

Control de versiones con TFS

Parte 3

Gestión de dependencias con NuGet

Spanish translation
by Juan María Laó Ramos

Visual Studio ALM Rangers



Microsoft



Visual Studio

La información contenida en este documento representa la visión de Microsoft Corporation sobre los asuntos analizados a la fecha de publicación. Dado que Microsoft debe responder a las condiciones cambiantes del mercado, no debe interpretarse como un compromiso por parte de Microsoft, y Microsoft no puede garantizar la exactitud de la información presentada después de la fecha de publicación.

Este documento es sólo para fines informativos. MICROSOFT NO OFRECE NINGUNA GARANTÍA, EXPRESA, IMPLÍCITA O LEGAL, EN CUANTO A LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO.

Microsoft publica este documento bajo los términos de la licencia Creative Commons Attribution 3.0 License. Todos los demás derechos están reservados.

© 2014 Microsoft Corporation.

Microsoft, Active Directory, Excel, Internet Explorer, SQL Server, Visual Studio, and Windows son marcas comerciales del grupo de compañías de Microsoft.

Todas las demás marcas son propiedad de sus respectivos dueños

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, you should not interpret this to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft grants you a license to this document under the terms of the Creative Commons Attribution 3.0 License. All other rights are reserved.

© 2014 Microsoft Corporation.

Microsoft, Active Directory, Excel, Internet Explorer, SQL Server, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Índice

Prólogo.....	4
Introducción	5
Entendiendo NuGet.....	6
Gestión de recursos compartidos con NuGet.....	6
Usando NuGet para la gestión de dependencias.....	11
Referencias adicionales.....	15
Hands-on Lab (HOL) – Usando NuGet.....	16
Ejercicio 1 – Buscar y encontrar un paquete NuGet	16
Ejercicio 2 – Usando la consola Package Manager Console	19
Ejercicio 3 – NuGet Package Manager.....	20
Conclusión	22

Prólogo

Desde la primera versión de la guía de Branching de TFS, han cambiado muchas cosas en el mundo del control de versiones. Las soluciones de control de versiones están por todas partes, y muchas de ellas incluyen integración con builds, seguimiento de proyectos, y otros servicios. Los sistemas de Control de Versiones Distribuidos ya no son un nicho, y han cambiado lo que para muchos desarrolladores significa “tener versionado el código”. Hay muchos más desarrolladores que usan control de versiones que antes – y eso es algo fantástico para miles de millones de usuarios finales de su software.

Más desarrolladores usando control de versiones también significa, ahora más que nunca, que la industria necesita guías sólidas, prácticas y fáciles de entender. Esta guía, y todas las que la precedieron, se esfuerzan en eso – ofrecer una guía sobre el control de versiones que todo equipo de desarrollo necesita para ser efectivo y a la vez accesible y flexible. En esta última edición, hemos racionalizado la orientación y simplificado conceptos con el objetivo de ayudar a equipos de todas las formas y tamaños para que consigan una estrategia que les permitan conseguir la flexibilidad y agilidad que los equipos de desarrollo modernos necesitan.

También quiero decir que esta guía no podría ir por esta versión sin los lectores. Gracias a todos los que habéis contribuido con el feedback y las ideas que nos han ayudado a darle forma a esta guía durante los últimos años. Como en versiones anteriores, si ves algo en esta guía que te gustaría cambiar o mejorar, ¡háznoslo saber!

¡Feliz versionado!

Matthew Mittrik – Program Manager, Cloud Dev Services

Introducción

El objetivo de esta guía es ofrecer una guía práctica sobre la gestión de dependencias y de recursos compartidos, usando NuGet con Visual Studio. Es una de las guías adicionales como se indica en el capítulo “¿Qué hay de nuevo?” en la guía **Control de Versiones con TFS Parte 1 – Estrategias de Branching**.

A quién va dirigida

En esta guía, nos centramos en **Garry**, el jefe de equipo, **Doris**, la desarrolladora, y **Dave**, el administrador del TFS. Leer [ALM Rangers Personas and Customer Profiles](#)¹ para más información sobre estos y otros personajes.

Visual Studio ALM Rangers

Los Visual Studio ALM Rangers ofrecen una guía profesional, experiencia práctica y proporcionar soluciones a la comunidad ALM. Son un grupo especial compuesto por miembros del grupo de producto de Visual Studio, de Microsoft Services, de Microsoft Most Valuable Professionals (MVP) y de Visual Studio Community Leads. La información sobre sus miembros está disponible aquí [online](#)².

