Discover developing an application
end-to-end natively

Multimobile Development

Building Applications for iPhone and Android

Matthew Baxter-Reynolds

Apress®

Multimobile Development

Building Applications for the iPhone and Android Platforms

Matthew Baxter-Reynolds

Apress®

Multimobile Development: Building Applications for the iPhone and Android Platforms

ISBN-13 (pbk): 978-1-4302-3198-1

ISBN-13 (electronic): 978-1-4302-3199-8

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

President and Publisher: Paul Manning
Lead Editor: Jonathan Hassell
Technical Reviewer: Matthew Fitchett
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Coordinating Editor: Anita Castro
Copy Editor: Mary Ann Fugate
Compositor: Lynn L'Heureux
Indexer: Potomac Indexing, LLC
Artist: April Milne
Cover Designer: Anna Ishchenko

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

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

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

0x34 0x4D 0x2B 0x45

Contents at a Glance

About the Author xiii

About the Technical Reviewer xiv

Acknowledgements xv

Chapter 1: Introduction 1

Chapter 2: The Six Bookmarks Server Service 5

Chapter 3: Application Architecture and Functional Specification 19

Chapter 4: Android: Installing the Toolset 35

Chapter 5: Android: Building the Logon Form and Consuming REST Services 55

Chapter 6: Android: An ORM Layer on SQLite 93

Chapter 7: Android: Pushing Changes to the Server 155

Chapter 8: iOS: Installing the Toolset 211

Chapter 9: iOS: Building the Logon Form and Consuming REST Services 237

Chapter 10: iOS: An ORM Layer on SQLite 291

Chapter 11: iOS : Pushing Changes Back to the Server 381

Chapter 12: iOS: MonoTouch 427

Index 447

Contents

About the Author ..... **xiii**

About the Technical Reviewer ..... **xiv**

Acknowledgements ..... **xv**

Chapter 1: Introduction **1**

What's the Purpose of This Book? 2

How Is This Book Structured? 2

Where Can You Get Help and Support? 4

Conclusion 4

Chapter 2: The Six Bookmarks Server Service **5**

Creating an API Account 5

Creating a User 6

The Users Service 8

RESTful Web Services 8

Testing the Calls 9

Examining Logon Operations 9

The Bookmarks Service 12

Adding Some Test Data 12

Working with OData 13

OData Queries 17

Issuing Updates over OData 18

Constraining Data to the Logged-On User 18

Conclusion 18

Chapter 3: Application Architecture and Functional Specification 19
A Word About Slates 19
Functional Specification 19
Logging On 20
Synchronizing 20
Navigator 21
Configuring Bookmarks 22
Configuring a Single Bookmark ("Configure Singleton") 23
Missing Functionality 23
Application Architecture and Technical Specification 24
Approach 24
Object-Relational Mapping 25
Server Communication 29
Technical Approach Broken Down by Platform 30
Conclusion 33

Chapter 4: Android: Installing the Toolset 35
Why Android First? 35
Installing the Toolset 35
Installing Java 35
Installing Eclipse 36
Installing the Android SDK 36
Installing the Android Development Tools (ADT) into Eclipse 37
Configuring the Emulator 41
Creating Our Android "Hello, World" Application 44
Saying "Hello, World" 47
Declarative Layout 47
Wiring Up the Button 49
Conclusion 54

Chapter 5: Android: Building the Logon Form and Consuming REST Services 55
Creating the Project 55
Conventions for Presenting Code 56
Calling RESTful Services 57
Issuing Web Requests 57
Authenticating Our API Account 62

Authenticating the User via "UsersService" 73
Setting "Allow Internet Access" Permission 74
Creating the Logon Form 75
Model/View/Controller 79
Logging On 86
"Remember Me" 89
Conclusion 91

Chapter 6: Android: An ORM Layer on SQLite 93
SQLite on Android 93
Entities 93
"EntityType" 94
Creating the Basic "Entity" Class 100
Setting Values in an Entity 102
Building "Bookmark" 107
Registering the "EntityType" 109
Displaying Some Fake Bookmarks 110
Creating the Form 110
Showing the Bookmarks 117
Wiring Up the Bookmarks 122
Building the "Sync" Class 124
Calling the Server's Bookmarks OData Service 124
Managing the Database 132
The "SqlStatement" Class and "ISqlStatementSource" Interface 134
Creating Tables 135
Examining the Database with Sqliteman 139
Writing Bookmarks to the Database 143
Reading Bookmarks and Displaying Them on the Navigator 148
Conclusion 153

Chapter 7: Android: Pushing Changes to the Server 155
Capturing Local Changes 155
Constraining SQL Filters 155
Excluding Deleted Entities from the View 161
Getting a Bookmark by Ordinal 162
Building the Configuration Form 163
Configuring Singletons 186

Pushing Changes to the Server 194
Detecting Local Changes 194
Issuing Server Requests to Insert, Update, and Delete 198
Update via "HTTP MERGE" and Insert via "HTTP POST" 199
Marking Fields As Being Available on the Server 199
Conclusion 209

Chapter 8: iOS: Installing the Toolset 211
iPad Development 211
Installing Xcode 211
An Objective-C Primer for .NET and Java Developers 211
Problems with Objective-C 212
Calling Methods (aka "Sending Messages") 213
Properties (and a Little Memory Management) 215
Methods 221
Namespaces 223
The Biggest Gotcha in Objective-C 223
"Hello, World" for iPhone 224
Building the User Interface 225
Creating a Windows and Showing the View 232
Conclusion 235

