1

Introduction To DotNetNuke

The first web pages were created in the early 1990s, and since then managing the content of those web pages has been key to the acceptance and expansion of the World Wide Web. In the early days, this information was not managed in a specialized editor, as it was later in the 1990s, but by the web browser itself.

As the web expanded, this editing functionality became more restrictive, limited only to the owners of a web page or server. Early web pages were limited to text-based content, and the editors were rudimentary as well, providing simple text entry formatted with the Hypertext Markup Language (HTML) syntax. HTML is the syntax used to code a web page, and web browsers, such as Internet Explorer or Firefox, parse the code to display content.

A lot has changed in the nearly 20 years since the first pages launched. Every person on the Internet has the ability to create and manage his or her own web page, or a collection of pages forming a website. The tools used to manage these pages have evolved over time into extremely powerful and easy-to-use applications, enabling people with all levels of knowledge the ability to manage and create pages.

This chapter provides an overview of some of the common tools, called content management systems, available to aid in the creation of a web page, and how DotNetNuke provides those tools.

This chapter answers the following questions:

- ❑ What is a content management system?
- ❑ What is open source?
- ❑ What is DotNetNuke, and what is the history behind it?
- ❑ What does DotNetNuke provide and how?

What Is a Content Management System?

Before we get into discussing what a content management system (CMS) is, let's cover some of the basics about web pages and how they are assembled and managed. The contents of a web page are usually stored on a web server as files. These files consist of HTML code to control the formatting of the web page. In order to view a web page, a request must be made through a web browser. A web browser will present the HTML information returned from the web server in a formatted manner to the visitor of the website. You can also view the source of the page to see the HTML content that is returned from the web server.

The tools used to manage HTML have progressed over the years, from simple text editors to full-blown applications that do most, if not everything, for the user when creating a web page. A CMS can be a mix of both of these extremes, providing you with a simple way to edit the text that makes up the HTML code for a web page, or to manage all your web page(s) in an automated fashion.

A CMS is an application that allows for the editing, publishing, and management of websites. In general, a CMS will provide you with the basic tools to create a new web page and manage the content on that web page. Most websites will consist of multiple web pages, and a good CMS will provide an organizational structure for these pages and allow you to manage that structure and provide navigation options to your website's users.

There are many other benefits to using a CMS. Most CMSs will provide dynamically generated content, meaning that they store the content that makes up a web page in a predefined format, usually using a templating system to display the content and a database to store the content. The content is retrieved from the database and then displayed to the visitor of the website by applying a template to that content prior to sending the information back to the browser.

This is an important part of a CMS because the separation of the content and the layout allow flexibility in the design and layout of your websites. Most CMS systems will allow you to change the look and layout of a web page through the selection of a new theme, sometimes known as a *new skin*. Applying a new theme to your page or website allows you to change the branding of a site without having to redo all the content contained on the pages. Being able to change the look of a single web page, or even your entire website, with a theme is an important feature to look for in a CMS.

As a CMS, DNN provides the ability to change the theme of a website or page through the use of skins. We will cover skins more in Chapter 3.

Another important feature of some content management systems is their extensibility. Extensibility within a CMS can mean a number of things. In regard to the topic of this book, we will refer to extensibility as the addition and customization of functionality for a website, such as adding the ability to have a weblog (blog), host a survey, or even an e-commerce application.

There are a number of CMSs available for use in building a website, among those are a number of open source solutions such as Joomla! and Drupal that run on Linux Apache, MySQL and PHP, or otherwise known as LAMP. There are even a few open source CMSs for the Microsoft .NET Framework other than DotNetNuke.

What Is Open Source?

DNN is the largest open source project built using Microsoft-based technologies. Open source, in simple terms, refers to an application — be it a web application such as a CMS, a desktop application that runs on a computer, or even an operating system for a computer that controls everything a computer does — that has the source code available, in most cases for free. Source code for a software program consists of the code written in a computer language that defines functionality to a computer and then tells the computer how to execute that functionality.

