Setting up your own SymbolSource Server: step-by-step

I've already explained why everyone should be using a symbols server. In that post, I explained that TFS comes with its own integrated symbols server, but at the moment it doesn't play very nice with NuGet packages. Or maybe you are not using TFS at all.

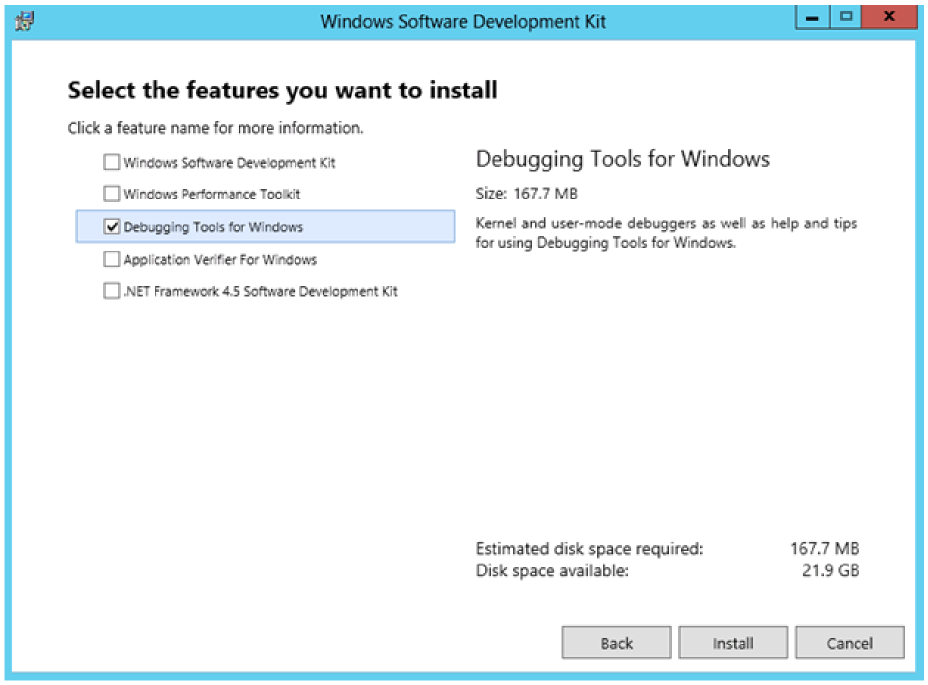
That's where SymbolSource comes in. NuGet has the ability to create symbols packages, so you can easily push them to a symbols package repository. This is exactly what SymbolSource offers, either as a public or private repository on SymbolSource.org, or as a separate cloud-based instance of SymbolSource where you also can manage users and repositories. Both plans are still in an unlimited free beta mode, so give it a try.

As much as I am a huge fan of the hosted SymbolSource offering, some people prefer to host code, symbols and packages on their own infrastructure, if only to troll some sys-admins or slow down development (if you're a manager, I'm talking about *security*).

With the release of the SymbolSource community edition, I figured it was about time to try it out for myself.

Prerequisites

Before you open Visual Studio and search NuGet for the 'symbolsource' keyword, you'll have to install the Debugging Tools for Windows. As I'm going to try it out on a Windows Azure Virtual Machine running Windows Server 2012 RC (living on the edge aren't we), I'll simply install the Windows SDK for Windows 8 Release Preview which contains the **Debugging Tools for Windows** as a standalone component.



Windows Software Development Kit

Select the features you want to install

Click a feature name for more information.

- [ ] Windows Software Development Kit
- [ ] Windows Performance Toolkit
- [✓] Debugging Tools for Windows
- [ ] Application Verifier For Windows
- [ ] .NET Framework 4.5 Software Development Kit

Debugging Tools for Windows

Size: 167.7 MB

Kernel and user-mode debuggers as well as help and tips for using Debugging Tools for Windows.

