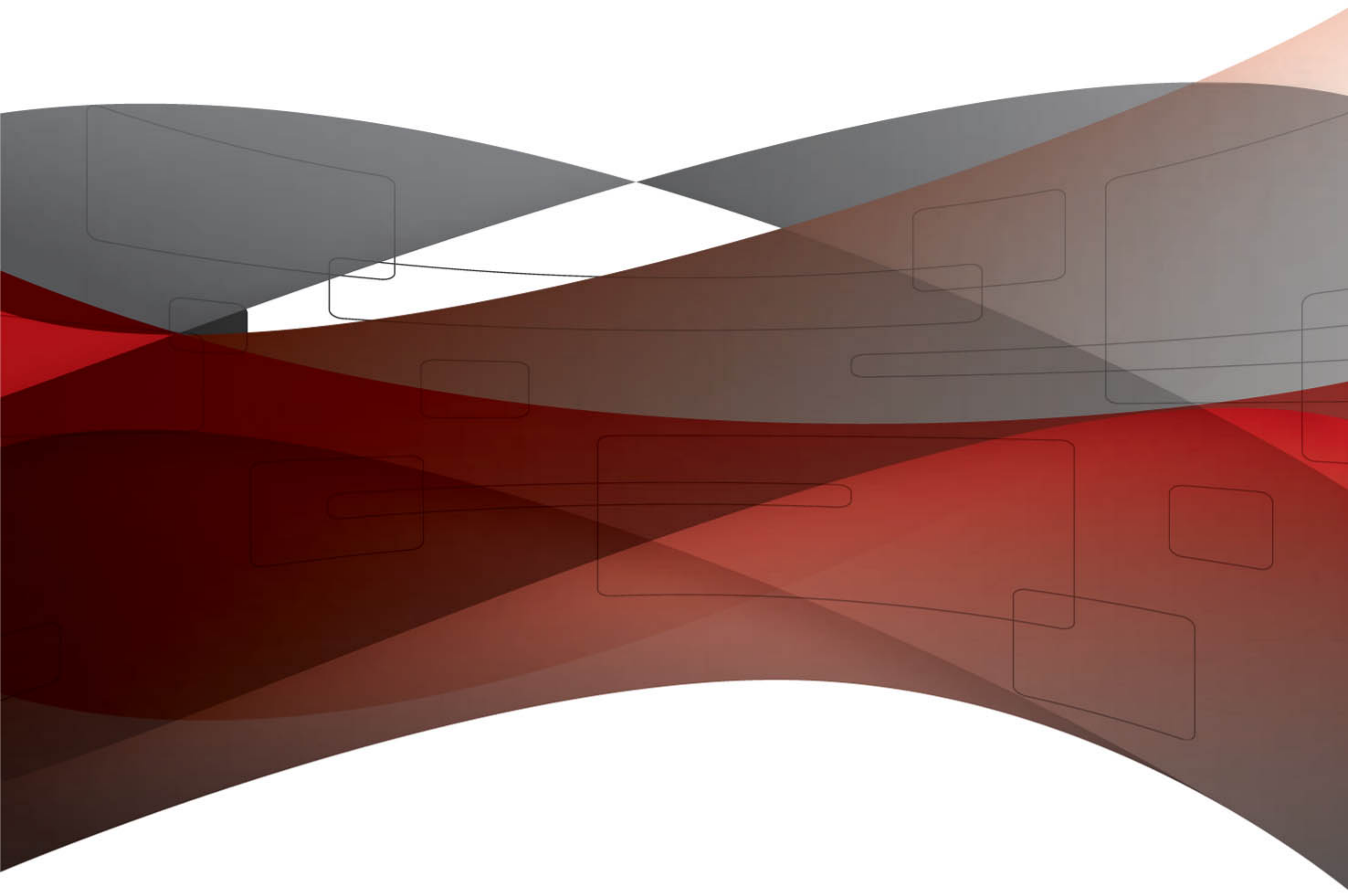


ADOBE® FLEX® 4.5 Fundamentals

Training from the Source

Michael Labriola, Jeff Tapper, and Matthew Boles

Foreword by Adam Lehman, Adobe Flash Builder Product Manager

Adobe

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

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

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

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

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*.