Open source has become very popular in recent years with the increased acceptance and use of the Linux operating system. Linux initially was a hobbyist's toy, but it has grown into an enterprise application that is used widely in business. Generally, open source projects are developed by volunteers, usually organized into a team that customizes, tweaks, and helps to promote the product. In most cases, these developers are not paid for the modification and enhancements put into open source projects. Although they might use that application in their business, and thus get paid by other means, the work for the application itself is not paid for by any general entity or business.

As Linux proved that the open source model for software could work, more and more applications were developed with open source in mind. Some notable examples of successful open source projects include SugarCRM, JBoss Portal, and Alfresco, all of which have received acclaim from the press and high acceptance in the business marketplace.

Software Licenses

Almost all software packages have a software license associated with them. These licenses generally define the terms of ownership and what the users of a software package are entitled to in regard to the software. In the early days, software licenses were included within the packaging, sometimes on the installation disks themselves.

Now software licenses tend to come into play during the installation process. The installation will usually provide a review of the license and require acceptance before the installation will complete. The next time you're installing a piece of software and see a message prompting you for acceptance, try reading the license, at least some of it; they can be quite long.

Common Retail Software Licenses

The source code for most retail software is not available and is closely guarded as intellectual property by the application's creator/owner. Software licenses for retail software generally limit you as to how you use the application, the number of computers you install the application on, and the liability that the creator/distributor of the software has. In most cases, these licenses will also include terms that define the software as owned by the creator and only available for you to use, not to own yourself.

Common Open Source Licenses

Open source applications, on the other hand, generally differ in many ways from their retail software counterparts. Software that is truly open source and freely available generally comes with very liberal software licenses. These liberal licenses allow for a wide range of scenarios for use of the open source software, from simply being able to use the software without charge to having the right to modify, rename, and redistribute or even sell the software without having to pay the creators or team that manages the software.

A permissive license in open source is considered fairly liberal in its terms because it usually allows you to modify and redistribute the code, and doesn't restrict you to using the same license in future iterations of the code. The BSD license is a commonly used permissive license, originally named after an open source operating system called Berkeley Software Distribution (BSD). The primary restriction with a BSD license is that the included copyright in the source code must remain in future iterations. This copyright is usually held by the owners of the open source project.

Another common type of open source license is known as a *copyleft license*. A copyleft license usually gives the same type of licensing as the BSD but has restrictions as to how the open source code can be redistributed and licensed. The GNU General Public License (GPL) is a common copyleft license. The primary restriction for open source projects released under the GPL is that any modifications and enhancements to the source code must also be released under the same software license. In other terms, any company making changes or enhancements to an open source project released under the GPL must also release that package under the GPL.

What Is DotNetNuke?

DotNetNuke (DNN) is an open source web application framework that can provide CMS functionality. Where DNN differs from other CMSs is the extensibility that it provides through the use of add-ons, known as extensions. DNN can be used for many different types of websites, from the simplest of personal websites to the most complex enterprise internal and external sites. Later in the book we will go into how to configure DNN to work in some common website scenarios.

DNN runs on Microsoft .NET, a platform designed to allow a variety of programming languages to be used for developing applications that can run on Microsoft-based operating systems. DNN is written in the VB.NET language, the latest iteration of Microsoft Visual Basic. DNN is the largest and most utilized open source project running on the Microsoft platform, with more than 700,000 registered users at DotNetNuke.com.

Because DNN runs on the .NET Framework, you can configure the application to run on web servers powered by Microsoft operating systems, including Windows XP, Windows 2000, Windows 2003, Windows Vista, and Windows Server 2008. DNN is a database-driven application, storing most of the pertinent information for the content and settings for a website in a defined database structure running on Microsoft's SQL Server, a database server. In most cases, the database server software can run on the same computer as the web server. Websites that receive a lot of traffic can experience performance issues, so putting the database on a standalone server is a common practice.

It is also possible to configure DNN to run on free tools that Microsoft provides. These tools can run on multiple versions of Windows but are mainly useful for versions such as Windows VISTA Home edition that don't provide the ability to install a web server as part of their core offering. These tools include a free version of the database server, web server, and other components necessary to run a web application. You can find these tools available from Microsoft's Web Platform Installer by visiting `www.microsoft.com/Web/downloads/application.aspx`.