Estimated disk space required:	167.7 MB
Disk space available:	21.9 GB

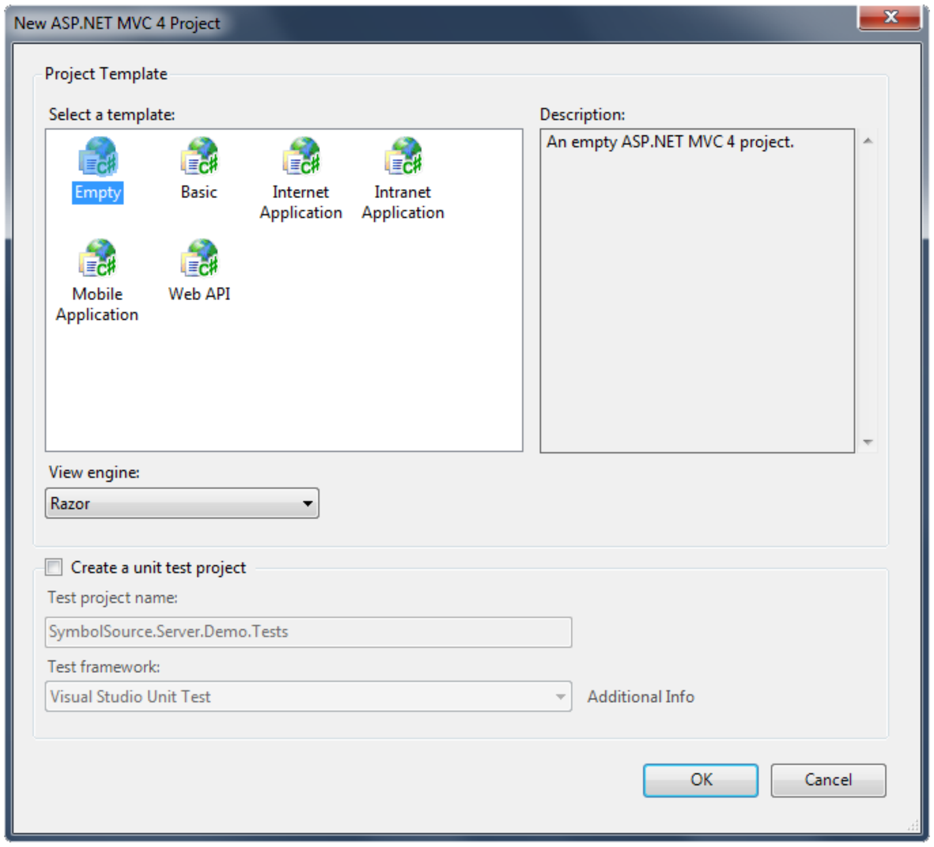
Keep track of the installation path as you will need it later on.

Setting up your basic SymbolSource server

If you are familiar with setting up your own NuGet server (if not, check out this book), you'll notice that setting up your own SymbolSource server is peanuts. Very similar to the NuGet.Server package, SymbolSource provides you with a SymbolSource.Server.Basic package on NuGet (they just released v1.1.0 so go get it while it's hot!). Let's get this thing rolling...

Creating the Web application

I started off an *empty* ASP.NET MVC4 application.


New ASP.NET MVC 4 Project

Project Template

Select a template: Empty, Basic, Internet Application, Intranet Application, Mobile Application, Web API

Description: An empty ASP.NET MVC 4 project.

View engine: Razor

Create a unit test project

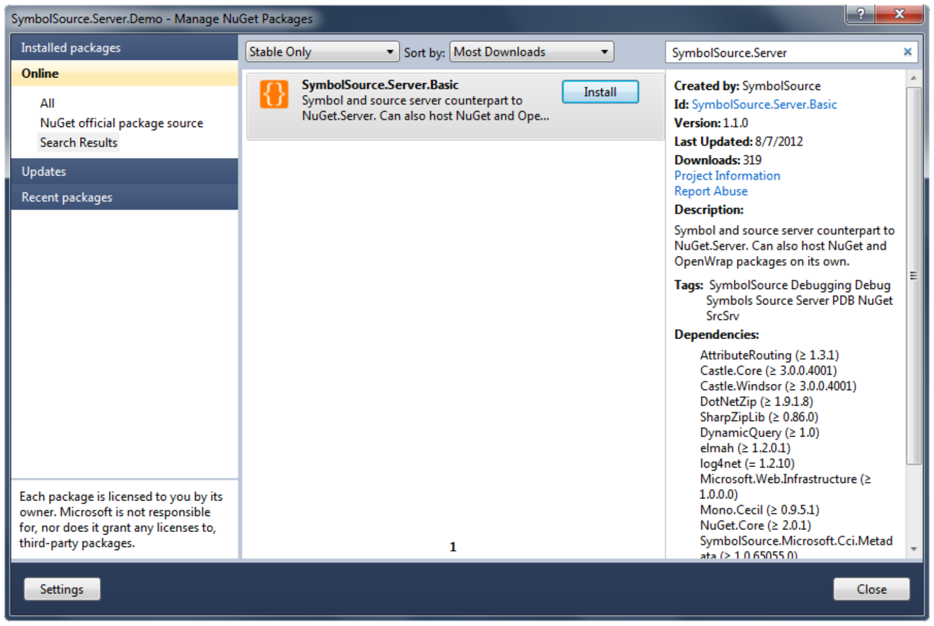
Test project name: SymbolSource.Server.Demo.Tests