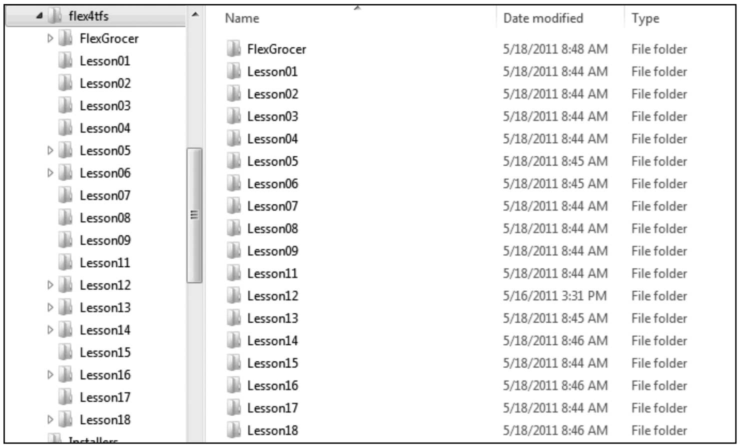
Italics are also used for placeholders, which indicate that a name or entry may change depending on your situation. For example, in the path *driveroot*:/flex4tfs/flexgrocer, you would substitute the actual name of your root drive for the placeholder.

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:



```
flex4tfs
  FlexGrocer
  Lesson01
  Lesson02
  Lesson03
  Lesson04
  Lesson05
  Lesson06
  Lesson07
  Lesson08
  Lesson09
  Lesson11
  Lesson12
  Lesson13
  Lesson14
  Lesson15
  Lesson16
  Lesson17
  Lesson18
```

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

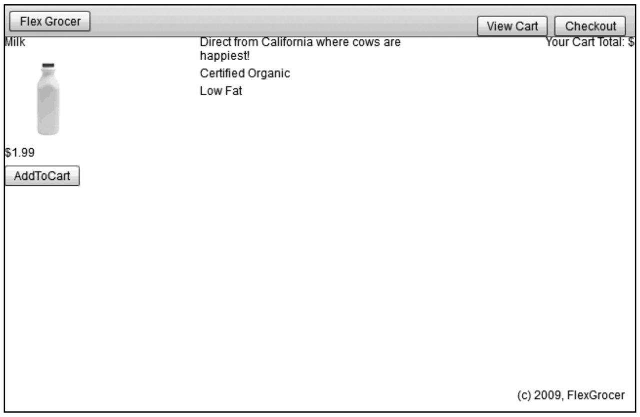
This lesson takes approximately 45 minutes to complete.

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.



Direct from California where cows are happiest!
Certified Organic
Low Fat

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.

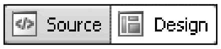
Displaying Images

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.

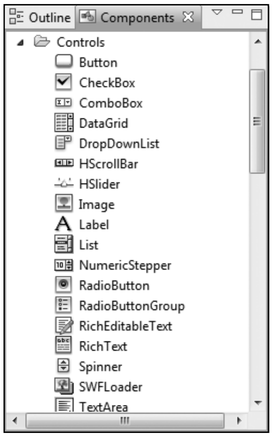
1 Open the FlexGrocer.mxml file that you created in the previous lesson.

  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.

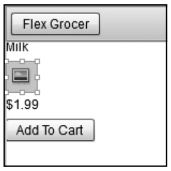
2 Switch Flash Builder to Design view by clicking the Design View button.



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

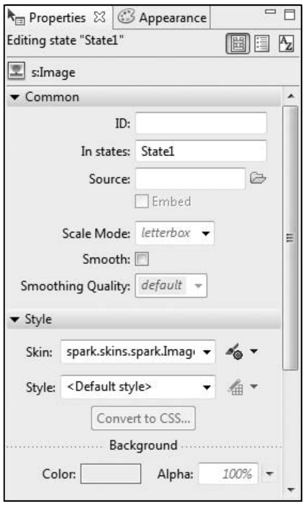


4 Select the Image control from the Controls folder and drag the control between the Milk and 1.99 Label controls you already added.



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.


Properties | Appearance
Editing state "State1"
s:Image
Common
ID:
In states: State1
Source:
Embed
Scale Mode: letterbox
Smooth:
Smoothing Quality: default
Style
Skin: spark.skins.spark.Imag
Style: <Default style>
Convert to CSS...
Background
Color:
Alpha: 100%

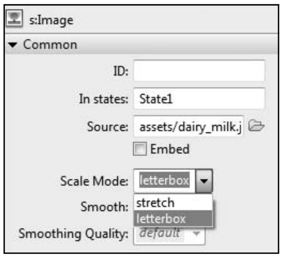
The Flex Properties view shows important attributes of the selected component—in this case, the Image control. You can see the `Source` property, which specifies the path to the Image file. The ID of the Image control references the instance created from the `<s:Image>` tag or Image class in ActionScript.

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`.