Chapter 9: iOS: Building the Logon Form and Consuming REST Services 237
Creating the Project 237
Creating the Logon Form 237
Creating the Logon Form User Interface 239
Showing the Logon Form 242
Special Note About Grouped Views 246
Conventions for Presenting Code in the iPhone Chapters 246
Calling the Services 247
Capturing the Logon Request 247
Calling the API Service 252
Building the Proxy Classes 253
Calling the Users Service 281
Notifying That Work Is in Progress 287
Conclusion 289

Chapter 10: iOS: An ORM Layer on SQLite....291
A Note About Content That Has Already Been Covered....291
Entities....291
The SBEntityType Class....292
The SBEntity Class....300
Setting Values in an Entity....303
Building SBBookmark....308
Creating SBEntityType Instances....311
Displaying Some Fake Bookmarks....313
Creating the View....313
Building the View Engine....317
Displaying Bookmarks....322
Handling Navigation....325
Building the Sync Class....327
Calling the Server's Bookmarks OData Service....327
Database Operations....344
Building SBDBHelper and Implementing Error Handling....347
Writing Bookmarks to the Database....361
Conclusion....380

Chapter 11: iOS : Pushing Changes Back to the Server....381
Configuring Bookmarks....381
Putting Data on the Table....384
Sorting the Bookmarks....387
Singleton View....388
Editing a Bookmark....393
Implementing the Delete Method....400
Adding a Bookmark....401
Deleting Bookmarks....403
Manually Syncing....406
Pushing Changes to the Server....407
Work Items....410
Issuing OData Change Requests....416
Flagging Fields As "Not on Server"....417
Issuing Requests....418
Modifying processWorkItems....424
Conclusion....426

Chapter 12: iOS: MonoTouch 427
Mono in the Big Picture 427
Chapter Structure 428
Installing MonoTouch 428
"Hello, World" 429
Inspecting the Code-Behind 432
Wiring Up the Button 434
Running the Project 435
Calling the Six Bookmarks API RESTful Service 436
Creating the Project 436
Building ServiceProxy Et Al. 437
Calling the Service Method 442
Conclusion 446

Index 447

About the Author

Matthew Baxter-Reynolds is an independent software development consultant, trainer, and author based in the UK, specializing in mobile technology solutions. He can be contacted via LinkedIn at `www.linkedin.com/in/mbrit`.

About the Technical Reviewer

In 2004, Matthew, with experience in VB.Net, joined a small e-commerce team to trial C# within a (then) small DVD and CD–focused e-commerce company.

Play.com went on to become one of Europe's largest e-commerce companies, with Matthew playing a major role as one of a handful of senior software developers. After six and a half enjoyable years, Matthew decided to move on to specialize in mobile technology, which he sees as a significant growth area for software developers and enterprises.

Working alongside Matthew Baxter-Reynolds, Matthew produced prototypes on a variety of technology platforms (Android, iPhone, Windows Phone 7, to name three) for a leading company in the mobile survey software market.

Matthew and his beautiful wife, Sarah, have a young boy, Isaac, and live in the beautiful town of Bury St. Edmunds. He enjoys films, games, music, and eating good food while drinking good beer, and he regularly practices muay thai.

His blog at `www.mattfitchett.com` covers all of the above, along with more mobile technology discussion.

Acknowledgments

With much thanks and appreciation to my wife, Andy, for the patience and support she has shown during writing and development of this book, Matt Fitchett for his excellent suggestions and review work, and Jonathan Hassell, Anita Castro, and the others at the Apress team for their sterling work in turning this book into reality.

CHAPTER 1

Introduction

For me, this book has become all about change. In the time that I have been watching the mobile computing marketplace and developing software solutions for it, there has never been a time when there has been a more rapid series of shifts and changes. A good friend of mine tells me that this is because of market consolidation. As of the time of writing (August 2010), we're looking at the time when the people who will be leaders in this space for the next 20 years jostle for position. There is a ton of money out there being spent, which is fantastic news for the typical reader of this book. Position yourself correctly and you could earn a seriously good living out of it all. .

To illustrate this point about change, I proposed this book to Apress in February 2010, and in just half a year between then and August 2010, here are just some of the changes that have happened. In a normal year, in a normal market, just a few of these things would be big news.

- Microsoft was still developing and building Windows Mobile 6.5. Windows Phone 7 had not been announced. No one really knew what sort of impact Windows Phone 7 will have.
- The iPad had not been announced, let alone sold over three million units (to date). (For me, this is perhaps the biggest change of all—the world will never be the same now that this class of device has been introduced.)
- The Pre was included in the original proposal. HP has now bought Pre, but the platform is now more or less obsolete.
- Just today, a research firm (Canalys) announced that Android's market share has grown 886 percent year-on-year.
- Canalys has also recently announced that 50 percent of BlackBerry users are looking to defect to iOS or Android.
- The image of Flash hadn't been damaged by Apple's insistence that it had no place on their platform.
- iPhone 4 had not been announced or released, and "Antennagate" had not happened.
- You couldn't multitask on an iPhone.
- iOS was still a trademark owned by Cisco.
- Gartner had not come out and likened Symbian to "re-arranging the deck chairs on the Titanic" in the face of the Android threat.
- The industry is starting to describe iPad-class devices as "slates" as opposed to "tablets."
- Even today, BlackBerry still hasn't given any formal details of version 6 of its platform, plus the BlackPad has been given as the name of its new slate only in the past few days. (Personally, I'm very excited about RIM's slate—it could be a real "game changer.")

