a child of the main application. Sometimes, positioning items correctly can be difficult in Design view, so switch to Source view and ensure the `VGroup` is positioned correctly. It should be just above the closing `</s:Application>` tag, so the end of the file reads like this:

```
</s:VGroup>
</s:HGroup>
<s:VGroup includeIn="expanded" width="100%" x="200">
</s:VGroup>
```

```
</s:Application>
```

- 3 Switch back to Design view. Ensure that the `expanded` state is selected in the States view. Drag an instance of the `RichText` control from the Controls folder of the Components view into the new `VGroup` you created in the previous step.



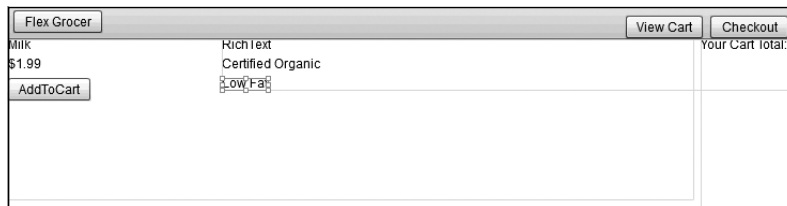
The RichText control enables you to display multiple lines of text, which you will need when you display the product description that will ultimately come from an XML file. You will use data binding in the next section to make this RichText control functional. For now, you are just setting up the layout.

- 4 Drag an instance of the Label control from the Components view to the bottom part of the VGroup container you created. Populate the text property with the words **Certified Organic**.

Later on, you will modify the `visible` property of this component so the contents of the text property are displayed only when a grocery item is certified organic.



- 5 Drag another instance of the Label control from the Components view to the bottom part of the VGroup container you created. Populate the text property with the words **Low Fat**.



Later, you will set the `visible` property of this label to `true` if the grocery item is low fat, or `false` if it is not.

- 6 Switch back to Source view. Notice that Flash Builder has added the RichText and the two Label controls you added in Design view.

Note that all the code created in Design view is displayed in Source view.

- 7 Locate the `<s:RichText>` tag in the expanded state and set the `width` property to 50%.

```
<s:RichText text="RichText" width="50%" />
```

- 8** Find the `<s:Image>` tag that is displaying the milk image. Add a `mouseover` event to the tag that will change the `currentState` to `expanded`. Remove the `includeIn` attribute.

```
<s:Image scaleMode="Letterbox"
         source="@Embed('assets/dairy_milk.jpg')"
         mouseOver="this.currentState='expanded'"/>
```

`mouseover` simply means that when the user rolls the mouse anywhere over the `dairy_milk.jpg` Image tag, the ActionScript will execute. In this ActionScript, you are referring to the `expanded` state, which you created earlier in this lesson.

If you had left the `includeIn` attribute in the image tag, the milk image would appear only in the initial state of `State1`. Therefore, when you mouse over the image and switch it to the `expanded` state, the milk bottle image will disappear. By removing the `includeIn` attribute, you are instructing the application to allow this image to be used in all states.

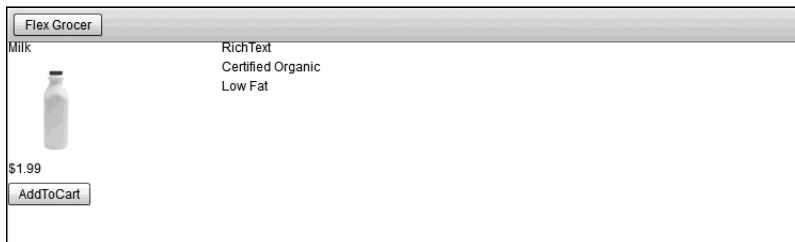
- 9** In the same `<s:Image>` tag, add a `mouseout` event that will change the `currentState` back to the initial `State1` state.

```
<s:Image scaleMode="Letterbox"
         source="@Embed('assets/dairy_milk.jpg')"
         mouseOver="this.currentState='expanded'"
         mouseOut="this.currentState='State1'"/>
```

When the user moves the mouse away from the `dairy_milk.jpg` image, the detailed state no longer displays, and by default the application displays only the images and labels for the control, which is expressed with an empty string.

- 10** Save and run the application.

When you roll the cursor over the milk bottle image, you see the RichText and Label controls you created in the `expanded` state.



Using Data Binding to Link a Data Structure to a Simple Control

Data binding enables you to connect controls, such as the text controls that you have already worked with, to an underlying data structure. Data binding is incredibly powerful because if the underlying data changes, the control reflects the changes. For example, suppose you create a text control that displays the latest sports scores; also suppose it is connected to a data structure in Flex. When a score changes in that data structure, the control that the end user views reflects the change. In this exercise, you will connect a basic data structure in an `<fx:Model>` tag to simple UI controls to display the name, image, and price for each grocery item. Later in the book, you will learn more about data models, the effective use of a model-view-controller architecture on the client, and how to connect these data structures with server-side data.

- 1 Be sure that `FlexGrocer.mxml` is open, and add an `<fx:Model>` tag after the comment in the `<fx:Declarations>` tag pair at the top of the page.

The `<fx:Model>` tag allows you to build a client-side data model. This tag converts an XML data structure into a format that Flex can use.

- 2 Directly below the opening `<fx:Model>` tag and before the closing `<fx:Model>` tag, add the following XML data structure. Your `<fx:Model>` tag should look as shown:

```
<fx:Model>
  <groceries>
    <catName>Dairy</catName>
    <prodName>Milk</prodName>
    <imageName>assets/dairy_milk.jpg</imageName>
    <cost>1.20</cost>
    <listPrice>1.99</listPrice>
    <isOrganic>true</isOrganic>
    <isLowFat>true</isLowFat>
    <description>Direct from California where cows are happiest!</description>
  </groceries>
</fx:Model>
```

You have defined a very simple data structure inline inside an `<fx:Model>` tag.

- 3 Assign the `<fx:Model>` tag an ID of `groceryInventory`. The first line of your `<fx:Model>` tag should look as shown:

```
<fx:Model id="groceryInventory">
```

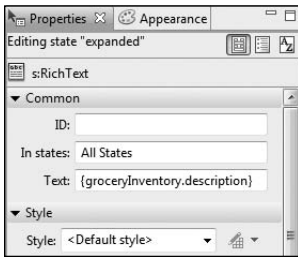
By assigning an ID to the `<fx:Model>` tag, you can reference the data with dot syntax. For example, to access the list price of the item, you could use `groceryInventory.listPrice`. In this case, that would resolve to 1.99.

4 Switch Flash Builder to Design view.

You can set up bindings between elements just as easily in Design view as you can in Source view.

