* 1. 

Hands-On Lab

ASP.NET MVC – NuGet

Lab version: 1.1.0

Last updated: 1/7/2011

Contents

[Overview 3](#_Toc282171614)

[Exercise 1: Creating a NuGet Package 4](#_Toc282171615)

[Task 1 – Adding a nuspec File 5](#_Toc282171616)

[Task 2 – Creating the NuSpec Script 6](#_Toc282171617)

[Task 3 – Generating the NuGet File 6](#_Toc282171618)

[Task 4 – Using Post-Build Events to Generate the Package 7](#_Toc282171619)

[Task 5 – Building and Packaging the Solution 9](#_Toc282171620)

[Exercise 2: Consuming NuGet Packages 10](#_Toc282171621)

[Task 1 – Consuming an External Library from Visual Studio 2010 NuGet Extension 11](#_Toc282171622)

[Task 2 – Running the Solution 13](#_Toc282171623)

[Summary 14](#_Toc282171624)

Overview

* 1. **Note:** This Hands-on Lab assumes you have basic knowledge of **ASP.NET**. If you have not used **ASP.NET MVC** before, we recommend you to go over **ASP.NET MVC Fundamentals** Hand-on Lab.
  2. With .NET you can develop class libraries, web projects, complete solutions, frameworks, application servers, web controls and helpers, etc. A live ecosystem of such packages (many of them open source projects) is available for you to download from the Internet and add to your own solutions. You can also create your own library and publish it. Nevertheless, this model implies some costs: you should locate the appropriate software, download and compile it, resolve dependencies of other packages, etc.
  3. With NuGet (<http://nuget.codeplex.com>) tool, you can download, create, and publish software packages, ready to be used by other developers, resolving the hassles of dependencies tracking. You can use a console tool to create a new package, upload it to a Microsoft server or to a server of your own, browse and install them from Visual Studio.
  4. In this Hands-on Lab, you will learn how to:
  + Make a NuGet package for MvcMusicStore application
  + Install NuGet Visual Extension tool in Visual Studio
  + Update an ASP.NET MVC application with a package downloaded from local NuGet Server

# System Requirements

* 1. You must have the following items to complete this lab:
  + ASP.NET and ASP.NET MVC 3
  + Visual Studio 2010
  + SQL Server Database (Express edition or above)

# Exercises

* 1. This Hands-On Lab is comprised by the following exercises:
  2. Exercise 1: Creating a NuGet Package
  3. Exercise 2: Consuming NuGet Packages
  4. Estimated time to complete this lab: **30 minutes**.
  5. **Note:** Each exercise is accompanied by an **End** folder containing the resulting solution you should obtain after completing the exercises. You can use this solution as a guide if you need additional help working through the exercises.

# Next Step

* 1. Exercise 1: Creating a NuGet Package

Exercise 1: Creating a NuGet Package

* 1. In this exercise you will learn how to package your solution using **NuGet**.
  2. For that purpose, you will run a script that will create the MusicStore package in your project folder. The script will be configured with a **nuspec** (NuGet specification) file, which will specify all the package settings and dependencies.
  3. **Note:** A **nuspec** file is a manifest in **XML** format that is used to build the package and is also stored in the package after the build process. It contains:

- The **metadata** for a package that includes the following fields: Id, version, authors (collection of author elements), description, language, tags (collection), licenceUrl (Uri), projectURL (Uri), iconUrl (Uri) and requrireLicenceAcceptance(boolean).

