VoiceXML:
10 Projects to Voice Enable Your Web Site
A Note from Gearhead Press

Gearhead Press is dedicated to publishing technical books for experienced Information Technology professionals—network engineers, developers, system administrators, and others—who need to update their skills, learn how to use technology more effectively, or simply want a quality reference to the latest technology. Gearhead Press emerged from my experience with professional trainers of engineers and developers: people who truly understand first-hand the needs of working professionals. Gearhead Press authors are the crème de la crème of industry trainers, working at the companies that define the technology revolution. For this reason, Gearhead Press authors are regularly in the trenches with the developers and engineers that have changed the world through innovative products. Drawing from this experience in IT training, our books deliver superior technical content with a unique perspective that is based on real-world experience.

Now, as an imprint of John Wiley & Sons, Inc., Gearhead Press will continue to bring you, the reader, the level of quality that Wiley has delivered consistently for nearly 200 years.

Thank you.

Donis Marshall
Founder, Gearhead Press
Consulting Editor, Wiley Computer Publishing
Gearhead Press Books in Print

(For complete information about current and upcoming titles, go to www.wiley.com/compbooks/)

Books in the Gearhead Press Point to Point Series

*Migrating to Microsoft Exchange 2000* by Stan Reimer  

*Installing and Configuring Web Servers Using Apache* by Melanie Hoag  

*VoiceXML: 10 Projects to Voice Enable Your Website* by Mark Miller  

Books in the Gearhead Press In the Trenches Series

*Windows 2000 Automated Deployment* by Ted Malone and Rolly Perraux  

*Robust Linux: Assuring High Availability* by Iain Campbell  
ISBN: 0-471-07040-8

*Programming Directory Services for Windows 2000* by Donis Marshall  
ISBN: 0-471-15216-1

*Programming ADO.NET* by Richard Hundhausen and Steven Borg  
ISBN: 0-471-20187-1

*Designing .NET Web Services Using ADO.NET and XML* by Richard Hundhausen and Steven Borg  

*Making Win32 Applications Mobile: Porting to Windows CE* by Nancy Nicolaisen  
ISBN: 0-471-21618-6

*Programming Windows CE Wireless Applications* by Barry Shilmover and Derek Ball  
ISBN: 0-471-21469-8

*Mastering SQL Server 2002 Security* by Mike Young and Curtis Young  

*Microsoft.NET Security Programming* by Donis Marshall  
Acknowledgments xv

Introduction xvii

Part One Introduction 1

Chapter 1 Introduction to VoiceXML 3
XML Overview 4
  XML Architecture 5
  XML versus HTML: Content versus Formatting 6
XML Elements 8
XML Syntax 9
DTDs: Defining the XML Elements 10
Displaying XML with XSLT 12
VoiceXML 13
  VoiceXML Working Draft Specification 13
  VoiceXML Syntax 14
  Processing VoiceXML 16
Final Thoughts 17

Chapter 2 Setting Up the Working Environment 19
VoiceXML Portals 20
  How It Works 20
VoiceXML Application Service Providers 21
  Setting Up a Developer Account at the BeVocal Cafe 22
    Create an Account 22
    Tools in the BeVocal Development Environment 23
  Testing Your Environment 35
Final Thoughts 35
Contents

Part Two  The Projects  37
Project 1  Building Basic Pages with VoiceXML  39
  Project Description  39
  Call Flow  41
  Create a Hello, World Application  42
    <form> Elements  43
    <block> Elements  43
    <prompt> Elements  44
    Verify that the Application Is Working  45
  Add Multiple Dialogs to the Template  46
    DOCTYPE  47
    <goto> Elements  47
  Accept User Input with <field> and Yes/No Grammar  49
    type and modal Attributes  50
    More on <field> Elements  50
    <if> Elements  51
  Set Up Error Handling  51
    <nomatch> and <noinput> Elements  54
  Form Interpretation Algorithm  56
    Basic Concepts of FIA  56
      Selection Phase  56
      Collection Phase  57
      Processing Phase  57
      Ending the FIA  57
    Analyzing the FIA Sequence  57
  Insert Meta Data for the Project Description  61
  Final Thoughts  62
Project 2  Creating Links  63
  Project Description  64
  Call Flow  65
  Create a Placeholder Template  65
    Review of <goto>  66
  Conditional Processing  68
  More Conditional Processing  70
  Expanding the VoiceXML Script  70
  Accessing External Resources  72
    Application Root Document  73
    Another External Data Source  75
  Setting Up the maincontent Form  77
    <link> Elements  78
      Reloading a Document  79
      Testing Scenarios  80
      Catching Events  81
Contents ix