5 Select the RichText control in the expanded state and be sure that the Flex Properties view is open. Modify the text property to `{groceryInventory.description}`.

Data binding is indicated by the curly brackets `{}`. Whenever the curly brackets are used, you use ActionScript instead of simple strings. Effective use of data binding will become increasingly important as you begin to work with server-side data.

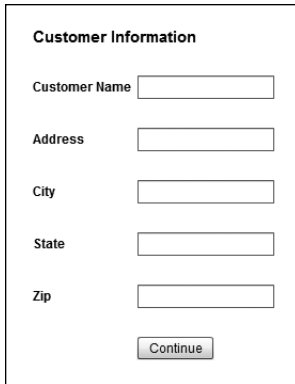


6 Save and run the application.

You should see the description you entered in the data model when you roll the cursor over the grocery item.

Using a Form Layout Container to Lay Out Simple Controls

Forms are important in most applications that collect information from users. You will be using the Form container to enable shoppers to check out their products from the grocery store. The Form container in Flex will handle the layout of the controls in this form, automating much of the routine work. With a Form container, you can designate fields as required or optional, handle error messages, and perform data checking and validation to be sure the administrator follows designated guidelines. A Form container uses three tags: an `<s:Form>` tag, an `<s:FormHeading>` tag, and an `<s:FormItem>` tag for each item on the form. To start, the checkout form will be built into a separate application, but later in the book, it will be moved into the main application as a custom component.



A form titled "Customer Information" with five text input fields and a "Continue" button. The fields are labeled "Customer Name", "Address", "City", "State", and "Zip".

Customer Information

Customer Name

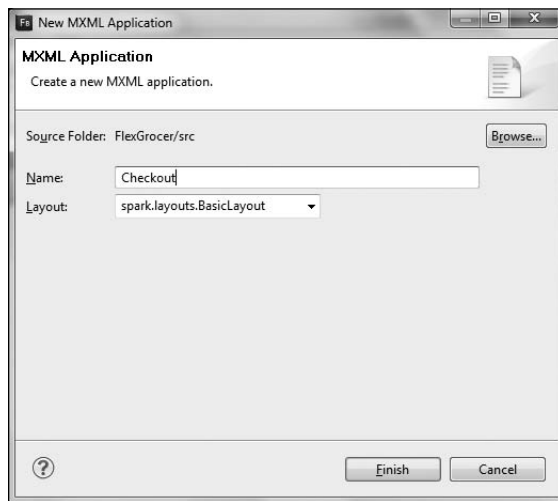
Address

City

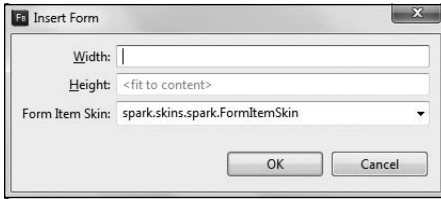
State

Zip

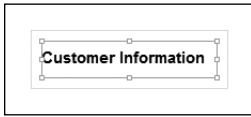
- 1 Create a new MXML application in your current project by choosing File > New > MXML Application. Name the application **Checkout**, and choose spark.layouts.BasicLayout as the layout for the new application. Then click Finish.



- 2 Switch to Design view, and drag a Form from the Layout folder of the Components view to the top left of the window. A dialog box will appear asking for the Width and Height of the form. Leave the default values and click OK.

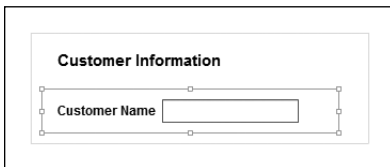


- 3 Drag a FormHeading component from the Layout folder in the Components view into the newly created form. Double-click the FormHeading, and change it to **Customer Information**.



A FormHeading is just a specialized label for Forms.

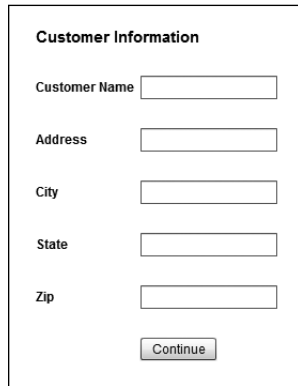
- 4 Drag a TextInput control from the Controls folder of the Components view and drop it just below the FormHeading. The TextInput and a label to the right of the TextInput both appear. Double-click the label and change it to **Customer Name**.



When adding controls to a form in Design view, Flash Builder automatically surrounds the control in a FormItem, which is why a label is appearing to the left of the control. If you switch to Source view, you can see the FormItem surrounding the TextInput. Back in Design view, notice how the left edge of the text input's label is aligned with the left edge of the FormHeading. As noted earlier, this is a feature of the Form and FormHeading classes, and it allows these items to always maintain the left alignment, regardless of the size of the FormItem labels.

- 5 Drag four more TextInputs to the form from the Components view. Change the labels of these to **Address**, **City**, **State**, and **Zip**. Drag a button below the last TextInput, and set its label to be an empty string (simply remove the default text). Click the button and change the button's text to **Continue**.

Due to the Form layout, selecting a discrete control such as the Button can be difficult. In this case, it is easiest if you attempt to click the very left side of the button. Remember, if you can't accomplish the desired effect in Design view, you can always do so in Source view.



Customer Information

Customer Name

Address

City

State

Zip

Each control is surrounded in its own FormItem and has its own label. Since you don't need a label next to the Continue button, you simply clear the text from the label on that form item.

- 6 Save and run the application.

What You Have Learned

In this lesson, you have:

- Learned how to load images at runtime with the Image control (pages 81–84)
- Learned how to display blocks of text (pages 85–87)
- Learned how to link simple controls to an underlying data structure with data binding (pages 88–89)
- Learned how to build user forms with a minimum of effort using the Form container (pages 89–92)