Colaboradores

Alan Wills, Anna Galaeva, Howard Dierking, Krithika Sambamoorthy, Stawinski Fabio, Taavi Kõosaar y Tommy Sundling

Un agradecimiento especial a los equipos e ALM Ranger que sentaron las bases con las versiones v1 y v2: Anil Chandr Lingam, Bijan Javidi, Bill Heys, Bob Jacobs, Brian Minisi, Clementino de Mendonca, Daniel Manson, Jahangeer Mohammed, James Pickell, Jansson Lennart, Jelle Druyts, Jens Suessmeyer, Krithika Sambamoorthy, Lennart Jansson, Mathias Olausson, Matt Velloso, Matthew Mitrik, Michael Fourie, Micheal Learned, Neno Loje, Oliver Hilgers, Sin Min Lee, Stefan Mieth, Taavi Koosaar, Tony Whitter, Willy-Peter Schaub, y la comunidad ALM.

Uso del código de ejemplo, erratas y soporte

Todo el código Fuente y la revisión de esta guía están disponible para su descarga a través del sitio [Version Control \(formerly Branching and Merging\) Guide](#)³. (Anteriormente conocida como Branching and Merging Guide). Puedes contactar con el equipo usando el foro de CodePlex.

Más ALM Rangers y otros Recursos

Understanding the ALM Rangers – <http://aka.ms/vsarunderstand>

Visual Studio ALM Ranger Solutions – <http://aka.ms/vsarsolutions>

¹ <http://vsarguidance.codeplex.com/releases/view/88001>

² <http://aka.ms/vsarindex>

³ <http://aka.ms/treasure18>

Entendiendo NuGet

Gestión de recursos compartidos con NuGet

Compartir y gestionar recursos entre proyectos puede llegar a ser complicado. *NuGet* nos ayuda a simplificar este proceso. Seguramente ya habréis usado NuGet en Visual Studio para descargar extensiones de la galería pública de NuGet. Pero debes saber que puedes configurarte un repositorio local para compartir recursos en tu proyecto y en tu compañía. “NuGet es una extensión de Visual Studio que hace realmente fácil el poder instalar y actualizar librerías y herramientas en Visual Studio” (<http://nuget.org>). Viene pre-instalado en Visual Studio 2012 y en versiones posteriores, y puedes instalarlo como extensión en versiones previas. También existe una herramienta “standalone”, NuGet.exe, que podemos usar desde fuera de Visual Studio.

Esto es lo que ofrece NuGet:

- Un paquete NuGet puede instalar recursos compartidos, actualizar referencias de proyectos, instalar extensiones de Visual Studio, y actualizar archivos de proyectos. Nos ofrece todo lo que necesitamos para realizar el desarrollo de un dominio particular.
- Repositorio privado – Puedes crear tu propio repositorio de paquetes con NuGet, tanto a través de archivos compartidos como con un servidor web.
- Repositorio público – También puedes acceder y contribuir a la [galería oficial de NuGet](http://nuget.org) ⁴. Así como a otros repositorios públicos como myget.org
- Comprobar fácilmente qué paquetes instalados están desactualizados
- Proceso sencillo de actualización cuando se actualizan los paquetes compartidos, incluidas las dependencias de esos paquetes
- Herramienta de línea de comandos que podemos ejecutar en el proceso de compilación para descargar las versiones adecuadas de los recursos compartidos

Esta guía está basada en la versión 2.7.1 de NuGet. Esta guía es complementaria a la [documentación de NuGet](http://docs.nuget.org/) ⁵. El objetivo de esta guía es ver cómo NuGet nos ayuda a gestionar recursos y dependencias compartidas cuando trabajamos en proyectos con Visual Studio y Team Foundation Server.

¿Cuándo deberíamos pensar en NuGet para compartir recursos?

- Si el proyecto ya usa NuGet para la gestión de dependencias externas (por ejemplo con librerías como Entity Framework, jQuery, o ASP.NET MVC), podemos simplificar la gestión usando la misma herramienta para las dependencias internas.
- Si varios proyectos comparten recursos NuGet puede ayudar a simplificar la gestión de esos recursos, incluyendo sus dependencias y las versiones.
- Si los recursos compartidos de un proyecto tienen dependencias complejas entre ellos.
- Si hay un equipo (o equipos) separados que crean recursos compartidos para que se usen en otros proyectos.