* 1. - An optional list of package’s **dependencies**. Each dependency element has the following attributes: id (the ID of a package that this package depends on), version (the required version of the dependency package. If you enter this value as a specific version number, such as 1.2.3, NuGet assumes that any package of that version or later is allowable. You can be more specific about which versions are allowable by using interval notation; for more information, see [Specifying Version Ranges](http://nuget.codeplex.com/wikipage?title=Version%20Range%20Specification)).
  2. - An optional list of **files** **to include** in the package. Each file element has the following attributes: src, target (both are paths).
  3. Other considerations:
  4. - If no version is specified, NuGet will assume that the latest version is being used.
  5. - All paths are relative to the **nuspec** file, unless an absolute file is specified.
  6. For more information, check [http://nuget.codeplex.com/documentation?title=Nuspec Format](http://nuget.codeplex.com/documentation?title=Nuspec%20Format).

Below is a sample nuspec file that uses most of the described elements:

* 1. XML
  2. <?xml version="1.0" encoding="utf-8"?>
  3. <package>
  4. <metadata>
  5. <id>NuGetSample</id>
  6. <version>2.0.0</version>
  7. <authors>Microsoft</authors>
  8. <description>This is an example of a nuspec file</description>
  9. <tags>NuGet nuspec hands-on-lab microsoft asp.net</tags>
  10. <language>en-US</language>
  11. </metadata>
  12. <dependencies>
  13. <dependency id="sample-package" version="3.0.0" />
  14. <dependency id="another-package" version="4.0.0" />
  15. </dependencies>
  16. <files src="..\sample\anyfile.dll"/>
  17. <files src="..\sample\requiredLibrary.dll" target="bin"/>
  18. </package>

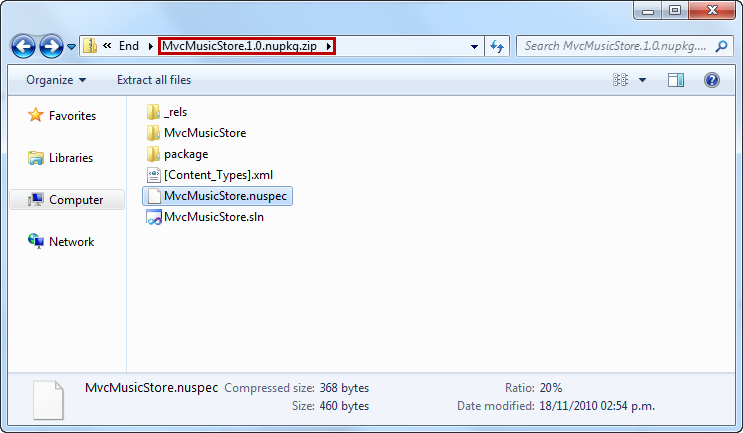
Task 1 – Adding a nuspec File

* 1. In this task, you will add a **nuspec** file that will contain a description of the package to be created.
  2. Open the **Source\Ex01-Creating NuGet Package\Begin\MvcMusicStore** folder in Windows Explorer.
  3. Create a new empty txt file and rename it to **MvcMusicStore.nuspec**.
  4. Open **MvcMusicStore.nuspec** in Visual Studio or any text/XML editor.
  5. Add the following XML specification to the **MvcMusicStore.nuspec** file:
     1. XML
     2. **<?xml version="1.0" encoding="utf-8"?>**
     3. **<package>**
     4. **<metadata>**
     5. **<id>MvcMusicStore</id>**
     6. **<version>1.0</version>**
     7. **<description>Music Store ASP.NET MVC Application</description>**
     8. **<authors>Microsoft</authors>**
     9. **<language>en-US</language>**
     10. **</metadata>**
     11. **</package>**
     12. **Note:** Language attribute is an ISO code. You could find more information about interntionalization in this article at [msdn](http://msdn.microsoft.com/en-us/library/ee796272%28CS.20%29.aspx).

