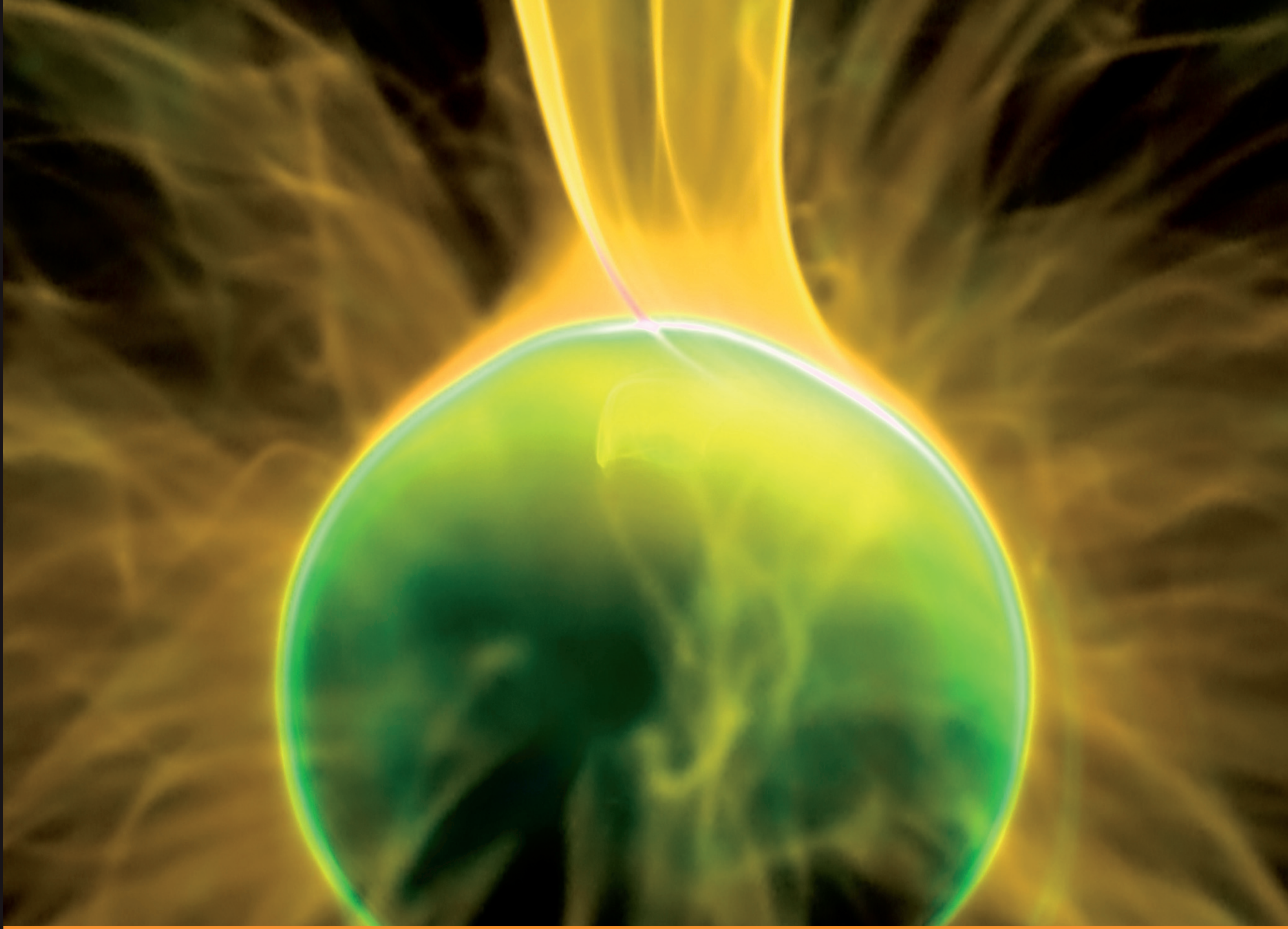


From Technologies to Solutions

Building Websites with VB.NET and

DotNetNuke 4

A practical guide to creating and maintaining your own DotNetNuke website, and developing new modules and skins

Daniel N. Egan

Michael A. Washington
Steve Valenzula

PACKT PUBLISHING

Building Websites with VB.NET and DotNetNuke 4

A practical guide to creating and maintaining your own DotNetNuke website, and developing new modules and skins

Daniel N. Egan

Michael A. Washington

Steve Valenzuela

[PACKT PUBLISHING

BIRMINGHAM - MUMBAI

Building Websites with VB.NET and DotNetNuke 4

Copyright © 2006 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, Packt Publishing, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: October 2006

First reprint: January 2007

Production Reference: 1100107

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 1-904811-99-X

www.packtpub.com

Cover Image by www.visionwt.com

Credits

Authors

Daniel N. Egan

Michael A. Washington

Steve Valenzuela

Additional Material

Charles Nurse

Reviewers

Jerry Spohn

Jim Wooley

Development Editor

Douglas Paterson

Technical Editors

Mithil Kulkarni

Bhushan Pangaonkar

Editorial Manager

Dipali Chittar

Indexer

Mithil Kulkarni

Proofreader

Chris Smith

Layouts and Illustrations

Shantanu Zagade

Cover Designer

Shantanu Zagade

About the Authors

Daniel Egan has held a variety of positions in the information technology and engineering fields over the last nine years. Currently, he is a System Development Specialist for Automated Data Processing's Southern California region, working extensively in database applications and web development. Daniel is an MCP and MCSD.