Como veis, en la mayoría de escenarios que involucran referencias NuGet encaja muy bien. Sin embargo, si estáis trabajando con una solución en la que los proyectos sólo se referencian entre sí, podéis olvidaros de NuGet. Sólo

⁴ <http://nuget.org/>

⁵ <http://docs.nuget.org/>

deberíais usarlo si tenéis planeado compartir esos proyectos más allá de la solución en la que están, ya que NuGet hará más fácil empaquetarlos y compartirlos de una manera más escalable y puede ser muy útil en otras soluciones. No es necesario decidirlo desde el principio, ya que convertir un componente en un paquete NuGet es realmente fácil.

Usando NuGet para compartir recursos

NuGet tiene los siguientes componentes:

- **Archivo de especificación NuGet (.nuspec)** – la especificación para la creación de un paquete NuGet. Define la información común sobre el paquete como el título, descripción, versión, etc. También define las dependencias con otros paquetes NuGet.
- **Paquete NuGet (.nupkg)** – un paquete en el formato NuGet (Open Packaging Conventions basado en Zip) que contiene los recursos compartidos y las librerías. El contenido del paquete está indicado por el [archivo de especificación NuGet](#)⁶.
- **NuGet feed** – Un directorio local, UNC share o un feed NuGet RSS creado con el componente NuGet.Server.
- **NuGet.exe** – una herramienta de línea de comandos.
- **NuGet Package Manager y NuGet Package Manager Console** – una extensión de Visual Studio con la que podemos gestionar los paquetes NuGet en una solución
- **NuGet.config** – [configuración de NuGet por máquina o por solución](#)⁷. Entre otras cosas, podemos definir los repositorios comunes (por ejemplo, los feeds internos de NuGet) para el equipo o la localización de paquetes para una solución concreta.

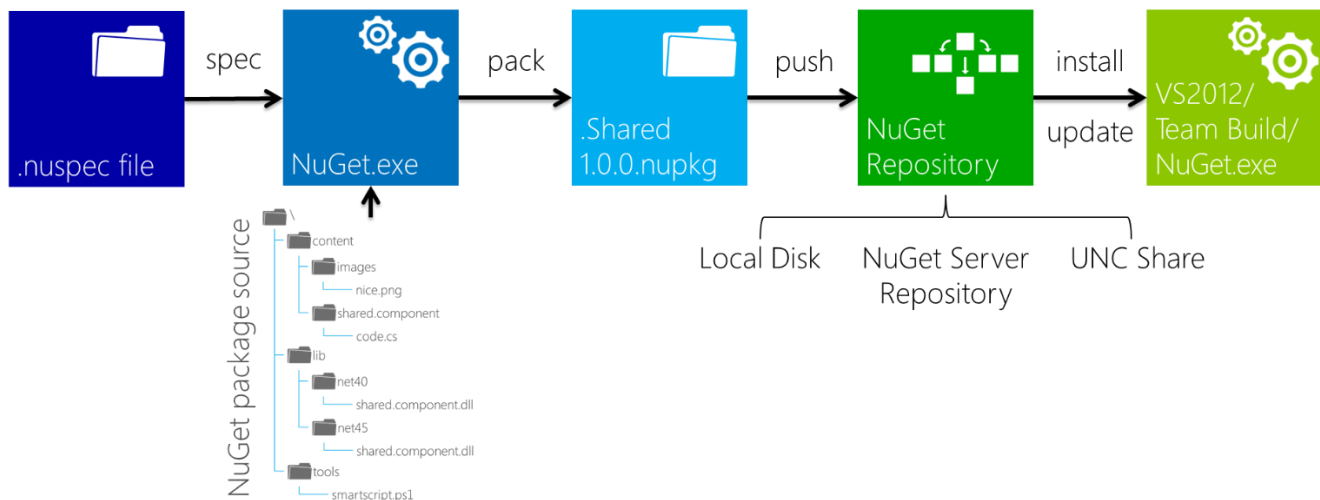


Figura 1 – El flujo general con NuGet

⁶ <http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#From a convention based working directory>