The combination of a Microsoft operating system and database server is a very common configuration in website-hosting businesses. This availability can provide you with many different options for website-hosting services, if you are not comfortable hosting your own website. We will cover these free tools and hosting providers in Chapter 2 when we go through the installation procedures for DNN.

Who Uses DotNetNuke?

DNN is flexible enough as an application that it is used both by individuals and by companies to build just about any type of website imaginable.

For an individual, DNN provides a number of easy-to-use modules that can be configured to run a weblog, family sites, photo galleries, community websites, and even sites for a small business. Both authors of this book use DNN to power their own personal websites. We also have experience configuring DNN to run in a wide variety of business scenarios, including retail sales, education, and healthcare markets.

From an enterprise level, DNN provides many of the tools necessary to manage an intranet website, as well as the flexibility and ease of use to develop and manage customer-facing websites. The tools we will cover in this book will provide you the basic functionality for enterprise websites. Another strength of DNN, although the topic is outside the scope of this book, is the ease of development for new extensions. Developers within business organizations can easily develop new extensions for DNN to meet specific business requirements.

What Does the License Allow?

DNN is released under a BSD license, meaning that it can be modified and redistributed in any number of ways, the only requirement being that the original copyright statements in the source code remain. This license has allowed many individuals and companies to utilize DNN in a variety of markets and projects. Many of these companies choose to rebrand DNN into a product of their own to fit into a vertical market. Doing so is completely legal because of the BSD license.

In addition to the license, there are a few other restrictions to the DNN brand. The terms ''DotNetNuke'' and ''DNN'' are trademarked entities owned by the DotNetNuke Corporation. The DNN Corp. has fairly liberal usage requirements for these brand items, the main restriction being that offshoots of the DNN code base cannot be branded as DotNetNuke or DNN. You can make changes to the source code and release it as another open source application; you just can't call this application DotNetNuke. We'll cover more later in the chapter about why it is in the DNN Corp. and the project's best interest to prevent such actions from occurring.

A Brief History of DotNetNuke

DNN has had an interesting history, most of which we have had the pleasure of experiencing. In 2001–2002 Microsoft was ramping up the marketing machine for their latest platform for software developers, the Microsoft .NET Framework. In order to gain acceptance for this new platform, Microsoft commissioned the development of some free tools for the development community. Now known as *starter kits*, these tools were sample applications to demonstrate the ease of use and power of this new platform.

One of these applications, the IBuySpy Portal, was a sample web portal that could be set up to run on ASP.NET and be used to create and manage a simple web portal. The application was distributed as a best practices implementation for software developers to use in designing their own application on the new .NET Framework. There were two versions of the IBuySpy Portal released, one written in VB.NET and one written in another .NET language, C#. Essentially, both applications were the same; they were just written in different languages to highlight best practice code for a wide range of developers.

ASP.NET is the Web side of the .NET Framework, providing the tools necessary for a web server, such as Microsoft's IIS, to execute code written in various .NET languages.

VB.NET is a popular programming language, the latest iteration of the popular Visual Basic language that was a foundation for the beginnings of Microsoft. DotNetNuke is written in VB.NET.

C# is another popular programming language. Although it's not the basis for DNN, there are a number of extensions for DNN written in C#.

From these initial releases of code, many developers jumped on the .NET bandwagon. On December 24, 2002 Shaun Walker released an enhancement to the IBuySpy Portal. He announced this in the forums located at `www.asp.net`, the primary community website where ASP.NET developers interacted, discussing changes and enhancements to the IBuySpy Portal. The enhancement, called IBuySpy Workshop (IBSW), included many changes and customizations of the original application to provide a more rounded and functional application.

It didn't take long for the application to gain steam within the Microsoft developer community as people realized the functionality that IBuySpy Workshop provided far exceeded the functionality of the standard IBuySpy Portal. Over the next few months, developers continued to make enhancements to IBSW and submit those enhancements back to Shaun, as well as post them in the forums for others to see. As these enhancements continued to roll in, a core group started to form as influential developers in the community.