In addition to his development work, he teaches a VB.NET Certification course at California State University, Fullerton as well as serves on its .NET Advisory board. He is also the founder and chief author of Dot Net Doc (`www.DotNetDoc.com`), a .NET and DNN developer resource website built using the DotNetNuke framework. He has written numerous articles on DotNetNuke and the underlying DNN architecture. He is also the founder of the LA/Orange County DNN Usergroup and is currenly working on two DNN-related projects: DNNUsergroup Online (`www.DNNUGOnline.com`), a portal designed to allow usergroups to broadcast their meetings online, and DotNetNuke Radio, a live internet radio show about DotNetNuke.

Michael Washington is a website developer and an ASP.NET, C#, and Visual Basic programmer. He is a DotNetNuke Core member and has been involved with DotNetNuke for over three years. He is the author of numerous DotNetNuke modules and tutorials. He is one of the founding members of the Southern California DotNetNuke Users group (`www.socaldug.org`). He has a son, Zachary, and resides in Los Angeles with his wife Valerie.

Steve Valenzuela is the manager of the University Extended Education (UEE) IT Department at California State University, Fullerton, where he has worked for the last five years. Steve has worked specifically with DotNetNuke for over two years, in that time re-designing and delivering various Extended Education websites on the DotNetNuke portal framework as well as designing and delivering custom modules that support the function of University Extended Education.

Charles Nurse has been developing software for more than 25 years. He is owner of his own consulting business, Keydance Computer Services, and has been a DotNetNuke developer for over three years, the last two years as a Trustee. He was lead developer on the .NET 2 version of DotNetNuke (DNN 4.0).

A native of Bristol, England, he obtained a Bachelor of Arts in Chemistry from Oxford University. In 1978, he moved to Canada to continue his studies at the University of Bristish Columbia where he obtained a Ph.D. (also in Chemistry), and where he met his wife Eileen. More recently (2003) he completed a Post Baccalaureate Certificate in Object Technology Programming at Simon Fraser University.

He is in the process of developing his own DotNetNuke Developer Resource site (`www.dnndevzone.com`) where he will be providing articles for developing for and with DotNetNuke.

He lives in Langley, BC, Canada with his wife and two children, both students at Simon Fraser University.

About the Reviewers

Jerry Spohn has been working with computers since the age of 11, at which he first began learning programming on a Commodore VIC 20. Times have changed, and he moved through the interesting world of IBM mainframes into PCs. After taking numerous courses on database design, programming, and object-oriented methodologies, he moved into Visual Basic and other Microsoft languages.

Jerry currently works as a Development Manager for a medium-sized software company in Pennsylvania. He also manages over 25 different websites using DotNetNuke, and is the owner of Spohn Software LLC, which does custom development across the entire Microsoft development toolset.

Jim Wooley began working on portals by building his own engine base on XML and XSLT. Just as he was about to release it, the IBuySpy Portal was released.

Promptly dumping his custom solution, he has been working on extending and deploying a number of IBuySpy and DotNetNuke portals. He is always striving to stay at the forefront of technology and enjoys the thrill of a new challenge. In addition, he attempts to pass on the insights he has gained by being active in the community, including leading the Atlanta VB Study Group and serving as INETA NorAm Membership Manager for the Georgia region.

Table of Contents

Introduction

DotNetNuke is a free, open-source evolution of Microsoft's celebrated ASP.NET reference implementation, the IBuySpy portal solution kit. DotNetNuke began life as a framework for constructing data-driven intranet and Internet portal applications, and has now developed into an advanced **web content management system** with tools to manage a dynamic and interactive data-driven website. The DotNetNuke portal framework allows you to quickly create a fully featured community-driven website, complete with standard modules, user registration, and integrated security. This free open-source application puts a staggering range of functionality into your hands, and, either by using it as is or by customizing it to your requirements, you are giving your projects a great head start.

Supported and tested by thousands of developers in the DotNetNuke community across the world, the DotNetNuke framework, on one hand, offers you the luxury of a well-tested and proven architecture, and on the other, the ability to manage your site through an easy web-based administration system.

The book is structured to help you understand, implement, and extend the DotNetNuke framework; it will take you inside DotNetNuke, allowing you to harness its power for easily creating your own websites.

What This Book Covers

Chapter 1 introduces DotNetNuke (DNN) and discusses the meaning and purpose of web portals, and the common aspects of successful web portals. It looks at different types of open-source web portals, and discusses why we selected DotNetNuke for this book. We then meet our fictional client Coffee Connections and, using user stories, gather the requirements needed to build this client's site.

In *Chapter 2* we see how to install a local version of DotNetNuke with Microsoft SQL Server and SQL Server 2005 Express, and cover setting the required permissions on your machine to run DNN properly.