This page intentionally left blank

Index

- @ (attribute) operator, 131
 - { } (braces), 81, 89, 98, 176–177
 - : (colon), 23
 - = (equal sign), 23
 - > (greater-than sign), 30
 - ? (question mark), 146
 - " (quotation marks), 68, 97
 - / (slash), 30, 31
 - _ (underscore), 177
 - " ; (escape code), 68
 - .. (descendant) operator, 132
 - . (dot) operator, 130
- A**
- AATC (Adobe Authorized Training Center), xxii
 - absolute positioning, 58, 59
 - ACA (Adobe Certified Associate), xxii
 - acceptDragDrop() method, 322
 - ACE (Adobe Certified Expert), xxii
 - ACI (Adobe Certified Instructor), xxii
 - action item controls, 303
 - ActionScript
 - Array instances, 184–185
 - classes, 142, 143, 427–429
 - components (See ActionScript components)
 - dispatching events in, 178–179
 - Drawing API, 404
 - event handling with, 98–99
 - and Flash Platform, 12, 13
 - for loops, 162
 - power of, 12
 - purpose of, 13
 - triggering validation from, 363
 - XML support, 127
 - ActionScript components, 419–452
 - adding functionality to, 439–450
 - choosing base class for, 426
 - complexity of building, 420
 - creating visuals for, 432–438
 - defining, 424–431
 - overview, 420–422
 - reasons for making, 422–424
 - specifying skin requirements for, 432–434
 - types of, 421
 - ActionScript Virtual Machine (AVM), 13
 - addData() method, 322
 - addEventListener() method, 275
 - addInts() method, 40
 - addItem() method, 157, 158, 160, 166–167
 - Add To Cart button, 158, 161, 167
 - addToTextArea event handler, 108
 - Adobe
 - certification levels, xxii
 - Certified Associate exams, xxi
 - Community Help, xx–xxii
 - Creative Suite, 14
 - Developer Connection, xxi
 - Flash (See Flash)
 - Flex (See Flex)
 - Labs, xxii
 - Marketplace & Exchange, xxii
 - open source site, 81
 - TV, xxi
 - Adobe Advanced Training series*, xviii
 - Adobe Flex 4.5 MXML and ActionScript Language Reference* (ASDoc), 41, 79, 208, 378
 - Adobe Integrated Runtime (AIR), 12
 - Adobe Training from the Source series*, xviii
 - AIR (Adobe Integrated Runtime), 12
 - AJAX (Asynchronous JavaScript and XML), 8, 9–10
 - anchors, layout, 59
 - animation tools, 12
 - API (application programming interface), 79
 - Application container, 46
 - application files. *See also* application applications
 - basic elements of, 22
 - comparing versions of, 31–32
 - creating, 21–22
 - naming, 21
 - organizing, 18–24
 - saving, 31
 - viewing, 18, 21–22
 - viewing errors in, 32–33
 - application programming interface (API), 79
 - applications. *See also* application files
 - building detail view of, 85–87
 - changing internal structure of, 73–75
 - controlling view states for, 68–70
 - customizing with skins, 399–417 (*See also* skins)
 - customizing with styles, 375–397 (*See also* styles)
 - debugging, 34–41
 - desktop, 4–5, 311
 - displaying images in, 81–84
 - displaying/managing data for, 144–150, 292
 - dividing into modules, 207
 - embedding fonts in, 388, 390
 - enterprise server, 10
 - evolution of, 3, 4–6
 - improving architecture of, 213–221
 - laying out interface for, 50–58
 - mainframe, 4
 - maintainability of, 5, 8

- minimum height/width for, 24
 - refactoring, 71–75, 101–103
 - rich Internet (*See* RIAs)
 - running, 28–33
 - saving, 187
 - tightly coupled, 262
 - viewing hierarchy of, 57
 - web (*See* web applications)
 - Web 2.0, xii
 - working with view states for, 63–70
 - Application tag, 22–23, 31
 - architecture
 - client-side, 95
 - loosely coupled, 262–263
 - model-view-controller (MVC), 88, 212–213
 - page-based, 4, 6–7, 8
 - service-oriented, 5
 - arguments, 147
 - ArrayCollection, 184–203
 - and cursors, 198
 - and data binding, 184–185
 - filtering items in, 202–203
 - populating, 185–192
 - sorting items in, 194–198
 - using data from, 192–193
 - vs. ArrayList, 246
 - ArrayList, 185, 246
 - Array notation, 192
 - arrays. *See also* ArrayCollection
 - and data binding, 183–184
 - determining number of items in, 202–203
 - for shopping cart items, 159–167
 - sorting items in, 194–198
 - using items in, 192–193
 - AS3 Drawing API, 404
 - ASDoc (*Adobe Flex 4.5 MXML and ActionScript Language Reference*), 41, 79, 208, 378
 - aspect ratio, 83
 - asynchronous component model, 439
 - Asynchronous JavaScript and XML (AJAX), 8, 9–10
 - attribute (@) operator, 131
 - attributes, 23, 30, 98
 - Auto-Generation, 181–183
 - AVM (ActionScript Virtual Machine), 13
- B**
- base class, 426
 - BasicLayout object, 47, 58, 90
 - [Bindable] metadata tag, 135–136, 145, 149, 157, 173, 176–177
 - bindings, 98. *See also* data binding
 - BitmapFill class, 408
 - boldface text/code, xvi
 - Boolean values, 151–152, 163
 - BorderContainer, 46
 - braces ({}), 81, 89, 98, 176–177
 - Breakpoint Properties view, 40
 - breakpoints
 - enabling conditional, 40
 - removing, 39, 107
 - setting, 34, 35, 104, 187
 - turning on/off, 39
 - Breakpoints view, 36
 - browsers, web, 4, 5
 - bubbling, event, 274–279
 - Build Automatically option, 28
 - buildProduct() method, 151, 152, 199
 - business applications
 - evolution of, 4–6
 - maintainability of, 5, 8
 - role of computers in, 3
 - button_clickHandler() method, 35, 37
 - Button control, 57
 - buttons
 - creating skin for, 400–404
 - customizing for different states, 407–413
- C**
- calculateSubtotal() method, 156, 204
 - calculateTotal() method, 164–165
 - camel case syntax, 382
 - capture phase, event flow, 275
 - CartGrid component, 295–296
 - Cascading Style Sheets. *See* CSS
 - case-sensitivity, 21, 50, 68
 - categories, filtering products by, 257–259
 - categories property, 231
 - CDATA (character data) blocks, 99, 102, 110
 - CD-ROM, xvii, xviii
 - centralized data processing, 4
 - change events, 258, 287–288, 442
 - change flags, 440
 - character data (CDATA) blocks, 99, 102, 110
 - Checkout application, 89–92
 - Checkout button, 56, 59, 345
 - checkout form, 89–92
 - checkout process, 335–358
 - adding billing info page to, 345–350
 - creating OrderEvent object for, 355–358
 - creating OrderInfo object for, 335–336
 - creating review page for, 350–355
 - creating user views for, 337–345
 - overview, 335
 - validating postal codes during, 369–372
 - CheckoutView component, 337–345
 - children, 46, 57, 128
 - child tags, 49
 - classes, 141–167. *See also* specific classes
 - and ActionScript, 142, 143
 - basics of building, 143
 - as blueprint for objects, 141
 - categories of, 274