Shaun, and others, realized that this free application could turn into something more. It was decided that the application was named too similarly to the IBuySpy Portal and should be renamed. After much discussion it was decided to name the product DotNetNuke, after the many other open source portal systems running on technologies other than Microsoft .NET. At the time, the other prominent portals were PHPNuke and PortalNuke, as well as a few others.

Who Develops DotNetNuke?

With a new name, it was time to form a team to help extend the functionality for DNN; thus, the Core Team was founded during the summer of 2003. The Core Team helped to provide the final few releases of what was then known as DotNetNuke 1.0.10 in the fall of 2003. They then began working on the next major version of DNN, a version that would be pivotal for a number of reasons.

DotNetNuke 2.0 was released in March of 2004 and promptly took the portal package to a whole new level. Some of the new features and flexibility added into this release included a totally new skinning engine, a very flexible extension installer, and a new backend model that allowed DNN to work with different types of databases. We'll go into more detail on these features later in the chapter as we discuss the different types of functionality that DNN provides.

DotNetNuke 3.0 came out just short of a year later, in March of 2005. This release had a number of enhancements targeted at extension developers. These enhancements provided methods for developers to access and modify information within the DNN framework, allowing for more functional extensions, empowering better websites using DNN and these extensions. This release of DNN also provided user integration with a new technology that Microsoft was preparing to release later in the fall. This integration with the technology prior to the official release provided Microsoft with some real-world testing, and provided DNN with some great promotion and support.

The 3.0 release of DNN also provided the abstraction of some of the key functionality provided by DNN into individual extensions. Prior to version 3.0, DNN provided a broad range of functionality, in a lot of cases more functionality than most websites needed. This functionality was built into the framework and was not easily removed for websites that didn't need this functionality. By abstracting these functional items into extensions, it was now possible for website administrators to remove unnecessary functionality from a website. This is an important item to note. As an administrator of a website, you might have multiple content authors working on your site. Limiting the amount of information and functionality that authors have access to can be an important part of maintaining a website. By limiting the tools available to authors, you will likely minimize the amount of work that administrators have to do cleaning up mistakes that result from authors' not knowing how to properly use some of the tools.

These abstracted extensions each turned into projects of their own. The DotNetNuke Core Team divided up these functional pieces into individual extensions that were then managed by smaller teams, called *project teams*. Each project team within the DNN open source project managed a particular extension, usually represented by at least one DNN Core Team member. These project teams provide the development, testing, and release of these extensions outside of the normal DNN release cycle. We spend the last half of this book covering some of these individual extensions and how you use them to build various types of websites.

Multiplatform Support

The fall of 2005 brought the release of two versions of DNN, DotNetNuke 4.0 and DotNetNuke 3.2. While numbered differently, in theory the two releases provided the same functionality. The difference between the two versions was the technology they ran on. Until this point, DNN had run on Microsoft's .NET Framework version 1.0 and 1.1, the version of .NET that DNN 3.2 ran on. In the fall of 2005, .NET 2.0 was released. DNN wanted to capitalize on the marketing behind the new .NET Framework, so DotNetNuke 4.0 was compiled for the new framework and released the same day that .NET 2.0 was released.

For the next year, two different versions of DNN were released in tandem, a version for .NET 1.1, and a version for .NET 2.0. In the fall of 2006 the final 3.*x* version of DNN was released as DNN 03.03.07. This ended the support for the .NET Framework 1.1, and all releases since have supported .NET 2.0. In the future, it is likely that .NET 2.0 support will be sunsetted, as the DNN platform moves to utilize features of .NET 3.5 that aren't available in 2.0. As of this writing, however, all current releases of DNN run on .NET 2.0. You will find .NET 2.0 support from nearly every hosting provider when you start looking for a company to host your website.

We'll talk more about configuring DNN in a hosting environment in Chapter 2.