- Steve Ballmer hadn't said that Apple had "sold more iPads than he would have liked and that "Microsoft-powered tablets are 'job one' urgency."
- We didn't know that Google could remote uninstall applications from any Android phone using a "kill switch."
- The UAE had not turned off BlackBerry Enterprise Services within the country.
- Motorola was looking very sick indeed, but is now looking much healthier thanks to the Droid and Droid X.
- MeeGo had not been announced (and as of the time of writing is not substantial enough to include in this book). My prediction, for what it's worth, is that this will get traction in spaces like automotive as opposed to slate or phone factors.
- Microsoft announced, launched, and killed a device called "Kin." To give you some idea of how much money is being thrown around, Microsoft attributes US $240 million of written-off monies to Kin. That's not small change.

In fact, this book has been difficult to write because of the velocity of all of this change. I'll be forever grateful to the team at Apress for managing to corral it into the place where, I hope, it's helpful and relevant to you, in spite of this almost constant upheaval in the market.

What's the Purpose of This Book?

In 2001, I set up a web site called .NET 247 (`www.dotnet247.com/`) that at the time achieved some success in the community that had sprung up around Microsoft's new software development toolset. The premise of the site was to help me as a developer migrate my knowledge from pre-.NET technologies (Win32, MFC, classic ASP, etc.) over to .NET. I found it frustrating that spinning up a thread or opening a file would be a few seconds' work prior to .NET, but in .NET it took hours of research.

With this book, I've looked to do a similar thing—answer the common questions and give you a leg up into understanding the platform so that you can get on and do the clever thing that only you've thought of. The idea of this book is not to go into masses of detail on every little thing; however, if you work through all of the different platforms in this book and its companion, you'll know enough to be proficient on any platform that you turn your hand to.

Specifically, what I've tried to concentrate on is the following:

- Getting to a point where you can compile and run an application on the emulator or device
- Showing how to build a user interface—specifically move between forms, handle events, get data on the screen, and capture input
- Showing how to connect to HTTP-based resources so that you can talk to services in the cloud
- Showing how to store and cache data locally for performance and for offline support
- Showing how to build a simple, but real, application that works end to end

How Is This Book Structured?

This book is split into three sections. There's an introduction section, which takes you through the background of the two applications that we're going to build. There is then a section on Android and another section on iOS. There is also a bonus chapter on using MonoTouch with iOS. (As of the time of

writing, MonoDroid for Android had not been released; hence there's no MonoDroid chapter, but it will obviously operate in a similar fashion to MonoTouch.)

In addition, this book has a sister book, which is structured similarly and takes you through building the same application that we're going to build in this book. The book's title—"Multimobile Development: Building native applications for Windows Phone, BlackBerry and generic applications using HTML5"—should tell you what you need to know.

■ **NOTE** The sister book also includes a chapter on Windows Mobile 6.5. Those of you looking to move an application away from Windows Mobile 6.5 to any platform, including iOS and Android, will want to read this. This chapter is available free of charge from the book's web site—see the following.

Each section starts with instructions on how to install the toolset that you are supposed to use with the platform. Some toolsets are very easy to install, while some have gotchas; thus the aim of the toolset installation chapter is mainly to cover the gotchas.

The next three chapters in each section take you through building what's called the "Six Bookmarks" application. This is a very simple application that is designed to show six buttons on the screen, and each button can be configured with a URL that invokes the device's default browser. The purpose of the application is not to be a fantastic piece of UI—it's designed to be a "carrier" to help you understand how to build all of the backend bits and pieces that you need to make an application functional. Figure 1-1 shows an example.

Figure 1-1. *The Six Bookmarks application running on an iPhone*

Each volume contains two chapters that are *essential* to following the work in the book, and I strongly recommend that you read them first.

To reduce the amount of work required to build the application, Six Bookmarks works on the assumption that there is a cloud-based service that holds the user's bookmarks. In order to use the software on a device, the user needs an account on this service. (This model will seem familiar to all readers of this book, I hope.) Chapter 2 discusses the structure of this service and familiarizes you with the service calls that make the application work.

The second important chapter is Chapter 3, which discusses the functional specification of the Six Bookmarks application and the technical architecture. Again, it's important that you read this in order to understand what it is that you are trying to build.

Where Can You Get Help and Support?

This book has a companion web site, located at `www.multimobiledevelopment.com/`, which hosts important resources that will support you in getting the most out of this book. Specifically, you will find the following:

- Downloads of all of the code for all of the platforms
- The Six Bookmarks cloud service implementation that you need to use to make the applications work
- A hosted version of the Six Bookmark HTML application (discussed in detail in Volume 2)
- Support forums (I'll be monitoring and contributing to these, so if you have a question or a problem, this is the best place to try.)

Finally, going back to my earlier point about the amount of flux in the market at the moment, I'll be updating the Web site to keep it up-to-date with changes in the toolsets and other movements within the industry.

Conclusion

Thanks for purchasing this book. Remember that if you do need help or support, then please visit the web site's discussion forums; but if you would like to contact me personally, you can find me at `www.linkedin.com/in/mbrit/`.