- classes (*continued*)
 - constructors of, 143
 - creating, 141, 154, 427–429
 - custom, 429–431
 - defining, 49–50, 143
 - naming, 143
 - properties of, 143, 147
 - reference document for, 41
 - vs. properties, 49–50
- class hierarchy, 208–209
- class instances, 49
- class keyword, 145
- class selectors, 382, 383, 384
- clickHandler() function, 99, 100
- client/server applications, 3, 4–5, 7
- client-side architecture, 95
- clone() method, 271
- code blocks, xvii
- code completion, 109, 110, 150
- code hinting, 29, 30, 98
- code line numbers, 27
- ColdFusion, xiv
- CollectionEvent class, 287–288
- collections
 - examples of, 194
 - filtering items in, 202–203
 - sorting items in, 194–198
 - using cursors in, 198–199
- colon (:), 23
- color
 - background, 376, 382, 393, 394, 404
 - highlight, 375, 402
 - label, 391, 392, 393
 - logo, 54
 - rollover, 379–381, 382, 383
 - text, 378
- colorName label, 171
- columns
 - in DataGrid, 292, 294, 297, 299
 - in layout objects, 46
- commitProperties() method, 440, 441, 443
- Community Help application, xx–xxii
- compiler, 176–183
- compiler arguments, 142
- compile-time errors, 33, 150
- components, 207–238. *See also* specific components
 - ActionScript (*See* ActionScript components)
 - advantages of, 212
 - applying styles to, 379–381
 - broadcasting events from, 263
 - changing look of, 400–404, 432–438
 - complexity of building, 420
 - creating, 105, 209–212, 230–238
 - creating directory of reusable, 221–230
 - declaring events for, 267–269
 - defining, 46, 424–431
 - drag-and-drop, 313
 - facilitating use of, 212
 - generic, 425
 - hierarchy of, 208–209
 - list-based, 321
 - to manage loading data, 230–238
 - and MVC architecture, 212–213
 - MXML, 208–213, 420
 - non-visual, 230–238
 - positioning, 46, 55
 - purpose of, 207
 - specifying skin requirements for, 432–434
 - types of, 421
 - visual, 213–230, 274
- Components view, 54
- composed containers, 72–73
- computer applications. *See* applications
- conditional breakpoints, 40
- configuration files, 23
- constraint-based layouts, 55, 58–63
- ConstraintLayout object, 47
- constructors, 143, 147, 148, 155–156
- containers
 - combining layout objects and, 48
 - composed, 72–73
 - finding, 65
 - positioning elements in, 58–59
 - purpose of, 46
 - size considerations, 62
 - types of, 46–47
- control bars, 51, 53, 57
- controllers, 212
- controls
 - accessing properties for, 81
 - APIs for, 79
 - assigning IDs to, 81
 - positioning, 64
 - simple (*See* simple controls)
- Cookbooks, xxi
- cookies, 7
- copyright label, 60
- createCursor() method, 199
- creationComplete event, 107–111
- Creative Commons License, 388
- CreditCardInfo component, 345–350
- cross-domain policy files, 122
- CSS (Cascading Style Sheets)
 - how namespaces are defined by, 389
 - inheritance, 381
 - standards, 384
 - styling with, 377–378, 382
- CSS files
 - creating SWF files from, 395–396
 - setting styles via, 386–394
 - sharing between applications, 386
- curly brackets ({}). *See* braces
- CurrencyFormatter class, 362–363, 364–368, 443
- currentState property, 68
- cursors
 - defined, 198
 - general steps for using, 198–199
 - refactoring to search with, 198–201
 - removing items with, 201–202
- custom classes, 429–431

- custom components. *See also* components
 - advantages of, 212
 - example, 210–211
 - facilitating use of, 212
 - and MVC architecture, 212–213
 - steps for creating, 209–210
 - ways of using, 207, 282
- custom event classes, 269–270
- D**
- data. *See also* datasets
 - allowing users to interact with, 292
 - in `ArrayCollections`, 192–193
 - creating components to manage loading, 230–238
 - debugging, 149
 - from event objects, 100–103
 - externalizing, 114–116
 - filtering, 202–203
 - manipulating shopping cart, 159–167
 - modifying, 109–111
 - nodes vs. attributes, 186
 - passing, when calling event handler, 99–100
 - passing, with events, 269–270
 - retrieving, 108, 120, 126
 - security issues, 122–123
- data binding, 169–205
 - and arrays, 183–184
 - binding simple controls using, 81
 - breaking complex, 173–176
 - breaking simple, 170–173
 - curly bracket syntax for, 81, 89 (*See also* braces)
 - as events, 179–181
 - and Flex formatters, 362, 363
 - implications of, 183–184
 - importance of, 170
 - linking data structure to simple control using, 88–89
 - populating text controls using, 81
 - proxying strategy, 184–185
 - purpose of, 88
 - two-way, 334
- `dataChanged` event, 249
- `dataForFormat()` method, 322
- `DataGrid`, 291–309
 - adding, 65–67
 - considerations, 292
 - default behavior of, 299
 - displaying shopping cart with, 292–308
 - dragging/dropping between List and, 315–321
 - purpose of, 292
- `DataGroup`, 245–255
 - creating custom renderer for, 450–452
 - creating `ProductList` from, 282–285
 - enabling virtualization for, 254–255
 - implementing `itemRenderer`, 246–251
 - purpose of, 242
 - simple example, 245–246
 - using in `ShoppingView`, 251–253
 - vs. `Group` class, 245
- data models, 88
- data nodes, 186
- data persistence, 15
- `dataProvider` property, 242
- `DataRenderer` class, 251, 256, 450
- datasets
 - defined, 241
 - populating List control with, 242–245
 - using `DataGroup` with, 245–246
 - virtualization of, 254
- data structures, 81
- data transfer objects (DTOs), 143
- debugger, 34–41, 104
- debugging. *See also* breakpoints; errors
 - data binding, 171–172, 175
 - data structures, 149
 - example, 34–41
 - and Local History feature, 31–32
 - rolling back to previous versions, 31–32
- Debug perspective, 105
- Debug view, 35, 38–39
- Declarations tag, 29
- `DefaultItemRenderer` class, 246
- default state, 63
- descendant (`.`) operator, 132
- descendant selectors, 382, 390–391
- Design button, 22
- Design view, 25, 53–56
- desktop applications, 4–5, 311
- detail view, 85–87
- DHTML (Dynamic HTML), 5, 9
- `dispatchEvent()` method, 263, 270
- dispatching events, 178–179, 183, 263–267
- display list, 274
- `DisplayObjects`, 274
- Document Object Model (DOM), 5
- `doDrag()` method, 322
- DOM (Document Object Model), 5
- dot-com boom, 7
- dot operator, 130
- double quotes (`"`), 97
- Drag and Drop Manager, 311, 312–313
- `dragComplete` event, 315
- `dragDrop` event, 316
- drag/drop operations, 311–331
 - between `DataGrid` and List, 315–321
 - and HTML, 4–5
 - phases, 312
 - in shopping cart, 326–331
 - terminology, 312
 - between two lists, 313–315
 - using non-drag-enabled component in, 321–326
- `dragEnabled` property, 313, 314, 315
- `dragEnter` event, 316
- `DragEvents`, 101

dragExit event, 316
 drag initiator, 312, 315
 DragManager class methods, 322
 dragOver event, 316
 drag proxy, 312
 drag source, 312
 DragSource class, 312, 314, 315, 322
 Drawing API, AS3, 404
 drawRect() method, 406
 dropEnabled property, 313, 315
 drop target, 312, 316
 DTOs (data transfer objects), 143
 dumb terminals, 4
 Dynamic HTML (DHTML), 5, 9
 dynamic interfaces, 71
 dynamic XML data, 133–137

