DotNetNuke Installation Guide

Charles Nurse

Version 1.0.03
Last Updated: June 20, 2006
Category: Installation

DotNetNuke Installation Guide

Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this document remains with the user.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, places, or events is intended or should be inferred.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Perpetual Motion Interactive Systems, Inc. Perpetual Motion Interactive Systems may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Perpetual Motion, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

DotNetNuke® and the DotNetNuke logo are either registered trademarks or trademarks of Perpetual Motion Interactive Systems, Inc. in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

DotNetNuke Installation Guide

Abstract

This document describes the Installation procedures for DotNetNuke versions 3.x and 4.x.

Contents

DotNetNuke Installation Guide

Warning

If you are upgrading you must read carefully the Chapters on Upgrade, especially on Configuring web.config for Upgrade.

Before proceeding any further, make a back-up copy of your web.config file. (We recommend you name it web.backup.resources or web_backup.config)

Introduction

With the release of version 3.0, DotNetNuke introduced an enhanced Installation and Upgrade procedure, and in subsequent versions, these procedures have been updated.

In version 2.1.2 and earlier, Installation and Upgrade was controlled by AutoUpgrade. In addition some resources (Skins, Modules, Containers) could be installed using the Web Upload control, which was part of the File Manager. Both of these processes have been significantly enhanced.

Types of Installation

There are three major types of installation / site provisioning available in DotNetNuke and these will be covered in this document

- Installation of DotNetNuke 3.x
- Upgrade of an existing installation to 3.x
- Installation of DotNetNuke 4.x
- Upgrade of an existing 3.x installation to 4.x
- Installation/Creation of new Portals
- Installation of Resources
  - Skins/Containers
  - Modules
  - Language Packs

Installation Features

There are also a number of “features” of the application that deal with Installation and Upgrade.

- AutoUpgrade – the ability to automatically detect whether an installation/upgrade is required and install or upgrade automatically
- Install Template – An xml based template that can control the installation, including host settings, desktop modules and portals
- DNNConfig file – the file saves the current database version, to avoid a call to the Database on every App_Start
- InstallationDate – determines whether to automatically generate "new" machine keys for a "new" installation
- New Install Page (Install.aspx) – this page provides feedback on the installation/upgrade process
- Install Folder – this folder provides administrators with the ability to bulk FTP new resources to a location on the file system, and through the Install page, bulk install these resources.
- Upload Resource – Admin or Host UI components that deal with resources that can be installed, have an "Upload" action button that can be used to launch the installation.

Installation of DotNetNuke 3.x

One of the biggest changes in version 3.x of DotNetNuke, is the enhancement to the Installation/Upgrade process. In earlier versions of DotNetNuke, an AutoUpgrade function detected the Database Version, compared it with the Assembly Version, and ran any incremental scripts required to bring the Database Version up to the current Assembly Version.

There was no conceptual difference between an Install and an Upgrade – only the number of scripts that were executed. So, for example, an upgrade from 2.1.1 to 2.1.2 required the execution of only the 2.1.2 script, whereas an upgrade from 2.0.1 required the execution of 5 scripts – 2.0.2, 2.0.3, 2.0.4, 2.1.1, 2.1.2. A fresh Installation of 2.1.2 required the execution of 19 version scripts.

DotNetNuke 3.x modifies this process so that fresh installs do not have to execute 34 scripts (as of 3.0.12), but instead execute 2 scripts DotNetNuke.Schema and DotNetNuke.Data. Upgrades still execute the incremental scripts.

Setting up your Site Ready for Installation

If you are installing a new instance of DotNetNuke then the first steps you need to accomplish will depend on whether you are creating a site on a local intranet or on a remote server. The complete process will be described here. If DotNetNuke is being installed into a Shared Hosting Plan, then the Hosting Provider has probably already set up the site and database for you.

1. Unzip the Package

The first step is to unzip the package to the physical location where the Application will reside. Starting with version 3.1, DotNetNuke is available in two packages:

- DotNetNuke_X.Y.Z_Install.zip - An Install Package – This package contains the ascx, aspx files as well as other content files and compiled assemblies (dlls) but does not contain any of the Visual Basic code files. Use this package if you don't expect to do any modifications to core files.
- DotNetNuke_X.Y.Z_Source.zip – A Source Package – This package contains all the files related to the core DotNetNuke project including the Visual Basic source files.

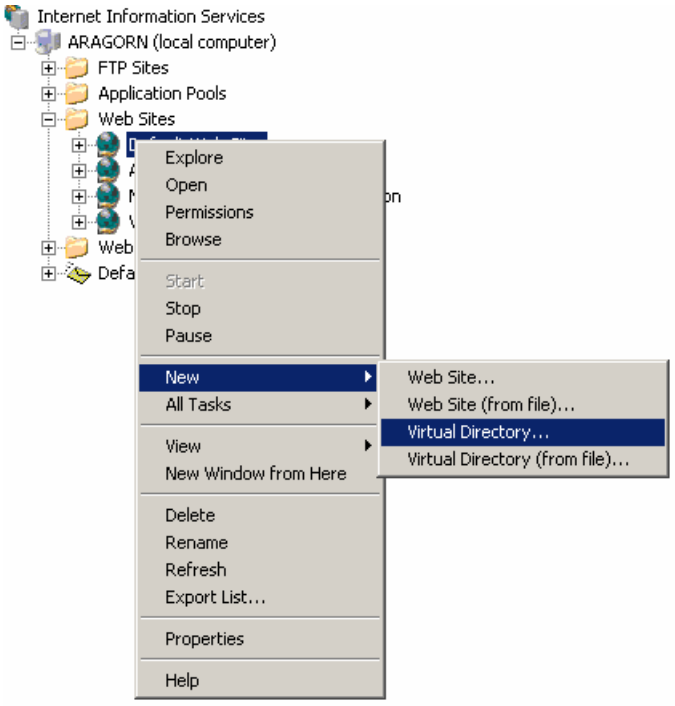
(where X.Y.Z is the version number, e.g. DotNetNuke_3.2.0_Install.zip).

On local intranet configurations you can place your website anywhere (e.g. C:\DotNetNuke) although by default IIS (Internet Information Server) expects the sites to be located at C:Inetpub\wwwroot\...

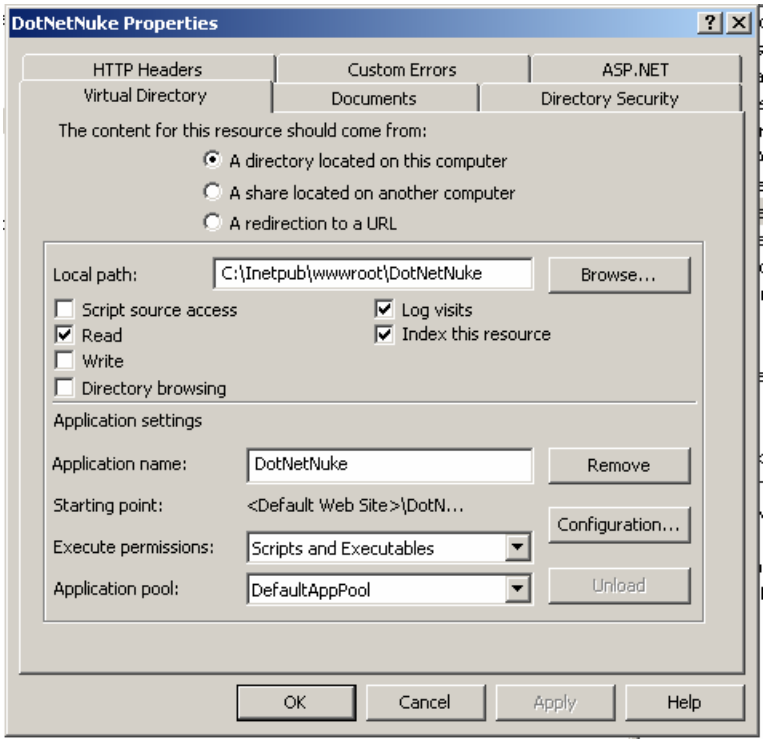
On remote hosting servers you will need to upload the files to your website following the procedures provided by your hosting provider.

2. Configuring IIS (Internet Information Server)

In a local intranet configuration, create a Virtual Directory in IIS called DotNetNuke that points to this physical directory. In IIS 6 right-click on "Default Web Site" and select New/Virtual Directory.



This will launch a wizard that will allow you to configure your Virtual Directory. Once you have created the Virtual Directory using the Wizard you can edit the properties of the new Virtual Directory, by right-clicking on the new Virtual Directory, and selecting Properties.

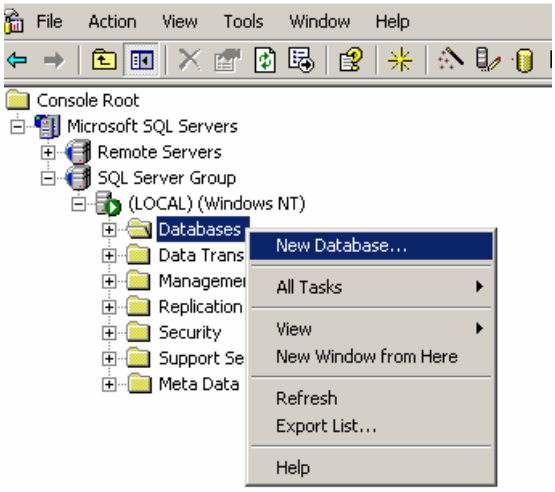


3. Create a Database in SQL Server

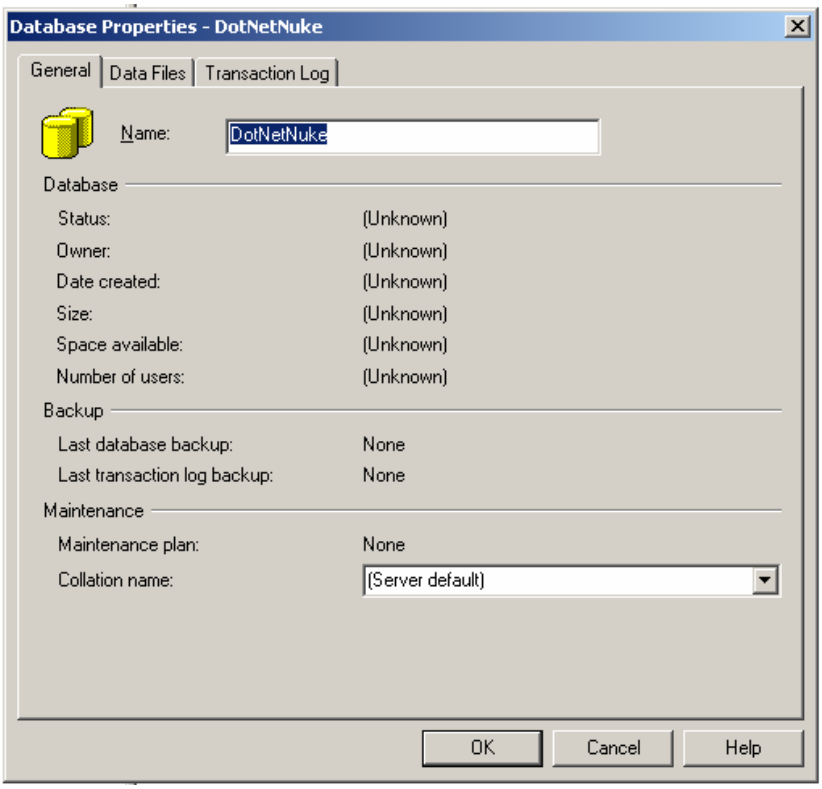
In a remote hosting environment, your hosting provider will probably have configured a SQL Server database for you, and will provide you with instructions regarding how to connect to the database.

In a local configuration you will need to create a new Database. How you create your local database depends on the client Database Tools you have available.

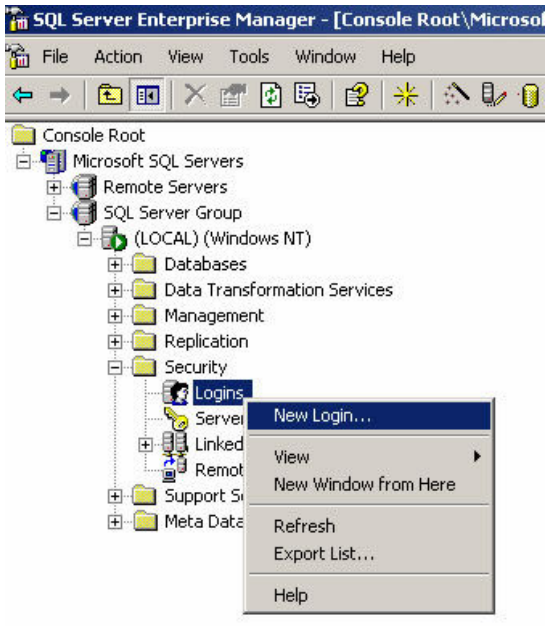
In Enterprise Manager (SQL Server 2000), you create a new Database by right-clicking on Databases and selecting New Database ...



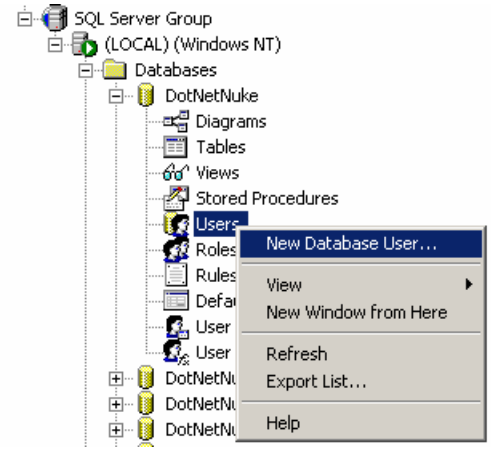
In the "New Database" dialog choose a name for your database (you can select any name, but the default connection string assumes the name DotNetNuke) and select Ok. You can also set a custom Collation name, and also choose the file name for the database, but the defaults for all these properties work.



The next step is to configure a database user for your DotNetNuke database. If you do not already have one, create a new SQL Server Login, by right-clicking on Security/Logins.