⁷ <http://docs.nuget.org/docs/release-notes/nuget-2.1#Hierarchical NuGet.config>

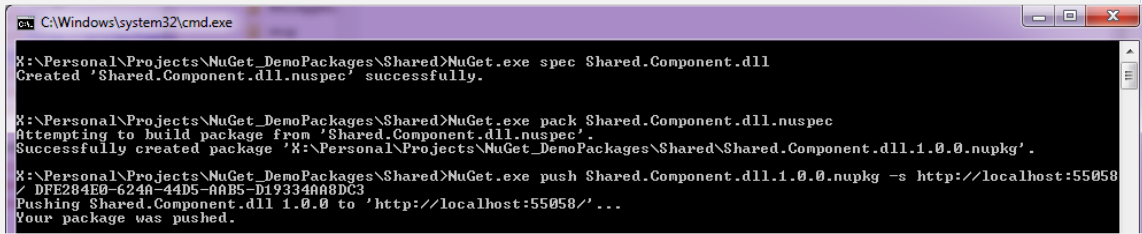
Creando un paquete NuGet

Paso	Instrucciones
1 Crear la estructura del paquete ☐ - Hecho	<p>Primero, crearemos un directorio para la estructura del paquete. Tenemos que seguir la estructura documentada de un paquete NuGet⁸. La estructura que vemos más abajo soporta tres tipos de recursos – assemblies (dll's), herramientas (scripts de PowerShell/programas que se puedan ejecutar desde el Package Manager Console), contenido (diferentes archivos que formarán parte del proyecto de destino en la instalación como imágenes, archivos de c#, archivos web, css) y compilación (archivos target de MSBuild que se incluyen en el proyecto). Nota: ten en cuenta que también podemos pasar el archivo de proyecto a un comando de paquete para ayudar a generar el nuspec basado en los datos del assembly automáticamente a través de la línea de comandos (Id al paso 3 más adelante)</p> <p>NuGet package structure</p> <ul style="list-style-type: none"> • \ • \lib\net40\Shared.Component.dll • \lib\net45\Shared.Component.dll • \tools\SmartScript.ps1 • \content\Images\nice.png • \content\Shared.Component\Code.cs <p>El directorio lib puede contener subdirectorios para versiones específicas del framework (como net40, net45). Cuando alguien instale vuestro paquete NuGet, el assembly NuGet usará las dependencias para enlazar con la versión de tu proyecto. Todas las versiones usarán las librerías localizadas en el directorio lib. En NuGet v2.7.1 y posteriores los directorios de contenido y de herramientas también pueden contener directorios para una versión concreta del framework.</p> <p>Crear esta estructura de directorios puede ser un paso automático de la compilación.</p>
2 Crea la especificación ☐ - Hecho	<p>Crea una especificación de paquete NuGet⁹. Podemos usar Nuget.exe para hacer esto (aunque hay otras formas). Digamos que queremos crear un paquete que sólo contenga Shared.Component.dll. Ejecutemos:</p> <pre>nuget.exe spec "Shared.Component.dll"</pre> <p>Que genera el archivo xml Shared.Component.dll.nuspec. Necesita algunas modificaciones para incluir los metadatos adecuados sobre el paquete. Podemos encontrar los detalles del archivo .nuspec en la documentación de NuGet¹⁰.</p>

⁸ http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#From_a_convention_based_working_directory

⁹ http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#Create_the_manifest

¹⁰ <http://docs.nuget.org/docs/reference/nuspec-reference>

Paso	Instrucciones
	<p>También hay un Explorador visual de paquetes¹¹ para ayudarnos en la creación y modificación de paquetes NuGet. En la siguiente imagen vemos un ejemplo de especificación para el componente "Shared.Component.dll".</p> <pre> 1 <?xml version="1.0"?> 2 <package > 3 <metadata> 4 <id>Shared.Component.dll</id> 5 <version>1.0.0</version> 6 <authors>ALM Ranger</authors> 7 <owners>ALM Ranger</owners> 8 <licenseUrl>http://licenseserver/view</licenseUrl> 9 <projectUrl>http://projectserver/view</projectUrl> 10 <iconUrl>http://coolicon/Shared.Component.gif</iconUrl> 11 <requireLicenseAcceptance>false</requireLicenseAcceptance> 12 <description>This is an internal shared component.</description> 13 <releaseNotes>Added something new. Removed something old. Left a few behind.</releaseNotes> 14 <copyright>Copyright 2012</copyright> 15 <tags>Net40 CSharp Common</tags> 16 <dependencies> 17 <dependency id="Shared.Core.dll" version="[1.0.2,3.0.5]" /> 18 </dependencies> 19 </metadata> 20 </package> </pre> <p>La especificación define la versión del componente y todas las dependencias que puede tener con otros paquetes NuGet. En el ejemplo, hay una dependencia con la versión 1.0.2 de Shared.Core.dll hasta la versión 3.0.5 (ambas inclusive). Podemos encontrar más información sobre el versionado en la documentación de NuGet¹².</p> <p>También podemos crear la especificación basándonos en un proyecto de Visual Studio usando el comando "spec" en el directorio del proyecto.</p>
3 Crea el paquete ☐ - Hecho	<p>Genera el paquete NuGet:</p> <pre>nuget.exe pack "Shared.Component.dll.nuspec"</pre> <p>Podemos saltarnos los tres primeros pasos usando el archivo de proyecto como entrada para el comando "pack" para generar la estructura, el archivo de especificación NuGet, y el paquete resultante se basará en la información del assembly y en los metadatos del proyecto.</p>
4 Copia, mueve o publica el paquete ☐ - Hecho	<p>Mueve o publica el paquete NuGet¹³ a un repositorio NuGet (feed).</p> <p>Esta captura de pantalla muestra el proceso completo.</p>  <p>La documentación de NuGet ofrece más información detallada sobre las transformaciones¹⁴ y los scripts de PowerShell automáticos que se ejecutan durante la instalación y desinstalación del paquete¹⁵.</p>