E

E4X (ECMAScript for XML), 127–133
 Eclipse platform, 14, 17, 36
 Eclipse project, 10, 14
 ECMAScript for XML (E4X), 127–133
 ECMA standard, 127
 e-commerce application. *See also* FlexGrocer application
 laying out, 50–58
 using drag and drop in, 311–331
 working with view states in, 63–70
 editors
 defined, 24
 example, 22
 expanding, 25
 inline, 296–299
 opening/closing, 24
 showing code line numbers in, 27
 viewing errors in, 32–33
 element selectors, 382
 embedded fonts, 388, 390
 embedded XML, 114–119
 @Embed directive, 84
 end users, 8

enterprise server applications, 10
 equal sign (=), 23
 error messages, 89
 errors. *See also* debugging
 and Build Automatically option, 28
 how Flash Builder reports, 27
 viewing, 32–33
 escape code (" ;), 68
 event-based programming model, 95, 96–97
 event bubbling, 274–279
 Event class, 100, 269–270
 event dispatchers, 96. *See also* dispatching events
 event flow, 275
 event handlers
 defined, 96
 naming, 104, 285
 passing data when calling, 99–100
 sending event objects to, 101–103
 for system events, 107–111
 event handling
 with ActionScript, 98–99
 example, 97–98
 overview, 96–97
 EventLab application, 274
 event listeners, 96, 179, 275, 288
 event objects. *See also* events
 generic, 101
 inspecting, 104–107
 using data from, 100–103
 events, 100–107. *See also* event handling; event objects
 communicating with, 445–448
 data binding as, 179–181
 declaring, 267–269
 defined, 100
 dispatching, 178–179, 183, 263–267
 drag initiator, 315
 drop target, 316
 inspecting, 104–107
 interpreting, 212

 listening to, 96, 179
 passing data with, 269–270
 purpose of, 80
 types of, 96, 107
 using data from, 100–103
 event subclasses, 270–274, 280
 event targets, 275
 eventText parameter, 108
 event variable, 105
 expressions
 E4X, 129, 133
 and loose coupling, 262
 maintaining, 262
 watch, 36, 117, 119, 136, 172, 191
 Expressions panel, 173, 191
 Expression Studio, 11
 Expressions view, 36, 118, 137, 191
 Extensible Application Markup Language (XAML), 11

F

factory methods, 150
 false/true values, 151–152, 163
 FAQs (frequently asked questions), xxi
 FedEx shipment tracking application, 8
 fill property, 408
 filterCollection() method, 258
 filtering, 202–203
 filterOrganic() method, 203
 findAny() method, 200
 findFirst() method, 199–200
 findLast() method, 200
 Flash Builder
 adjusting layout of views in, 26
 basic vocabulary for, 18
 creating projects in, 18–24
 debugger, 34–41
 deleting projects in, 41–42
 displaying code line numbers in, 27
 enabling conditional breakpoints in, 40
 help/support, xxi

- importing projects into, 41
 - laying out interface in, 45
 - naming projects in, 18–19
 - and object-oriented best practices, 95
 - purpose of, 14, 17
 - using Auto-Generation with, 181–183
 - versions, 14
 - viewing/correcting errors in, 32–33, 34–41
 - workbench, 24–27
- Flash Catalyst, 14
- Flash Debug perspective, 26
- Flash Platform, 12–15
- Flash Player
- and AIR, 12
 - and application height/width, 24
 - compilation process, 14
 - evolution of, 12
 - how it works internally, 439
 - popularity of, 12
 - and runtime-loaded CSS files, 395
 - sandboxes, 122–123
 - as single-threaded virtual machine, 439
 - versions, 13–14
- Flash Professional, 12–13
- Flash Text Engine, 81
- Flex
- application architecture, 212–213
 - application development, 18–24
 - applications (*See* applications)
 - basic vocabulary for, 18
 - Community Help, xx–xxii
 - compiler, 176–183
 - as component-based development model, 207
 - components, 421–422
 - event-based programming model, 95, 96–97
 - getting started with, xii–xiii, 17–43
 - home page, xxii
 - key technologies, 13–14
 - language tags, 23
 - and object-oriented programming, 41
 - positioning of components in, 46
 - purpose of, xii, 13
 - resources, xx–xxii
 - Spark components, 23
 - versions, xii, 13
 - working with view states in, 63–70
- FlexGrocer application
- adding events to, 95
 - building checkout process for, 335–358
 - controlling view states for, 68–70
 - creating list of products for, 185–191
 - creating/organizing files for, 18–24
 - customizing with skins, 399–417 (*See also* skins)
 - customizing with styles, 375–397 (*See also* styles)
 - defining product section for, 57–58
 - displaying images in, 81–84
 - displaying/managing data for, 144–150
 - externalizing data in, 114–116
 - formatting currency for prices in, 364–369
 - implementing checkout process for, 333–358
 - implementing drag/drop in, 326–331
 - improving architecture for, 213–221
 - laying out interface for, 50–58
 - manipulating shopping cart data for, 159–167
 - modifying, 30–33
 - overview, xv
 - Product data structure for, 141
 - providing product/category information for, 230–238
 - refactoring, 73–75, 101–103
 - running, 28–29
 - validating postal codes for, 369–372
 - visual shopping cart for, 169 (*See also* shopping carts)
 - website for, xvi
 - working with view states for, 63–70
- flexgrocer.com, xvi
- Flex Properties view, 82–83
- Flex SDK, xii, xiii, 14, 19, 23, 80
- flow-driven programming model, 96
- fonts, embedded, 388, 390
- for loops, 162–163, 164–165
- Formatter classes, 364–368
- formatters
- for displaying currency, 364–368, 443
 - examples of, 362–363
 - purpose of, 361, 363
- Form container, 47, 81, 89–92
- form fields, 89
- FormHeading component, 91
- FormItem component, 90
- FormLayout object, 47
- forms, 89, 91
- Forums, Adobe, xxii
- FreeBSD, 11
- frequently asked questions (FAQs), xxi
- functions. *See also* specific functions
- parameters vs. arguments for, 147
 - private vs. protected, 267
 - scope of, 102
 - vs. methods, 144
- </fx:Declarations> tag, 29
- FXG Graphics, 404–405
- FXG specification, 407
- <fx:Metadata> tag, 267–268
- <fx:Model> tag, 88
- fx namespace, 23
- FXP files, 34, 41
- <fx:Script> block, 99
- <fx:Style> tag, 382–383
- <fx:XML> tag, 149, 151