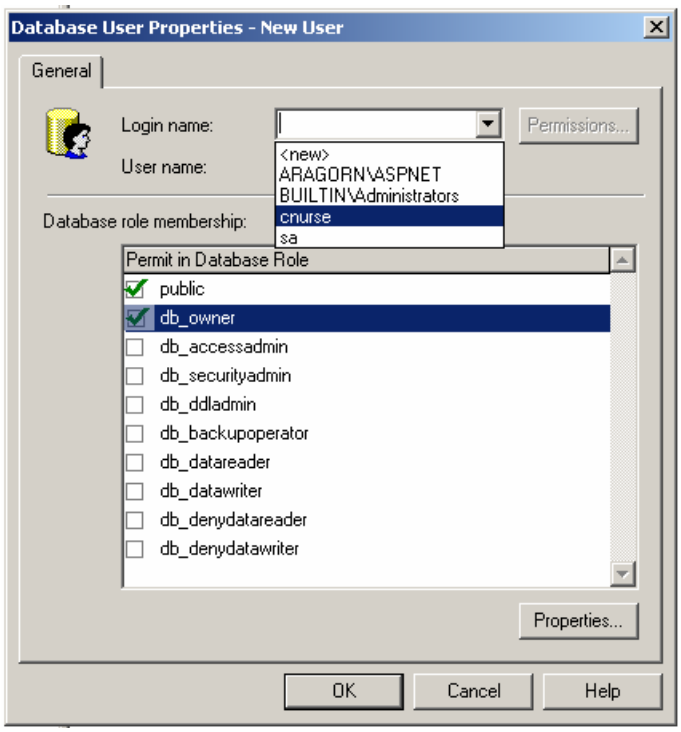


Once you have created the Login, right click on Users (under your new DotNetNuke Database) and select New User.

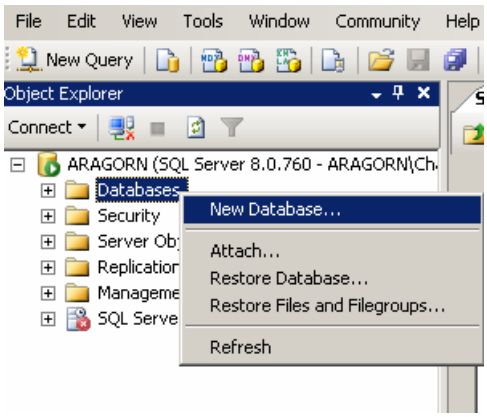


In the New User dialog, select the Login you would like to use from the drop-down combo, and set the public and db_owner permissions (you don't actually require full

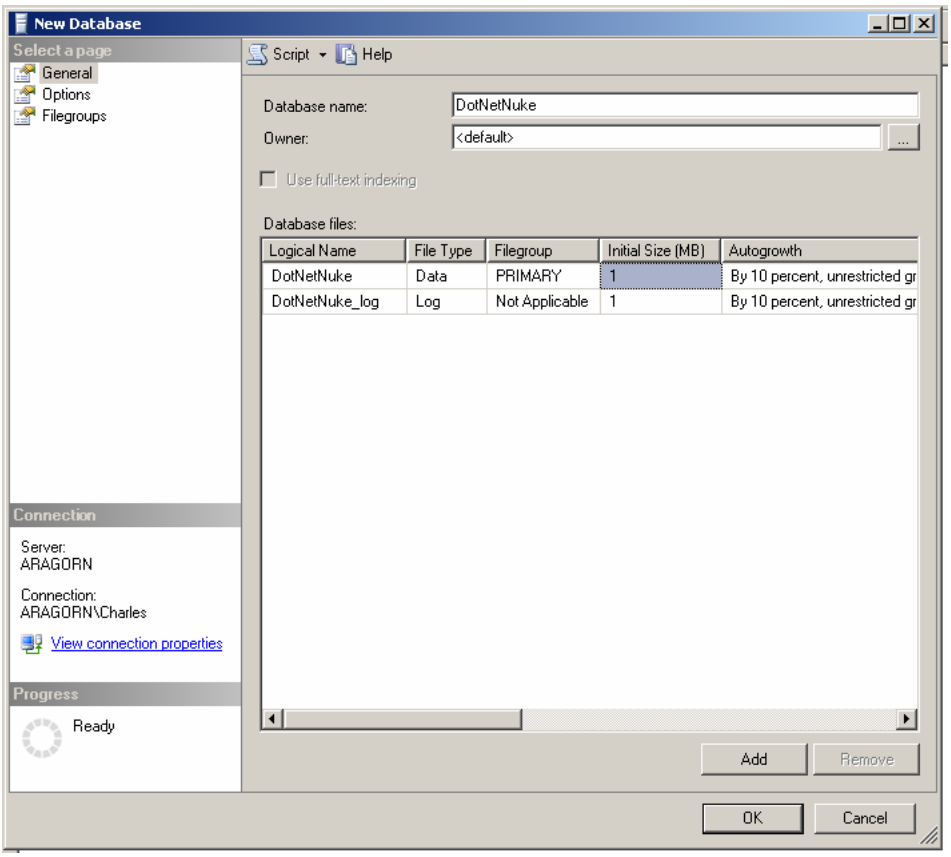
db_owner permissions – see later in the section on configuring the data provider in web.config for the required minimum permissions)



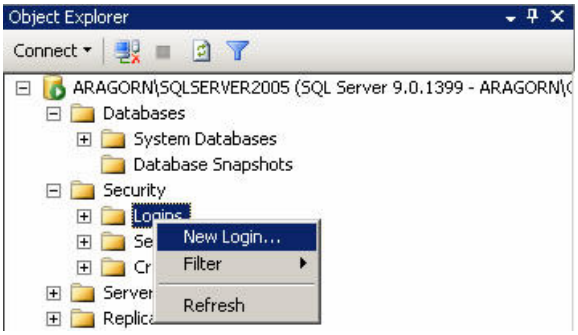
The process is similar in SQL Server Management Studio (SQL Server 2005). Again right-click on Databases and select New Database.



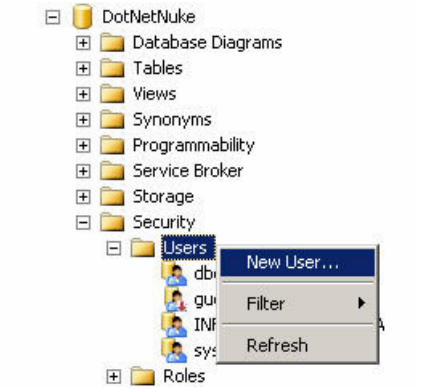
In the New Database dialog enter a name for the Database.



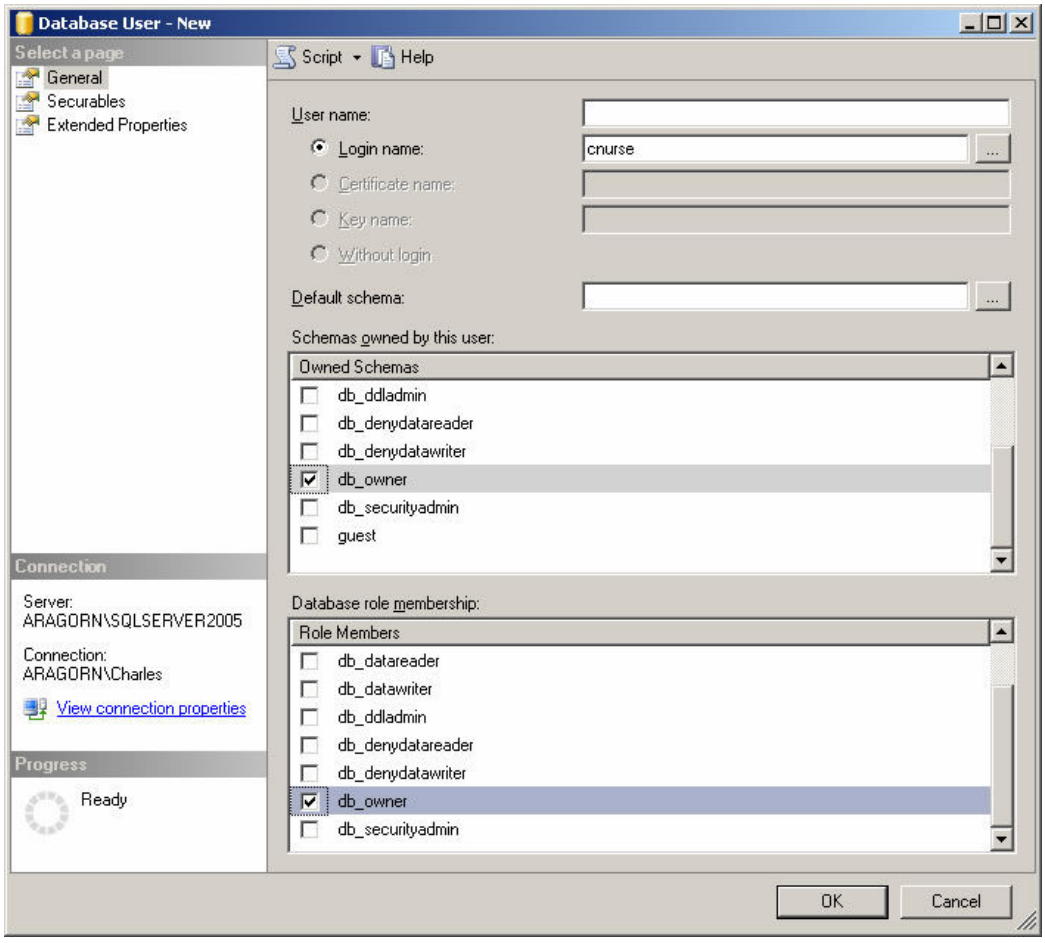
To create a new Login right-click on Security/Logins.



To add a new User for your newly created Database again right-click on Users for your Database (in SQL Server Management Studio this is under the Security node under your Database)



and again make sure that the db_owner permissions are set .

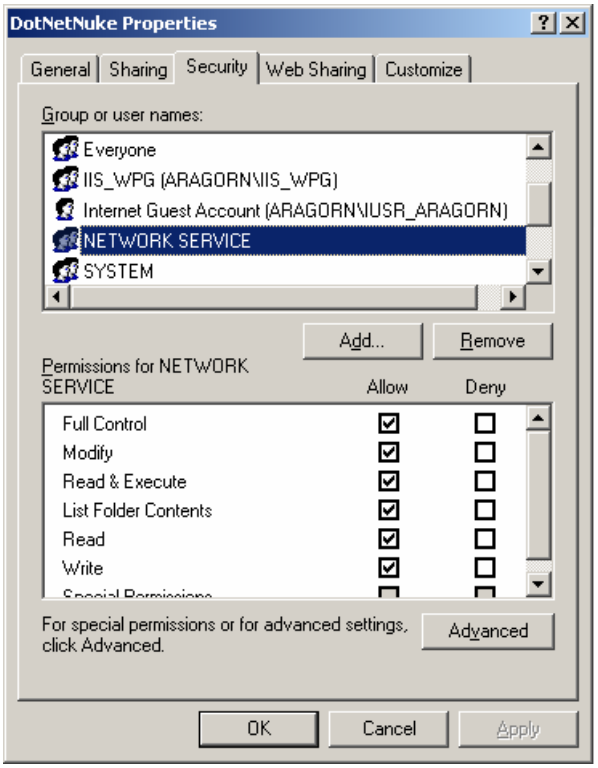


4. Set File Permissions

The next step in preparation for installing DotNetNuke is to set the NTFS file permissions for the Asp.NET worker process. This step is required as DotNetNuke needs to be able to create directories and files. The User Account that needs to be given permissions is different depending on which version of Windows you are using.

- In Windows 2000/IIS5 the account to use is the {Server}\ASPNET User Account,
- in Windows 2003/IIS6 the account is the NT AUTHORITY\NETWORK SERVICE User Account.

To set the correct File permissions use File Explorer to locate the physical folder you used in (1) above. Right-click on the folder, and select "Properties", and choose the Security Tab.



The required permissions for the various directories in DotNetNuke are as follows:

Root (and all child folders) – Read Access

Root – Write Access (and Modify Access if expecting to create child portals)
DesktopModules – Write, Modify Access (if installing 3rd party Private Assemblies)
Portals (and all child folders) – Write, Modify Access (to allow File Manager access and Skin/Container access)

If you are likely to be installing Language Packs you will also need Write and Modify Access to all child folders of the root.

Configuring web.config for Installation

All ASP.NET Applications require a web.config file. In versions prior to 3.1, this file was part of the zip package. In v3.1 and later versions this file is not part of the zip package. If you are installing v3.1 or later, before doing anything else rename **release.config** to **web.config**.

Before browsing to your site and initiating the installation, there are a number of **web.config** settings, that you will need to set. In the <**appSettings**> group of settings, there are five settings that control the Installation/Upgrade.

```
<appSettings>
   <add key="SiteSqlServer" value="Server=(local);Database=DotNetNuke;uid=;pwd=;" />
   <add key="MachineValidationKey" value="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902" />
   <add key="MachineDecryptionKey"
value="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902F8D923AC" />
   <add key="MachineValidationMode" value="SHA1" />
   <add key="InstallTemplate" value="DotNetNuke.install.resources" />
   <add key="AutoUpgrade" value="true" />
   <add key="UseDnnConfig" value="true"/>
   <add key="InstallMemberRole" value="true" />
      <!-- Show missing translation keys (for development use) -->
   <add key="ShowMissingKeys" value="false" />
   <add key="EnableWebFarmSupport" value="false" />
   <add key="EnableCachePersistence" value="false" />
</appSettings>
```

The first of these settings ("**SiteSqlServer**") defines the database you will be using (ie. **<add key="SiteSqlServer" value="Server=(local); Database=DotNetNuke; uid=;pwd=;" />**).

Here, replace the Server, Database, uid, and password by the relevant values for your database. In Shared Hosting environments, you will have been provided these values by your Hosting provider. On your local machine, you should not need to change the Server setting. However, in some situations Server might need to be set to the Machine Name of your local machine.

The second of these settings ("**InstallTemplate**") defines an install template. We will describe this template in detail later in this document. At this stage it is only important to realize that you can set this value to control what options are installed, but the default template will work for the vast majority of situations.

The third setting ("**AutoUpgrade**") determines whether the Upgrade/Install process runs automatically. By default this value is set to true. Whatever the setting, DotNetNuke will determine the current Database Version and Assembly Version and redirect to the Install page if they are different. If set to true, the Installation or Upgrade will proceed automatically. If set to false, the Install page will report the differences and stop.

The fourth setting ("**UseDnnConfig**") which is also set to true by default, tells DotNetNuke whether to read the current Database version from the dnn.config file in the /Install folder. As upgrades do not happen on every Application Start using this file can reduce the number of Database hits, and improve performance. At the end of Installation/Upgrade the current version is written to this file. If the Assembly Version is the same as the version in this file, nothing will happen. If it is different then AutoUpgrade will confirm that the Database Version is indeed different and proceed accordingly.