In *Chapter 3* we cover users, roles, and pages. Users are the individuals who visit or administer your portal, and their power depends on the roles that they have been assigned. We discuss how each page of your portal can be administered differently, laying the foundation for the rest of the book. From defining users, to registration, to security roles, this chapter will help you to begin administering a DNN portal.

In *Chapter 4* we cover the standard modules that come pre-packaged with DotNetNuke. We cover their basic uses as well as situations they may be used in. You will use these modules to build your portal's content.

Chapter 5 introduces the administrative functions available to the host and admin logins. These are special logins that have access to all areas of your portal, and are used to secure your site and make changes to its content. This chapter takes you through the tools to make sure you are comfortable with all that is available to you.

Understanding the core architecture of DNN is essential if you want to extend the system or even modify the existing code. In *Chapter 6* we learn how the DotNetNuke framework builds the pages, and the major classes that drive it.

In *Chapters 7* and *8* we take the knowledge we learned in the last chapter and use it to build a custom module. You will learn everything you need to know to start building your own modules so you can extend the capabilities of your portal. After creating your user controls, you will create your data access and business logic layers. In *Chapter 8* you will learn about the DotNetNuke **Data Access Layer** (**DAL**) and the **DAL+**, which take much of the routine work out creating custom modules. We finish our look at development by seeing how to package your module for distribution.

Chapter 9 talks about skins. A skin is the outer layer of your site, and defines the look and feel of the portal. In this chapter we design a custom skin for the Coffee Connections site. You will learn the skills needed to skin both your portal and your module containers.

When you finally have your portal the way you want it to look and function, you are ready to deploy it, and that is what *Chapter 10* shows you how to do. The chapter advises on what you should look for in a web host and helps to steer you clear of common deployment mistakes.

In *Chapter 11* we show you how to take advantage of one of the most exciting features of DotNetNuke: **multiple portals**. These are additional portals that use the same underlying database, but can contain different content. So instead of just having one website, you can create as many as you need using just one DotNetNuke installation. From parent portals to child portals, this chapter gives you the information necessary to create new portals from scratch or to use the new template structure built into the framework.

What You Need for Using This Book

This book has been written both for the beginner wanting to set up a website and also for ASP.NET developers with a grasp of VB.NET. No prior knowledge of DotNetNuke is assumed. To work with the DotNetNuke code, you will need access to Visual Studio .NET 2005 or Visual Web Developer 2005 Express.

This book uses the DotNetNuke open-source project available from `http://www.DotNetNuke.com`. To install and run DotNetNuke, you will need:

- The .NET Framework 2.0
- One of Windows Server 2003, Windows 2000, or Windows XP operating systems
- An installation of SQL Server 2005 or SQL Server 2005 Express Edition
- Visual Web Developer 2005 Express

You can download SQL Server 2005 Express Edition for free from `http://msdn.microsoft.com/vstudio/express/sql/download/`. Visual Web Developer 2005 Express can be downloaded for free from `http://msdn.microsoft.com/vstudio/express/vwd/download/`.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

There are three styles for code. Code words in text are shown as follows: "We then use the `Add` method of this object to add an item to the menu ".

A block of code will be set as follows:

```
Label1.Text = "Hello World!"
                Throw New Exception("Something didn't work right.")
            Catch exc As Exception
                Exceptions.ProcessModuleLoadException(Me, exc)
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items will be made bold:

```
Label1.Text = "Hello World!"
                Throw New Exception("Something didn't work right.")
            Catch exc As Exception
                Exceptions.ProcessModuleLoadException(Me, exc)
            End Try
```

New terms and **important words** are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes for example, appear in our text like this: "clicking the **Next** button moves you to the next screen".

[Tips, suggestions, or important notes appear in a box like this.]

Reader Feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an email to `feedback@packtpub.com`, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com` or email `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer Support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the Example Code for the Book

Visit `http://www.packtpub.com/support`, and select this book from the list of titles to download any example code or extra resources for this book. The files available for download will then be displayed.

[The downloadable files contain instructions on how to use them.]

Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in text or code—we would be grateful if you could report this to us. By doing this you can save other readers from frustration, and also help to improve subsequent versions of this book.

If you find any errata, report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **Submit Errata** link, and entering the details of your errata. Once your errata have been verified, your submission will be accepted and the errata added to the list of existing errata. The existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with some aspect of the book, and we will do our best to address it.

1
What is DotNetNuke?

From company intranets to mom and pop shops to local chapters of the 4H club, most organizations are looking to have a presence on the World Wide Web. Open-source web portals answer this demand by providing easy-to-install-and-use websites that are not only extremely functional but also free. Whether it is to sell services or to have a place to meet, web portals play an important part in communications on the Web.

In this chapter, we will first discuss what web portals are and what successful web portals have in common. We will explore different types of open-source web portals and discuss why we selected DotNetNuke for our project over other available portals. In addition, we will cover the benefits gained by using an established program as a framework and the benefits of DotNetNuke specifically. We will then introduce Coffee Connections, our fictional client. We will get a brief overview of Coffee Connections, determine the specific requirements for its website, and gather the requirements using user stories. This will give you a general overview of what to expect from this book and how to best use it depending on your role and experience with web portals and Visual Basic .NET.

Open-Source Web Portals

So what does it actually mean to have a web portal? We begin the chapter with an explanation of what a portal is, and then go on to the features of a web portal and reasons for selecting open-source web portals.

What is a Web Portal?

You have decided to start a portal and first need to find out what makes a web portal. Does throwing up a few web pages with links to different topics make it a web portal? A portal, in its most basic sense, aims to be an entry point to the World Wide Web. Portals will typically offer services such as search engines, links to useful pages, news, forums, and email, all in an effort to draw users to their site. In most cases, portals provide these services free in the hope that users will make the site their home page or at least come back often. Successful examples include Yahoo! and MSN. These sites are horizontal portals because they typically attract a wide audience and primarily exist to produce advertising income for their owners. Other web portals may focus on a specific group of users or be part of a corporate intranet. They will most often concentrate on one particular subject, like gardening or sports. This type of portal is a vertical portal because they focus inward and cater to a selected group of people.

The type of portal you create depends on the target audience you are trying to attract. You may discover that the portal you create is a combination of both horizontal and vertical portals in order to address specific needs, while simultaneously giving a broader range of services to your visitors. Whatever type of portal you decide on, horizontal or vertical, they both will share certain key characteristics and functionality that guarantee users will return to your site.

Common Portal Features

What makes a great portal? Is it a free prize giveaway, local weather forecasts, or sports scores for the teams you watch? While this package of extras might attract some users, you will certainly miss a large group of people who have no interest in these offerings. There are as many web portals to choose from as programming languages they are written in. However, one thing is for certain: there are particular services your portal should incorporate in order for it to be successful and attract a wide audience.

- **A Gateway to the World Wide Web**: Web portals are the way we start our day. Most of us have set up our home page to one web portal or another and whether you start at MSN, Yahoo!, or Apple, you will notice some common features. Local weather forecasts, movie reviews, or even maps of your community are a few features that make the web portal feel comfortable and tailored for you. Like reading the morning newspaper with a cup of coffee, it gives you a sense of home. Web portals attempt to be the place where all of your browsing starts.
- **Content Management**: Content management has come a long way from the days of paper memos and sticky notes. Computers have done away with the overflowing file cabinets holding copies of every document that crossed our desks. Little did we realize that even though we would be solving one problem, another one would rise in its place. How many times have you searched your computer wondering where you saved the document your boss needs right now? Then once you find it, you need to make sure that it is the correct version. Alternatively, if you run a Soccer Club, how do you ensure that all of your players can get a copy of the league rules? One of the

commonest uses for a web portal is content management. It allows users to have one place to upload, download, and search for a file that is important to them or their company. It also alleviates the problem of having more than one copy of a document. If the document is stored only in one location, you will always have the current copy.

- **Community Interaction**: People have always found a place to meet. From the malt shop on Main Street to your local church, people like to find others who have the same interests. This is one of the main drawing powers of a web portal. Whether you are a Christian looking for other Christians (`http://www.christianwebsite.com/`) or someone who is interested in **Personal Digital Assistants** (**PDAs**) (`http://www.pdabuzz.com`) there is a web portal out there for you. Web portals offer different ways for users to communicate. Among these are discussion forums that allow you to either post a question or comment to a message board or comment on the posts of others. Chat rooms take this a step further with the ability to talk to one or more persons "live" and have your questions answered immediately.