Matthew Baxter-Reynolds, August 2010

CHAPTER 2

The Six Bookmarks Server Service

We're going to talk more about the architecture and specification of the Six Bookmarks application in Chapter 3. In this chapter, we're going to look at the Six Bookmarks service. To support this book, I have set up a server with REST-based (aka "RESTful") services that allow the application to log on, retrieve bookmarks over the OData protocol, and post updates back, again using the OData protocol. (We'll talk more about OData later on.)

As discussed previously, Six Bookmarks is a commercial product provided in two ways—once as a commercial product and once as an open-source product. In this book, we're going to be accessing a service based on the open-source version of the code. Both applications communicate with a publically accessible server. The open-source server operates a sandbox, and in order to complete the work in this book, you'll need your own account.

NOTE It's currently very popular to talk about the "cloud" and storing things "in the cloud." The Six Bookmarks server service is one of these "cloud" services—I've provided a server hosted on the public Internet that allows you to store bookmarks "in the cloud" and retrieve bookmarks "from the cloud."

We will not be covering how to build this service in the book; however, the source code for the service be downloaded from the source repository at `http://code.multimobiledevelopment.com/`. This code and all of the other code downloads are distributed under the Mozilla Public License 1.1. More information on this can be found here: `www.mozilla.org/MPL/MPL-1.1-annotated.html`.

Creating an API Account

To create an API account, visit the services web site at `http://services.multimobiledevelopment.com/`. You will find a link on that page entitled **Register a new API account**. Click this to access a standard registration form as shown in Figure 2-1:

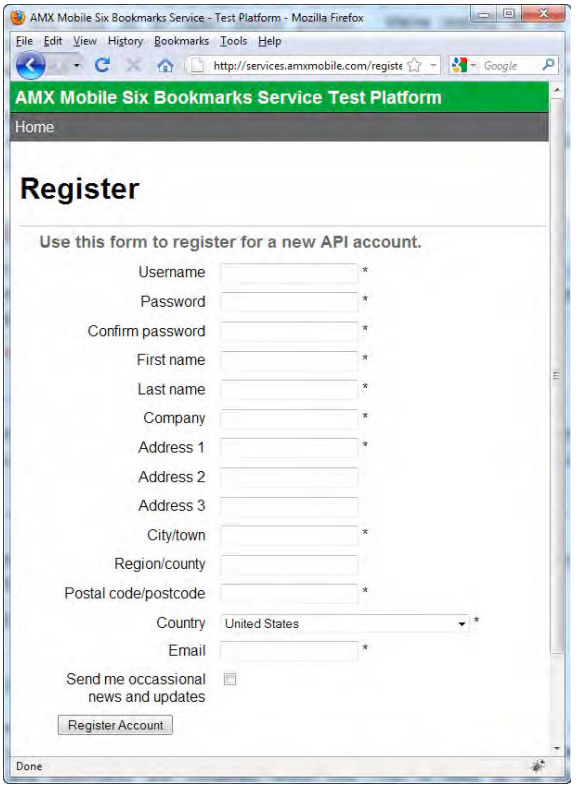

Figure 2-1. *The service registration form*

■ **NOTE** The site at `http://services.multimobiledevelopment.com` is a live work-in-progress. Some of the screenshots presented here may differ from the current reality of the site as you see it today. Also, the site you are using is not secured when accessed over HTTPS, as this is a test site not intended for production use. Were you to build a similar thing for production applications, it would be essential that you secure the site using HTTPS.

Go ahead and create your account. Please provide a valid email address, as you will need this should you need to reset your password in the future. (You will not get spammed.)

Registering your account will automatically log you on.

Creating a User

The purpose of registering for an account is to partition off a private section of the database for you to keep your own data in. A single SQL Server database exists on the server, and everyone's users and bookmarks are contained within this. This is likely to be slightly different for your own applications. For this book, we need to provide you with a sandbox service that makes it easier for you to work with the

chapters on the actual application creation on the devices; however, in production applications, you typically do not need this. I have to hive off individual readers' data into separate "virtual databases" to prevent corruption of data and weird behavior, and with potentially tens of thousands of you out there doing this, it's impractical to create physically separate databases.

Under the covers, you're going to be working with three tables: **ApiKeys**, **Users**, and **Bookmarks**. This entity-relationship diagram (ERD) shown in Figure 2-2 illustrates:

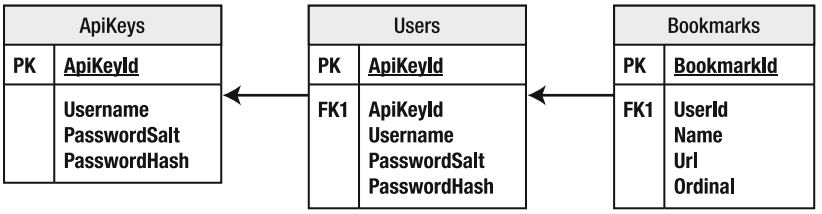

Figure 2-2. *ERD showing relationship between the ApiKeys, Users and Bookmarks tables*

When you register for an API account, you do not get any users created for you. A user in this context relates to someone who would use an instance of the mobile Six Bookmarks applications. To create a user, click on the **Manage Users** link. You will be presented with a message that indicates no users are available as per Figure 2-3.