The fifth setting ("**InstallMemberRole**") optionally installs the MemberRole provider's scripts. As these scripts require a higher SQL Server permission setting, some hosting providers may not allow these scripts to be run. If your hosting provider does not provide these permissions you will need to ask them to install the scripts, before you install DotNetNuke. Also, if you are sharing a Database with other applications that use the MemberRole provider (for instance Community Server), then depending on whether the other application is already installed you may not want to re-run these scripts.

Data Provider Settings

There are two Data Access settings that it is important to review. Much further down the web.config file the data providers are defined.

```
<data defaultProvider="SqlDataProvider">
   <providers>
      <clear />
      <add name="SqlDataProvider"
         type="DotNetNuke.Data.SqlDataProvider, DotNetNuke.SqlDataProvider"
         connectionStringName="SiteSqlServer"
         upgradeConnectionString=""
         providerPath="~\Providers\DataProviders\SqlDataProvider\"
         objectQualifier=""
         databaseOwner="dbo" />
   </providers>
</data>
```

The **objectQualifier** attribute is used to control the prefix applied to every object in the SQL Server database. By default this is set to an empty string "", so for instance the "Portals" table is created as "Portals". If, for example, the **objectQualifier** was set to "dnn_" then the "Portals" table would be created as "dnn_Portals" and stored procedures etc that referenced the Portals table wold reference it as dnn_Portals. Setting this value is recommended, as it allows DotNetNuke to exist within an environment where multiple applications might need to use the same database.

The **databaseOwner** attribute is used to identify the user that "owns" the database objects. By default it is set to "**dbo**" and in most cases this attribute should be left this way. However, some SQL Server DBAs (in particular in Shared Hosting Environments) may not grant the user you identify in the connection string db_owner privileges. In this case, set the **databaseOwner** attribute to your login user. Note this user will need to have the following SQL Permissions, at a minimum:

db_datareader (necessary for dotnetnuke)
db_datawriter (necessary for dotnetnuke)
db_ddladmin (neccessry for MemberRoles)
db_securityadmin (necessary during installation of MemberRoles)

The previous two settings are only relevant for the default MS SQL Server Data Provider. If you are using an alternate data provider, you will have to refer to its documentation to configure the provider properly.

Navigation Provider Settings

In DotNetNuke 3.2 the main navigation system was abstracted into Navigation Providers.

```
      <navigationControl defaultProvider="SolpartMenuNavigationProvider">
```

```
         <providers>
            <clear/>
            <add name="SolpartMenuNavigationProvider"
type="DotNetNuke.NavigationControl.SolpartMenuNavigationProvider,
DotNetNuke.SolpartMenuNavigationProvider"
providerPath="~\Providers\NavigationProviders\SolpartMenuNavigationProvider\"/>
            <add name="DNNMenuNavigationProvider"
type="DotNetNuke.NavigationControl.DNNMenuNavigationProvider,
DotNetNuke.DNNMenuNavigationProvider"
providerPath="~\Providers\NavigationProviders\DNNMenuNavigationProvider\"/>
            <add name="DNNTreeNavigationProvider"
type="DotNetNuke.NavigationControl.DNNTreeNavigationProvider,
DotNetNuke.DNNTreeNavigationProvider"
providerPath="~\Providers\NavigationProviders\DNNTreeNavigationProvider\"/>
            <add name="DNNDropDownNavigationProvider"
type="DotNetNuke.NavigationControl.DNNDropDownNavigationProvider,
DotNetNuke.DNNDropDownNavigationProvider"
providerPath="~\Providers\NavigationProviders\DNNDropDownNavigationProvider\"/>
         </providers>
      </navigationControl>
```

By default the defaultProvider attribute is set to the SolpartMenuNavigationProvider, but if you would like to use any of the other providers, then you can change the defaultProvider attribute to be the same as the name attribute of one of the other providers.

Member Roles Prototype Settings

DotNetNuke 3.0 includes a prototype of the member/roles provider that will be included in the upcoming release of ASP.NET 2.0 (codenamed “Whidbey”). There are a number of settings that can be configured for this provider in the web.config file. However, DotNetNuke only really supports modifying 4 of the parameters.

```
   <memberrolesprototype>
      <membership userIsOnlineTimeWindow="15">
         <providers>
            <add name="DNNSQLMembershipProvider"
               type="DotNetNuke.Security.Membership.DNNSQLMembershipProvider,
DNNSQLMembershipProvider"
               connectionStringName="SiteSqlServer"
               enablePasswordRetrieval="true"
               enablePasswordReset="true"
               requiresQuestionAndAnswer="false"
               minRequiredPasswordLength="4"
               minRequiredNonalphanumericCharacters="0"
               requiresUniqueEmail="false"
               passwordFormat="Encrypted"
               applicationName="/"
               description="Stores and retrieves membership data from the local Microsoft
SQL Server database" />
         </providers>
      </membership>
      <roleManager
```

```
         cacheRolesInCookie="true"
         cookieName=".ASPXROLES"
         cookieTimeout="30"
         cookiePath="/"
         cookieRequireSSL="false"
         cookieSlidingExpiration="true"
         createPersistentCookie="false"
         cookieProtection="All">
         <providers>
            <add name="DNNSQLRoleProvider"
               type="DotNetNuke.Security.Role.DNNSQLRoleProvider, DNNSQLRoleProvider"
               connectionStringName="SiteSqlServer"
               applicationName="/"
               description="Stores and retrieves roles data from the local Microsoft SQL
Server database" />
         </providers>
      </roleManager>
      <profile enabled="true">
         <providers>
            <add name="AspNetSqlProvider"
               type="DotNetNuke.Users.Profile.DNNSQLProfileProvider,
DNNSQLProfileProvider"
               connectionStringName="SiteSqlServer"
               applicationName="/"
               description="Stores and retrieves profile data from the local Microsoft
SQL Server database" />
         </providers>
         <properties>
            <add name="FirstName" type="string" allowAnonymous="true" />
            <add name="LastName" type="string" allowAnonymous="true" />
            <add name="Unit" type="string" allowAnonymous="true" />
            <add name="Street" type="string" allowAnonymous="true" />
            <add name="City" type="string" allowAnonymous="true" />
            <add name="Region" type="string" allowAnonymous="true" />
            <add name="PostalCode" type="string" allowAnonymous="true" />
            <add name="Country" type="string" allowAnonymous="true" />
            <add name="Telephone" type="string" allowAnonymous="true" />
            <add name="Fax" type="string" allowAnonymous="true" />
            <add name="Cell" type="string" allowAnonymous="true" />
            <add name="Website" type="string" allowAnonymous="true" />
            <add name="IM" type="string" allowAnonymous="true" />
            <add name="TimeZone" type="integer" allowAnonymous="true" />
            <add name="PreferredLocale" type="string" allowAnonymous="true" />
         </properties>
      </profile>
      <anonymousIdentification
         enabled="true"
         cookieName=".ASPXANONYMOUS"
         cookieTimeout="100000"
         cookiePath="/"
         cookieRequireSSL="false"
         cookieSlidingExpiration="true"
         cookieProtection="None"
         domain="" />
   </memberrolesprototype>
```

minRequiredPasswordLength – controls the minimum password length. This has been set to 4 in the default web.config, in order to support the historic use of host/host for the username and password of the default superuser account. Increasing the default

will require that the passwords provided in the <**superuser**> and <**administrator**> nodes of the install template be modified.

minRequiredNonalphanumericCharacters – controls the minimum number of non-alpha-numeric characters required in the password. This has been set to 0 as historically DotNetNuke has not required non-alphanumeric characters. Again, setting this value to any positive number will require the <**superuser**> and <**administrator**> nodes of the install template be modified.

cookieName – both the rolemanager and anonymouIdentification section of the provider allow you to set the cookie name. This is probably not necessary, but is available.

Running v3.2 on .NET 2 under Medium Trust

DotNetNuke 3.2, while compiled against ASP.NET 1.1 is compatible with .NET 2.0. For the most part if you running v3.2 on .NET 2, you do not have to do anything different, that is, unless you or your hosting provider has configured the site for Medium Trust.

If you are running DotNetNuke 3.2 on .NET 2.0 under Medium Trust, you need to add an attribute setting to all the config sections of your web.config, as shown below. This attribute is not needed (in fact it will cause an error) if you are running on .NET 1.1.

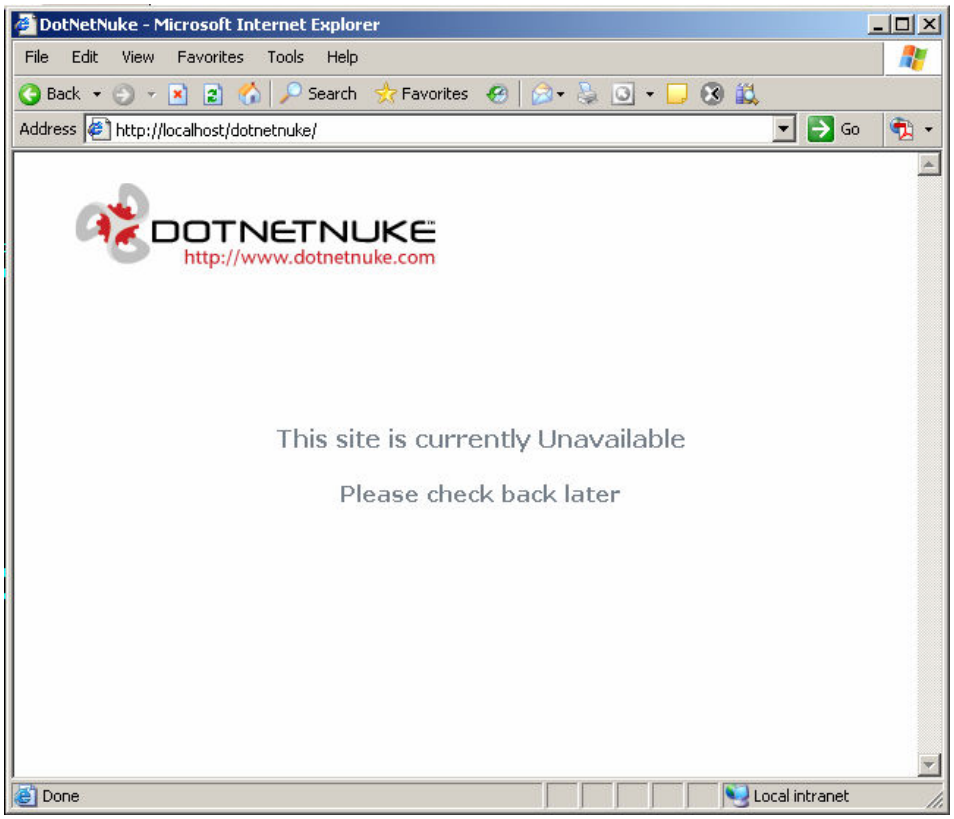
```
   <configSections>
      <sectionGroup name="dotnetnuke">
         <section name="data" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="logging" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="scheduling" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="htmlEditor" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="navigationControl" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="searchIndex" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="searchDataStore" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="friendlyUrl" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="caching" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
         <section name="authentication" requirePermission="false"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"/>
      </sectionGroup>
      <sectionGroup name="memberrolesprototype">
         <section name="membership" requirePermission="false"
type="Microsoft.ScalableHosting.Configuration.MembershipConfigHandler, MemberRole,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=b7c773fb104e7562" />
```

```
        <section name="roleManager" requirePermission="false"
type="Microsoft.ScalableHosting.Configuration.RolesConfigHandler, MemberRole,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=b7c773fb104e7562" />
        <section name="profile" requirePermission="false"
type="Microsoft.ScalableHosting.Configuration.ProfileConfigHandler, MemberRole,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=b7c773fb104e7562" />
        <section name="anonymousIdentification" requirePermission="false"
type="Microsoft.ScalableHosting.Configuration.AnonymousIdConfigHandler, MemberRole,
Version=1.0.0.0, Culture=neutral, PublicKeyToken=b7c773fb104e7562" />
      </sectionGroup>
   </configSections>
```

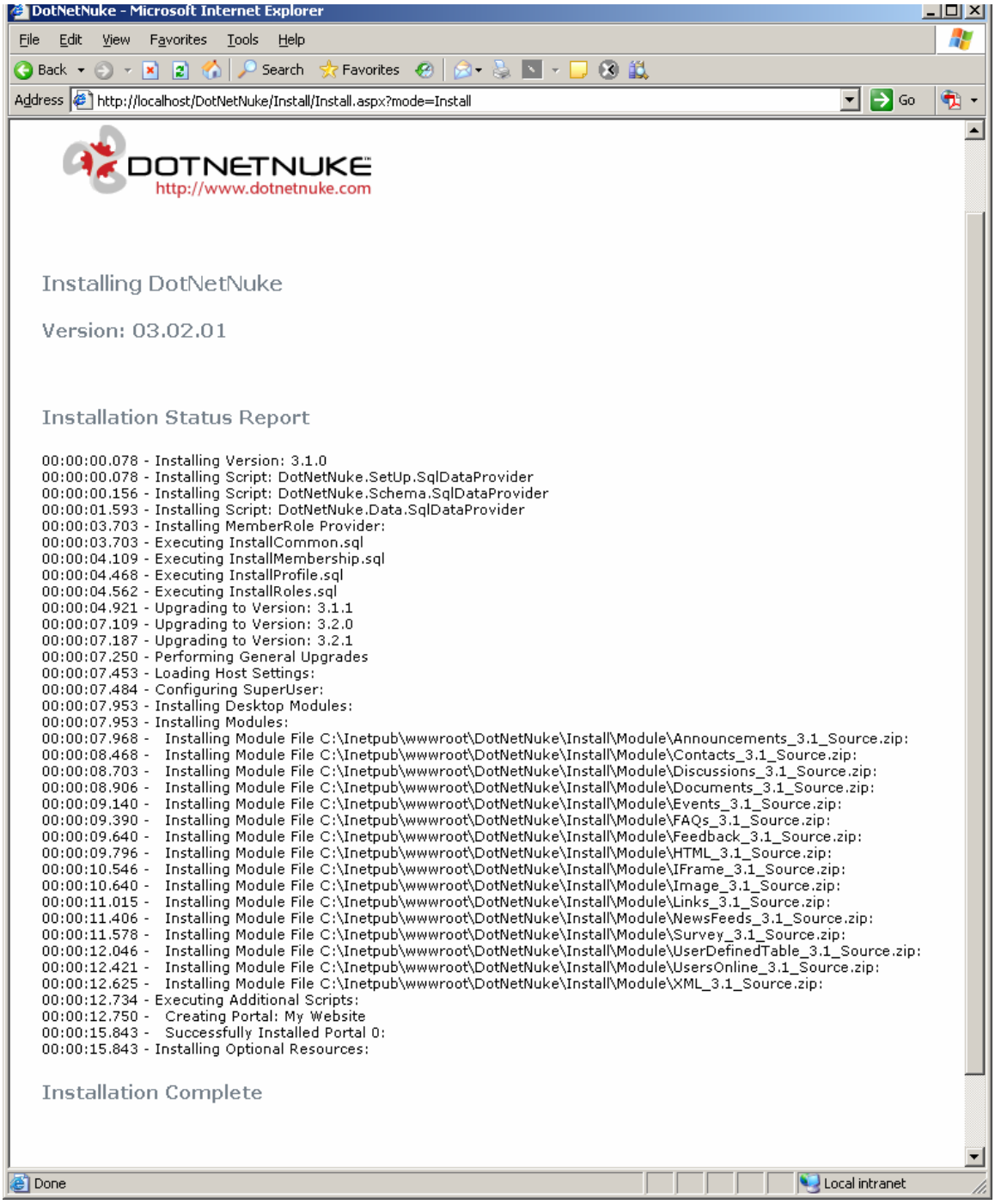
Installing DotNetNuke

So we have configured our web.config file and our install template and we are ready to install DotNetNuke 3.x. The installation is quite straight forward. Browse to the root folder – in a local environment this will probably be http://localhost/DotNetNuke, while on a remote installation it will be www.mydomain.com.