G

Generate Getter/Setter dialog box, 182, 249, 428
 getItemAt() method, 192–193
 getItemInCart() method, 163
 getters/setters, 177–178, 181–183, 249, 428
 Google Maps, 10
 “go to definition” shortcut, 171
 graphical elements, 46, 404–405
 graphics editing programs, 404–405
 graphics property, 404–405
 greater-than sign (>), 30
 GridColumn class, 296–299
 Group container, 46, 245
 Group tag, 48–49

H

handleCreationComplete() method, 109, 149, 152
 handleViewCartClick() method, 102, 103, 104
 hasFormat() method, 322
 HGroup container, 72–73
 HorizontalLayout object, 47
 HostComponent metadata, 433
 HTML (Hypertext Markup Language)
 and drag/drop operations, 5
 latest version of, 10
 limitations of, 4–5, 10
 as page-based architecture, 4
 HTML 5 specification, 10
 HTTP (Hypertext Transport Protocol), 5, 7
 HTTPServices
 accessing data retrieved from, 121–122
 creating, 120, 230–231
 retrieving XML data via, 124–126
 Hypertext Markup Language.
 See HTML
 Hypertext Transport Protocol (HTTP), 5, 7

I

IDataRenderer interface, 246, 247
 IDE (integrated development environment), 14
 ID selectors, 382, 392
 IFill interface, 408
 IFormatter interface, 362
 IFrames, 9
 Image control, 81–84
 images
 displaying, 81–84
 editing, 404–405
 loading at runtime, 81
 loading at start of application, 84
 scaling, 83
 implicit getters/setters, 177–178, 181–183
 importing
 classes, 109–110, 215
 projects, 41, 68
 import statements, 110, 157, 304
 includeIn property, 67, 87, 334
 inheritable style properties, 381
 inheritance
 hierarchy, 263
 and invalidateProperties() method, 440
 and protected functions, 267
 style, 376, 381, 384
 inline editors, 296–299
 inline item renderers, 303–304
 inline styles, 379–381
 instance methods, 151
 instances
 Array, 184, 194
 bindable, 194
 DataGroup, 242
 event, 276, 277
 Label, 193
 List, 242, 313
 Object, 184
 ObjectProxy, 118, 135
 validator, 371
 vs. properties, 49
 XML, 184

integers, unsigned, 155
 integrated development environment (IDE), 14
 interactivity, 5, 63
 interfaces, 199. *See also* UI
 Internet
 applications (*See* web applications)
 dot-com boom, 7
 explosive growth of, 4
 security issues, 122–123
 invalidateProperties() method, 440
 isItemInCart() method, 163
 italics, xvii
 itemEditor property, 297
 item editors, 296–299
 itemRenderer class, 246–251, 256
 item renderers
 for displaying products, 299–302
 how they work, 246
 implementing, 246–251
 inline, 296–299, 303–304
 items. *See also* products
 adding to shopping cart, 159–161
 displaying based on category, 257–258
 dragging to shopping cart, 311, 326–331
 finding in shopping cart, 161–163
 updating quantity of, 161, 164–166
 IT organizations, 8
 IValidator interface, 363
 IViewCursor interface, 198, 199

J

Java, xiv
 Java applets, 9
 JavaFX, 10
 JavaScript, 5, 9
 Java Virtual Machine, 10
 just-in-time (JIT) compilation, 14

K

keyboard shortcuts, xvii

L

Label component, 378

Label controls, 57, 60, 80

LabelFunction property

displaying subtotal with,
305–308

purpose of, 242

using with lists, 242–245

Label property, 49

Label tag, 29

LastResult property, 121–122

layout anchors, 59

layout objects, 46, 47–48

Layout property, 48

layouts, 45–63. *See also* containers
adding elements in Design view,
53–56

combining containers and, 48

constraint-based, 55, 58–63

for e-commerce application,
50–58

purpose of, 46

starting in Source view, 51–53

lessons

directory structure for, xviii

list of, xiv–xv

minimum system requirements,
xix

standard elements used in,
xvi–xvii

Letterbox scale mode, 83, 84

Linear Gradient class, 408–409

line breaks, 31

Linux, 11

list-based components, 321

List controls. *See also* lists

displaying category data with,
137–139

dragging/dropping between,
313–315

dragging/dropping between
DataGrid and, 315–321

populating with dataset,
242–245

using LabelFunction with,
242–245

lists. *See also* List controls

formatting data in, 244–245

responding to user's choice
from, 257–259

virtualization with, 255–257

Local History feature, 31–32

loops, 162–163, 164–165

loosely coupled architecture,
262–263

M

Mac OS-based systems

manifest files for, 23

and Silverlight, 11

system requirements, xix

Macromedia, xii, 12, 13

mainframes, 4

Major League Baseball application, 8

manifest files, 23

menu commands, xvii

messaging, 15

metadata tags, 267

methods. *See also* specific methods

creating objects with, 150–153

defining, 143

DragManager class, 322

DragSource class, 322

factory, 150

instance, 151

overriding, 271

parameters vs. arguments for, 147

static, 151

vs. functions, 144

microcomputers, 4

Microsoft

Expression Studio, 11

Silverlight, 11

minHeight attribute, 31

minWidth attribute, 31

MLB.TV Media Player, 8

models, 212

model-view-controller (MVC)
architecture, 88, 212–213

Moonlight, 11

mouseDown event, 315

MouseEvent properties, 104–107

mouseMove event, 315

mouseout event, 87

mouseover event, 87

multiDisplay() function, 244

MVC (model-view-controller)
architecture, 88, 212–213

mx components, 20

mx.formatters package, 362

MXML

case-sensitivity of, 21

class instances vs. properties in,
49–50

compiling, 149

components, 208–213, 420
(*See also* components)

creating applications in, 18–24

creating classes in, 149

decoding tags in, 49–50

formatting rules/standards, 30

item renderers, 303–304

purpose of, 13

MXMLC compiler, 395

MXML Component dialog box, 214

mx.validators package, 363–364

N

name collision, 147, 148

@namespace declaration, 389

namespaces

fx namespace, 23

how CSS defines, 389

s namespace, 23

Spark, 23, 387, 389

styles and, 383–384

navigation system

importance of, 333

and loose coupling, 262–263

using states as basis for, 334,
337–345

nested quotes, 97, 99

.NET, xiv, 11
 new keyword, 151
 New Package command, 221
 New State icon, 64
 newTotal variable, 164, 165
 nodes, 128, 186
 non-visual components, 230–238