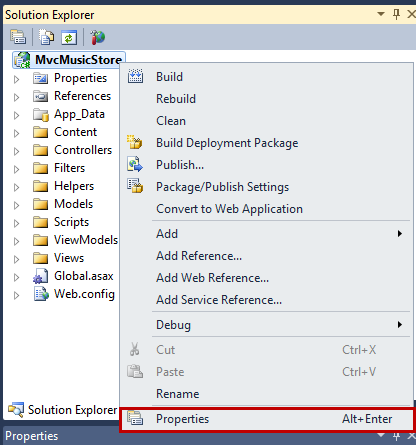
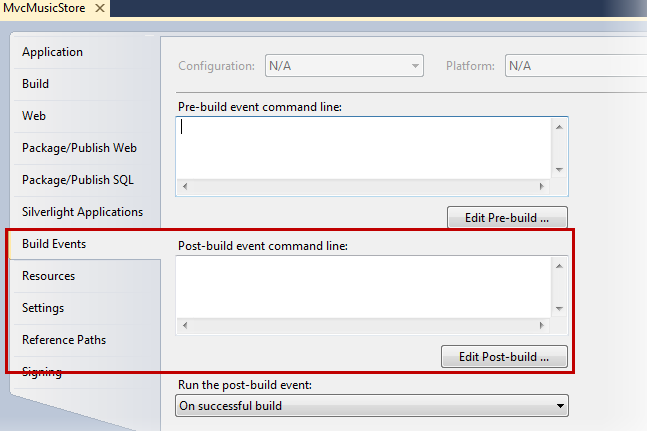
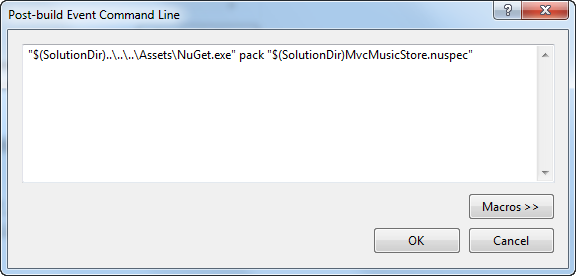
Task 2 – Creating the NuSpec Script

* 1. In this task, you will create a script that will automatically use NuGet executable with **MvcMusicStore.nuspec** specifications:
  2. Open the **Source\Ex01-Creating NuGet Package\Begin** folder in Windows Explorer.
  3. Create a new txt file and rename it to **CreateNuGetPackage.cmd**.
  4. Open **CreateNuGetPackage.cmd** in Visual Studio or any text editor, and add the following lines:
     1. cmd
     2. **@echo off**
     3. **..\..\Assets\NuGet.exe pack MvcMusicStore\MvcMusicStore.nuspec**
     4. **Note:** Package file will be automatically named with **nuspec** attributes as *<name>.<version>.nupkg*
     5. The [**NuGet.exe**](http://nuget.codeplex.com/releases/52018/download/184132) command-line utility is a wrapper for the PackageBuilder API used to create packages. Therefore, as an alternative you can use the API directly to create packages.

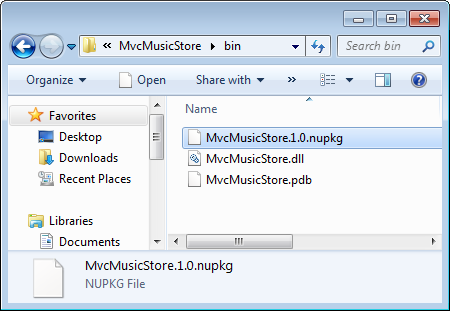
Task 3 – Generating the NuGet File

* 1. Execute the **CreateNuGetPackage.cmd** file by double-clicking it. After that, check that a new package file **MvcMusicStore.1.0.nupkg** was generated in the same folder.
  2. Rename **MvcMusicStore.1.0.nupkg** as a .zipfile.
  3. Open it with Windows Explorer to see its content. Note that the **NuGet** package contains the **nuspec** file used to build the package as well as the solution files.
     1. 
     2. Figure
     3. .nuget file content