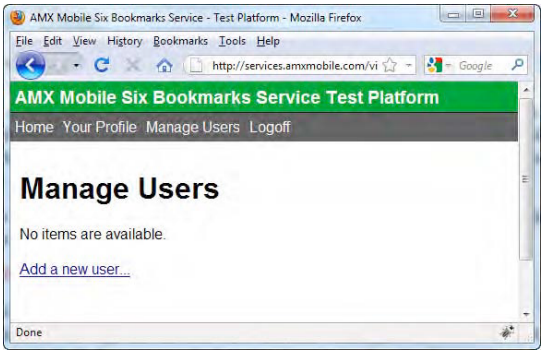

Figure 2-3. *The "Manage Users" page showing no available users*

Click on the **Add a new user** link to enter a new user. Figure 2-4 illustrates.

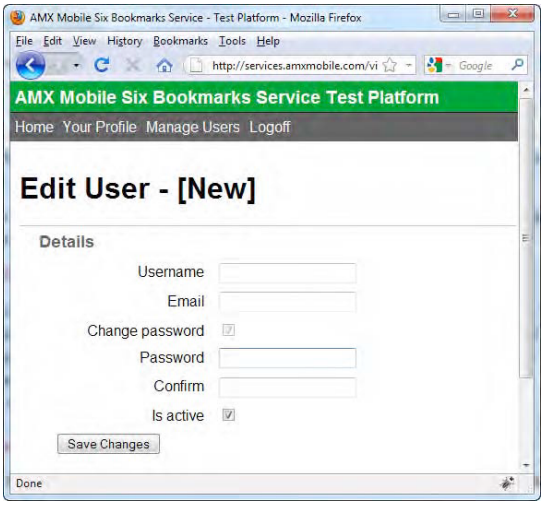

Figure 2-4. *The "Edit User" page*

You'll need to create at least one user in order to proceed to the next section.

The Users Service

The "users" service is a RESTful web service that provides a capability to log on a user. (This book deals only with logging users on; however, the service is capable of other functions, including registering users.)

It's important to familiarize yourself with how the service works, as it will aid in understanding the flow of the applications that we will build in later sections.

RESTful Web Services

A "RESTful" web service is a service that is based on the principle of REST, which stands for "Representational State Transfer." It is not a formal, standardized protocol, but rather a set of principles or constraints that describes the shape and operational usage of a service that you can get data from or provide data to. It is a very natural way of working with remote services, which is why they are so popular and prevalent. That naturalness translates into being very easy to build, and equally very easy to consume.

One common and straightforward way of structuring a RESTful web service is to request data using an HTTP GET request and retrieving results back as XML. The HTTP request can be a GET request, including parameters specified in the query string. Alternatively, the parameters can be made via a POST request that works by passing up XML.

Let's continue this by looking in more detail at the logon operation on the Users service.

Testing the Calls

The API relies on sending up custom HTTP headers, and as such we can't test it using a regular web browser. Rather than asking you to build some custom code to call the service, you can download a test harness for trying the service. You can download this from the source repository at `http://code.multimobiledevelopment.com/`. Look for a file in the **Downloads** section of the form `Amx.Services-<Version>-TestClient.zip`. This is a .NET application.

If you download the utility and run it, you'll see you have an area to enter a URL and an area to enter two values: `API username` header and `Token` header. We'll talk about these values later, but essentially they provide additional information to the service to help guide the response.

Examining Logon Operations

The first thing we can try and do with our Users service is log on a user. Ultimately, a successful logon will return a token that we can use in subsequent requests.

If you open the test harness, the URL will be given as follows:

```
http://services.multimobiledevelopment.com/services/apirest.aspx?operation=➥
logon&password=APIPASSWORD
```

Click the **Send Request** button and you'll see a result like Figure 2-5.

Figure 2-5. *An example of a failed request to the API service*

You can see in the response that an error has been returned.

The protocol for the REST services exposed by the service is that exceptions are returned back in the `Error` element, and the `HasException` element is set to `true` if an error has been returned. (The value shown in the XML is `1`, but the `datatypes` schema is used to indicate that this is a Boolean value.)

■ **NOTE** This error notification and transmission is just how I have designed the service—it doesn't follow that all RESTful web services will use this approach. It's down to the owner of the service to design a protocol that is sensible and logical to use within the loose construct of what a RESTful service typically looks like.

Referring back to Figure 2-5, the error indicates that "Neither a logon token nor API key were provided in this request." What this is telling us is that the headers have not been provided to the server.

To call the operations on the server, we need a token. In order to get a token, we need to call the server, so we have a chicken and egg situation! However, one operation on the server does not need a token—this is the `Logon` operation on the API service, which is used solely to obtain a token for use with the other methods.

Obtaining a Token

By default, when you start the harness, it will be set to connect to the API service and to call the `Logon` method. First, into the `API username header` text box, enter the username of the account you created in the first part of the chapter. Second, modify the password value in the URL to be the username on your account.

If you click `Send Request` now and the details are correct, you'll see something similar to the image shown in Figure 2-6.