¹¹ <http://docs.nuget.org/docs/creating-packages/using-a-gui-to-build-packages>

¹² <http://docs.nuget.org/docs/reference/versioning>

¹³ <http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package>

¹⁴ <http://docs.nuget.org/docs/creating-packages/configuration-file-and-source-code-transformations>

¹⁵ http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#Automatically_Running_PowerShell_Scripts_During_Package_Installation_and_Removal

Obtener una actualización de un paquete NuGet

NuGet permite la actualización de paquetes tanto a nivel de proyecto como a nivel de solución. Sin embargo, deberíamos hacer algunas comprobaciones para comprobar que la actualización de un paquete no causará problemas inesperados en la solución o proyecto. Una aproximación es usar el “what if” de la consola de PowerShell para ver las dependencias que se actualizarán sin hacer la operación.

Recomendamos:

- Actualizar los paquetes a nivel de solución para mantener la misma versión en todos los proyectos de la solución.

No recomendamos:

- Actualizar todos los paquetes de la solución a la vez, a menos que hayamos hecho las comprobaciones para todos los paquetes.
- Tener diferentes versiones de un mismo paquete NuGet para diferentes proyectos en la misma solución. Puede ocasionar problemas que resultarían difíciles de tracear. En casos excepcionales, tener diferentes versiones puede ser necesario, pero debería ser sólo temporalmente. Sin embargo, si los paquetes son cosas como scripts en aplicaciones web como jQuery, puede estar bien en tu escenario.

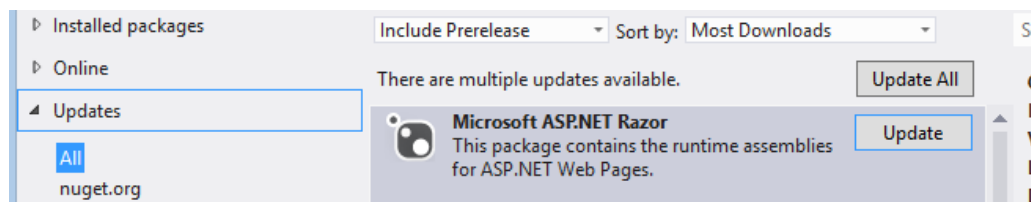


Figura 23 – Actualizaciones disponibles desde la ventana Manage NuGet Packages

Antes de actualizar, comprueba las notas de release del paquete NuGet. ¿Contiene cambios funcionales que pueden romper tu código? Si tiene nuevas dependencias ¿cómo afectarán a tu proyecto?

A partir de NuGet v2.8 es posible usar el script estándar de PowerShell [-whatif switch](#)¹⁶ en una consola de PowerShell con los comandos `install-package`, `uninstall-package` y `update-package` que nos dará información necesaria para saber si debemos aplicar el cambio.

Las convenciones del [versionado semántico](#)¹⁷ nos ofrecen una guía para ver el impacto que un cambio tendrá en nuestro proyecto. Básicamente, en el número de versión {Major}. {Minor}. {Patch}:

- Las actualizaciones Major indican que puede haber cambios incompatibles de la API
- Las actualizaciones Minor indican que contienen cambios compatibles con versiones anteriores
- Las actualizaciones Patch indican que sólo contienen correcciones de bugs

Creando un feed de NuGet

Un repositorio NuGet es un feed que las herramientas NuGet Package Manager, NuGet Package Console, NuGet.exe, y otras herramientas de NuGet pueden consumir. El feed es un repositorio centralizado para acceder paquetes NuGet que hayamos incluido en el repositorio. Hay tres tipos de feeds de NuGet, pero sólo uno de ellos soporta las capacidades de un feed rss.