Task 4 – Using Post-Build Events to Generate the Package

* 1. In this task, you will use Visual Studio Post-Build events to automatically generate the NuGet package each time the solution is built. In order to do that, you will include the NuGet script into post-build events set up. Therefore, NuGet.exe will get executed in the same way as before, but you will not need an extra step each time you want to package:
  2. Open the begin solution at **Source\Ex01-Creating NuGet Package\Begin\MvcMusicStore**.
  3. Open **MvcMusicStore** project Properties from Solution Explorer:
     1. 
     2. Figure 2
     3. Project properties
  4. Open the **Build-Events** tab and click on **Edit Post-Build**.
     1. 
     2. Figure 3
     3. Adding Build Events
  5. Copy the following line into the **Post-build Command Line** dialog and click **OK**:
     1. Post-build Command
     2. **"$(SolutionDir)..\..\..\Assets\NuGet.exe" pack "$(SolutionDir)MvcMusicStore.nuspec"**
     3. 
     4. Figure 4
     5. Creating a Post-build event

Task 5 – Building and Packaging the Solution

* 1. In this task, you will build the solution and verify that the package is generated after the build.
  2. Build the solution or press **CTRL+F5**.
  3. Check that the package was generated at the **bin** folder:
     1. 
     2. Figure 5
     3. Post-build NuGet generated at \bin folder
  4. Rename **MvcMusicStore.nupkg** as a .zipfile.
  5. Open it with Windows Explorer to see the content. Note that each **nuget** package contains the **nuspec** file used to build the package.

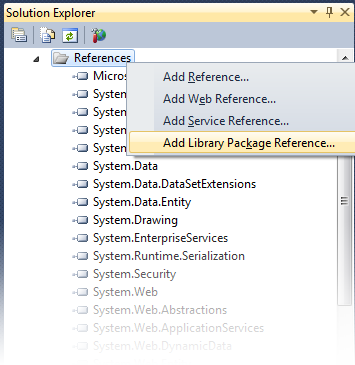
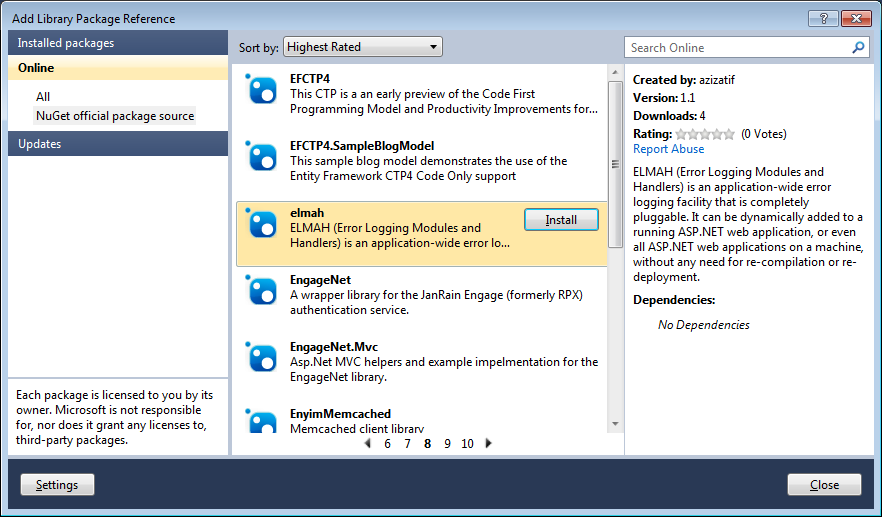
# Next Step