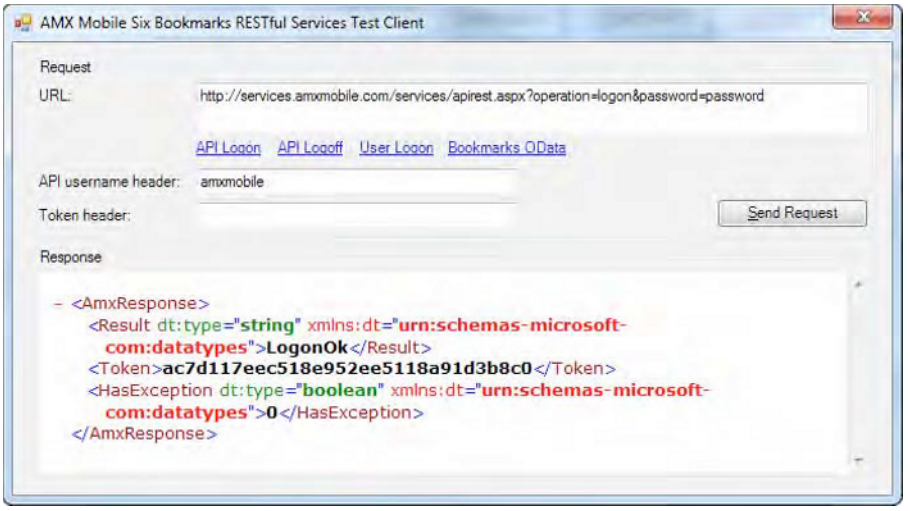

Figure 2-6. *An example of the result of a successful request to the API service*

You'll see in this case an error has not been returned. The `Result` element will be set to `LogonOk` or `InvalidPassword`. (Any other errors will result in an exception being returned.)

The most important element here is the `Token` value. This is the token that we'll use in all other requests. Copy the token into the clipboard, and then paste it into the `Token header` field. We'll use this later.

Logging On the User

Now that we have obtained a token to use and authenticated the API, we can actually log on the user. We've used the API service so far—we're now going to use the Users service.

If you click the `User Logon` link on the test harness, the URL will be rewritten to the following:

```
http://services.multimobiledevelopment.com/services/usersrest.aspx?operation=↩
logon&username=USERNAME&password=PASSWORD
```

This URL is configured to call the Users REST service. If you replace the `USERNAME` and `PASSWORD` placeholders in that string, and assuming you have copied the token into the `Token header` field, and click **Send Request**, you'll get a response like Figure 2-7, which, apart from the URL, looks identical to Figure 2-6.

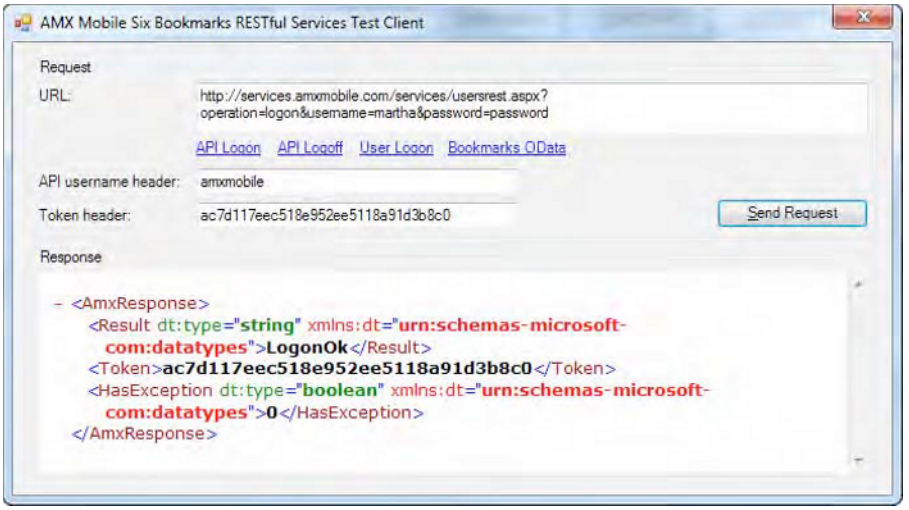

Figure 2-7. *An example of a response from a successful request to the Users service*

Assuming this works, you'll see another `LogonOk` response. What a `LogonOk` tells you here is that the token is now bound to the user you authenticated. (This is important—this means that you cannot use the same token with different users. This will never be a problem on a mobile device as you are not multiuser, but in a web application, it is worth considering.) Other results you can get back from the service are `InvalidUsername`, `InvalidPassword`, or `AccountInactive`.

Cleaning Up

To clean up the service, we have to log off of the API. This is done via the `Logoff` operation. Click the **API logoff** link on the harness, and the URL will be rewritten once again. Click the **Send Request** button, and you'll see a response much like in Figure 2-8.

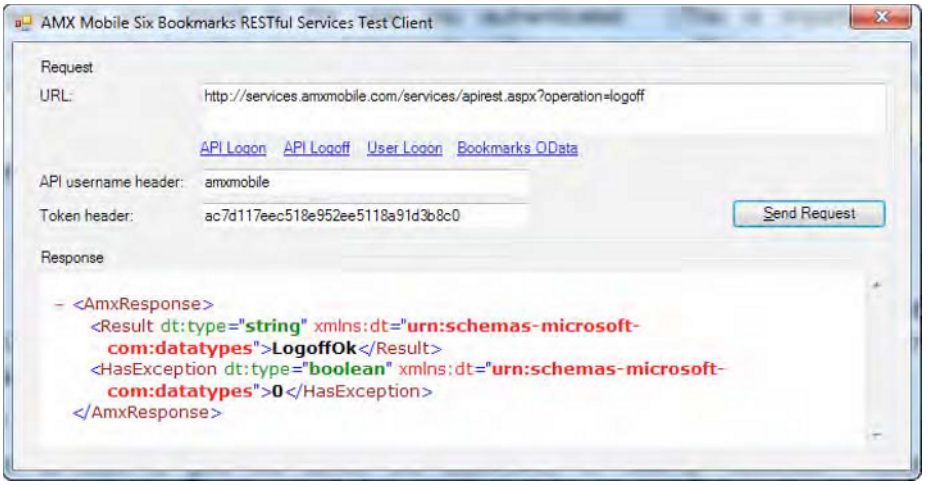

Figure 2-8. *An example of a successul "Logoff" call to the API service*

This operation is used to give the server the opportunity to clean up resources related to the token. (Specifically, it deletes a row in the database.) We'll look at token cleanup in more detail when we build the native device applications.