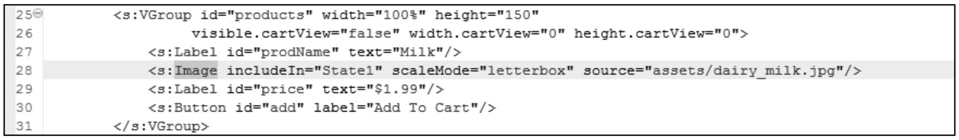
In an ideal world, all the images that you use in the application would be a perfect size, but this is not always the case. Flex can scale the images in two ways. You can choose letterbox to keep the aspect ratio of the original images correct even as their size is adjusted, or you can choose stretch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


s:Image
Common
ID:
In states: State1
Source: assets/dairy_milk.j
Embed
Scale Mode: letterbox
Smooth:
Smoothing Quality: default

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.



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

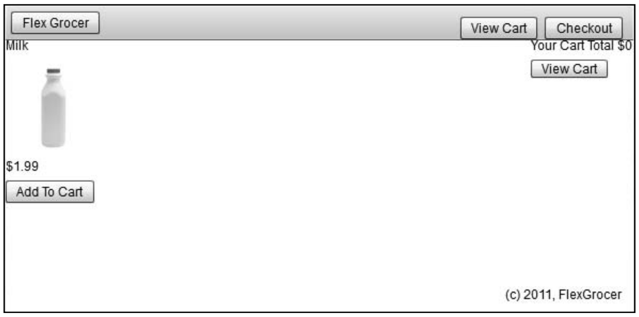
9 In the `<s:Image>` tag that you added, insert an `@Embed` directive to the Image control.

```
<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

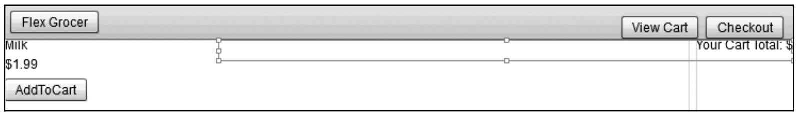
In this exercise, you will use a rollover event to display a detailed state of the application. You will explore different simple controls to display text and review how application states work.

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.

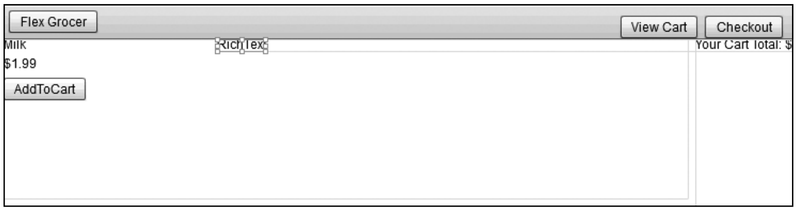


This new VGroup needs to be a child of the main application. Sometimes, positioning items correctly can be difficult in Design view, so switch to Source view and ensure the VGroup is positioned correctly. It should be just above the closing `</s:Application>` tag, so the end of the file reads like this:

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

</s:Application>
```

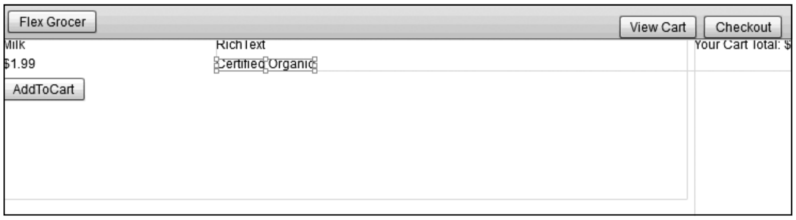
3 Switch back to Design view. Ensure that the expanded state is selected in the States view. Drag an instance of the RichText control from the Controls folder of the Components view into the new VGroup you created in the previous step.



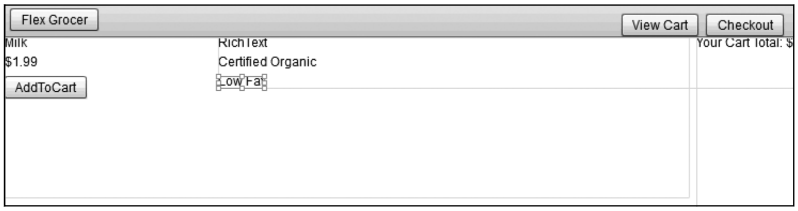
The RichText control enables you to display multiple lines of text, which you will need when you display the product description that will ultimately come from an XML file. You will use data binding in the next section to make this RichText control functional. For now, you are just setting up the layout.

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.