The AutoUpgrade function will detect that the Database is empty. If the "**AutoUpgrade**" setting is set to false then an "Under Construction" will indicate that the site is not available. The administrator can then trigger the Install by using the url – "**Install/Install.aspx?mode =Install**".



If '**AutoUpgrade**" is set to true then the Installation process starts automatically. No matter which way the installation is triggered, feedback similar to the following will be displayed.



Lets have a look at what happens in this default installation.

- The Installer reports the version of the template being installed

- The Installer reports that the installation scripts were executed. (any incremental scripts for later versions would then be executed)
- The Installer reports that the scripts required for the MemberRole Provider were executed, as well as an extra script required to fix the default names of some of the constraints
- The sections of the Install template are then parsed
  - Host Settings
  - SuperUser
- The modules are then Installed
- The portal(s) are created.
- Optional Resources are then Installed, for example any additional skins, containers etc.

Upgrade to DotNetNuke 3.x

Backup your site

Before upgrading, if you can, backup your entire site. At a minimum, backup your web.config.

Preparing for Upgrade

Once you have backed up your web.config file, you can extract the new version of DotNetNuke over your existing version. Any files in your current installation that are no longer required, should be removed by the upgrade process.

Please refer to the previous chapter for a review of the packages that are available.

Configuring web.config for Upgrade

As described in the section on Installing DotNetNuke v3.x, the zip package no longer includes a copy of **web.config**. The reason for this will become clear when you read this section on upgrading an existing DotNetNuke site (especially version 3.0.13 or later).

As described above DotNetNuke version 3 added the memberRole prototype. Three keys in the AppSettings control the password encryption for this component.

```
    <add key="MachineValidationKey" value="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902" />
    <add key="MachineDecryptionKey"
value="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902F8D923AC" />
    <add key="MachineValidationMode" value="SHA1" />
```

These keys may look innocuous enough, but if your **MachineValidationKey** and **MachineDecryptionKey** values are not the same as those above then you must be very careful, when setting up the web.**config** file for upgrade.

In order to successfully upgrade you must follow the procedure outlined below

- Make a back-up copy of your existing web.config file (eg web.backup.resources). We cannot emphasise enough how important it is that you have a backup. If you lose the original value of the MachineKeys, then none of your users will be able to log in.
- Once you are sure that you have safely backed up your web.config file, rename release.config to web.config.
- Replace the following keys in your new web.config, with the values in your backup.
  SiteSqlServer
  MachineValidationKey
  MachineDecryptionKey
  InstallationDate (this may not be present in web.config, so add the key that is in web.backup.resources to web.config)
- Make any other changes to web.config that you made to support additional providers etc.

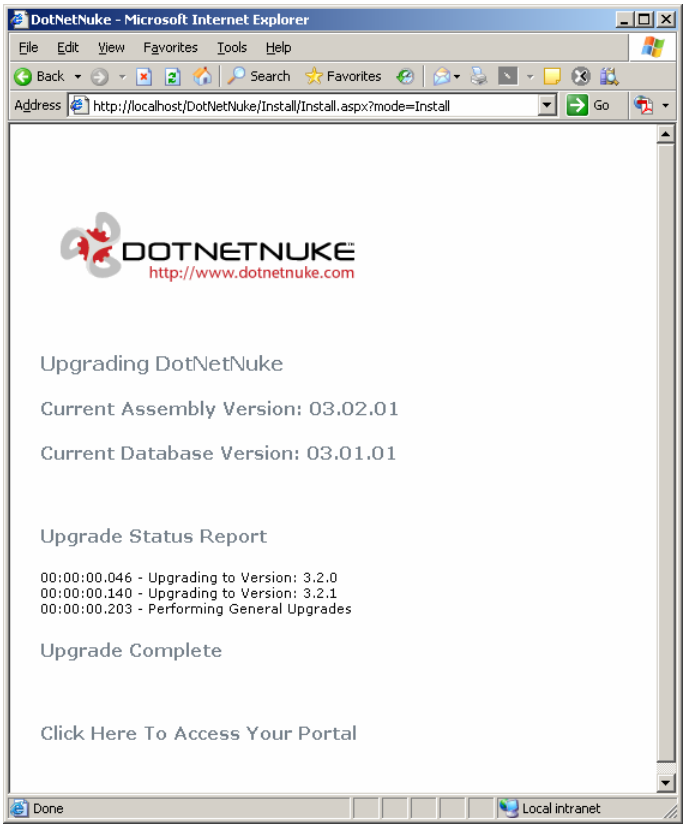
You can, of course, update your previous version of web.config (after backing it up), by moving any new configuration sections from the release.config to your web.config. Different versions of DotNetNuke introduced new provider sections and depending on which version you are upgrading from you will need to copy the following sections from release.config to web.config.

	Version Introduced	Add if upgrading from Version (or earlier)
Navigation Provider Section	3.2.0	3.1.1
Caching Provider Section	3.1.0	3.0.13
DBLoggingProvider	3.1.0	3.0.13

Upgrading DotNetNuke

You should now be able to browse to the site and trigger the Upgrade process. **Do not browse to your site unless you are sure that the MachineKeys in your new web.config are the same as they were in the old file that you backed-up.**

When you do browse to your site, you will get an upgrade report similar to the one shown below.



Lets have a look at what happens in this upgrade.

- The Installer reports the new version of DotNetNuke being installed (in this case 3.2.1)
- The Installer reports the current version of the installation (in this case 3.1.1)
- The Installer reports that a number of incremental scripts were executed.
- The Installer reports that General Upgrades were performed

- Although not shown in this example, if any optional resources (Modules, Skins Containers) were included in the Install folder they would then have been installed at the end of the upgrade.

Installation of DotNetNuke 4.x

Introduction

DotNetNuke 4.0 is the first version of DotNetNuke to be designed for ASP.NET 2.0. Version 3.2 will run on ASP.NET 2, but requires Visual Studio 2003 and .NET 1.1 for any development work.

As has been discussed in many of the ASP.NET related forums, .NET 2 (and Visual Studio 2005) introduced a different paradigm for developing and running websites. The main issue that we had to address is that there no longer is a web-project in .NET2/Visual Studio 2005. Instead, .NET 2 treats all files that are contained under a website's Virtual Directory as part of the website.

In addition Visual Studio 2005 (VS2005) provided a much enhanced Project and Item Template system, which is also supported in the Express line of products, in particular, Visual Web Developer Express (VWD). This feature has allowed us to develop a DotNetNuke Starter Kit, which simplifies the creation of DotNetNuke sites.

This chapter describes the differences in the procedures used for this version. If you need more information on a topic not covered then please review the Chapter on "Installation of DotNetNuke v3.x".

Which package should I use?

DotNetNuke v4.x comes in three separate packages

- DotNetNuke_X.Y.Z_Install.zip - An Install Package – This package contains the ascx, aspx files as well as other content files and compiled assemblies (dlls) but does not contain any of the Visual Basic code files for the providers and httpModules.

- DotNetNuke_X.Y.Z_Source.zip – A Source Package – This package contains all the files related to the core DotNetNuke project including all the Visual Basic source files.
- DotNetNuke_X.Y.Z_StarterKit.vsi – A Visual Studio 2005 Installer package – This package contains a Project Template (based on the Install package) a Module Item Template (in both Visual Basic.NET and C#), and a Skin Item Template. The supporting class libraries, providers and HttpModules are provided as pre-compiled assemblies (dlls).

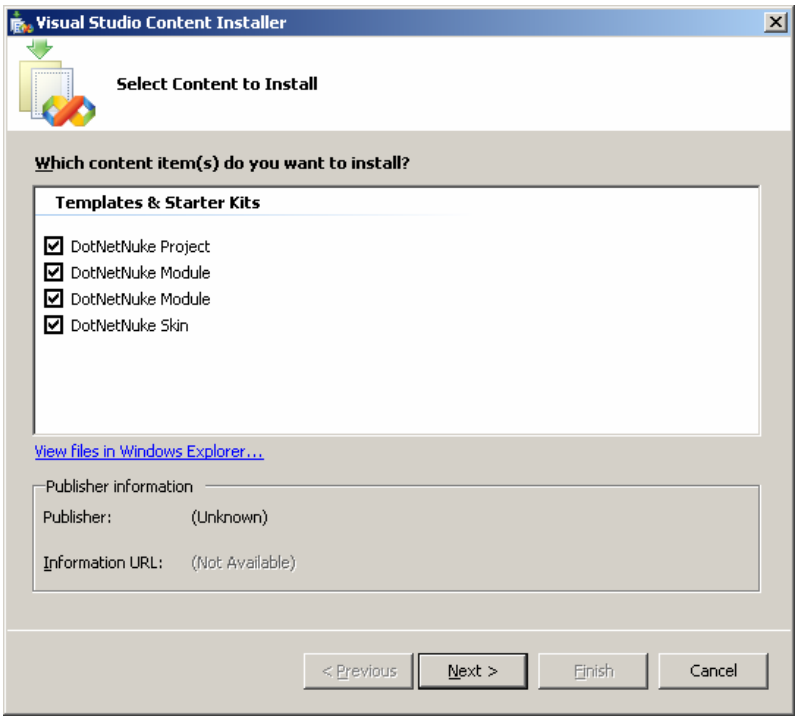
In deciding which package to use, you also need to determine what your development environment will be. This is because Visual Web Developer Express (VWD) does not support any project type other than web-sites. Thus if you expect to do most of your development in VWD rather Visual Studio 2005 (VS2005), then you will not be able to use the Source package.

Using the Starter Kit

1. Installing the Starter Kit

If you decide to use the Starter Kit, you will need to first install it. The package will only install if you have already installed VS2005 or VWD on your local machine.

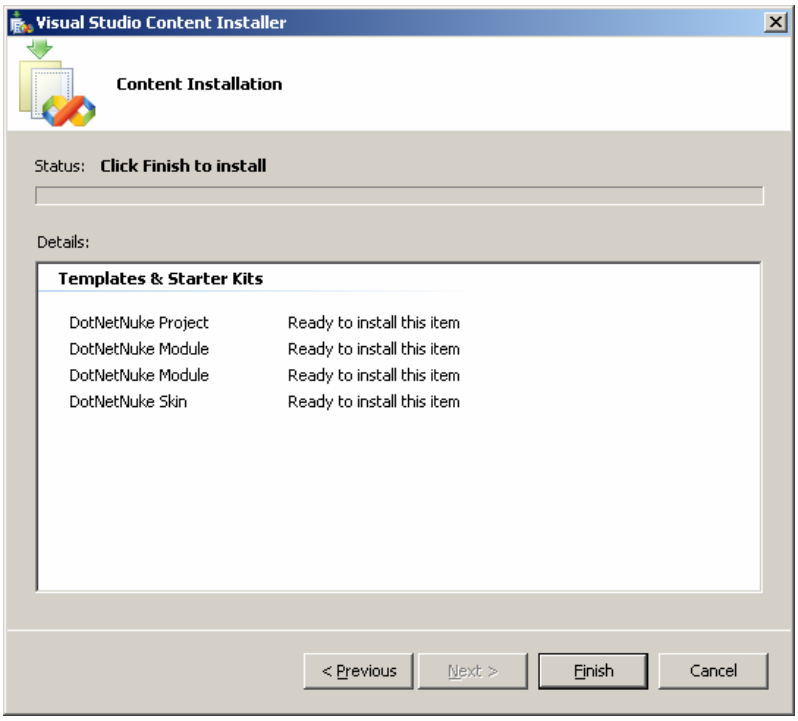
To install the Starter Kit, double-click the file in Windows File Explorer. The Visual Studio Content Installer will launch and will look something like this.



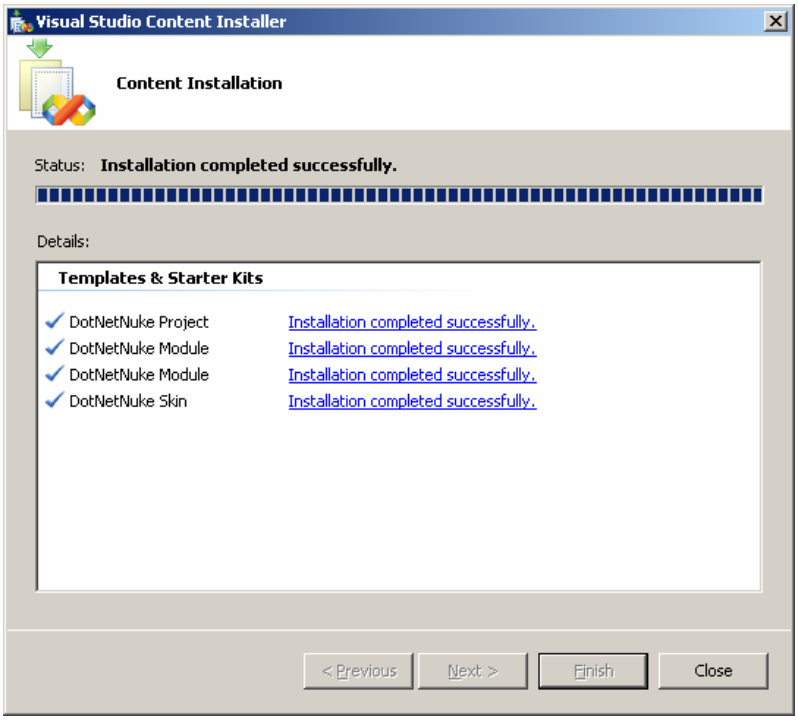
Select the templates you would like to install – by default they are all selected - and click Next – there are two DotNetNuke Module templates (one for VB and one for C#).