Test framework: Visual Studio Unit Test

Next I ran the following script in the NuGet Package Manager Console:

```
Install-Package SymbolSource.Server.Basic
```

For those who prefer clicking there way through the UI, you can achieve the same by right clicking your Web project's references, select Manage NuGet Packages, and query the NuGet official feed for the term "SymbolSource.Server".


SymbolSource.Server.Demo - Manage NuGet Packages

Stable Only | Sort by: Most Downloads | SymbolSource.Server

SymbolSource.Server.Basic
Symbol and source server counterpart to NuGet.Server. Can also host NuGet and Ope...

Created by: SymbolSource
Id: SymbolSource.Server.Basic
Version: 1.1.0
Last Updated: 8/7/2012
Downloads: 319
Project Information
Report Abuse
Description:
Symbol and source server counterpart to NuGet.Server. Can also host NuGet and OpenWrap packages on its own.
Tags: SymbolSource Debugging Debug Symbols Source Server PDB NuGet SrcSrv
Dependencies:
AttributeRouting (≥ 1.3.1)
Castle.Core (≥ 3.0.0.4001)
Castle.Windsor (≥ 3.0.0.4001)
DotNetZip (≥ 1.9.1.8)
SharpZipLib (≥ 0.86.0)
DynamicQuery (≥ 1.0)
elmah (≥ 1.2.0.1)
log4net (= 1.2.10)
Microsoft.Web.Infrastructure (≥ 1.0.0.0)
Mono.Cecil (≥ 0.9.5.1)
NuGet.Core (≥ 2.0.1)
SymbolSource.Microsoft.Cci.Metadata (≥ 1.0.65055.0)

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Installing this package injects a ton of dependencies into your consuming Web application project. This isn't a bad thing, really, because you get a whole lot of functionality by simply installing this single NuGet package. It even comes with ELMAH logging preconfigured, so you get logging out of the box.

Note: Don't forget about security! See http://code.google.com/p/elmah/wiki/SecuringErrorLogPages for more information on remote access and securing ELMAH.

Now's the time you need to configure the installation path of *Debugging Tools for Windows*. To do so, open the Web.config file and adjust the section. You'll find a predefined SrcSrvPath element with the default installation path for a x64 machine running Windows 8 or Windows Server 2012. If you chose a different installation path, or if you are running a previous version of Windows, simply replace the value with your installation path.

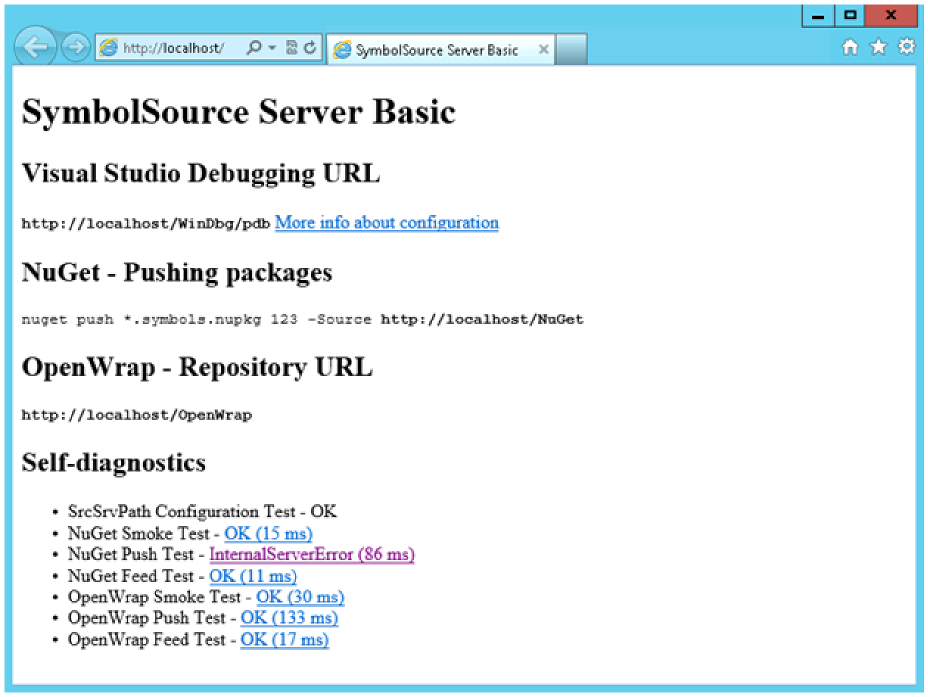
Windows Server 2012 / Windows 8:

```
<add key="SrcSrvPath" value="C:\Program Files (x86)\Windows Kits\8.0\Debuggers\x64\srcsrv" />
```