- **Directorio local** — la forma más sencilla de crear un feed es crear un directorio local en disco y copiar el archivo .nupkg en él. Podemos configurar esta localización en Visual Studio como el lugar donde estarán los paquetes NuGet que nuestros proyectos consumirán.

¹⁶ http://docs.nuget.org/docs/release-notes/nuget-2.8#Preview_NuGet_Operations_With_-whatif

¹⁷ <http://semver.org/>

- **Directorio compartido**— Usar un directorio compartido como UNC. Copia los nuevos paquetes en el directorio compartido, y ya estarán disponibles. De nuevo, es tan sencillo como configurar Visual Studio. La figura 3 muestra la UI desde la cual podemos añadir nuevos feeds.
- **Servidor NuGet** – Con nuestra propia aplicación web NuGet, podemos obtener notificaciones de actualizaciones y otras ventajas. Hay un paquete NuGet que te puede ayudar. Podéis encontrar [información más detallada en la web de documentación de NuGet](http://docs.nuget.org/docs/creating-packages/hosting-your-own-nuget-feeds)¹⁸. También es posible [clonar toda la galería de nuget.org](https://github.com/NuGet/NuGetGallery/blob/master/README.markdown)¹⁹.

Después de haber configurado alguno de estos repositorios, podemos configurar nuestras instalaciones de Visual Studio para que las usen abriendo Options, Package Manager, Package Sources. Por defecto, Visual Studio 2013 lee el feed público de NuGet.org.

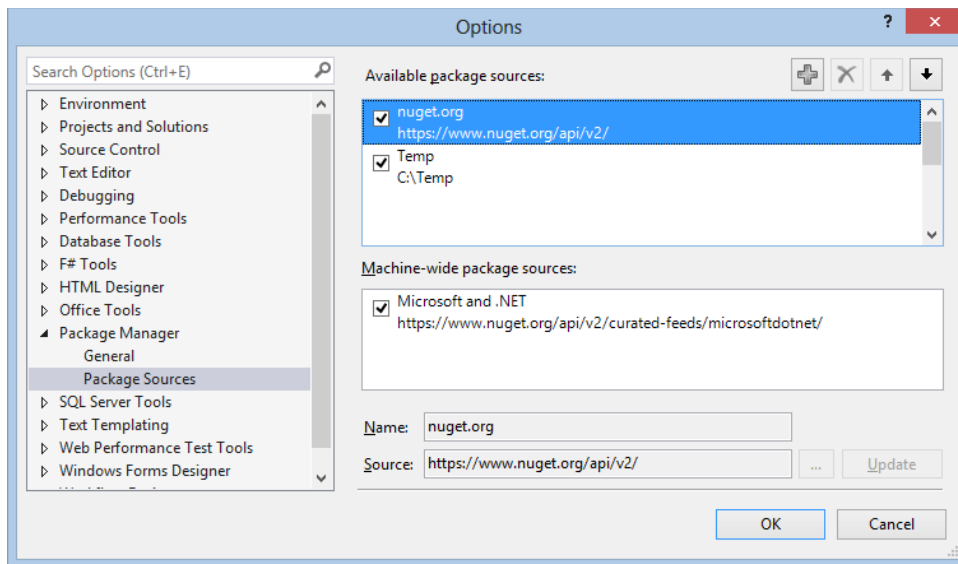


Figura 3 – Opciones de configuración de NuGet Package Manager en Visual Studio 2013

Usando NuGet para la gestión de dependencias

NuGet hace muy fácil compartir recursos. NuGet lleva ya unos años dando vueltas por ahí. Está constantemente evolucionando, madurando, y [obteniendo más y más experiencia](http://stats.nuget.org/)²⁰ en la comunidad de desarrollo de software.

Localización de los recursos compartidos en el control de versiones

¿Deberíamos incluir los paquetes de NuGet en nuestro control de versiones?

Para los fuentes del paquete, la respuesta es obviamente sí – así como harías para el código de cualquier assembly.