  One of the most interesting ways to express your opinions or communicate your ideas to others on a web portal is to use a **blog**. A blog (also known as a weblog) is sort of like a diary on the Web, except you do not lock it when you are done writing in it. Instead, you make all your thoughts and observations available to the world. These blogs range in topic from personal and comical (`http://weblog.herald.com/column/davebarry/`) to technical (`http://weblogs.asp.net/scottgu`) and, in recent years, have exploded on the scene as the de facto way to communicate on the Internet. Most web portals will offer at least one of these ways to communicate.
- **Security & Administration**: Web portal security not only manages who can access particular sections of the site but also enables administrators to access, add, and change content on the site. Most web portals use a **WYSIWYG** (what you see is what you get) style editor that allows users to add and edit content without needing to know programming or HTML. It is as simple as adding content to a text file. Having users authenticate with the portal allows you to tailor the site to individuals so that they can customize their experience.

Why DotNetNuke?

When the time comes to decide how you want to build your portal, you will have to make many decisions: Do I create my portal from scratch? If not, which web portal framework should I use? What type of hardware and software do I have available to me? Moreover, what is my skill level in any particular platform? In this section, we will discuss some of the better-known portals that are available.

For our portal, we have decided that it would be counter-productive to start from scratch. Instead, we will be using an already developed framework in designing our portal. We will have many options from which to select. We will discuss a few of our options and determine why we believe DotNetNuke fits us best.

PHP-Nuke

Most likely the grandfather of DotNetNuke (in name at least) is PHP-Nuke (`http://www.phpnuke.org`). PHP-Nuke is a web portal that uses **PHP** (a recursive acronym for Hypertext Preprocessor) pages to create dynamic web pages. You can use it in a Windows environment but it is most comfortable in a Linux/Unix environment. PHP is an open-source, HTML-embedded scripting language, which is an alternative to Microsoft's **ASP** (**Active Server Pages**) the precursor to ASP.NET, which is the programming language used in DotNetNuke. PHP-Nuke, like DotNetNuke, is a modular system that comes with pre-built standard modules and allows you to enhance the portal by creating custom modules. Since we will be using a Windows platform, and are more comfortable using ASP.NET, this choice would not fit our needs.

Metadot

Metadot Portal Server is another open-source portal system available to those looking to create a web portal. Metadot states that "its user friendly environment" allows non-technical individuals to create powerful websites with just a "few clicks of the mouse". Like PHP-Nuke, Metadot runs primarily on the Linux operating system (although, it supports Windows as well), Apache web server, and a MySQL database. It uses Perl as its scripting language. For the same reasons as PHP-Nuke, this framework will not fit our needs.

Rainbow

Similar to DotNetNuke, the Rainbow project is an open-source initiative to build a CMS (content management system) based on the IBuySpy portal using Microsoft's ASP.NET. In contrast to DotNetNuke, the Rainbow Project used the C# implementation of IBuySpy as its starting point. It does run on Windows and uses ASP.NET, but our language of choice for this project is VB.NET so we will rule out Rainbow.

DotNetNuke

So why did we select DotNetNuke as the web portal of choice for this book? Well here are a few reasons for selecting DotNetNuke:

- **Open-source web portal written in VB.NET:** Since we wanted to focus on building our web portal using the new VB.NET language, this was an obvious choice. DotNetNuke was born out of a best-practice application called IBuySpy. This application, developed for Microsoft by Scott Stanfield and his associates at Vertigo Software, was created to highlight the many things that .NET was able to accomplish. It was supposed to be an application for developers to use and learn the world of .NET. IBuySpy was an application by the original author of DotNetNuke (formerly IBuySpy Workshop), Shaun Walker of Perpetual Motion Interactive Systems Inc. He originally released DotNetNuke 1.0 as an open-source project in December 2002. Since then DotNetNuke has evolved to version 4.x and the code base has grown from 10,000 to over 120,000 lines of managed code and contains many feature enhancements over the original IBuySpy Starter Kit.
- **Utilizes the new ASP.NET 2.0 Provider Model**: With the release of ASP.NET version 2.0, Microsoft debuted a new provider pattern model. This pattern gives the developer the ability to separate the data tier from the presentation tier and provide the ability to specify your choice of databases. The DotNetNuke framework comes pre-packaged with an SQL Data Provider (Microsoft's SQL Server, MSDE, or SQLExpress). You can also follow this model to create your own data provider or obtain one from a third-party vendor. In addition, the DotNetNuke framework also uses many of Microsoft's building-block services like the **Data Access Application Block** for .NET (`http://www.microsoft.com/downloads/details.aspx?FamilyID=F63D1F0A-9877-4A7B-88EC-0426B48DF275&displaylang=en`) introduced by Microsoft in its Patterns and Practices articles.
- **Contains key portal features expected from a web portal**: DotNetNuke comes pre-packaged with modules that cover discussions, events, links, news feeds, contact, FAQs, announcements, and more. This gives you the ability to spend your time working on specialized adaptations to your site. In addition to this, the DotNetNuke core team has created sub-teams to maintain and enhance these modules.
- **Separates page layout, page content, and the application logic**: This allows you to have a designer who can manage the "look and feel" of the site, an administrator with no programming experience who can manage and change the content of the site, and a developer who can create custom functionality for the site.
- **Ability to "skin" your site**: Separating the data tier from the presentation tier brings us to one of the most exciting advancements in recent versions of DotNetNuke, *skinning*. DotNetNuke employs an advanced skinning solution that allows you to change the look and feel of your site. In this book, we will show you how to create your own custom skin, but you will also find many

custom skins free on websites like core team member Nina Meiers' eXtra Dimensions Design Group (`http://www.xd.com.au`), and Snowcovered (`http://www.snowcovered.com`). These give you the ability to change the look and feel of your site without having to know anything about design, HTML, or programming.