Earlier versions of Windows:

```
<add key="SrcSrvPath" value="C:\Program Files\Debugging Tools for Windows (x86)\srcsrv" />
```

You can now start the application and browse the home page shown below.



SymbolSource Server Basic

Visual Studio Debugging URL

`http://localhost/WinDbg/pdb` More info about configuration

NuGet - Pushing packages

`nuget push *.symbols.nupkg 123 -Source http://localhost/NuGet`

OpenWrap - Repository URL

`http://localhost/OpenWrap`

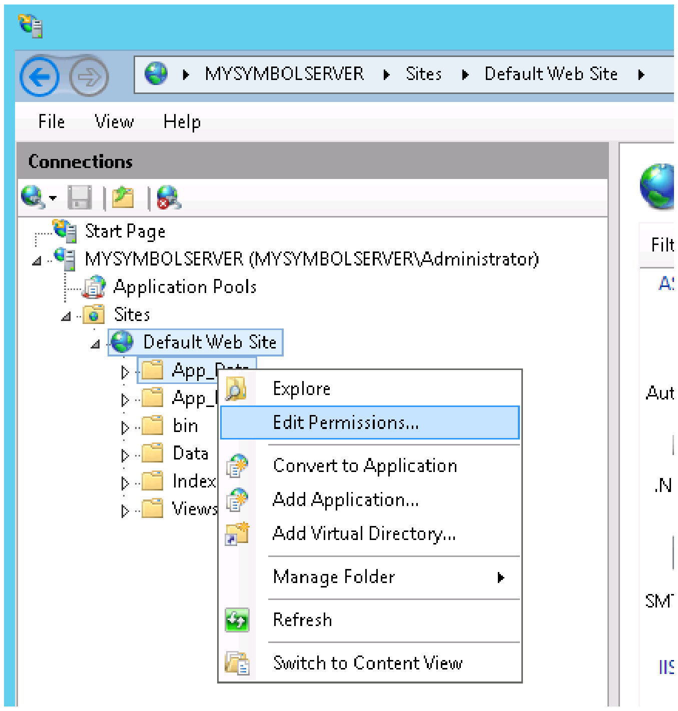
Self-diagnostics

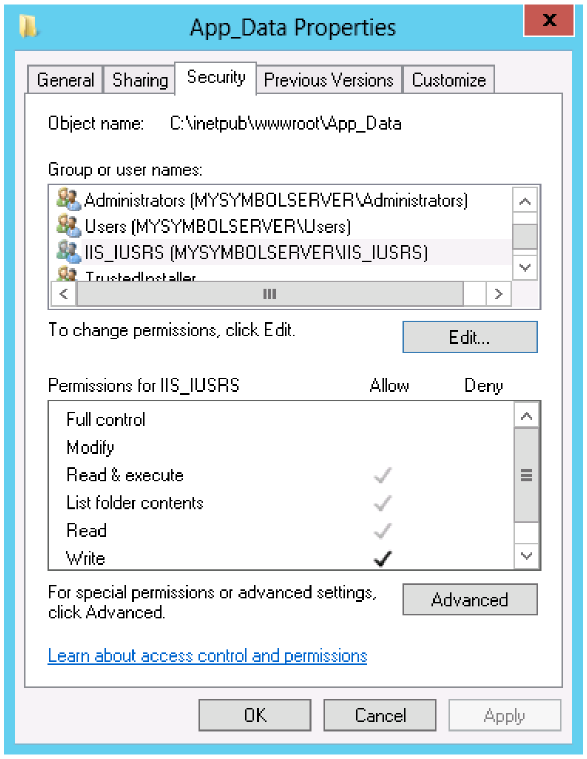
- SrcSrvPath Configuration Test - OK
- NuGet Smoke Test - OK (15 ms)
- NuGet Push Test - InternalServerError (86 ms)
- NuGet Feed Test - OK (11 ms)
- OpenWrap Smoke Test - OK (30 ms)
- OpenWrap Push Test - OK (133 ms)
- OpenWrap Feed Test - OK (17 ms)

All the information you need, including all required URLs, is listed on the application's start page.