Another change for the DNN projects came in the fall of 2006. While the project had always been managed by Shaun Walker and three or four individuals on the board of directors, the copyright, trademarks, and other materials for DNN were owned by Shaun's consulting company, Perpetual Motion Interactive Systems (PMIS). Although this provided a great background for the project in its infancy, as the project grew and more individuals and businesses started to rely on DNN on a larger scale, there were discussions of what would happen to the project if something were to happen to Shaun. Being the sole owner of PMIS, if he were to get hit by a bus, the common what-if scenario in the IT world, what would happen to DNN as a whole?

To prevent this potential scenario, the members of the board of directors on the DNN Core Team formed a corporation licensed in the state of Delaware, the DotNetNuke Corporation. The process of transferring the ownership of the copyrights and trademarks for DNN to the DNN Corporation started at this time. Now if something were to happen to one individual, DNN can live on due to the management of the DNN Corporation.

The DotNetNuke Corporation

The DotNetNuke Corporation is tasked with managing the DNN project, continuing the growth and development of the project, and protecting the trademarks and copyrights associated with the project. This is a necessary and important role in an open source project. Due to the open license that DNN is released under, it is possible for anyone to take the source code and re-release that code in just about any way they deem fit. However, they are not allowed to utilize any DNN trademarks in doing so; the corporation is responsible for policing and protecting this usage.

As we've discussed, the DNN project has been around for a number of years and has quite a bit of brand recognition/value within many circles. If someone were to take the source code for DNN and re-release it under the same name, the value of the brand would become diluted. The potential for confusion exists within the community if there appear to be two projects called DotNetNuke.

Another potential issue that the DNN Corporation is tasked with is to the prevent something known in the open source world as a *fork*. A fork is when an individual or group decides to take the source code for an existing project and start up their own open source project based on that code. Although this is completely legal based on the license for DNN, it has the potential to fragment the strong community that has been built around DNN.

Since its inception the DotNetNuke Corporation has been working to secure funding to help continue to drive the DNN project. As of the time of this writing the DNN Corp has recently secured a round of venture capital funding that will allow them to continue to grow. For a more thorough history of DNN and the funding process, check out *Professional DotNetNuke 5* (Wiley, 2009).

DotNetNuke Professional and Community Editions

The DotNetNuke Corporation recently announced and released a new product called DotNetNuke Professional Edition. This product is a licensed application sold and supported by the DNN Corporation. At the same time, the free version of DNN was rebranded as the DotNetNuke Community Edition.

There are a few differences between these two editions, the biggest being the cost associated with a license of the Professional Edition. The Community Edition remains free. The Professional Edition has a few enhancements over the Community Edition, as well as services provided with purchase, such as a year of free upgrades and product support provided by the DotNetNuke Corporation.

The Community Edition remains the most adopted and widely used version of DNN and will continue to grow just as it always has. The content in this book applies to both the Community and Professional versions of DNN. In most cases, you will find that the Community Edition provides everything you need.

If you work for a company that is opposed to using open source software, the Professional Edition might provide your management team with the assurances that DNN can be used in enterprise environments.

What Is the DotNetNuke Ecosystem?

DNN has a very extensive community that has grown with the project since its inception. That community includes individuals and businesses that provide a lot of information, services, and products for DNN. Because of DNN's extension model, it is very easy for administrators to plug new extensions into their website. Combined with high developer acceptance of the platform, this has allowed an extensive ecosystem to spring up around the platform. It is easy for individuals and businesses to find support and products for DNN though the members of this ecosystem.

Some of the strength of this ecosystem is the availability of thousands of modules and skins for DNN, in addition to the modules and skin packages that come with the packages available from `dotnetnuke.com`. Some of these modules and skins are offered for free, whereas others have a price associated to them. A lot of the individual developers/businesses sell these modules on their own websites, but many of them can also be found at the DotNetNuke Marketplace (`http://marketplace.dotnetnuke.com`).