Creating Menus 82
  <enumerate> Elements 83
  <choice> Elements 84
Link Menus as Hierarchies 88
  Creating Pauses 90
Modularize Processing with Subdialogs 91
  <subdialog> 92
  <return> Elements 93
  Using Returned Values 93
Handle Requests Dynamically with <submit>
  Processing User Input with a Script 95
Final Thoughts 97

Project 3 Contact Us 99
  Project Description 100
    Call Flow 101
Repeating Content and Controlling Application Flow 101
  <var> Elements 102
  Predefined Grammars 104
  Pauses in Output 104
Recording a Message 105
  Menu Choices 106
  Recording a Message 107
    <record> Element Attributes 109
    Confirming the Recording 109
    Processing with <subdialog> 110
Saving a Recording 111
  Dynamically Creating a Filename 114
Transferring a Call 115
  The <transfer> Element 116
    Attributes of the <transfer> element 117
    Checking the Return Results 118
Completing the Script 118
Final Thoughts 123

Project 4 User Authentication 125
  Project Description 126
    Application Call Flow 127
Testing a Caller’s Phone Number 128
    <say-as> Elements 131
Activate Password Protection 132
    <audio> Elements 134
    Mapping Grammar Slots 134
    Error Checking 135
    <log> Elements 135
    Processing the Output 136
Contents

Voice Prints 136
   Making a Voice Print 136
   Unique Keys 139
   <assign> Elements 140
   Taking Voice Samples 140
   <register> Elements 140
   Completing the Voice Print Sampling 142
Authenticating a User through Voice Recognition 142
   <verify> Elements 145
   <catch> Elements 145
   Shadow Variables 146
Final Touches to the Application 146
Final Thoughts 149

Project 5  Templates and Multiple Documents 151
   Project Description 152
      Application Call Flow 153
   Create the Root-Level Document 154
   Creating Templates 156
      Templates versus Objects 156
      Creating the Placeholder Templates 157
      The Greeting Template 158
         Scope of Variables 159
      The Goodbye Template 159
      The Menu Template 160
         Completing the Templates 163
   Secure Login 166
      Scope of Grammars 168
      Login Template 168
      Authentication Routine 170
   Final Thoughts 171

Project 6  Telephone Dialer 173
   Project Description 174
      Application Call Flow 175
   Build a Contact List 176
   Map a Name to a Telephone Number 178
      Filling Slots 178
   Confirm User Input 180
      Field-Level Shadow Variables 181
      Check for User Input 182
   Group Items in a Grammar 182
   Create Multiple Tokens as a Single Name 183
      Grammar Operators 184
   Refine the Contact List 184
      Optional Tokens 185
      Creating Nicknames 185
      Primary Selections 186
Allocate Multiple Phone Numbers for One Contact 186
  Create a Memorable Token 187
  Add Optional Words to the Phrase 187
Control Data Type Speech Output 188
Transfer the Call to the Contact Number 189
Place Another Call without Disconnecting 190
Modularize the Application 191
Localize Help and Error Messages 193
Insert Meta Data 195
Put It All Together 196
Final Thoughts 199

Project 7 Report Recorder 201
  Project Description 202
    Application Call Flow 203
  Develop a Simple Grammar 203
    reporterName and reporterId Grammars 204
    Slot Comparison for Authentication 207
  Limit Login Attempts 207
    Set the Variables 210
    Error Check Field Input 211
  Develop an Eternal Grammar File 211
    Create a List of Beach Names 211
    Grammar Rules 212
    External Grammar Files 214
    Calling an External Grammar 215
    Dynamically Generate a URI 216
      Define a Global Variable 216
      Generate a URI 220
    Map Reporters to Beaches’ Names 220
  Record a Report 221
  Save the Recording 225
    Pass Field Names and Variables to the Subdialog 225
    Process the Recording 225
    Test the Script 227
  Access the Recordings 227
  reportRecorder.vxml 228
  Final Thoughts 233

Project 8 Quiz Generator 235
  Project Description 236
    Application Call Flow 237
  Create a Simple Quiz 238
    The Basic Template 238
  Add Questions to the Quiz 241
    Process the First Field 241
    Use Blocks for Transitions and Multiple Outputs 244
    Add Questions 244
xii Contents