- **Visual Studio Debugging URL** - You need to configure this URL in the Debug settings of your Visual Studio development environment. More info and a recommended setup are documented on the SymbolSource Web site.
- **NuGet Symbols Packages Repository URL** - This is the NuGet package repository where you should push your symbols packages to.
- **OpenWrap Repository URL** - Yep, SymbolSource supports both NuGet and OpenWrap!
- **Self-diagnostics** - Very useful and very nice from the SymbolSource guys to provide this. This will help you verify whether the prerequisite is installed and your Web site is configured correctly to provide a fully functioning SymbolSource basic server. Shield this info from the public! No one needs to know...

If you look at the previous image, you'll notice that the NuGet push test failed with an InternalServerError. Checking this on the server itself reveals that the application has no access to the App_Data folder. To fix this, simply add write permissions for the IUSR on the **App_Data** folder through IIS Management Console. Make sure you do the same for the **Data** and the **Index** folder.




App_Data Properties

General | Sharing | Security | Previous Versions | Customize

Object name: C:\inetpub\wwwroot\App_Data

Group or user names:

Administrators (MYSYMBOLSERVER\Administrators)
Users (MYSYMBOLSERVER\Users)
IIS_IUSRS (MYSYMBOLSERVER\IIS_IUSRS)
TrustedInstaller

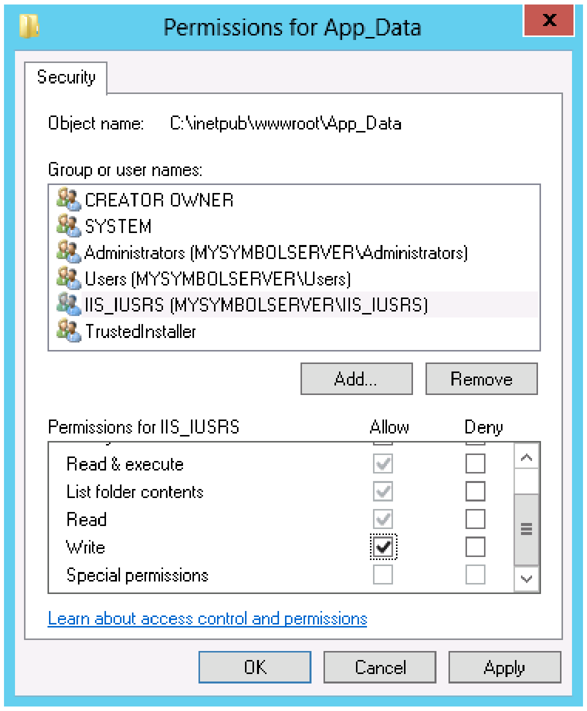
To change permissions, click Edit. [Edit...]

Permissions for IIS_IUSRS	Allow	Deny
Full control		
Modify		
Read & execute	✓	
List folder contents	✓	
Read	✓	
Write	✓	

For special permissions or advanced settings, click Advanced. [Advanced]

Learn about access control and permissions

[OK] [Cancel] [Apply]