En los proyectos que usen esos paquetes hay dos opciones:

- No añadir los paquetes NuGet al control de versiones. En su lugar, usaremos el patrón de NuGet Package Restore, que veremos en el próximo capítulo.
- Añadir los paquetes NuGet al control de versiones con nuestro código. Configurar NuGet para que se descargue una versión de cada paquete y lo mantenga en la localización que podemos compartir con el resto del team Project.

¹⁸ <http://docs.nuget.org/docs/creating-packages/hosting-your-own-nuget-feeds>

¹⁹ <https://github.com/NuGet/NuGetGallery/blob/master/README.markdown>

²⁰ <http://stats.nuget.org/>

Tipos de recursos que podemos compartir con NuGet

Los paquetes NuGet pueden [contener cuatro tipos de recursos](#)²¹ – assemblies, herramientas, contenido (como imágenes, css, JavaScript, vistas de Razor, archivos C#) y compilaciones (archivos targets de MSBuild que podemos incluir en el proyecto). Es posible crear paquetes que contengan sólo assemblies o sólo archivos o ambos. Además podemos estructurar [cada assembly o contenido para que NuGet lo instale sólo en versiones específicas del .Net Framework](#)²². Otro aspecto importante a recordar es que los paquetes NuGet pueden contener scripts de instalación/desinstalación y scripts de cambio de configuraciones por paquete incluso por versión del framework.

Patrón de gestión de dependencias con feeds centrales de NuGet

El patrón principal para compartir recursos y gestionar dependencias con NuGet es usar feeds internos y externos de NuGet. Por defecto, NuGet conoce el feed de la galería pública de NuGet, donde podemos encontrar los paquetes públicos de NHibernate, jQuery, Entity Framework, ASP.NET MVC. Usando las herramientas de NuGet (como el Package Manager) podemos instalar nuevos paquetes NuGet al proyecto o actualizar los que ya tiene. La extensión de NuGet para Visual Studio nos notificará cuando haya versiones nuevas de los paquetes que estén instalados. Cuando instalamos un paquete NuGet en nuestro proyecto, NuGet instalará automáticamente todas sus dependencias.

Este patrón tiene tres hilos:

- **Ofrecer un feed NuGet.** Puede ser tanto un directorio local, un directorio compartido o un repositorio real de un Servidor de NuGet. Para tener un feed que esté disponible para todo el equipo usaremos un directorio compartido o un servidor de NuGet. Es muy fácil publicar un paquete en un directorio compartido ya que sólo hay que copiarlo al directorio. Crear un servidor de NuGet requiere un pequeño trabajo de desarrollo (o al menos hacer un clon de la galería de nuget.org), y usar la herramienta de NuGet para publicar paquetes. La principal ventaja de usar un servidor de NuGet es que puede notificar a los usuarios cuando aparezcan nuevas versiones y escalará mejor para buscar y navegar, a medida que se vayan incluyendo paquetes al repositorio ya que los metadatos del paquete están disponibles.
- **Calcular el feed** con paquetes NuGet. Usando la herramienta de línea de comandos nuget.exe, podemos automatizar la creación y publicación de nuevos paquetes con la ayuda de MSBuild o Team Foundation Build.
- **Escuchar el feed.** En caso de que planeemos guardar los paquetes en nuestro control de versiones, cada proyecto que use el paquete, sólo un miembro debería ser el responsable suscribirse al feed, instalar el paquete y actualizar las futuras versiones. Ese usuario debería instalar el paquete en su propio entorno, y subirlo al control de versiones, como si fuese un cambio más. Los demás miembros del equipo obtendría el paquete a través del sistema de control de versiones – no tienen que suscribirse a ningún feed. Este método asegura que todos los miembros del equipo están usando la misma versión de los assemblies u otros recursos del paquete. Además, puedes poner los permisos necesarios para prevenir que otros miembros del equipo actualicen el paquete.

También puedes decidir que los paquetes no se suban al sistema de control de código y usar el “Patrón de gestión de dependencias que se usa en el Package Restore.” En este escenario todos los entornos de desarrollo accederán al feed y se actualizarán los paquetes que falten cuando sea necesario. Manteniendo las dependencias, esto significa no tener binarios en el control de versiones que aporta varias ventajas.