The Bookmarks Service

The final service exposed from the server is the Bookmarks OData service. OData is an up-and-coming data format that is currently being pitched as the proposed *de facto* standard for data interchange in a Web 2.0 world. My opinion is that it is a decent standard with a good, practical working method, and hence I've chosen to use it in this book to bridge the gap between relational data stored in the cloud and data stored on the device.

■ **TIP** You can find out more about OData at the official site: `http://www.odata.org/`.

Adding Some Test Data

In order to see how the OData service works, you're going to need some test data. There's an interface on the service that lets you maintain the bookmarks against a user.

Log on to `services.multimobiledevelopment.com` and find a user that you want to work with. Click on the **Manage this user's bookmarks** link at the bottom of the page. You will see an interface that allows you to define bookmarks. Figure 2-9 illustrates.

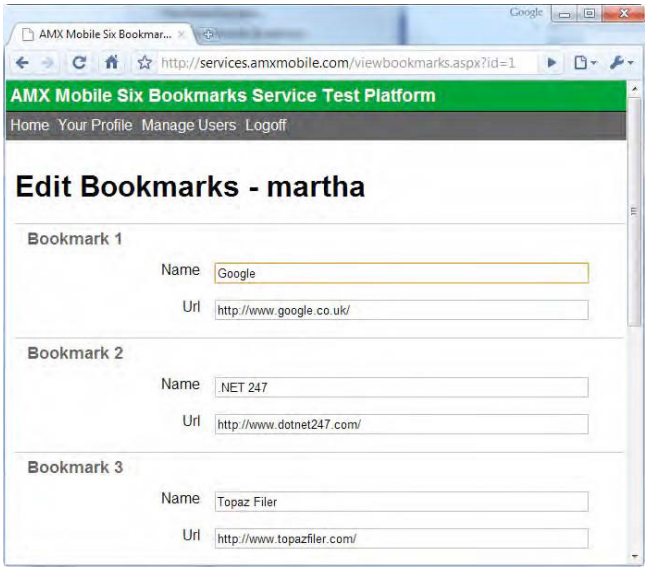

Figure 2-9. *The "Edit Bookmarks" screen showing three bookmarks*

Add a number of bookmarks and click **Save Changes**.

Working with OData

Now that we have some bookmark data, we can look at using the Bookmarks service. We're going to be using the test harness again, and you will need a token—so if you do not currently have a token, go through the steps described previously to obtain one.

On the harness, if you click the **Bookmarks OData** link, you'll get a rewritten URL, like this one:

```
http://services.multimobiledevelopment.com/services/bookmarks.svc/
```

Click **Send Request** and you'll get a response like Figure 2-10. You should note that the test harness continues to send up the special headers. The service call would be rejected should these headers be missing or incorrect.

■ **NOTE** The OData standard allows for data to be returned either in Atom or JSON format. Atom format is the most relevant here—JSON is typically used when working with Ajax calls from a web page. The actual format of the data is not important—what is important is that OData is built on open standards. (Notably, Microsoft sees OData as a core data protocol going forward, starting with a full implementation in .NET 3.5 SP1 and support on the Azure platform.)