O

object-oriented programming (OOP), xiv, 41, 95, 102, 141, 215
 ObjectProxy, 184
 objects
 building method to create, 150–153
 classes as blueprint for, 141
 converting XML to, 117–119, 133
 data transfer, 143
 event (See event objects)
 OOP. See object-oriented programming
 Open Font Library, 388
 Open Perspective button, 22, 26
 Open Perspective icon, 107
 open source site, 81
 OrderEvent object, 355–358
 OrderInfo object, 335–336, 355
 Outline view, 57, 65

P

Package Explorer, 22, 25, 41, 145
 package keyword, 145
 packages, 143, 145, 221
 page-based architecture, 4, 6–7, 8
 Panel container, 46
 parameters, 147
 partAdded() method, 445–448
 partRemoved() method, 445, 448–449
 PC manifest files, 23
 personal computers, 4
 perspectives, 26, 36, 107
 PHP, xiv
 postal-code validator, 369–372

prefix, 23
 prepare method, 300, 301
 private functions, 267
 private keyword, 158
 Problems view, 22, 32
 ProductEvent class, 280–286, 304–305
 ProductItem components
 breaking out, 221–230
 cleaning up appearance of, 376–377
 creating instances of, 228
 productivity, 8
 ProductList component
 creating, 282–284, 426
 styling labels in, 391
 using, 284–285
 product nodes, 186
 products. See also items
 adding/removing from shopping cart, 284–286, 304–305
 creating, 189
 displaying names of, 148–149
 filtering based on category, 257–259
 keeping track of shopping cart, 154–159
 product section, 57–58
 programmatic graphics, 404–405
 programming languages, xiv
 Project menu, 28
 projects. See also applications
 creating, 18–24
 deleting, 41–42
 importing, 41, 68
 naming, 18–19, 41
 overwriting, 41
 viewing, 21–22
 viewing errors in, 32–33
 properties. See also specific properties
 creating, 215
 declaring, 143
 vs. class instances, 49
 Properties panel, 53, 57

protected functions, 267
 proxies
 array, 184–185
 drag, 312
 pseudo-selectors, 382, 393–394
 public properties, 146–147

Q

question mark (?), 146
 Quick Fix tool, 146, 180
 quotation marks ("), 68, 97

R

RadialGradient class, 408
 redundancy, 71
 refactoring, 71–75
 applications, 73–75, 101–103
 benefits of, 71–72
 defined, 71
 to search with cursor, 198–201
 ShoppingCart class, 287–288
 ShoppingCartItem class, 204–205
 remote XML data, 110–139
 dynamic, 133–137
 embedded XML, 114–116
 searching with E4X, 127–133
 security issues, 122–123
 XMLListCollection, 137–139
 XML loaded at runtime, 119–123
 Remove button, 303–304
 removeItem() method, 304
 Reset Perspective command, 53
 ResultEvents, 101
 result handler, 122
 Resume button, 38
 return types, 99, 102
 RIAs (rich Internet applications), 3–15
 advantages of, 7–8
 and drag/drop technique, 311
 examples of excellent, 8
 functions of, 5–6
 goals of, 6–7

- technology choices, 8–14
 - vs. traditional web applications, 6–7
 - RichEditableText control, 81
 - rich Internet applications. *See* RIAs
 - RichText control, 80, 81, 86, 109
 - rollover event, 85–87
 - root nodes, 128–129
 - Run button, 22, 28, 29
 - runtime
 - changing CSS at, 395–397
 - changing skins at, 448
 - loading images at, 81, 84
 - styling at, 395–397
 - XML loaded at, 119–123
- S**
- sandboxes, 122–123
 - <s:Application> tag, 22–23, 31
 - satellite navigation system, 262–263
 - <s:BasicLayout/> tag, 52
 - scalar values, 98
 - Scale Mode menu, 83, 84
 - scope, 102
 - Script blocks, 99, 102
 - scroll bars, 24
 - Scroller tag, 48–49
 - scrolling content, 48–49
 - SDK (software development kit), xii, xiii, 14, 19, 23, 80
 - searches
 - array, 246
 - with cursor, 198–201
 - descendant, 190
 - XML (with E4X), 127–133
 - security issues, 122–123
 - security sandboxes, 122–123
 - selectability, 255
 - selectedIndex property, 255
 - selectedItem property, 255
 - self-closing tags, 31, 51
 - send() method, 120, 126
 - servers, 4. *See also* client/server applications
 - server-side languages, xiv
 - server-side objects, 6
 - server technologies, 15
 - service-oriented architecture (SOA), 5
 - setStyle() method, 381
 - setters/getters, 177–178, 181–183, 249, 428
 - <s:FormHeading> tag, 89
 - <s:FormItem> tag, 89
 - <s:Form> tag, 89
 - shipment tracking application, 8
 - ShoppingCart class. *See also* shopping carts
 - building, 154–159
 - refactoring, 287–288
 - replacing Array in, 194
 - ShoppingCartItem class, 154–159, 204–205
 - shopping carts
 - adding items to, 63, 159–161, 284–286
 - displaying with DataGrid, 291, 292–308
 - dragging items to, 311, 326–331
 - finding items in, 161–163
 - formatting list data for, 244–245
 - keeping track of items in, 154–159
 - manipulating data in, 157, 159–167
 - removing items from, 201–202, 284–286
 - updating quantity of items in, 164–166
 - updating totals in, 287–288
 - ShoppingList component, 425–452
 - adding functionality to, 439–450
 - checking functionality of, 429–431
 - choosing base class for, 426
 - creating class for, 427–429
 - creating custom renderer for, 450–452
 - creating skin for, 434–438
 - defining interface for, 425
 - specifying skin requirements for, 432–434
 - ShoppingView class, 251–253
 - Show Line Numbers command, 27
 - Show View command, 25
 - Silverlight, 11
 - simple controls, 79–92
 - linking data structure to, 88–89
 - overview, 80–81
 - purpose of, 80
 - tools for laying out, 79
 - using Form container to lay out, 89–92
 - ways of using, 79, 80
 - SkinnableComponent class, 421–422, 445
 - SkinnableContainer, 46
 - SkinnableDataContainer, 47
 - skinning, 46, 432
 - SkinPart metadata, 432
 - skin parts, 401, 434
 - skins, 399–417
 - changing at runtime, 448
 - creating, for Application component, 413–416, 434–438
 - creating, for FlexGrocer button, 400–404
 - creating, for ShoppingList component, 434–438
 - creating renderer for, 450–452
 - customizing button states with, 407–413
 - errors, 401
 - purpose of, 376
 - relationship between states and, 404–413
 - role of, in Spark components, 400–404
 - vs. styles, 376
 - SkinState metadata, 432–433
 - <s:Label> tag, 29
 - slash (/), 30, 31
 - <s:List> control, 137–138
 - s namespace, 23
 - SOA (service-oriented architecture), 5
 - software development kit (SDK), xii, xiii, 14, 19, 23, 80