Object name: C:\inetpub\wwwroot\App_Data

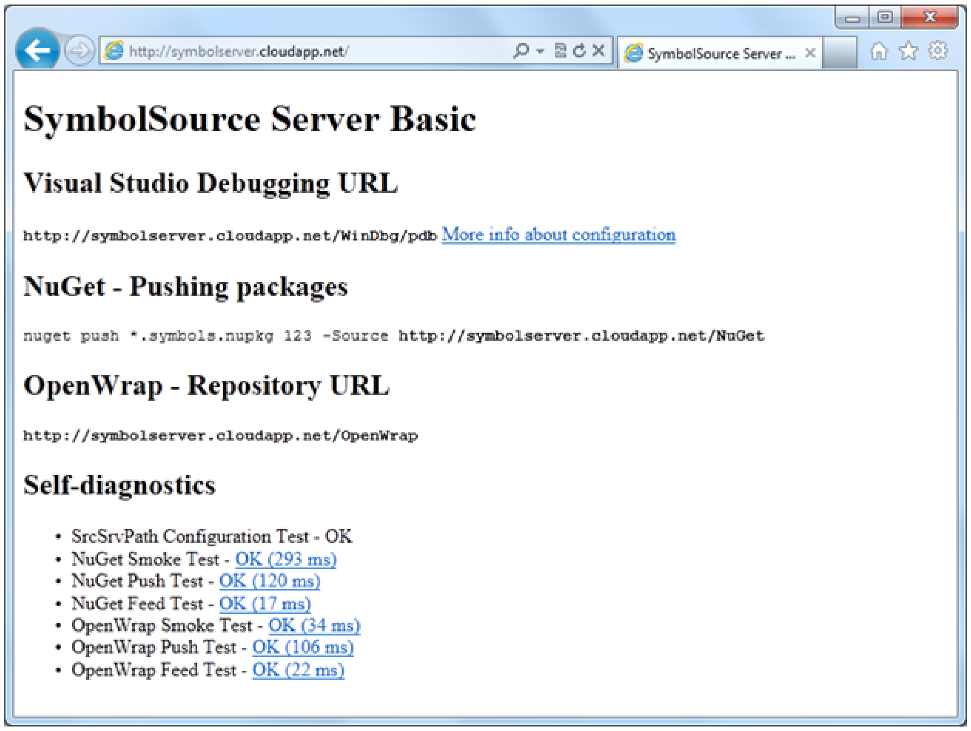
Group or user names:
- CREATOR OWNER
- SYSTEM
- Administrators (MYSYMBOLSERVER\Administrators)
- Users (MYSYMBOLSERVER\Users)
- IIS_IUSRS (MYSYMBOLSERVER\IIS_IUSRS)
- TrustedInstaller

Permissions for IIS_IUSRS	Allow	Deny
Read & execute	✓	
List folder contents	✓	
Read	✓	
Write	✓	
Special permissions		

There's one last thing you need to do: modify your web.config and add the following elements:

```
<system.webServer>
 <modules runAllManagedModulesForAllRequests="true" />
</system.webServer>
```

If all went well, your home page should indicate that your server has been configured correctly.



SymbolSource Server Basic

Visual Studio Debugging URL

`http://symbolserver.cloudapp.net/WinDbg/pdb` More info about configuration

NuGet - Pushing packages

```
nuget push *.symbols.nupkg 123 -Source http://symbolserver.cloudapp.net/NuGet
```

OpenWrap - Repository URL

`http://symbolserver.cloudapp.net/OpenWrap`

Self-diagnostics

- SrcSrvPath Configuration Test - OK
- NuGet Smoke Test - OK (293 ms)
- NuGet Push Test - OK (120 ms)
- NuGet Feed Test - OK (17 ms)
- OpenWrap Smoke Test - OK (34 ms)
- OpenWrap Push Test - OK (106 ms)
- OpenWrap Feed Test - OK (22 ms)

Creating and pushing symbols packages

A symbols package has the *.symbols.nupkg extension and contains only DLL, PDB, XMLDOC and source files. To create them, add the -Symbols option to the NuGet pack command. You should see two packages being created instead of one. A detailed description on the symbols package contents and structure can be found on the online NuGet Documentation.

Quoting from the SymbolSource documentation:

"At this moment, the server accepts any NuGet API key, but you are free to secure it using any of the IIS authentication capabilities."

This means you can easily push the symbols package using the following command:

```
nuget push {packageID}.symbols.nupkg {API key} -source {url}
```

The server will ignore your API key (as it accepts anything at the moment). Replace the {url} placeholder with the NuGet symbols packages repository URL you found on your server's home page.

That's it! Happy packaging!

Sharing is caring

Tweet Follow @xavierdecoster

G+1 +6 Recommend this on Google

Like One person likes this.

NUGET

Posted by Xavier Decoster on Aug 7 2012

Last revised: 27 Oct, 2012 10:16 PM

Comments for this thread are now closed. ✕

15 Comments xavierdecoster.com 1 Login

Recommend Share Sort by Oldest

James R Counts • 4 years ago

Hey Xavier. Nice article!