As the packages have not been signed with a Digital Certificate, you will then get the following warning.

Select Yes and proceed to the next screen

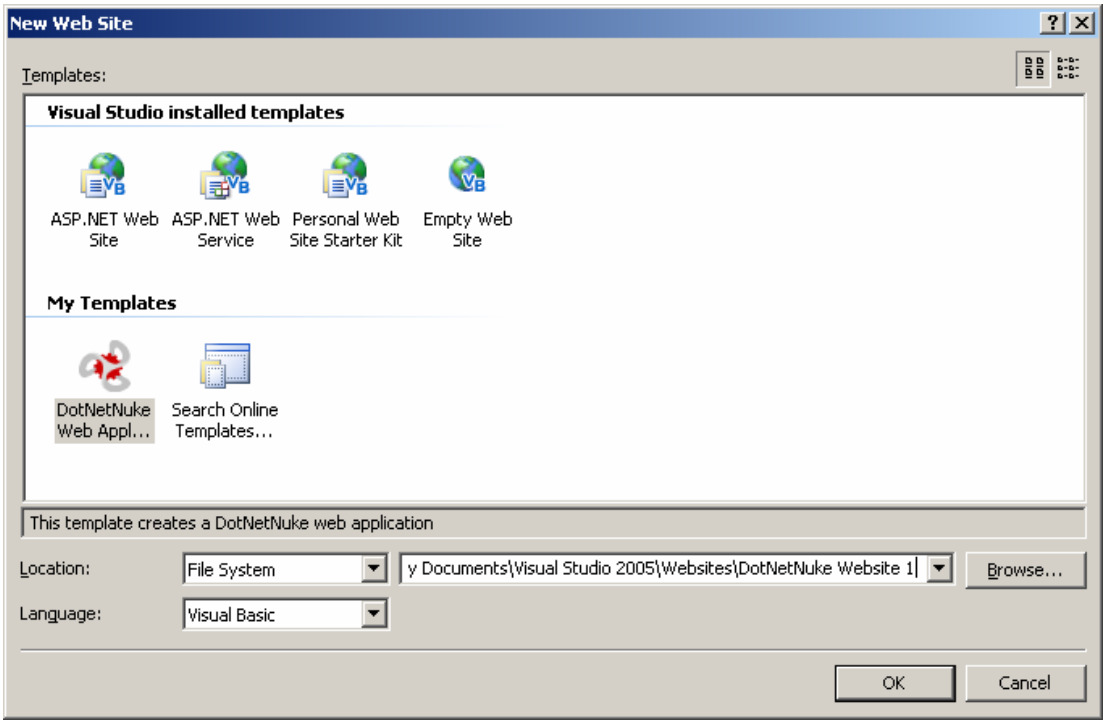


Finally click Finish to install the Templates.



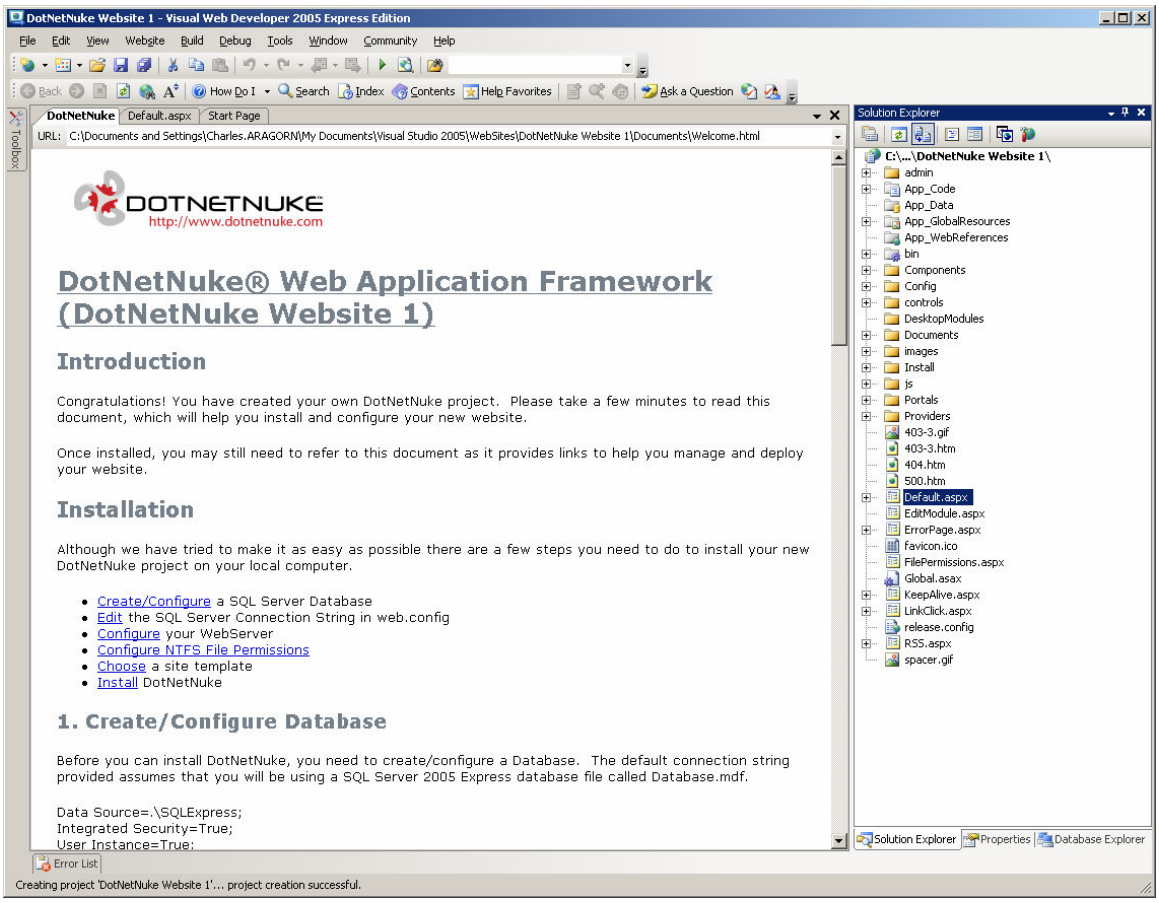
2. Creating a DotNetNuke Site from a Project Template.

Once you have installed the template you can now create a new DotNetNuke project. In VS2005 or VWD select New Website … from the File menu.



Select the DotNetNuke template, choose the Location i.e. File System or HTTP and enter a name in the text box. In the example above I have selected the File System option, which will use VS2005/VWD's built in File System based Web Server, and named the site – "DotNetNuke Website 1".

When you select OK a website will be created in the folder you selected, and the "welcome.html" File will automatically open in the Internal Browser.



This “Welcome.html” provides instructions on how to complete the Installation on your local machine.

Using the Install or Source Packages

1. Unzip the Package

As with Installing DotNetNuke 3.x, if you decide to use either of the zip packages (Install.zip or Source.zip) you will need to unzip them to a location on your computer where the Application will reside.

However, as we have reorganized the Folder structure for all the Class Library support projects the procedure is a little different, if you decide to use the Source Package.

If you expect to only use the Install package then follow the procedure outlined for extraction of a version 3.x package.

Even if for now you are using the Install Package, if you expect to use the Source Package in the future then use the following procedure.

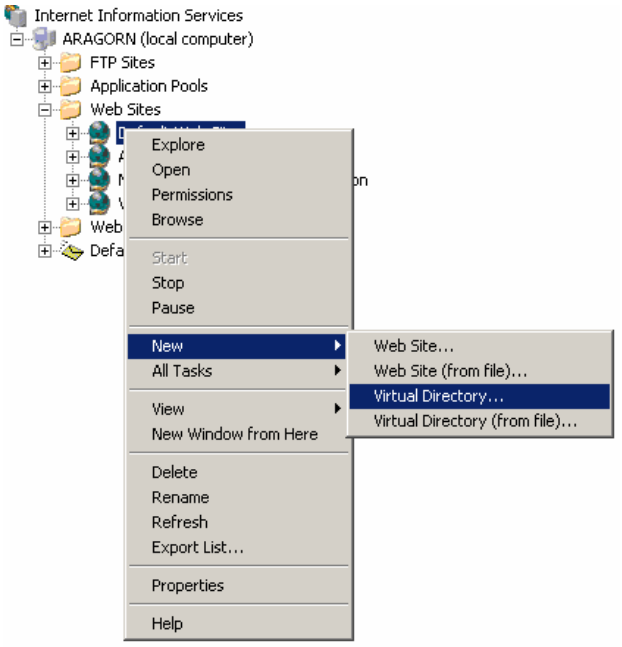
- Create a directory where you would like the Application to reside (eg C:\DotNetNuke_2)
- If you are using the Install Package,
    i. create a sub-directory called Website (eg C:\DotNetNuke_2\Website)
    ii. extract the Install zip package to the sub-directory you just created

  If you are using the Source Package extract the Source zip to the first folder you created (eg C:\DotNetNuke_2), the sub folders will automatically be created.

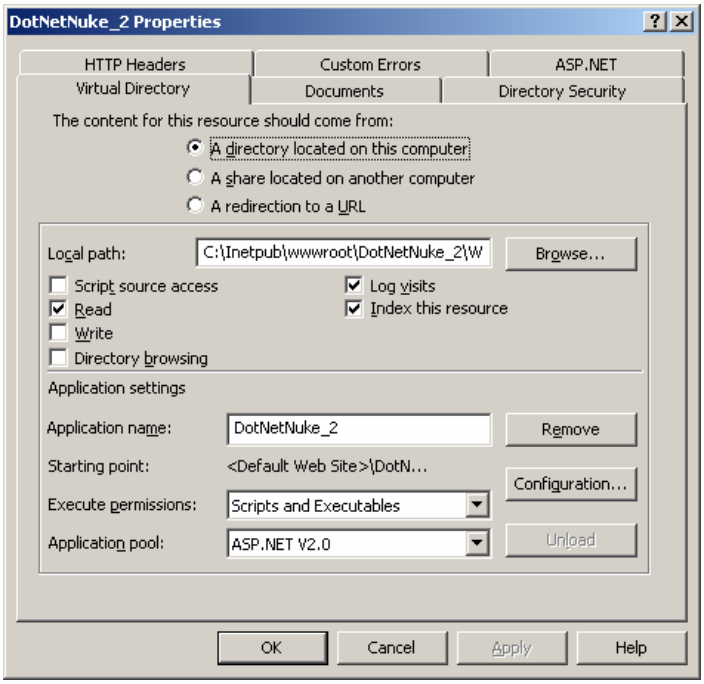
2. Configuring IIS (Internet Information Server)

In a local intranet configuration, create a Virtual Directory in IIS called DotNetNuke_2 that points to the appropriate physical directory.

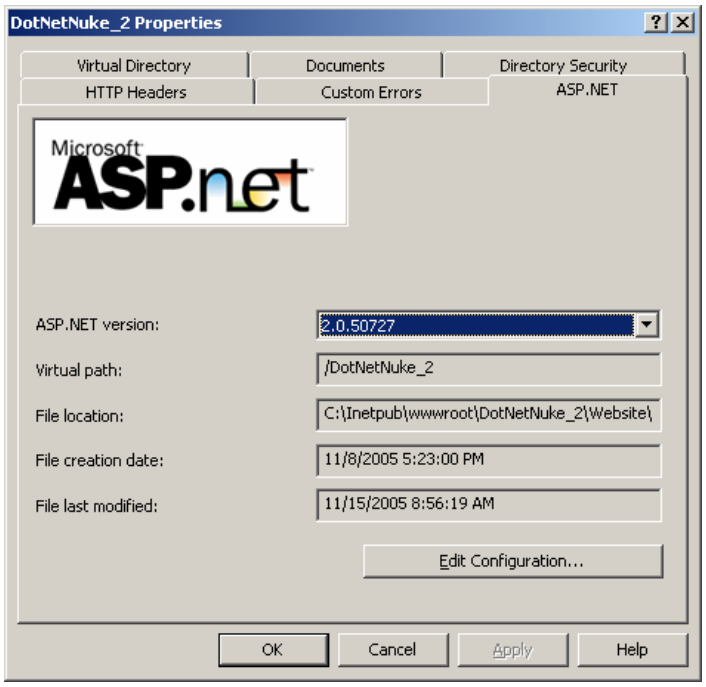
The appropriate directory will depend on the process you followed above. If you are using the Install package then point IIS to the location where you extracted the Install Package. If you are using the Source package then point IIS to the location of the \Website subdirectory.



This will launch a wizard that will allow you to configure your Virtual Directory. Once you have created the Virtual Directory using the Wizard you can edit the properties of the new Virtual Directory, by right-clicking on the new Virtual Directory, and selecting Properties.



For DotNetNuke 4.x you must make sure that the Virtual Directory is configured for .NET2.



Configuring web.config for Installation

While the web.config file for DotNetNuke v4.x differs quite a bit from that used in DotNetNuke 3.x, most of the differences are due to .NET2 and not to DotNetNuke v4.x.

We will therefore only focus here on differences between v3.x and v4.x. Please refer to the Chapter on Installing DotNetNuke v3.x for more information on DotNetNuke specific web.config settings.

As with v3.x, before doing anything else rename **release.config** to **web.config** (if you are required to run under Medium Trust then rather than use **release.config**, rename **development.config**).