This ecosystem didn't form over night; it has taken time to grow and continues to grow today. Most members of the ecosystem got their start in the DotNetNuke Forums (`http://forums.dotnetnuke.com`), assisting other customers and learning as much about the platform as they were able to. Because of the continual growth of the platform, it is important for those providing products or services to keep a finger on the pulse of the project and users in the community, so you will still find those community members/vendors posting frequently in the forums.

One of the strengths of DNN is the breadth of knowledge available from its community members. As the project has grown, so has the community itself. You will find great blog posts from active members in the community that provide insight into the platform and how best to use the different tools that DNN provides. You can find blog posts at `DotNetNuke.com` as well as `DotNetNukeBlogs.com`, a website run by this book's coauthor, Chris Hammond. In addition to the helpful blog posts, you will find a great group of people that provide support to DNN users in the forums. If you run into a problem with DNN, that is the first place you should look for an answer.

What Does DotNetNuke Provide and How?

DNN enables you to create web pages and add content to those pages through the use of extensions. There are a few types of extensions; the one we will primarily use through this book is the module extension. DNN provides an extensive list of modules out of the box, offering a wide range of functionality that you can choose to implement on your website. There are 25 modules available for installation, either during the DNN installation process or after your website is up and running. Chapter 2 covers the DNN installation process.

With the advent of DotNetNuke version 5, the term extensions has been entered into the vocabulary of a DNN website. In previous versions of DNN, there were many different types of extensions, but they were not grouped together to be called extensions.

DNN's modules range from the Text/HTML module, a simple module that provides the most basic CMS functionality within DNN, to the Store module, which provides a very capable e-commerce solution for your website. Before we get into this extended functionality, however, it's important to have a basic understanding of how DNN works.

When someone visits a website, the information is downloaded from the web server to the user's web browser for display. How that page is actually generated depends on the type of web server it is running on and what, if any, CMS is being used to manage the website. Some CMSs will generate individual files for each web page; these files are known as *static files* and are stored as files on the hard drive of the web server. They are static because once they've been generated by the CMS, they exist entirely on the server. In the future, the CMS may update the files, but that depends on the requirements of the CMS system.

Other CMSs will actually create the web pages dynamically when requests are made from visitors of the website. In most cases, the information used to generate the web pages is stored in a database. This database may be located either on the web server itself or on a specific database server. CMSs that dynamically generate the web pages returned to visitors allow for websites that can be highly flexible in the layout and formatting of the content to be displayed on the web pages.

DNN is a CMS that generates the web pages using this dynamically driven model. Almost all the content that is entered through the web browser interfaces for DNN will end up being stored in a database. The primary database that DNN supports is Microsoft SQL Server 2000/2005/2008 or SQL Server Express. It is possible to get DNN to run on other databases such as MySQL, but that discussion is beyond the scope of this book.

When managing your DNN website, you will provide most of the maintenance through your web browser. Only rarely will you have to maintain anything outside of the browser — for example, installing DNN (see Chapter 2). The following sections provide a brief overview of the basic functionality that the DNN framework provides to website administrators. This will give you a better understanding of how things will fit together over the next few chapters.

Security

Before we talk about the content on your website, it is important to understand that DNN provides administrators with the ability to define multiple levels of permissions for their websites. You can create a website that is completely open to the public for viewing or one that requires users to register and log in, in order to view it. Once you have users logging in to your website, you can take the security further and provide individuals or groups with the ability to view or edit content. DNN's security model is extremely flexible, even allowing you to limit users or groups to edit only parts of a web page instead of the whole thing. Chapter 3 covers the security functionality within DNN in more detail.

Pages

Most websites you run across on the Internet have multiple pages, different files on the web server that provide different content for each of the pages that are loaded by request. DNN is no different. As a site administrator, you can create and manage pages within your website. As you'll see in Chapter 3, adding pages is easy to do through the web interface. You can choose a predefined template for the page that consists of a defined layout, skin, and even already populated content. Pages within DNN, sometimes referred to as *tabs*, have multiple properties, such as title, layout, descriptions, keywords, as well as permissions. These can be defined at or after creation. Another important aspect to pages within DNN is the impact that they have on the navigation and organizational structure of your website. We'll cover all these aspects in more detail throughout the book.