Insert Recorded Prompts 245
Download the Star Trek Sound Files 245
Insert <audio> Elements 246
Record the Prompts 247
Complete the VoiceXML Template 248
VoiceXML Script Generation with XML and XSLT 255
Set Up the XML/XSLT Work Environment 255
Create the Data File (XML) 256
Analyze the Data 256
Create an XML Skeleton 258
Define a Link 258
Define a Help Message 259
Greeting Block 260
Define Questions <form> 261
Define Goodbye <form> 265
The Final XML Data File 265
Build the Stylesheet (XSLT) 269
Test the Basic Application 270
Insert XSLT Templates into the Stylesheet 271
Document Root Template 271
Quiz Template 272
Link Template 273
Output 274
Help Template 276
Power of Template Processing 277
Greeting Template 278
ErrorNoInput Template 280
Noinput Template 280
Instructions and confirmBeginning Template 281
SeriesOfQuestions Template 281
Set Template 282
Question Template 282
Answers Template 283
Answer Template 283
Checking an Answer 284
Goodbye Template 285
Test the Completed XSLT Stylesheet 285
Final Thoughts 289

Project 9 San Francisco Arts Calendar 291
Project Description 292
Application Call Flow 293
Build a Skeleton 294
Create a Greeting Subdialog 296
Create a Goodbye Subdialog 296
Create the CalendarMenu Subdialog 297
  Request a Discipline 298
  Request a Date 299
  Using Return Values from an Object 301
  Implied Error Checking 301
  Use <subdialog> to Call the Database 303
  Test Script 303
  Retrieve Another Event 304
  Completed Script 305
Access the SFArts Calendar Database 307
  Create a Template of Desired Output 308
The Cold Fusion Template 309
Final Thoughts 313

Project 10 FAQ Engine 315
Project Description 316
  Application Call Flow 317
Set Up the Brainhat Server 318
  About Brainhat 318
  The Brainhat Server 318
  Install Brainhat 318
  Test the Installation 319
Build Vocabularies in Brainhat 321
  Vocabulary Overview 321
  Add to the Vocabulary 323
  Create a New Definition 324
  Create a Standalone Vocabulary 325
Model Scenarios to Build Dialog Context 326
  Create a Simple Scenario 326
  Give Brainhat a Personality 327
  Activate a Scenario 328
  View the Log File 329
Develop an FAQ Application 329
  Create a List of Questions 330
  Build the Scenario and Define the Vocabulary 330
    Defining a New Word 331
    Synonyms 333
  Extend the Scenario 335
    Defining Facts 335
    Build Propositions 337
    Context-Sensitive Inferences 338
    Batch File Testing 339
    Intuitive Dialogs 341
    Handling Questions on Cost 342
    Prepared Responses 343
    Final Scenario 344
Contents

Set Up the VoiceXML Entrance 347
   Start the Server in VoiceXML Mode 347
   Create Simple Access Script 349
      Caching Documents and Grammars 350
      Create a VoiceXML Grammar 351
      Processing Input 352
   Final Thoughts 354

Glossary 355

Index 359
When I was first approached by Donis Marshall to write this book, I thought, “No problem. I’ll whip it out in a couple of months and be done with it.” Anyone who has ever attempted to write a book knows how naïve and uncomprehending that thought was. Over the past ten months, I’ve needed the support of my friends and family who kept checking in periodically to see if I was “still alive.” Yes, I am, and I sincerely thank you for worrying. Can we go to Disneyland now?

The short list for thanks starts with Donis Marshall. Thanks for allowing me to rewrite, change, be late...whatever it took to get the book right.

Bryan Michael at The BeVocal Cafe saw the value of this project right away and didn’t hesitate with any type of resources that were necessary to make my job a little easier. The development team at BeVocal was indispensable when I needed quick turnaround confirmation for technical questions. A special thank you to Mukund Bhagavan, Kesava Neeli, and Vavnnet Raja for responding at all hours of the day and night.

Kevin Dowd and Larry Larson provided invaluable technical support when generating the projects for the Brainhat FAQ Engine and the San Francisco Arts Calendar. Thanks also to Richard Weiss for introducing me to VoiceXML and to Tony Caviglia for contributing ideas for the Telephone Dialer project.

Finally, this book would never have been completed without the unwavering support of my wife, Rosemary. Thank you for believing it could be done and making it happen.
Playing with a new technology is fun. I have been a teacher in one form or another for over 20 years, but it still gets me excited when I see something that seems so obvious and so simple that it is shocking it hasn’t been done before. That’s the way I feel about VoiceXML.