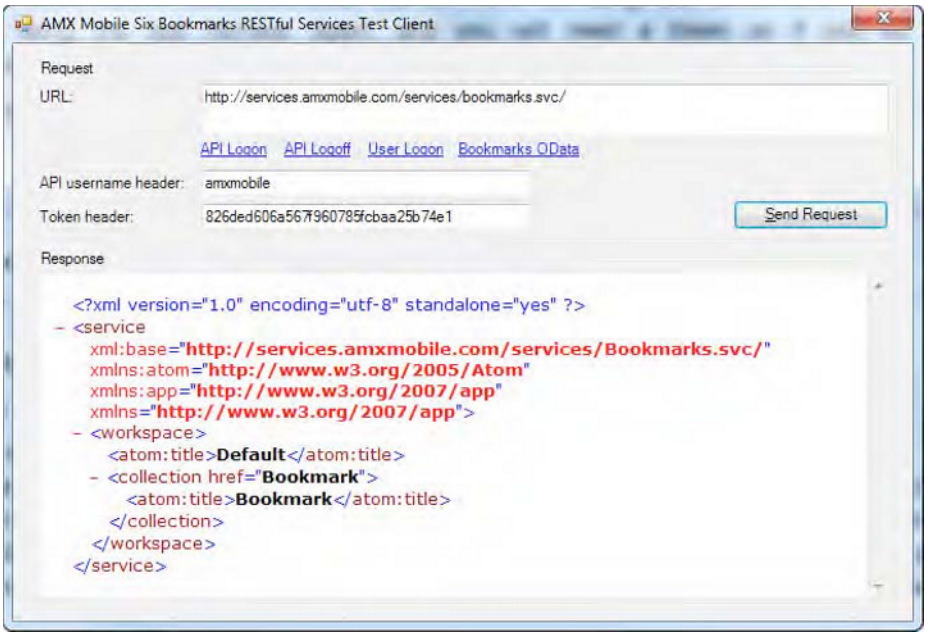

Figure 2-10. *An example of a successful call to the Bookmarks OData service*

The preceding output is telling us that the Bookmarks service is about to return data of type `Bookmark` (look for the `//collection/atom:title` element in the XML). Thus, if we issue this URL, again using the test harness, we'll get back some bookmarks. Here's the URL:

```
http://services. multimobiledevelopment.com/services/ bookmarks.svc/Bookmark
```

From this point, I'm going to show you the XML output as a listing, rather than screenshots. This will make it easier to follow the discussion.

In the following example, three bookmarks are returned from this call, and these are shown in the following listing. (Your output will vary depending on the bookmarks you've set up against the user that you've logged in as, obviously.) Here's the listing:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<feed xml:base="http://services. multimobiledevelopment.com/services/Bookmarks.svc/"↩
 xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"↩
 xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"↩
 xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Bookmark</title>
  <id>http://services. multimobiledevelopment.com/services/bookmarks.svc/Bookmark</id>
  <updated>2010-04-18T10:54:32Z</updated>
  <link rel="self" title="Bookmark" href="Bookmark" />
```

```
<entry>
    <id>http://services. multimobiledevelopment.com/services/Bookmarks.svc/Bookmark(1002)</id>
    <title type="text"></title>
    <updated>2010-04-18T10:54:32Z</updated>
    <author>
      <name />
    </author>
    <link rel="edit" title="Bookmark" href="Bookmark(1002)" />
    <category term="AmxMobile.Services.Bookmark" scheme="http://schemas.microsoft.com/↩
ado/2007/08/dataservices/scheme" />
    <content type="application/xml">
      <m:properties>
        <d:BookmarkId m:type="Edm.Int32">1002</d:BookmarkId>
        <d:UserId m:type="Edm.Int32">1001</d:UserId>
        <d:Name>.NET 247</d:Name>
        <d:Url>http://www.dotnet247.com/</d:Url>
        <d:Ordinal m:type="Edm.Int32">1</d:Ordinal>
      </m:properties>
    </content>
  </entry>
  <entry>
    <id>http://services. multimobiledevelopment.com/services/Bookmarks.svc/Bookmark(1001)</id>
    <title type="text"></title>
    <updated>2010-04-18T10:54:32Z</updated>
    <author>
      <name />
    </author>
    <link rel="edit" title="Bookmark" href="Bookmark(1001)" />
    <category term="AmxMobile.Services.Bookmark" scheme=↩
"http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
    <content type="application/xml">
      <m:properties>
        <d:BookmarkId m:type="Edm.Int32">1001</d:BookmarkId>
        <d:UserId m:type="Edm.Int32">1001</d:UserId>
        <d:Name>Google</d:Name>
        <d:Url>http://www.google.co.uk/</d:Url>
        <d:Ordinal m:type="Edm.Int32">0</d:Ordinal>
      </m:properties>
    </content>
  </entry>
```

```
  <entry>
    <id>http://services.multimobiledevelopment.com/services/Bookmarks.svc/Bookmark(1003)</id>
    <title type="text"></title>
    <updated>2010-04-18T10:54:32Z</updated>
    <author>
      <name />
    </author>
    <link rel="edit" title="Bookmark" href="Bookmark(1003)" />
    <category term="AmxMobile.Services.Bookmark" scheme=↩
"http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
    <content type="application/xml">
      <m:properties>
        <d:BookmarkId m:type="Edm.Int32">1003</d:BookmarkId>
        <d:UserId m:type="Edm.Int32">1001</d:UserId>
        <d:Name>Topaz Filer</d:Name>
        <d:Url>http://www.topazfiler.com/</d:Url>
        <d:Ordinal m:type="Edm.Int32">2</d:Ordinal>
      </m:properties>
    </content>
  </entry>
</feed>
```

Thanks to the clarity of the Atom format, it's very easy to understand the format of the data, even though the dataset is an unfamiliar one. Each of the `feed/entry` elements contains a single data item (which we'll be calling an "entity" throughout to keep in line with nomenclature on the object relational mapping structures that we're going to be using). The `m:properties` element within them contains the data. (For information, this maps 1:1 to the structure of the table used to store the bookmarks.)

An interesting element here is the `ID` element against each entry. These provide a URL that can be used to access an individual item. (Although, remember that you need to pass up the special headers in order for the service to return the data.)

Pick the ID of an item in your set of bookmarks and issue a request for it, passing in the token, e.g.:

```
http://services.multimobiledevelopment.com/services/bookmarks.svc/Bookmark(1003)
```

This time you will just see that item, as per the following listing:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<entry xml:base="http://services.multimobiledevelopment.com/services/Bookmarks.svc/"↩
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"↩
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"↩
xmlns="http://www.w3.org/2005/Atom">

  <id>http://services.multimobiledevelopment.com/services/Bookmarks.svc/Bookmark(1003)</id>
  <title type="text"></title>
  <updated>2010-04-18T10:55:13Z</updated>
  <author>
    <name />
  </author>
```