The final thing to keep in mind with your pages is that they do not actually store content, other than the items discussed earlier in this section. Pages have a skin applied to them that defines either a single pane or multiple panes for use on the page. A *pane* is part of a DNN skin that provides a location for placing content.

Extensions, Modules, and Skins

Extensions enable you to add functionality to your DNN website with minimal effort. Although there are multiple types of extensions, we will be focusing primarily on the skin and module extensions. Extensions are installed using compressed ZIP files, known as packages, which you upload through the browser through the extension installer. Chapter 3 covers installing extensions in more detail.

Skin packages provide the layout and design for your DNN pages. Because of the flexibility built into DNN, it is possible to completely overhaul the look of your website simply with a few clicks of the mouse. Beside the graphical and style elements of a skin package, the important aspect of skins is the number of panes they provide. A skin for a DNN page controls the number and placement of panes on a page.

A simple skin might contain a single pane in which all of the content for that page resides, whereas a more complex skin can have multiple panes with different formatting applied to each pane to control the look and feel of the content within that pane. Chapters 3 and 4 cover how to utilize skins on a portal and page level, as well as how to arrange content within your skins. We will not get into the development of custom skins in this book. If you are interested in custom skin development, check out *Beginning DotNetNuke Skinning and Design* (Wiley, 2007).

The Module type of extension provides the functionality and content on your DNN website. There are numerous types of modules available for DNN, providing countless ways to configure and manage different content on the site. DNN comes with a selection of modules that you can install during the initial setup or after your website is up and running. Modules, like the other extensions within DNN, are uploaded through the browser in a compressed ZIP file. DNN extracts the file in this ZIP folder and runs any database scripts necessary to install the module on your instance of DNN. Once a module has been installed, it will be available for placement on a page. We'll cover this process for using modules in Chapter 4 and then cover specific modules in detail in the later chapters of the book.

The primary focus of this book is on using the available resources that DNN provides. The following table lists the modules that are included in the download of DotNetNuke 5, along with a brief description of each. We will cover a number of these modules and their usage throughout the book.

Available Modules

Module Name	Description
Adsense	The Adsense module enables you to easily set up and configure Google's Adsense advertising tools on the pages of your website.
Announcements	The Announcements module is a simple module that allows you to enter a title and description, and link to another resource. The module provides a list of recent announcements with the ability to restrict them by date, as well as archive the content.
Blog	The DotNetNuke Blog module provides an easy-to-use blogging tool. It can be configured to provide a single weblog or multiple weblogs to users, depending on the needs of your website.
Documents	The Documents module is a useful tool if you are providing downloadable documents to visitors of your website. The module can be configured to provide download links to users, as well as track the metrics from those downloads.
Events	This module can be used to provide a list of events, in either a calendar format or a listing format. Events can have registration and notification options created for them as well; this is useful for managing websites for organizations that schedule meetings for groups of people.
FAQs	The FAQ module enables you to create a list of frequently asked questions and answers for users of your website.
Feedback	The Feedback module is a useful module for providing users with the ability to send administrators an e-mail with by providing a simple form on the website. It also has some useful reporting tools.
Form and List	The Form and List module is a very useful, yet complex, module that enables you to create forms in which data can be entered, and then display that data in a formatted manner for display to your website users. This module can be useful for reporting scores from events, or other statistical information.
Forum	The Forum module provides advanced community features such as thread posts and replies, thread status, e-mail notifications and a number of other functions. This module is a must for a community website.
Help	The Help module is useful for creating documentation on a website. You can create nested categories and organize content in a formal structure.
Text/HTML	The Text/HTML module is one of the most used modules for DNN. It provides you a simple What You See Is What You Get (WYSIWYG) editor that provides a rich text entry interface, easily allowing users to style content with basic HTML tags without having to know HTML syntax.
IFrame	The IFrame module enables you to use an HTML feature to display the contents of another website in a frame on a different web page. You can control the width, height, title border, and scrolling of the frame.