There are three major differences between the 3.x web.config and the 4.x web.config:

1. In addition to the connection String setting under <**appSettings**> required in 3.x, 4.x also requires the same connection String to be set under the <**connectionStrings**> section

```
  <connectionStrings>
    <!-- Connection String for SQL Server 2005 Express -->
    <add
      name="SiteSqlServer"
      connectionString="Data Source=.\SQLExpress;Integrated Security=True;User
Instance=True;AttachDBFilename=|DataDirectory|Database.mdf;"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

   Note that the name of the connectionString "SiteSqlServer" is the same name as the key used in < **appSettings** >, and that the < **appSettings** > setting is also required to support legacy modules.

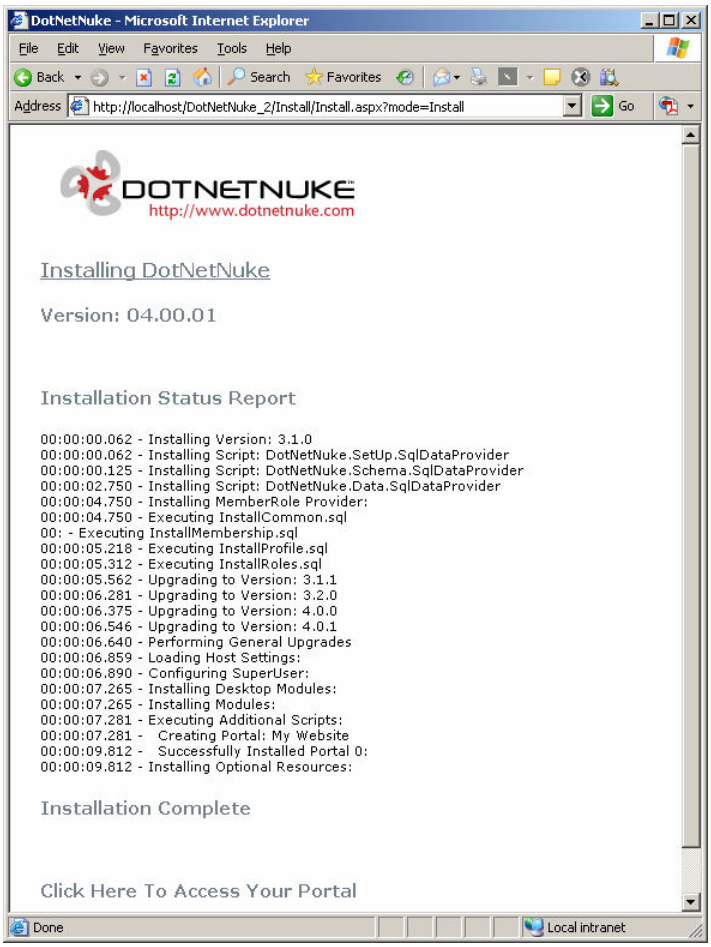
2. The machine key settings which were in the < **appSettings** > settings – MachineValidationKey, MachineDecryptionKey and MachineValidationMode – are now in the <**machineKey**> setting in <**system.web**>

```
    <machineKey
      validationKey="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902"
      decryptionKey="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902F8D923AC"
      decryption="3DES"
      validation="SHA1"/>
```

3. The settings for the Membership elements have been moved from the <memberrolesprototype> node to the <system.web> node, although there are no changes in the settings themselves. If you wish to modify the settings please read the more detailed section under Installation of DotNetNuke 3.x.

Installing DotNetNuke

So we have configured our web.config file and our install template and we are ready to install DotNetNuke 4.x. The installation process for version 4.x is exactly the same as for version 3.x. Browse, to the Virtual Directory you used to configure the site, and the installation should proceed, as below. (Note: in the example below there were no module zip files in the Install/Module folder, so no modules were installed automatically.



Upgrade to DotNetNuke 4.x

Backup your site

Before upgrading, if you can, backup your entire site. At a minimum, backup your web.config.

Preparing for Upgrade

If you have read the previous chapter you will know that DotNetNuke v4.0 introduced a new folder structure, when working with the Source version. This is due to the fact that the 30 Class Library projects can not reside in the main Website folder or any of its children.

This introduces a level of complication for anyone who wishes to upgrade from a v3.x Source development setup to a v4.x Source development setup.

If you are upgrading a v3.x Source development setup to v4.x Install only or you are upgrading a v3.x Install setup to v4.x Install only then you can follow the normal upgrade process.

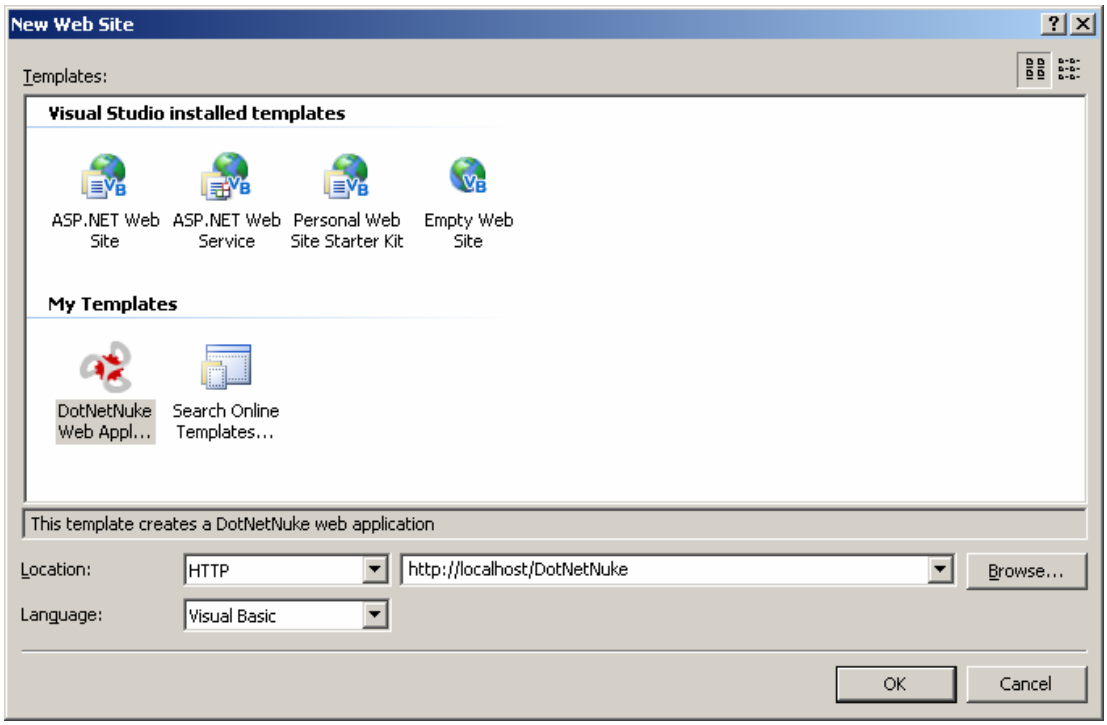
Upgrading v3.x to v4.x Install only

If you are not worried about the source of the Class Library projects you can upgrade to the v4.0 Install package.

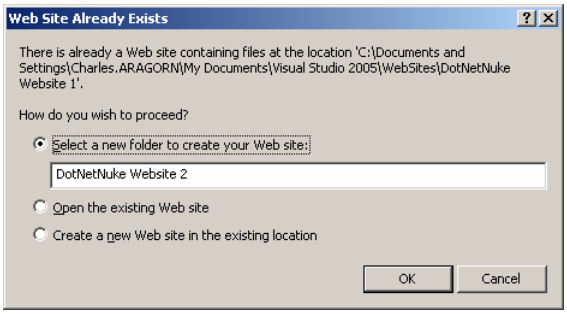
Once you have backed up your web.config file, there are two ways you can use to extract the “Install” version over your existing v3.x site.

1. Extract the Install.zip package over your existing version.

2. In VS2005 or VWD select New Website ... and follow the procedure described in the previous chapter, except that you will need to select the location of your existing Website;



As the site already exists the New Web Site Wizard will warn you that a site already exists in that location and provide you with 3 options.



In this case, select the last option, to create a new Website in the existing location.

When you have finished this you are ready to configure your web.config file for v4.x.

Upgrading v3.x to v4.0 Source

The situation is a little trickier if you want to upgrade to the Source version of DotNetNuke v4.0.

It is therefore recommended, that rather than overwriting in some way your old "site", that you follow the instructions in the previous chapter to create a new DotNetNuke v4.x site, and move any custom content you have on your old site, such as Modules, Skins and Portal content, as well as your web.config file, to the new site.

Then in IIS re-point the Virtual Directory to the Website folder of the DotNetNuke v4.x site.

This is the safest and simplest way to move your development from v3.x to v4.x, and once you have completed this you are ready to configure your web.config file.

Configuring web.config for Upgrade to v4.x

The easiest way to upgrade your web.config file is to copy the settings you have modified to the release.config file – by following the procedure below:

- Make a back-up copy of your existing web.config file (eg web.backup.resources). We cannot emphasise enough how important it is that you have a backup. If you lose the original value of the MachineKeys, then none of your users will be able to log in.
- Once you are sure that you have safely backed up your web.config file, rename release.config to web.config.
- Replace the following keys in your new web.config, with the values in your backup.
  SiteSqlServer
  InstallationDate (this will not be present in release.config, so add the key that is in web.backup.resources to your new web.config)
- Set the <connectionStrings> to the same values as you just copied from your old web.config

```
<add
  name="SiteSqlServer"
  connectionString="???????"
  providerName="System.Data.SqlClient" />
```

- Set the values of the machineKey element to the values that used to be in the appSettings node as follows.

```
<machineKey
  validationKey="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902"
  decryptionKey="F9D1A2D3E1D3E2F7B3D9F90FF3965ABDAC304902F8D923AC"
  decryption="3DES"
  validation="SHA1"/>
```

where the "validationKey attribute should have the value that was in the MachineValidationKey setting and the decryptionKey attribute should have the value that was in the MachineDecryptionKey setting.

- Make any other changes to web.config that you made to support additional providers etc.

Upgrading DotNetNuke

You should now be able to browse to the site and trigger the Upgrade process. **Do not browse to your site unless you are sure that the MachineKeys in your new web.config are the same as they were in the old file that you backed-up.**

When you do browse to your site, you will get an upgrade report similar to that shown in the Chapter on Upgrading to v3.x.

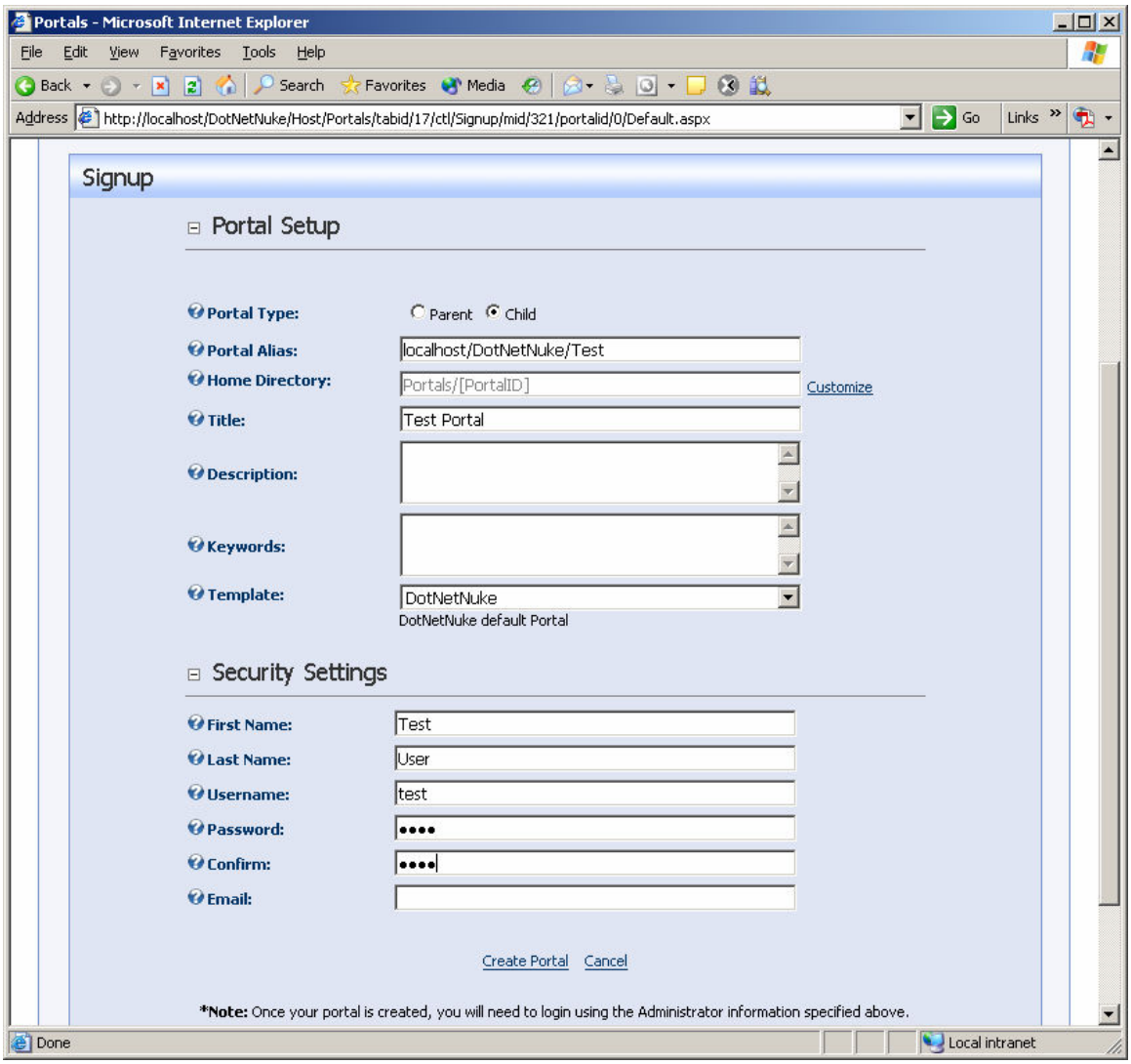
Installation of Optional Resources and Modules

Optional Resources such as language packs, skins and Private Assemblies can also be installed automatically during the Installation Process. The /Install folder has six sub-folders, Container, Language, Module, Portal, Skin and Template. The zip package – described later in this document – for the appropriate resource is placed in the appropriate folder, and installed automatically by the Installer.

From version 3.1 forward, the core Desktop Modules are all located by default in the /Install/Module folder so that they all install automatically

Installing Additional Portals

There are two ways to add additional portals once DotNetNuke has been installed. The first, most obvious method is to login as Host, select the Portals option from the Host menu, and enter the relevant information for the Portal.



There is, however, a second method for installing portals, that can batch install multiple portals, in one operation.

This second approach is modeled after the installation of the portals during the DotNetNuke installation process. A template file "**portals.resources**" can be placed in the /Install/Portal folder. This file is similar to the install template, except that the only child nodes processed are the <portals><portal> nodes.

Example Portals.resources File