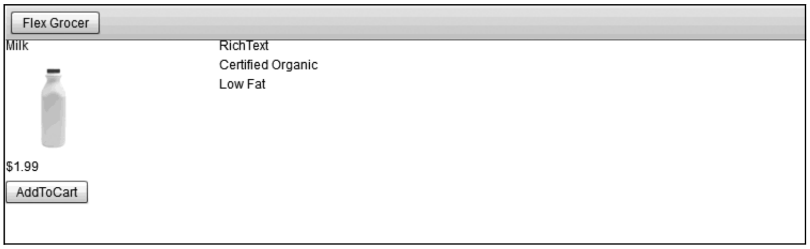
9 In the same `<s:Image>` tag, add a `mouseOut` event that will change the `currentState` back to the initial State1 state.

```
<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">
```

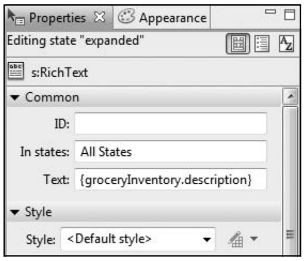
   By assigning an ID to the `<fx:Model>` tag, you can reference the data with dot syntax. For example, to access the list price of the item, you could use `groceryInventory.listPrice`. In this case, that would resolve to 1.99.

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.



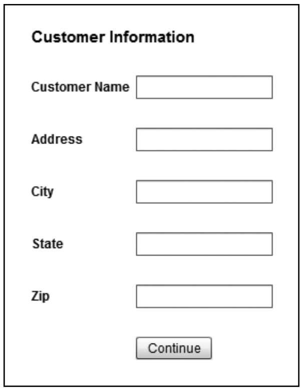
Editing state "expanded"
s:RichText
Common
ID:
In states: All States
Text: {groceryInventory.description}
Style
Style: <Default style>

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.



Customer Information

Customer Name

Address

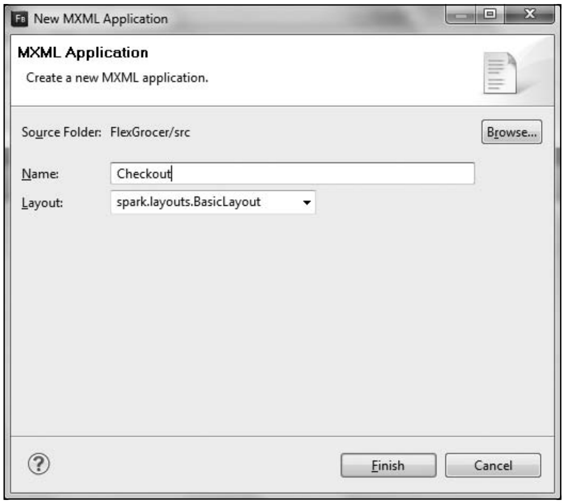
City

State

Zip

Continue

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.



New MXML Application

MXML Application

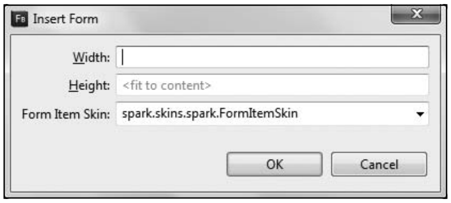
Create a new MXML application.

Source Folder: FlexGrocer/src

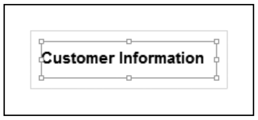
Name: Checkout

Layout: spark.layouts.BasicLayout

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.

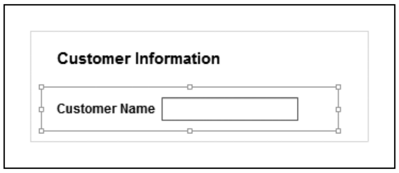


3 Drag a FormHeading component from the Layout folder in the Components view into the newly created form. Double-click the FormHeading, and change it to **Customer Information**.



A FormHeading is just a specialized label for Forms.

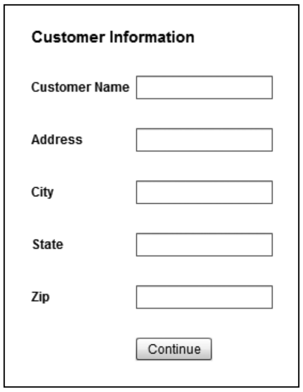
4 Drag a TextInput control from the Controls folder of the Components view and drop it just below the FormHeading. The TextInput and a label to the right of the TextInput both appear. Double-click the label and change it to **Customer Name**.



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.



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