VoiceXML makes it possible for anyone who can build a basic Web page to create a voice Web page. The first time I heard about VoiceXML, it took me less than an hour to get my first application up and running. It’s just that simple. Simplicity, however, doesn’t mean that big things can’t be done. If you have called 1-800-555-1212 recently, you have heard a VoiceXML application written by TellMe.com handling the incoming calls through VoiceXML.

Who Should Read This Book

VoiceXML: 10 Projects to Voice Enable Your Website is an outgrowth of my experiments while playing with VoiceXML. One of the problems with many technology books I have seen is that they give you all of the tools, but they don’t tell you how to use them. Context is one of the most important parts of learning. I think that people beginning to learn a technology can learn much more from creating and implementing projects than by learning about function sets and where things are on a menu bar.

This book is for beginning voice developers. It assumes no knowledge of VoiceXML or previous experience with the technology. If you are a Web developer or have developed even one Web page, this book is for you. VoiceXML terminology is similar to HTML, but it has a much smaller element set: There are less than 50 elements to learn. The first five projects will bring you up to speed with everything you will need to know when voice enabling your current Web site.
Database managers will find the book interesting if they want to see how to output a database query as a vocal response. Most of the data on the Web is stored in databases, whereas most of the people in the world have access to a telephone. VoiceXML acts as the glue joining the two technologies.

System administrators will be excited to know that VoiceXML can be used to call them on the phone when their systems need help. The application can be set up to call, not beep them, with a system specific message output vocally. With a little more work, simple administration of the system can be handled through the phone call, making it unnecessary to access a computer to make updates.

CGI programmers will have a field day with VoiceXML. Any type of output that is currently generated on a Web site through Perl, ASP, JSP, PHP, or other dynamic content generator can be output as a voice response to be accessed over the telephone.

However, this book isn’t for everyone. It is not a reference manual and does not cover advanced techniques of VoiceXML. If you have already developed extensive voice applications or have been a developer on one of the voice portals such as BeVocal, TellMe, or HeyAnita, this probably isn’t the book for you. You’d be better off with a detailed examination of the VoiceXML 2.0 specification at the W3C, tearing apart the documentation on the Nuance site for developing complex grammars, or using Bruce Balentine’s How to Build a Speech Recognition Application style guide. VoiceXML: Professional Developer’s Guide by Sharma and Kunins has also been getting good reviews as a solid reference manual.

But if you are just getting started with VoiceXML, keep reading.

**How This Book Is Organized**

This book shows how to build basic VoiceXML applications from existing Web sites through a series of 10 step-by-step projects. Each project begins with a problem statement and a possible solution. The project description follows and includes a brief overview and a list of the components and steps it will take to complete the project. The implementation of the project goes step by step, each phase ending in a hands-on exercise, confirming that the application is working before proceeding to the next stage.

The book is divided into two sections: two chapters of introduction and environment and the 10 projects. Chapter 1 is a general overview of VoiceXML and how it fits into XML technologies. This can be used as an introduction to the technologies or as a brief overview for developers who have previous XML/VoiceXML experience. Chapter 2 looks at the BeVocal Cafe voice portal environment that is used for developing the projects. The tools of the portal are analyzed and basic setup procedures are covered.

Next are the 10 VoiceXML projects. The first five projects take your existing Web site and output it as VoiceXML. Project 1 begins with a basic Hello World application and expands it to a complete VoiceXML application. Projects 2 and 3 add vocal links and contact information. Voice authentication and login processes are covered in Project 4. Project 5 reexamines the first four projects and breaks them into small, reusable templates.

After building the first five projects, you will have a solid understanding of how VoiceXML applications are created. The last five projects are applications built on the foundations of the previous five projects. Projects 6 through 10 can be done in any
order; however, from user response, I’d suggest doing Project 6, the telephone dialer project, first because it is pretty impressive when you are trying to explain to your boss or your in-laws how the technology can be used.

Project 6 is a handy project that can be used as a phone book telephone dialer, placing calls through vocal requests. The Report Recorder in Project 7 allows surfers (wave surfers, not Web surfers) to call in reports on beaches from around the world and have the reports generate a wav file that can be heard on a Web site or over the phone. Project 8 uses XML/XSLT technology to create phone quizzes that can be used for games or customer surveys. Project 9 uses the database calendar at the San Francisco Arts Organization to dynamically answer phone requests for entertainment information regarding dates and types of events. The final project, the FAQ engine, is an experimental project for putting a voice interface on an inference engine that can handle questions through natural language processing. The projects are followed by a glossary of terms and acronyms.