- **Supports multiple portals**: Another advantage of using DotNetNuke as your web portal of choice is the fact that you can run multiple portals using one code base and one database. This means you can have different portals for different groups on the same site but still have all of the information reside in one database. This gives you an advantage in the form of easy access to all portal information, and a central place to manage your hosting environment. The framework comes with numerous tools for banner advertising, site promotion, hosting, and affiliate management.
- **Designed with an extensible framework**: You can extend the framework in a number of ways. You can modify the core architecture of the framework to achieve your desired results (we will discuss the pratfalls of doing this in later chapters) and design custom modules that "plug in" to the existing framework. This would be in addition to the pre-built modules that come with DotNetNuke. These basic modules give you a great starting point and allow you to get your site up and running quickly.
- **Mature portal framework**: As of the writing of this book, DotNetNuke is on version 4.2. It means that you will be using an application that has gone through its paces. It has been extensively tested and is widely used as a web portal application by thousands of existing users. What this affords you is stability. You can be comfortable knowing that thousands of websites already use the DotNetNuke framework for their web portal needs.
- **Active and robust community**: Community involvement and continuing product evolution are very important parts of any open-source project and DotNetNuke has both of these. The DotNetNuke support forum is one of the most active and dynamic community forums on the ASP.NET website. There are currently over 280,000 users registered on the DotNetNuke website. At the time of writing, the much-anticipated DotNetNuke version 4.2 had just been released, and has brought about a significant number of improvements over its previous releases. The core team continues to move forward, always striving towards a better product for the community.
- **Recognized by the Microsoft team as a best-practices application**: In March 2004 at the VSLive conference in San Francisco, the premiere conference for Visual Studio .NET Developers, DotNetNuke 2.0 was officially released, and showcased for the public. This gave DotNetNuke a great leg up in the open-source portal market and solidified its position as a leader in the field.

Benefits of Using an Established Program

Whether you are building a website to gather information about your soccer club or putting up a department website on your company's intranet, one thing is certain— to write your web portal from the ground up, you should plan on "coding" for a long time. Just deciding on the structure, design, and security of your site will take you months. After all this is complete, you will still need to test and debug. At this point, you still have not even begun to build the basic functionality of your web portal.

So why start from scratch when you have the ability to build on an existing structure? Just as you would not want to build your own operating system before building a program to run on it, using an existing architecture allows you to concentrate on enhancing and customizing the portal for your specific needs. If you are like me and use Visual Studio to do your development, then you already adhere to this concept. There is no need for you to create the basic building blocks of your application (forms, buttons, textboxes, etc.); instead you take the building blocks already there for you and assemble (and sometimes enhance) them to suit your needs.

The DotNetNuke Community

The DotNetNuke community has one of the most active and dynamic support forums on the ASP.NET website and has over 280,000 users registered on the DotNetNuke website.

Core Team

The core team comprises individuals invited to join the team by Shaun Walker, whom they affectionately call the "Benevolent Dictator". Their invitations were based on their contributions and their never-ending support of others in the DotNetNuke forum. Each team member has a certain area of responsibility based on his or her abilities. From database functionality and module creation to skinning, they are the ones responsible for the continued advancement of the framework. However, not being a member of the core team does not mean that you cannot contribute to the project. There are many ways for you to help with the project. Many developers create custom modules they make freely available to the DotNetNuke community. Other developers create skins they freely distribute. Still others help answer the many questions in the DotNetNuke forum. You can also be a contributor to the core architecture. You are welcome to submit code improvements to extend, and/or expand the capabilities of DotNetNuke. These submissions will be evaluated by the core team and could possibly be added to the next version.

The DotNetNuke Discussion Forum

When the DotNetNuke project started, one of the things that helped to propel forward its popularity was the fact that its forums were housed on the ASP.NET forums website (`http://www.asp.net/forums/showforum.aspx?forumid=90`). With well over 200,000 individual posts in the main DotNetNuke forum alone, it was, and continues to be one of the most active and attentive forums on the ASP.NET forums website (`http://www.asp.net/forums/`). Beginning sometime after the version 3.x release, the DotNetNuke team puts its finishing touches on its own forum module. It now utilizes this module for most new DotNetNuke questions (`http://www.dotnetnuke.com/tabid/795/Default.aspx`). In both forums, you will find help for any issue you may be having in DotNetNuke.

The main forum is where you will find most of the action, but there are also sub-forums covering topics such as Core Framework, Resources, Getting Started, and Custom Modules. You can search and view posts in any of the forums but will need to register if you want to post your own questions or reply to other users' posts. The great thing about the forums is that you will find the core team hanging out there. Who better to answer questions about DotNetNuke than those who created it? However, do not be shy, if you know the answer to someone else's question feel free to post an answer. That is what the community is all about: people helping people through challenging situations.

The Bug Tracker

Like any application there are bound to be a few bugs that creep into the application now and then. To manage this occurrence, the DotNetNuke core team uses a third-party bug tracking system called *Gemini*, by CounterSoft. The bug tracker is not for general questions or setup and configuration errors; questions of that nature should be posted in the discussion forum. You can view the status of current bugs at the Gemini site (`http://support.dotnetnuke.com`), but will not be able to add new bugs to the system. Reporting a bug is currently done by posting to the DotNetNuke forum. Follow the guidelines currently posted there (`http://www.asp.net/forums/ShowPost.aspx?tabindex=1&PostID=752638`). To summarize: you need to first search the bug tracker to make sure that it has not already been reported. If you cannot find it in the system you will need to supply the forum with exactly what you did, what you expected to have happen, and what actually happened. Verified bugs will be assigned to core team members to track down and repair.