```
<dotnetnuke>
```

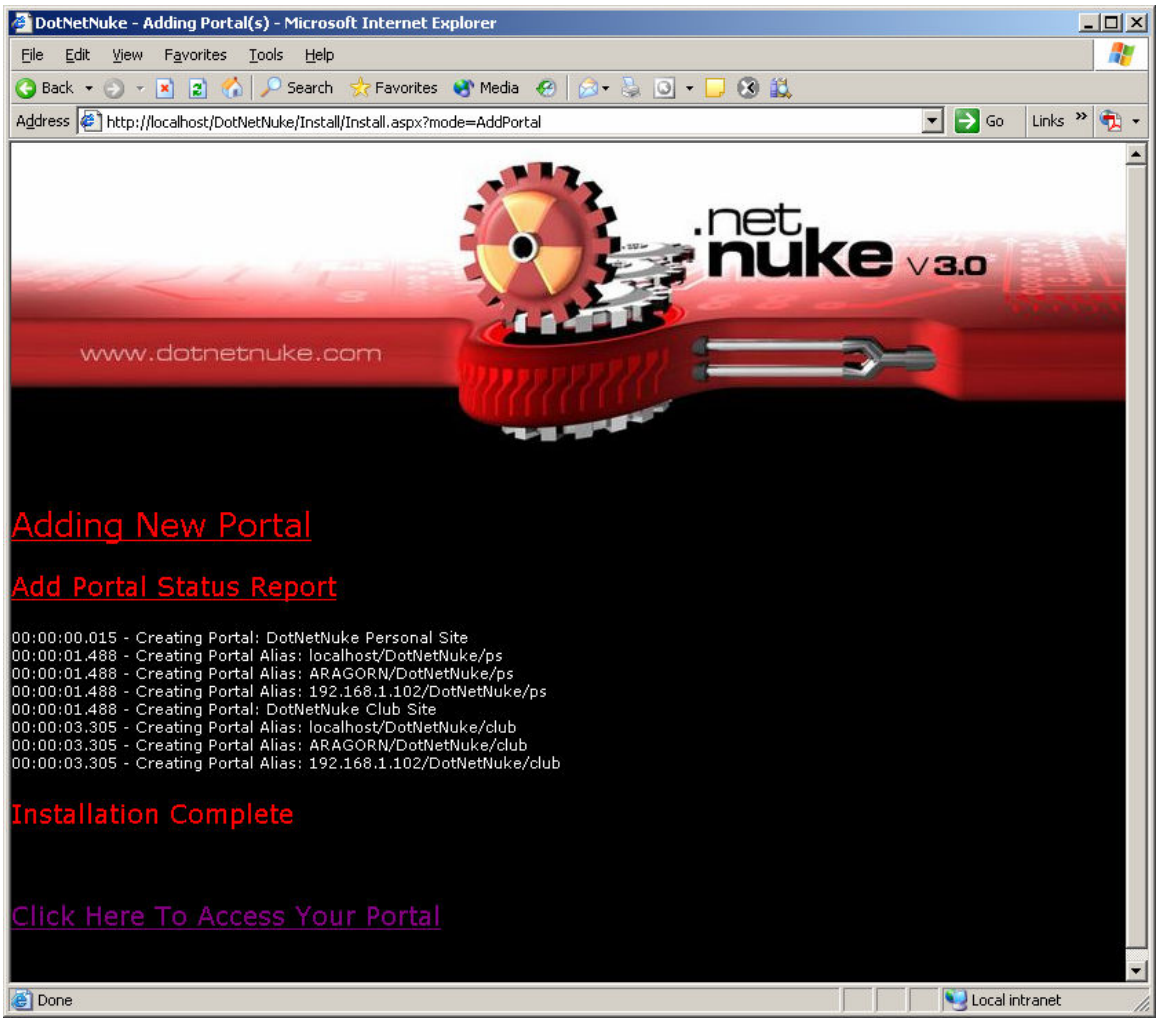
```
   <portals>
      <portal>
         <portalname>DotNetNuke Personal Site</portalname>
         <administrator>
            <firstname>Administrator</firstname>
            <lastname>Account</lastname>
            <username>admin</username>
            <password>admin</password>
            <email></email>
         </administrator>
         <description>Default Personal Site</description>
         <keywords>Default, DotNetNuke, CMS, Web, Future</keywords>
         <templatefile>Personal Site.template</templatefile>
         <portalaliases>
            <portalalias>localhost/DotNetNuke/ps</portalalias>
            <portalalias>ARAGORN/DotNetNuke/ps</portalalias>
            <portalalias>192.168.1.102/DotNetNuke/ps</portalalias>
         </portalaliases>
         <ischild>true</ischild>
      </portal>
      <portal>
         <portalname>DotNetNuke Club Site</portalname>
         <administrator>
            <firstname>Administrator</firstname>
            <lastname>Account</lastname>
            <username>admin</username>
            <password>admin</password>
            <email></email>
         </administrator>
         <description>Default DotNetnuke Club Site</description>
         <keywords>Default, DotNetNuke, CMS, Web, Future</keywords>
         <templatefile>Club or Organization Site.template</templatefile>
         <portalaliases>
            <portalalias>localhost/DotNetNuke/club</portalalias>
            <portalalias>ARAGORN/DotNetNuke/club</portalalias>
            <portalalias>192.168.1.102/DotNetNuke/club</portalalias>
         </portalaliases>
         <ischild>true</ischild>
      </portal>
   </portals>
</dotnetnuke>
```

Once the file is saved to the Install/Portal folder, the installation of the additional portals defined in this file can be triggered in one of two ways.

1. Using the scheduled event “Install Resources”

2. Manually browsing to the Install page with the following url “Install/Install.aspx?mode=AddPortal”.

When you use the latter method the Install page again provides feedback, similar to that shown below.

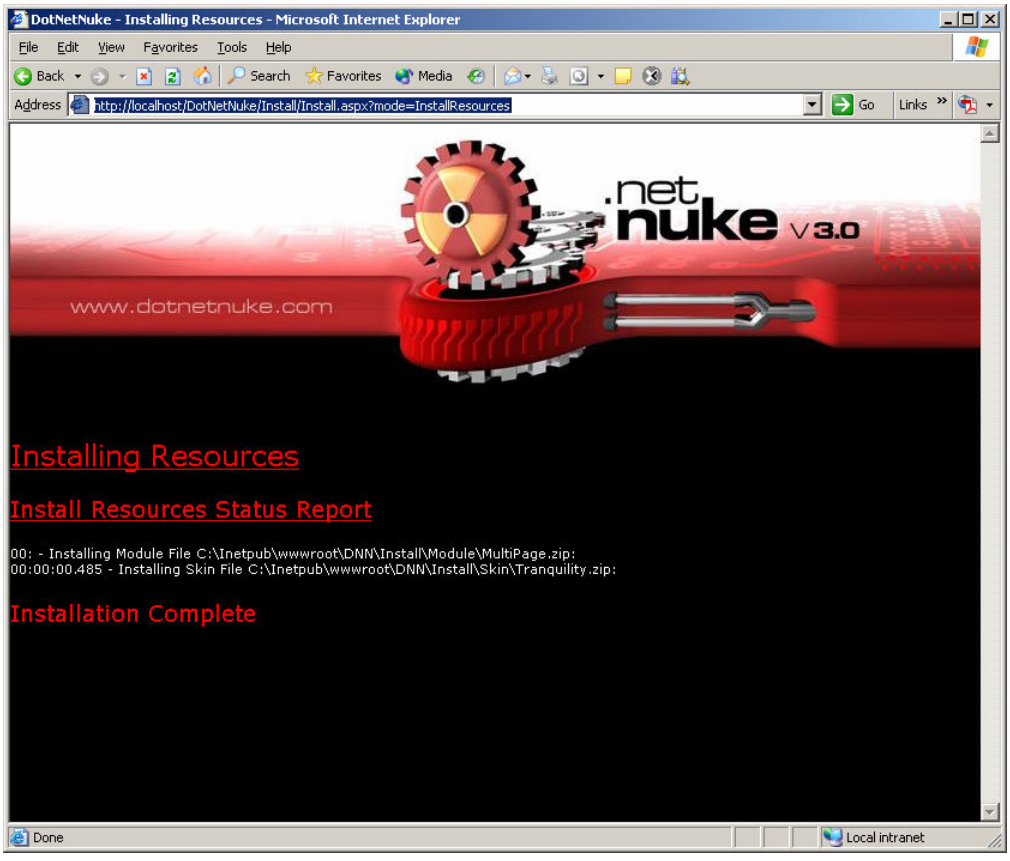


Installing Additional Resources

Additional resource such as Skins, Modules (packaged as Private Assemblies) and Language Packs can also be installed in two ways.

Through the Appropriate Admin/Host menu in the Application. Each resource can be installed through the User Interface element appropriate for that resource. Thus a Module is installed through the Host/Module Definitions menu.

Adding the Installation package to the appropriate folder under /Install. The /Install folder has a number of sub-folders. Zip files placed in these folders cam be installed by browsing to the url "**Install/Install.aspx?mode=InstallResources**", or through the Install Resources Scheduled Event. If installed using the Install page, a status report similar to the one shown below is produced.



The Resource needs to be packaged as a zip file before being installed. As this document focuses on the installation we do not describe here how to package the resource. This is covered in other documents.

The following are the resources that can be installed, using both of these approaches

Modules

- Through Application UI Host/Module Definitions, Select the "Upload New Module" action from the Action menu or from the link at the bottom of the control
- Through Installer by placing zip file in /Install/Module.
- Module Development and Packaging are described in the "DotNetNuke Module Developer's Guide"

Skins/Containers

- Through Application UI Skins menu option on either the Host or Admin menus, Select the "Upload Container" or "Upload Skin" actions from the Action menu or from the link at the bottom of the control
- Through Installer by placing zip file in /Install/Skin or /Install/Container.
- Skin/Container Development and Packaging are described in the "DotNetNuke Skinning" document.

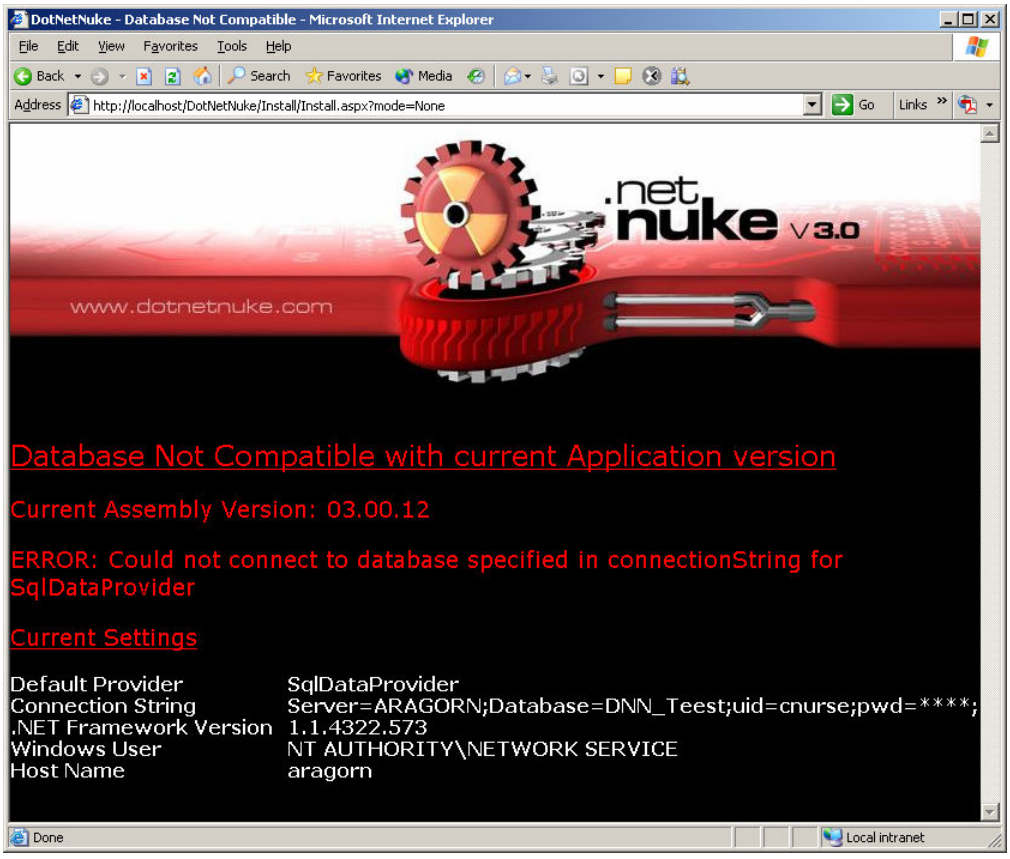
Language Packs

- Through Application UI Host/Languages, Select the "Upload Language Pack" action from the Action menu or from the link at the bottom of the control
- Through Installer by placing zip file in /Install/Language.

What went wrong?

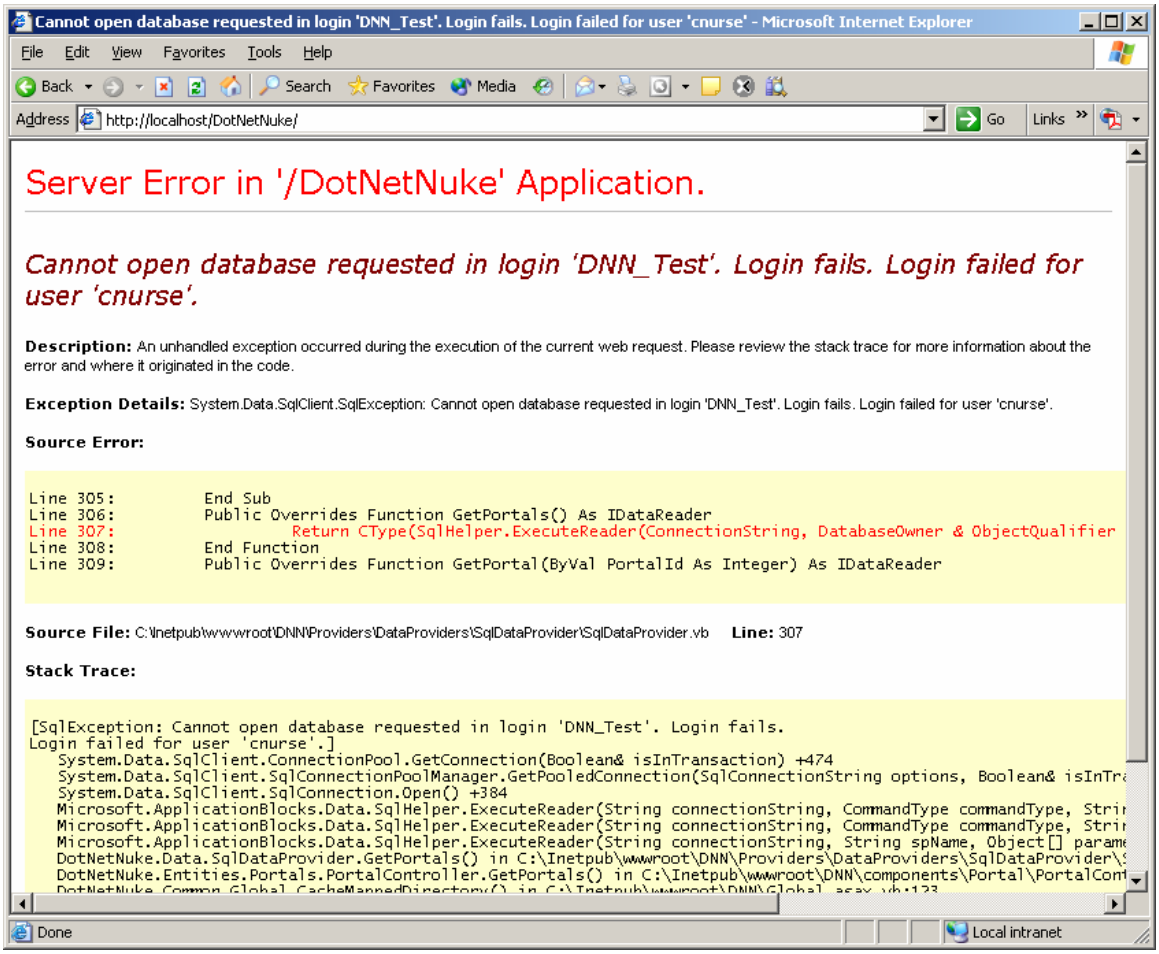
In this section we will attempt to provide answers to some common failure scenarios.

ERROR: Could not connect to database specified in connectionString for SqlDataProvider.



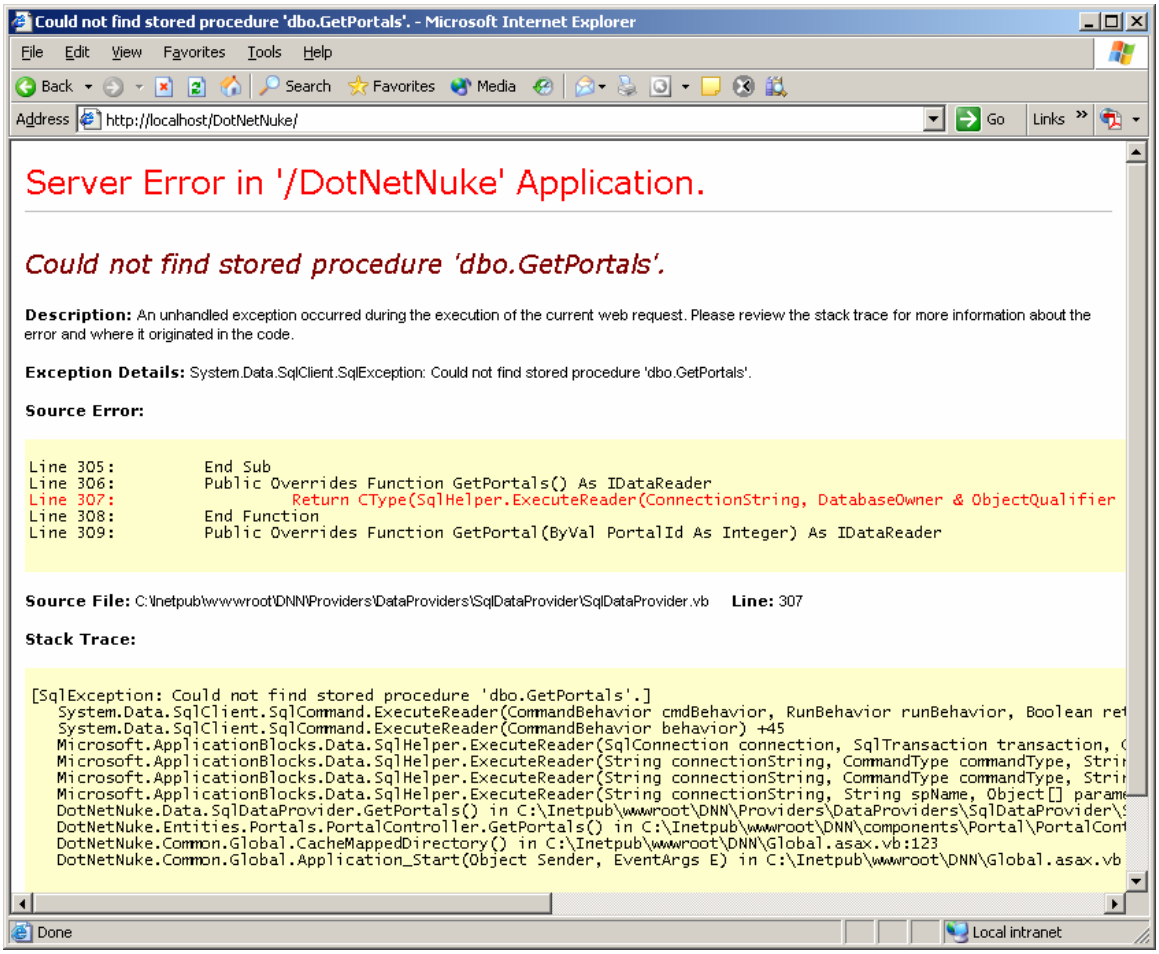
This error is caused by an incorrect connection string – in the example above the Database is mis-spelt, it should be DNN_Test not DNN_Teest

Cannot open database requested in login '{Database}'. Login fails. Login failed for user '{user}'



This error is caused when the user specified in the connection string cannot log in to the database – in this case “cnurse” has not been created in the SQL Server database. It can also occur if the user credentials provided in the connection string are incorrect (eg the password is not entered correctly)

Could not find stored procedure 'dbo.GetPortals'



This error can be caused by an incorrect {**objectqualifier**}. The database version as saved in the **dnn.config** file and the assembly/application version are in sync, but since the {**objectqualifier**} is incorrect the application is trying to execute an unknown stored procedure. In this case, the {**objectqualifier**} is defined as an empty string as shown by the stored procedure being called, and the database was created using "dnn_"

It can also be caused if you are attempting to carry out a fresh installation (new Database) into a root folder where a previous installation of DotNetNuke existed. In this scenario the **dnn.config** file is reporting that the Database Version and Assembly Version are in sync, but this is not the case as we are using a new Database. To solve this error, delete the dnn.conifg file from the /Install folder.

These scenarios are actually quite similar. In both cases the **dnn.config** file is reporting that the Application and Database are in sync, when in fact they are not. In the first case it is because the {**objectqualifier**} is incorrect, in the second the database is actually empty.

Appendix - DotNetNuke Install Template

As mentioned previously, a clean installation uses a template file to control the installation. In web.config the "**InstallTemplate**" setting determines which file is used, so you can either edit the default file **DotNetNuke.install.resources** or create your own and edit the "**InstallTemplate**" setting. The install template is an xml file located in the /Install folder.

The following sections provide an explanation of the nodes/attributes in this file. The Appendix includes a copy of the template that is packaged in the Release.

<dotnetnuke> Node

The <**dotnetnuke**> node is the root node of the template. All other nodes are children of <**dotnetnuke**>. There are 7 immediate children

- **<description> child**
  Provides a description of the template.
- **<scripts> child**
  Defines the script files that builds the Database (these files must be located in the folder indicated by the "providerPath" attribute of the default Data provider in web.config) - see later
- **<version> child**
  The database version that the <**scripts**> files build. This is usually the current version, but if the version is less than the current version then the appropriate upgrade scripts are run to make the database version = app version. For example if the version of the template is 3.0.9 and the Assembly Version is 3.0.12 then the scripts defined by <**scripts**> are run followed by 3.0.10, 3.0.11 and 3.0.12.
- **<superuser> child**
  Information about the superuser (Host Account) - see later
- **<settings> child**
  The default HostSettings for the application – the child nodes of this node correspond to the HostSettings table in the database, and are managed under the Host/Host Settings menu (after installation)
- **<desktopmodules> child**
  The desktop modules to install for this installation – see later

- **<portals> child**
  The portal(s) to be installed – see later

<scripts> Node

The < **scripts** > node defines the scripts that build the Database. It can contain any number of <script> child nodes.

- **<script name="xxx"> child**
  Each <**script**> child node defines a single script that is to be executed. The name attribute can be used to identify the script file (ie what it does). The element value is the filename of the script file.

<superuser> Node

The <superuser> node configures the default superuser (or host account). It has 7 immediate children.

- **<firstname> child**
  The first name of the superuser
- **<lastname> child**
  The last name of the superuser
- **<username> child**
  The user name of the superuser
- **<password> child**
  The password of the superuser – note that the password must meet the requirements configured in the <**memberrolesprototype**> settings above
- **<email> child**
  The email address of the superuser
- **<locale> child**
  The locale (culture) of the superuser
- **<timezone> child**
  The timezone (expressed in minutes from GMT) of the superuser

<portals> Node

The <portals> Node configures the default portals. It can contain a number of <**portal**> nodes which have 8 immediate child nodes.

- **<portalname> child**
  The name of the portal (will display in the TitleBar)
- **<administrator> child**
  Configures the Admiistrator User for the portal - has the same schema as the <superuser> node – see above
- **<description> child**
  A description of the portal (used in the Html Page META Tags)
- **<keywords> child**
  keywords for the portal (used in the Html Page META Tags)
- **<templatefile> child**
  The filename of the template file used for the portal (must be physically located in the Portals/_default directory)
- **<homedirectory> child**
  Sets the portal's home directory (usually left blank – defaults to Portals/[PortalId]
- **<portalaliases><portalalias> child**
  Portal Aliases for the portal (usually left blank – defaults to the location where the installation runs)
- **<ischild> child**
  Must be blank (future use)

Example Template