Tools You Will Need

The basic projects in this book can be run through one of the online voice portals such as BeVocal, TellMe, or HeyAnita with no additional software. For Projects 6 through 10, you will need the additional software, most of which is freely downloadable and available on the Internet.

What’s on the Web Site

An accompanying Web site can be found at http://cafe.bevocal.com/wiley/10projects. It contains the latest code for each of the projects, a list of enhancements to the projects that have been suggested by other readers, and links to online resources for further reading.

A special newsgroup has been set up to handle questions, allowing developers to exchange information on how they have expanded the ideas in the book and used them in their own applications. You can log in to the newsgroup at news://cafe.bevocal.10projects.

My goal on the Web site is to set up a community of developers who would like to extend the projects and add to their usefulness. If you would like to participate, please log in and join us.

Final Thoughts

I hope I have gotten you excited enough to begin your first project with VoiceXML. With this book and access to the Internet, you could have your first project up and running within an hour.

I look forward to seeing you on the Web site and to your participation in the growing community of VoiceXML developers.
PART One

Introduction
If you have ever designed a Web page, you already have the basic skills needed for developing a VoiceXML application. VoiceXML is a simple, tagged language that is similar to HTML in structure. In fact, it consists of less than 50 elements, making it much smaller than the HTML element set.

HTML gives visitors who come to your site the ability to view content, click links, and move through the pages. VoiceXML will let your visitors do this verbally over the phone. Think of VoiceXML as the HTML of the telephone. It guides users through the content of your site by using a telephone instead of a computer. The VoiceXML application will not be a replacement for your existing Web site, but it will offer your visitors additional access to your site through a highly accessible medium, the standard telephone.

The data on your Web site is the content for dialogs between the user and your Web server. VoiceXML acts as the interpreter between any telephone and your Web site content. Instead of using HTML to access your content visually, VoiceXML creates a verbal interface to the data through a voice gateway, usually housed on a voice portal.

A complete industry called voice portals has sprung into existence within the past two years. Voice portals are service providers who provide space for your voice applications, much as an Internet service provider (ISP) does for HTML Web pages. The voice portal provides the gateway and processing architecture for handling VoiceXML applications that you develop. Portals contain development tools, space allocation, example tutorials, and detailed documentation for the creation of VoiceXML applications. We will be using the services of one of the portals for development of the 10 projects.
A point of confusion among most beginning voice developers is about the telephone itself. VoiceXML allows access to your data through any phone. There is no special hardware attached to your phone, and it is not a secret, James Bond–enabled device.

The second question that usually comes up is, “Well, can I talk to the computer to access the Web site or does it have to be over a telephone?” VoiceXML is a telephone technology and is not normally used for accessing data vocally on a local computer. We will be looking at development environments that can run on your local computer, but these are mainly for VoiceXML testing, not accessing data and Web site content.

To understand how VoiceXML works, we’ll first take a brief look at XML and its technologies; then we’ll examine how VoiceXML fits in.

**XML Overview**

XML (eXtensible Markup Language) is a nonproprietary, vendor-neutral way to store and transport data. The XML standard is maintained by the World Wide Web Consortium (W3C), a group that develops common protocols for ensuring interoperability across the Web. The XML technical recommendation can be found at [www.w3.org/TR/REC-xml](http://www.w3.org/TR/REC-xml).

Unlike HTML, the XML language allows the developer to mark up content with definitions of what the data is, not what it should look like. Isolating the data from formatting makes it accessible to any application that understands the XML tag set that is applied to the data. This makes it possible to have one source of data that can be output to multiple formats such as HTML, XML, PDF, and VoiceXML. If the underlying data changes, applications accessing the data will not have to be rewritten to accommodate the new content.

XML is a relatively new technology. It allows developers to create their own tag sets instead of relying on predefined elements such as those in HTML. The most important concept of XML is that the data is marked up with tags that define or describe the data, not what it should look like when it is rendered to output. Instead of using tags to define content as bold or italic, it can describe what the data is, using element names such as authorName or bookTitle. Using tags to describe the data allows the calling application to decide how to format the output without affecting the data. With XML, it is now possible to have one central data source that supplies multiple applications.

Developers were told it was an “HTML killer” when XML first hit the marketplace. Magazine articles insinuated that all Web sites were going to have to be reformatted in XML or left at the side of the road, watching the XML train roar by. As with most hype when a new technology comes out, this turned out to be far from the truth. HTML is alive and well, formatting the content for display in Web browsers on millions of sites, as it will be for many years to come. XML does have its place as a presentation medium when used in conjunction with XSLT, the transformation and formatting language, but it was found to have a better use in the data exchange field.