DotNetNuke Project Roadmap Team

If you want to find out what is in the works for future releases of DotNetNuke then you will want to check out the DotNetNuke Project Roadmap (`http://www.dotnetnuke.com/Development/Roadmap/tabid/616/Default.aspx`). The main purpose of this document is as a communication vehicle to inform users and stakeholders of the project's direction. The Roadmap accomplishes this by using User Stories. User Stories are closely related to Use Cases with the exception that they take the view of a fictitious customer requesting an enhancement. The priority of the enhancements depends on both the availability of resources (core team) and the perceived demand for the feature.

The License Agreement

The license type used by the DotNetNuke project is a modified version of the BSD (Berkeley Software Distribution) license. As opposed to the more restrictive GPL (GNU General Public License) used by many other open-source projects, the BSD license is very permissive and imposes very few conditions on what a user can do with the software; this includes charging clients for binary distributions, with no obligation to include source code. If you have further questions on the specifics of the license agreement, you can find it in the documents folder of the DotNetNuke application or on the DotNetNuke website.

Coffee Connections

Wherever your travels take you, from sunny Long Beach, California, to the cobblestone streets of Hamburg, Germany, chances are that there is a coffee shop nearby. Whether it is a Starbucks (located on just about every corner) or a local coffee shop tucked neatly in between all the antique stores on Main Street, they all have one thing in common, coffee, right? Well yes, they do have coffee in common, but more importantly, they are places for people with shared interests to gather, relax, and enjoy their coffee while taking in the environment around them. Coffee shops offer a wide variety of services in addition to coffee, from WiFi to poetry readings to local bands; they keep people coming back by offering them more than just a cup o' Joe.

But how do you find the coffee shops that have the type of atmosphere you are looking for? In addition, how do you locate them in your surrounding area? That's where Coffee Connections comes in; it is its desire to fill this void by creating a website where coffee lovers and coffee shop regulars can connect and search for coffee shops in their local area that cater to their specific needs. Coffee Connections has a vision to create a website that will bring this together and help promote coffee shops around the world. Users will be able to search for coffee shops by zip code, types of entertainment, amenities, or name. It will also allow its customers to purchase goods online and communicate with others through chat rooms and forums.

Determining Client Needs

In any project, it is important to determine the needs of the client before work begins on the project. When designing a business-driven solution for your client your options range from an extensive **Request for Proposal** (**RFP**) and case modeling, to user stories and **Microsoft Solutions Framework** (**MSF**). To determine the needs and document the requirements of Coffee Connections we will use user stories.

We selected **User Stories** as our requirements collection method for two reasons. First, the DotNetNuke core team uses this method when building enhancements and upgrading the DotNetNuke framework. Thus using user stories will help to give you a better understanding of how the core team works, the processes team members follow, and how they accomplish these tasks in a short amount of time. Second, it is a very clean and concise way to determine the needs of your client. We will be able to determine the needs of Coffee Connections without the need for pages and pages of requirement documents.

What is a User Story?

User stories were originally introduced as part of **Extreme Programming**. Extreme Programming is a type of software development based on simplicity, communication, and customer feedback. It is primarily used within small teams when it is important to develop software quickly while the environment and requirements of the program rapidly change. This fits the DotNetNuke project and the DotNetNuke core team well.

User stories provide a framework for the completion of a project by giving a well-designed description of a system and its major processes.

The individual stories, written by customers, are features they wish the program to possess. Since the user stories are written by the customer, they are written in the customer's terminology and without much technical jargon. The user stories are usually written on index cards and are approximately three sentences long. The limited space for detail forces the writer to be concise and get to the heart of the requirement. When it is time to implement the user story, the developer will sit down with the customer—in what is referred to as an iteration meeting—to go over particular details of each user story. Thus, an overview of a project is quickly conceptualized without the developer or customer being bogged down in minor details.

User stories also help in the creation of **acceptance tests**. Acceptance tests are specified tests performed by the user of a system to determine if the system is functioning correctly according to specifications the user presented at the beginning of the development process. This assures that the product performs as expected.

Advantages of Using User Stories

There are many different methods of defining requirements when building an application, so why use user stories? User stories fit well into **Rapid Application Development** (**RAD**) programming. Software and the computer industry in general change on a daily basis. The environment is fast moving and in order to compete in the marketplace it is important to have quick turn around for your product. User stories help to accomplish this in the following ways:

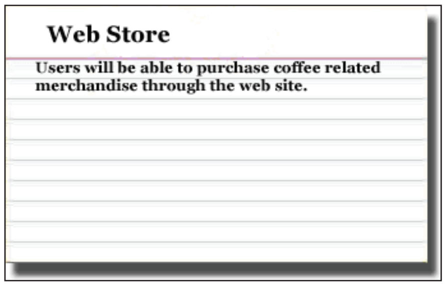
- **Stressing the importance of communication**: One of the central ideas behind user stories is the ability to have the users write down what exactly is expected from the product. This helps to promote communication by keeping the client involved in the design process.
- **Being easily understandable**: Since user stories are written by the customer and not by the developer, the developer will not have the problem of "talking over the head" of the customer. User stories help customers know exactly what they are getting because they personally write down what they want in terms that they understand.
- **Allowing for deferred details**: User stories help the customer as well as the developer understand the complete scope of a project without being bogged down by the details.
- **Focusing on project goals**: The success of your project depends less on creative coding strategies and more on whether you were able to meet the customer's goals. It is not what you think it should do but what the customer thinks it should do.