Module Name	**Description**
Links	The Links module provides an easy interface to display a list of links in multiple formats. You can choose a list or a drop-down list. You also get an easy entry and maintenance screen for the links without having to know HTML to generate the lists.
Map	The Map module allows for simple mapping functionality within DNN and can be useful for tracking meeting locations.
MarketShare	The MarketShare module enables you to link to content at `http://marketplace.dotnetnuke.com` while earning referral credit ($) from any sales generated from your links.
Media	The Media module enables you to display images, videos, and audio files on a website without having to know the HTML necessary to include such content.
NewsFeeds	The NewsFeeds module enables you to easily display and format RSS streams from other websites; these generally consist of latest news or blog feeds from the sites that distribute them.
Reports	The Reports module enables you to generate reports using SQL queries against the website's database. These can be useful for seeing the statistics of your website and its visitors.
Repository	The Repository module is a complex module that allows for a number of different configurations. In general, it is a way to manage a collection of items, such as contacts, photos, videos, and even articles.
Store	The Store module is DNN's e-commerce component; it can be configured in a variety of ways and can track products and purchases.
Survey	The Survey module enables you to configure and track results for surveys on your website.
User Defined Table	The UDT module is the basis for what has become the Form and List module. The module allows you to setup simple and complex HTML tables, allowing for data entry and display of this customized data.
Users Online	The Users Online module allows you to track and display the current user activity on your website.
Wiki	The Wiki module provides basic wiki functionality to the visitors of your website, allowing for user-generated and -moderated content.
XML	The XML module enables you to display the contents of an XML file, applying style to it through the use of an XSL file. It is not a commonly used module but can be very flexible for someone with familiarity in Extensible Stylesheet Language (XLS).

In addition to the modules provided with the core package for DNN, thousands of third-party modules are available; some of these are free and some are for sale.

Upgrades

DNN is built to be very flexible, enabling you to implement upgrades to new versions of the framework, as well as to upgrade the extensions installed on a website. Upgrading extensions follows the same process as installing them. The framework checks to see if an upgrade is being performed and completes the appropriate actions. Chapter 9 covers upgrades to the platform in more detail, and Chapter 9 discusses upgrading the functionality of modules and skins.

Users Management

Users on your website can interact with content and functionality in different ways. Some of these interactions will require users to be logged in, whereas others will not. DNN provides a customizable registration system, allowing you to define user profile fields that must be filled in during the registration process.

Common Misconceptions

Because DNN is an open source project and available for free, there are quite a few common misconceptions about the software and its use. As mentioned previously, the ecosystem around DNN is quite active and healthy. Because of the open source nature of the project, many people expect everything regarding DNN to be free, including support, customizations, implementation, and training. As with most things in life, that is just not the case. Although you can find a lot of great free products for DNN, and free support on the DNN Forums (`forums.dotnetnuke.com`), in most cases you will find that businesses supporting or providing services for DNN are not free, and why should they be? These companies are providing services and products; the fact that they are for an application that comes with a free license doesn't mean that they have to release their products and services for free.

Keep that in mind when you are getting into a project that requires work above and beyond the scope of what DNN provides out of the box. With that in mind we lead on to another common misconception. While DNN provides a whole world of functionality out of the box, with the core functionality and core modules, you won't necessarily be able to accomplish all the tasks your website needs with these free tools.

The ecosystem for DNN exists to provide the knowledge and products necessary to accomplish tasks with DNN that are not provided with these out-of-the-box tools. While it is often possible to accomplish a whole host of functionality with unique configurations for free tools, it will likely save you time, and ultimately money, to invest in third-party extensions or services.

Summary

We've covered a lot of information about content management systems, open source projects, licenses, and DotNetNuke in this first chapter. The community and developers behind DNN have worked hard to make it a very functional product. While we will be the first to admit that it does contain some things that are not always the easiest to comprehend, we hope that this book will make using DNN easier for everyone. We hope this chapter has whetted your appetite for using DotNetNuke, as well as for the rest of our book. In Chapter 2, we will cover the basics of installing DNN so that you can get your website up and running.