The most powerful use of XML is in the exchange of data between applications. XML and its derivatives are plain ASCII text. As such, the data is vendor-neutral because most, if not all, major applications can read ASCII. If an industry agrees on a
set of XML elements to define the data for that industry, any data that is marked up with those elements will be able to be exchanged between applications. This applies even across platforms. The data can be created in a Unix environment, processed on a Windows box, and stored on a Mac.

XML is an extremely flexible content storage and exchange format. In practice, if your Web site does not exchange or store information with other applications, you are better off staying with the basic HTML that is already on your site. But if you intend to exchange data with companies within your industry, XML is quickly becoming the de facto standard.

**XML Architecture**

On the simplest level, an XML application is a three-tiered architecture. The first tier is the data layer. The data may be stored in a flat file or in a database. The application layer imports the data and transforms it to the correct output format. This may be done by processing requests from the client and providing dynamic content for formatting or by generating output directly from the application. The display layer can be any mechanism that understands the application’s output. Figure 1 shows the structure of a simple XML application.

![XML Architecture Diagram](image)

*Figure 1* XML systems architecture.
XML is the data layer. It can be stored in various formats such as flat files, relational databases, or even as a stream from another XML application.

An optional component of the system is a DTD, or Document Type Definitions. The DTD holds the definitions of the elements and attributes that can be used when marking up the XML data. The data is compared against the DTD to verify that all elements are defined and the hierarchy of the elements within the data is correct. Most applications begin with a DTD to verify input and test the application during production and then remove it for optimization when the application goes live.

The second tier of the application is the formatting and transformation of the data into a specific output. Because XML is a storage mechanism, not a formatting language, the transformation layer uses a stylesheet to mark up the data for various types of output.

Multiple stylesheets can be used against the same data for outputting to various devices. This is a godsend for documentation managers. In the past, if the same data was to be included in a book, printed through a PDF file, and incorporated into a Web page, three copies of the data would have to be made, one for each of the outputs. It was virtually impossible to keep the three copies in sync. When changes were made to the original data file, the copies of the data would not be updated. With XML, there is only one source of data for all types of output. There are no copies of the data, just the original being accessed by different output type’s stylesheets.

The third layer is the output mechanism. The output can be to a computer monitor, a PDA, an Adobe Acrobat PDF file, or a stream to another XML application. The important concept to remember is that the XML data is not changed by the different types of output. The data is stored as XML, formatted and transformed using stylesheets, and then is output to the selected device.

**XML versus HTML: Content versus Formatting**

As a Web developer, you know that data is incorporated into the HTML page along with the tags that format the data. This is a major problem when the data is needed for other projects. Any application that needs access to the content of your Web page has to figure out a way to strip out the HTML and leave the underlying data. Even when all the tags are stripped out and the data lays bare, there is no way for the calling application to know what the data means. The data and the formatting of the data are combined in such a way that the content of your Web site is virtually useless to other applications.

HTML tags tell us nothing about the content of the data, just what the output should look like when displayed in the browser. In Listing 1, the title, the headline of the page, and the information for each train is surrounded by HTML tags that dictate color, line breaks, and other types of formatting. Any application that wants to access the data will have to figure out a way to strip out the formatting and leave the content.

The HTML example causes two problems. If the Manhattan Transportation Authority (MTA) decided to change the content of the page, the calling application that is trying to
access the data would have to know about the changes and go back to get a new set of
data. In addition, if there were changes to the formatting of the MTA page, the calling
application would have to be rewritten to strip off the new formatting.

By isolating the data from any formatting, XML does away with these problems.
Listing 2 shows the same data stored in XML. The data is surrounded by tags that
describe what the data is, not what it should look like. Any application that under-
stands the XML tag set of the MTA can now access the data and output it in any desired
format. MTA can change content, add or delete items, and update the schedule with-
out affecting any application that is using the data.

Listing 1  Data stored as HTML.
8  Chapter 1

Listing 2  Data stored as XML.

XML Elements

XML data is surrounded by tags called elements. The element names are chosen and
defined by the application developer. Many industry leaders are starting to develop
XML tag sets that will define the data for an entire industry. To exchange information
between companies in an industry, applications within the loop must have access to
the defined tag set and apply it to their data.

The XML element definitions are stored in a DTD document or a schema. In theory,
each industry agrees on the definition of the elements that are most applicable for that
industry and then publishes that definition in a public repository. Anyone who wants