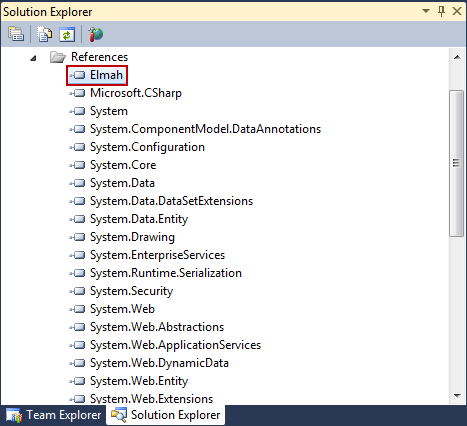
1. Exercise 2: Consuming NuGet Packages

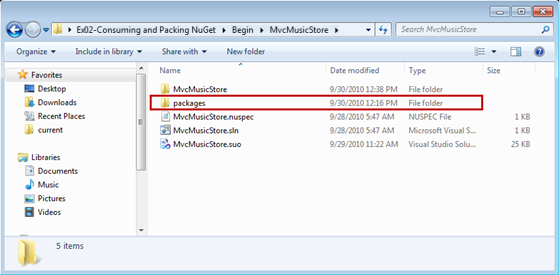
Exercise 2: Consuming NuGet Packages

* 1. In this exercise, you will learn how to consume external libraries in your project by using NuGet public repositories. You will be able to add any packaged library from the cloud into your project.
  2. With that goal in mind, in this exercise you will add an ELMAH (Error Logging Modules and Handlers) Library to your project, enabling you to see the error log generated by the library.
  3. **Note: ELMAH (Error Logging Modules and Handlers)** is a pluggable library for error and exception logging that can be dynamically added to an ASP.NET web application.
  4. To read more about **ELMAH**, see this article at [msdn](http://msdn.microsoft.com/en-us/library/aa479332.aspx).

Task 1 – Consuming an External Library from Visual Studio 2010 NuGet Extension

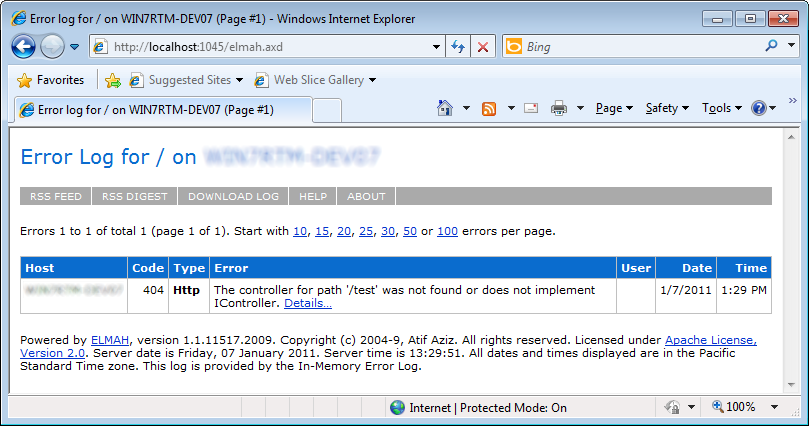
* 1. In this task, you will include **ELMAH** library into your project to add an error log.
  2. Open the begin solution at **Source\Ex02-Consuming and Packing NuGet\Begin\MvcMusicStore**.
  3. From Solution Explorer, open References folder and right-click to add **Library Package Reference.**
     1. 
     2. Figure
     3. NuGet Tool – Adding a package reference
  4. Select **ELMAH** from the package list and click **Install**.
     1. 
     2. Figure
     3. NuGet Tool – Installing a package reference



* + 1. Figure
    2. Package Manager Options – Elmah Library added to the solution
    3. 
    4. Figure
    5. Project folder– Elmah Library download location

Task 2 – Running the Solution

In this task, you will run the solution to see ELMAH log working:

* 1. Press **F5** to run **MvcMusicStore** solution.
  2. Browse an inexistent URL like **/test**. You should see a “*The resource cannot be found*” error message.
  3. Browse **/elmah.axd** to see the log that will include a line for the error logged when browsing an inexistent URL:
     1. 
     2. Figure
     3. Elmah Library log view

# Next Step

Summary

Summary

* 1. By completing this Hands-On Lab you have learned how to use NuGet Packages in your solution:
  + Create a NuGet package for your application
  + Add a Package Reference from Visual Studio