Coffee Connections User Stories

Below you will find the user stories for Coffee Connections. From these stories, we will use DotNetNuke to build the customer's website. The title of the card is followed by a short description of what is needed. Throughout the book, we will refer back to these as we continue to accomplish the project goals for Coffee Connections.

Title	Description
Web Store	Users will be able to purchase coffee and coffee-shop-related merchandise through the website.
Coffee Shop Search	Users will be able to find coffee shops in their area by searching a combination of zip code, coffee shop name, amenities, or atmosphere and rating.
Coffee Finder Additions	Users will be able to post coffee shops they find and give a description of the coffee shop for other users to see.
Coffee Shop Reviews	Users will have the ability to rate the coffee shops that are listed on the website.
Site Updates	Administrators will have the ability to modify the site content easily using a web-based interface.
Coffee Chat	Users will be able to chat with people from other coffee shops on the site.
Coffee Forum	Users will be able to post questions and replies in a Coffee Shop Forum.

When referring back to the user stories later in the book, we will use a card to compare and determine if we have met the customer's needs.



Web Store

Users will be able to purchase coffee related merchandise through the web site.

Summary

In this chapter, we have discussed the meaning and purpose of web portals, and what successful web portals have in common, looked at different types of open-source web portals, and discussed why we selected DotNetNuke. We then met our fictional client Coffee Connections, and using user stories, gathered the requirements to build its site.

The next chapter will cover the always-enlightening task of installing the software. We will cover what we need to run DotNetNuke and describe the process of installing the framework.

2

Installing DotNetNuke

In previous versions of DotNetNuke (version 3.0), whether you were a developer or just wanted to set up a quick and easy website, you needed to download the entire code base and install all of it up to your server. While the ability to download the code has not disappeared, the core team also allows you to download a slimmed-down version that only contains the files that are needed to upload and work with a basic DotNetNuke site.

In this chapter, we will cover the steps necessary to set up a non-developer version of the website on your local machine. We will show you how to set up the DotNetNuke portal and database by using Microsoft SQL Server 2005 Express Edition. Finally, we will log in as an administrator and change the default passwords.

Installing DotNetNuke (Local Version)

Before you begin installing DotNetNuke, you will need to determine if you have the .NET 2.0 Framework installed. The easiest way is to browse to the following location `C:\WINDOWS\Microsoft.Net\Framework` and look for a folder that starts with V2.0 (for example: v2.0.50727). If you do not see this folder, then you will have to download the 2.0 version of the .NET Framework. You can find the files at the .NET Framework home site (`http://msdn.microsoft.com/netframework/`). For our examples, we will be using Windows XP Professional, IIS 5.1, and version 2.0 of the .NET Framework.

In this section of the book, we will only be using the Install Package, which only contains the items that are needed to deploy to a web host: we will be using IIS to host our site. **IIS** stands for **Internet Information Services** and is the web server application that will run our web portal. If you have downloaded the Source Package and use Visual Studio 2005 then you do not need IIS to work with DotNetNuke. We will also be using SQL Server 2005 in this discussion. DotNetNuke will work easily with SQLExpress. We will discuss installing and working with the Source Package and SQLExpress when we discuss building custom modules. If you haven't installed IIS then make sure that it is installed prior to the .NET Framework.

Clean Installation

If this is the first time you are installing DotNetNuke, or you do not want to upgrade from a previous version, then you will want to perform a clean installation. This means that you will have to build your DotNetNuke instance from scratch. This chapter will walk you through all the steps necessary to accomplish this task. If you wish to upgrade DotNetNuke from a previous version, please refer to the *Upgrading* section towards the end of the chapter.

Downloading the Code

Before we start installing our web portal, we need to download the source code. Go to the DotNetNuke website `http://www.DotNetNuke.com`. You will be required to register before you can download the code. This step is simple, just click on the **Register** link in the upper right-hand corner, and fill in the required information. Provide a working email address, as the registration process will send an email that includes a verification code.

Once you receive the email you may continue to the DotNetNuke site, log in, and download the code. You will find the DotNetNuke source by clicking on the **Downloads icon**. If you want the documentation that comes with DotNetNuke, you will need to download both the Install Package and the Documentation Package. While the file is downloading, take time to explore what the DotNetNuke site has to offer. You will find information that will help you as you build your portal.

When you are downloading, you will also see a Starter Kit for DotNetNuke. The Starter Kit is used to help Visual Studio Developers work with DotNetNuke. We will discuss this download in the Module Development chapter.

Once you have the Install Package downloaded from the site, you can double-click on the ZIP file to extract its contents. Where you extract the file is entirely up to you. Most of the documentation you come across will assume that you extract it to `C:\DotNetNuke` so for consistency's sake we will do the same.

Setting Up a Virtual Directory

After you unzip the files, you will need to set up a virtual directory in IIS. If IIS is not already installed on your system, you can install it by going to **Control Panel | Add Remove Programs | Add Remove Windows Components.**

For more information on installing and using IIS, `http://www.IISFaq.com`, (which utilizes the DotNetNuke framework for its portal) should suffice.

A virtual directory is a **friendly name**, also called an **alias**, that allows you to separate a physical folder from a web address and defines the application's boundaries. A virtual directory is needed if your files are not located in the home directory. The home directory for IIS is found at `C:\Inetpub\wwwroot` (if installed at the default location). The virtual directory, or alias name, is used by those accessing your website. It is the name they type in the browser to bring up your portal so select a simple name.

The following table shows examples of mapping between physical folders and virtual directories. As you can see, we will need to set up a virtual directory for DotNetNuke since its location is outside the home directory, in `C:\DotNetNuke`.