I followed your instructions but hit a road block trying to deploy to Server 2008 R2. No matter what I try (including just copying the entire solution onto the server), only "SrcSvrPath" shows up as "OK" all other tests are "Not Found".

Clearly I'm missing something. Any ideas?

• Share ›

Xavier Decoster Mod ➦ James R Counts • 4 years ago

Hi James,
You should get more details about the Not Found errors when clicking those links.
Anything of interest in there that might point us towards the root cause of the issue?

• Share ›

Andy • 4 years ago

Hi Xavier,

This all looks like just what I need, however I can't seem to get it working. Whenever I push a package the the cmd prompt says that the package was pushed but the App_Data folder just get's a new error*.xml document. The error is "The controller for path '/Nuget' could not be found or it does not implement IController." Don't suppose you have any idea what's wrong?

• Share ›

TripleEmcoder ➦ Andy • 4 years ago

That error might not necessarily indicate a problem. We have a minor issue with routing to fix. But anyway packages should appear in Data. Is it still empty after pushing? Does the diagnostic homepage indicate any problems?

• Share ›

Greg Roberts • 3 years ago

Xavier, trying to raise someone at symbolsource@symbolsource.org about hosting an inhouse symbol server with nuget, can you help with a contact ? Thanks

• Share ›

Xavier Decoster Mod ➦ Greg Roberts • 3 years ago you could also try pinging @TripleEmcoder on Twitter

• Share ›

Matthieu Penant • 3 years ago

Hi Xavier,

I followed your procedure, all the tests are passing server-side, but when I try to push a package, I end up with this message : "The remote server returned an error: (506) Package submission failed: The given path's format is not supported". Any idea about what is missing ? All paths involved have nothing special as far as I know.

• Share ›

Chris N → Matthieu Penant • 3 years ago

Hi Matthieu,
I ran into the same problem. Do you happen to find a solution, yet?

• Share ›

Xavier Decoster Mod → Chris N • 3 years ago

It is very likely you guys are already using a more recent version of the package than the one I used when writing this post. A blind shot at the issue: does any of the configured paths end with a trailing slash (or not)? Could you share the exact push command you are trying to perform? You might also want to try reaching out to **@TripleEmcoder**, as I personally haven't had this issue before.

• Share ›

Chris N → Xavier Decoster • 3 years ago

Hi Xavier,

Thanks for reply. I found that even there was an error, the package was pushed to the server correctly. However I tried to create the symbol server on Windows Server 2008 R2, and I ran the Self-diagnostics, the NuGet Push Test show InternalServerError even though I had all the folders permission set to write. Click on the InternalServerError I get:

The remote server returned an error: (405) Method Not Allowed.
Description:
An unhandled exception occurred during the execution of the current web
request. Please review the stack trace for more information about the error and
where it originated in the code.

Exception Details:
System.Net.WebException: The remote server returned an error: (405) Method
Not Allowed.

Source Error:

---

see more

2 • Share ›

Chris N → Chris N • 3 years ago

I'm able to get it running, nut the Visual Studio Debugging URL is not working. Nothing get download in the catch folder.

• Share ›

Xavier Decoster Mod → Chris N • 3 years ago

As you are not running on Windows Server 2012, did you install the correct version of Debugging Tools for Windows?

• Share ›

Chris N → Xavier Decoster • 3 years ago

I've tried both Debugging Tools for Windows 7 and 8. Neither of them work.

• Share ›

Chris N → Chris N • 3 years ago

Hi Xavier,

I'm able to get it working now. It was a native project type that I used for testing, and even though the symbol package got pushed to the server, the source code never did.

I still have this one problem that I couldn't figure out is how to change the location to store the data? Instead of store data in the Data folder, how do I change that to store the data on a different location on the network?

I'm really appreciated.

CN

1 • Share ›

Xavier Decoster Mod Chris N • 3 years ago

Didn't try this myself yet, but you could try making the Data directory a Virtual Directory?

• Share ›

ALSO ON **XAVIERDECOSTER.COM**

Generated AssemblyVersion for NuGet package on TFS Build

2 comments • 4 years ago

Avat Xavier Decoster — The nuget.settings.targets file has been merged into the nuget.targets file in the latest version of the nuget.build package which is used under the hood of the pkg …

NuGet $version$ token explained

2 comments • 4 years ago

Avat Xavier Decoster — The -Version option on the command line will always override the version specified in the nuspec

Subscribe Add Disqus to your site Add Disqus Add Privacy

Disqus

Log in.

;