```
<dotnetnuke>
   <description>This is the default DotNetNuke Host Installation Template</description>
   <scripts>
      <script name="Setup">DotNetNuke.SetUp.SqlDataProvider</script>
      <script name="Schema">DotNetNuke.Schema.SqlDataProvider</script>
      <script name="Data">DotNetNuke.Data.SqlDataProvider</script>
   </scripts>
   <version>03.00.12</version>
   <superuser>
      <firstname>SuperUser</firstname>
      <lastname>Account</lastname>
      <username>host</username>
      <password>host</password>
      <email>host</email>
      <locale>en-US</locale>
      <timezone>0</timezone>
```

```
   </superuser>
   <settings>
      <ControlPanel>Admin/ControlPanel/IconBar.ascx</ControlPanel>
      <Copyright>Y</Copyright>
      <DemoPeriod></DemoPeriod>
      <DemoSignup>N</DemoSignup>
      <DisableUsersOnline>Y</DisableUsersOnline>
      <EncryptionKey Secure="True"></EncryptionKey>
   <FileExtensions>swf,jpg,jpeg,jpe,gif,bmp,png,doc,xls,ppt,pdf,txt,xml,xsl,css,
zip</FileExtensions>
      <HostCurrency>USD</HostCurrency>
      <HostEmail></HostEmail>
      <HostFee></HostFee>
      <HostPortalId>0</HostPortalId>
      <HostSpace></HostSpace>
      <HostTitle>DotNetNuke</HostTitle>
      <HostURL>http://www.dotnetnuke.com</HostURL>
      <PaymentProcessor>PayPal</PaymentProcessor>
      <PerformanceSetting>3</PerformanceSetting>
      <ProcessorPassword Secure="True"></ProcessorPassword>
      <ProcessorUserId Secure="True"></ProcessorUserId>
      <ProxyPassword Secure="True"></ProxyPassword>
      <ProxyPort></ProxyPort>
      <ProxyServer></ProxyServer>
      <ProxyUsername Secure="True"></ProxyUsername>
      <SiteLogBuffer>1</SiteLogBuffer>
      <SiteLogHistory>0</SiteLogHistory>
      <SiteLogStorage>D</SiteLogStorage>
      <SkinUpload>G</SkinUpload>
      <SMTPServer></SMTPServer>
      <SMTPAuthentication></SMTPAuthentication>
      <SMTPUsername Secure="True"></SMTPUsername>
      <SMTPPassword Secure="True"></SMTPPassword>
      <UseCustomErrorMessages>Y</UseCustomErrorMessages>
      <UseFriendlyUrls>Y</UseFriendlyUrls>
      <UsersOnlineTime>20</UsersOnlineTime>
      <SchedulerMode>2</SchedulerMode>
      <AutoAccountUnlockDuration>10</AutoAccountUnlockDuration>
   </settings>
   <desktopmodules/>
   <portals>
      <portal>
         <portalname>DotNetNuke Default Portal</portalname>
         <administrator>
            <firstname>Administrator</firstname>
            <lastname>Account</lastname>
            <username>admin</username>
            <password>admin</password>
            <email></email>
         </administrator>
         <description>Default DotNetnuke Portal</description>
         <keywords>Default, DotNetNuke, CMS, Web, Future</keywords>
         <templatefile>DotNetNuke.template</templatefile>
         <portalaliases>
            <portalalias></portalalias>
         </portalaliases>
         <ischild>false</ischild>
      </portal>
   </portals>
</dotnetnuke>
```

Additional Information

The DotNetNuke Portal Application Framework is constantly being revised and improved. To ensure that you have the most recent version of the software and this document, please visit the DotNetNuke website at:

http://www.dotnetnuke.com

The following additional websites provide helpful information about technologies and concepts related to DotNetNuke:

DotNetNuke Community Forums
http://www.dotnetnuke.com/tabid/795/Default.aspx

Microsoft® ASP.Net
http://www.asp.net

Open Source
http://www.opensource.org/

W3C Cascading Style Sheets, level 1
http://www.w3.org/TR/CSS1

Errors and Omissions

If you discover any errors or omissions in this document, please email marketing@dotnetnuke.com. Please provide the title of the document, the page number of the error and the corrected content along with any additional information that will help us in correcting the error.

Appendix A: Document History

Version	Last Update	Author(s)	Changes
1.0.0	March, 2005	Charles Nurse	• Initial 3.0 Installation Guide
1.0.1	June, 2005	Charles Nurse	• Updated for version 3.1.0
1.0.2	Aug 16, 2005	Shaun Walker	• Applied new template
1.0.3	Nov 22, 2005	Charles Nurse	• Updated for versions 3.2/4.0