- software upgrades, 4
 - SolidColor class, 408
 - someColor property, 171–172
 - SortField objects, 194–197
 - Source button, 22
 - Source view, 25, 51–53
 - Spark classes, 23
 - Spark components
 - and embedded fonts, 390
 - namespace for, 23, 387, 389
 - role of skins in, 400–404, 413
 - vs. MX components, 20
 - spark.formatters package, 362
 - Spark namespace, 23, 387, 389
 - spark.validators package, 363–364
 - <s:states> tag, 337
 - stateless protocols, 7
 - states
 - controlling, 68–70
 - creating, 64
 - creating navigation structure using, 334, 337–345
 - maintaining, 7
 - naming, 68
 - relationship between skins and, 404–413
 - setting properties for, 67
 - state selectors, 393–394
 - static methods, 151
 - Step Into button, 37, 40
 - Step Over button, 37, 40
 - stretch scale mode, 83
 - style inheritance, 381
 - StyleManager, 396–397
 - styleName property, 383
 - style properties, 379, 381
 - styles, 375–397. *See also* skins
 - assigning multiple, 382, 397
 - complete list of, 378
 - CSS inheritance for, 381
 - overriding, 397
 - purpose of, 376
 - at runtime, 395–397
 - setting with CSS files, 386–394
 - setting with <fx:Style> tag, 382–383
 - setting with setStyle() method, 381
 - setting with tag attributes, 379–381
 - vs. skins, 376
 - ways of applying, 377–378
 - subclasses, 270–274, 280
 - subtotals, 156, 204, 305–308
 - Support pages, xxi
 - SWF files, 29, 395–396
 - system events, 96, 107–111
 - system requirements, xix
- T**
- tag attributes, setting styles via, 379–381
 - tags
 - choosing attributes for, 30
 - Form container, 89
 - selecting, 30
 - self-closing, 31, 51
 - target phase, event flow, 275
 - target property, 100, 105
 - targets, event, 275
 - text
 - controls, 80–81
 - displaying blocks of, 85–87
 - styles for manipulating, 378–379
 - TextArea component, 80, 108
 - TextInput control, 80, 91, 247
 - Text Layout Framework (TLF), 81
 - this keyword, 105, 147
 - tight coupling, 262
 - TileLayout object, 47
 - timestamp property, 270–274
 - timestamps, 269
 - TLF (Text Layout Framework), 81
 - toString() method, 148–149, 150, 151, 156, 160
 - total property, 165, 166
 - trace() method, 149, 150, 153, 156
 - training centers, Adobe, xxii
 - Training from the Source* series, xviii
 - transfer objects, 143
 - true/false values, 151–152, 163
 - tutorials, xxi
 - two-way bindings, 334
 - type property, 100
 - type selectors, 382, 383–385
- U**
- UI (user interface), 45–76
 - arranging elements in, 58
 - drag-and-drop technique, 311 (*See also* drag/drop operations)
 - dynamic, 71
 - for e-commerce application, 45
 - HTML limitations, 4–5
 - laying out, 50–58
 - tools for creating, 11
 - UIComponent class, 208, 263, 421
 - underscore (_), 177
 - unique identifier (UUID), 41
 - uni tRPCResult() handler, 122
 - Universal Resource Identifier (URI), 23
 - unsigned integers, 155
 - updateItem() method, 164
 - URI (Universal Resource Identifier), 23
 - URLs, 23, 120
 - UserAcknowledgeEvent class, 270–274
 - user events, 96, 97, 107
 - user frustration level, 8
 - user input forms, 334
 - user interface. *See* UI
 - users, collecting information from, 89
 - UUID (unique identifier), 41

V

Validator classes, 363, 364, 369–372

validators
 for checking postal codes, 369–372
 examples of, 363–364
 purpose of, 361, 363

value objects, 143–150, 153

values
 attribute, 98
 Boolean, 151–152, 163
 scalar, 98
 setting/reading, 177
 true/false, 151–152, 163

variables
 controlling, 363
 integer, 35
 name collision among, 147
 naming, 147
 in RIAs, 7
 showing current state of, 36

Variables view, 36, 37–38, 105

vector graphics, 404–405

VerticalLayout object, 47, 57

VGroup container, 72–73, 85

video publishing, 15

View Cart buttons, 101–103

views
 adjusting layout of, 26
 displaying list of, 25
 grouping, 36
 in MVC architecture, 212, 213
 (See also specific views)
 opening/closing, 25

view states, 63–70
 controlling, 68–70
 creating, 63–67
 defined, 63
 naming, 68

virtualization

implementing, 254–255
 with List class, 255–257
 power of, 255
 purpose of, 253–254

visual components, 213–230, 274

void return type, 99, 102

W

WarningDialog application, 264–267

watch expressions, 36, 117, 119, 136, 172, 191

Web 2.0, xii

web applications. *See also* applications
 connectivity issues, 6
 and drag/drop technique, 311
 and event-based programming, 95
 evolution of, 4–6
 flow for traditional, 6–7
 inefficiencies of, 6
 maintaining state in, 7

web browsers, 4, 5

Web Services, 6

Window menu, 25

Windows-based systems
 manifest files for, 23
 and Silverlight, 11
 system requirements, xix

Windows Presentation Foundation (WPF), 11

workbench, 24–27

workflow engines, 15

World Wide Web, xiv. *See also* Internet

WPF (Windows Presentation Foundation), 11

X

XAML (Extensible Application Markup Language), 11

XML
 ActionScript support for, 127
 and AJAX, 9
 and code completion, 150
 converting to objects, 117–119, 133
 embedded, 114–119
 and Flex, 13
 formatting rules/standards, 30
 loaded at runtime, 119–123
 namespaces, 23
 nomenclature, 23
 terminating tags in, 31
 vs. XMLList, 128

XML class, 127

XML data
 accessing returned, 121–122
 dynamic, 133–137
 remote (See remote XML data)
 retrieving via HTTPService, 124–126
 security issues, 122–123

XMLDocument class, 127

XMLHttpRequest, 9

XMLList, 128

XMLListCollection, 133, 135, 137–139

Z

zip-code validator, 369–372

ZipCodeValidator class, 369–372