Podemos configurar Visual Studio para que trabaje con más de un repositorio NuGet a la vez. Podemos añadir paquetes tanto de repositorios internos (visibles sólo dentro de la compañía) como externos. Un escenario podría ser una organización que quiere “aprobar” los paquetes de un feed público e incluirlos en el feed privado

²¹ http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#From_a_convention_based_working_directory

²² http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#Supporting_Multiple_.NET_Framework_Versions_and_Profiles

de manera que los desarrolladores sólo trabajen con los paquetes aprobados. En tu repositorio interno, puedes incluir paquetes de componentes internos que dependen de paquetes de cualquier repositorio (incluso de feeds externos de NuGet) y NuGet instalará automáticamente esas dependencias. Además cuando busquemos un paquete, NuGet lo buscará en cada repositorio. Si permites la restauración de paquetes, NuGet se comportará de la misma forma en la búsqueda secuencial en cada repositorio para obtener los paquetes dependientes.

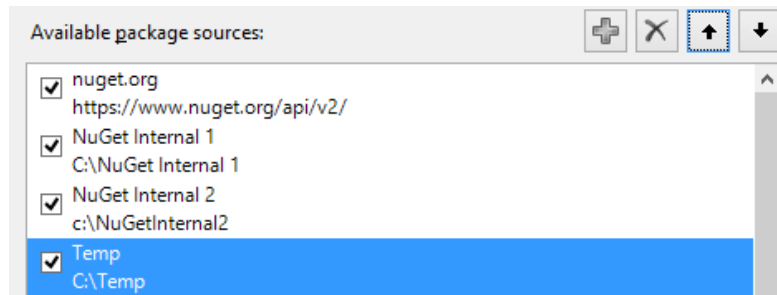


Figura 47 – Lista de fuentes de paquetes disponibles

La localización de los paquetes NuGet descargados

Por defecto, NuGet descarga todos los paquetes en el directorio de la solución. Sin embargo, dependiendo de la estructura de control de versiones y los requisitos del proyecto, podremos querer guardar los paquetes en un sitio diferente. Podemos modificar el directorio de descarga por defecto (y otras cosas) modificando el archivo [nuget.config](#)²³ (Figura 5) y localizarlo en otro sitio dentro de la estructura de directorios.

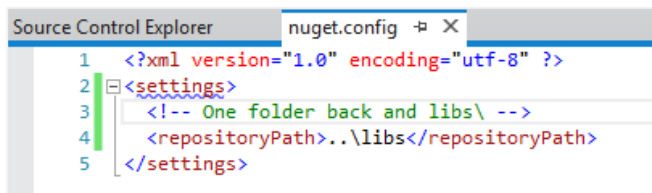


Figura 5 – Archivo de configuración Nuget.config con un directorio de repositorio diferente

Cuando añadimos un paquete al proyecto, NuGet lo incluirá en el directorio con el siguiente patrón de nombres "PackageName.Version."

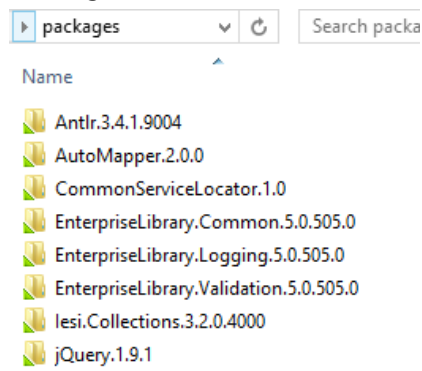


Figura 69 – Directorio de "Paquetes" y su contenido

Con este patrón, no añadiremos los paquetes al control de código. En su lugar, cada entorno de desarrollo descargará los paquetes del repositorio según lo que hayamos definido en el packages.config por proyecto en la

²³ http://docs.nuget.org/docs/reference/nuget-config-file#NuGet_config_extensibility_point

solución. A partir de la versión v2.7 de NuGet se habilitó la opción Package Restore en Visual Studio por defecto. Por defecto, se descargarán los paquetes NuGet que falten (Options -> Package Manager – Figura 7)

[Automatic Package Restore](#)²⁴ restaurará los paquetes que falten antes de compilar cualquier proyecto con MSBuild. De esta manera nos aseguramos de que NuGet h descargado todos los elementos necesarios (incluyendo las propiedades y targets que se han incluido en el paquete NuGet) y los coloca en su posición correcta. NuGet hace esto usando los eventos de compilación de Visual Studio

Para Team Build 2013 y Visual Studio Online por defecto funciona así, sin embargo, en versiones anteriores de [Team Build es necesario hacer una pequeña modificación para integrar el uso de la línea de comandos del Package Restore de NuGet](#)²⁵.

Fíjate que si has habilitado la el Package Restore de NuGet a nivel de solución antes de la versión 2.7, recomendamos que migréis a la última versión del ["Automatic Package Restore"](#)²⁶.

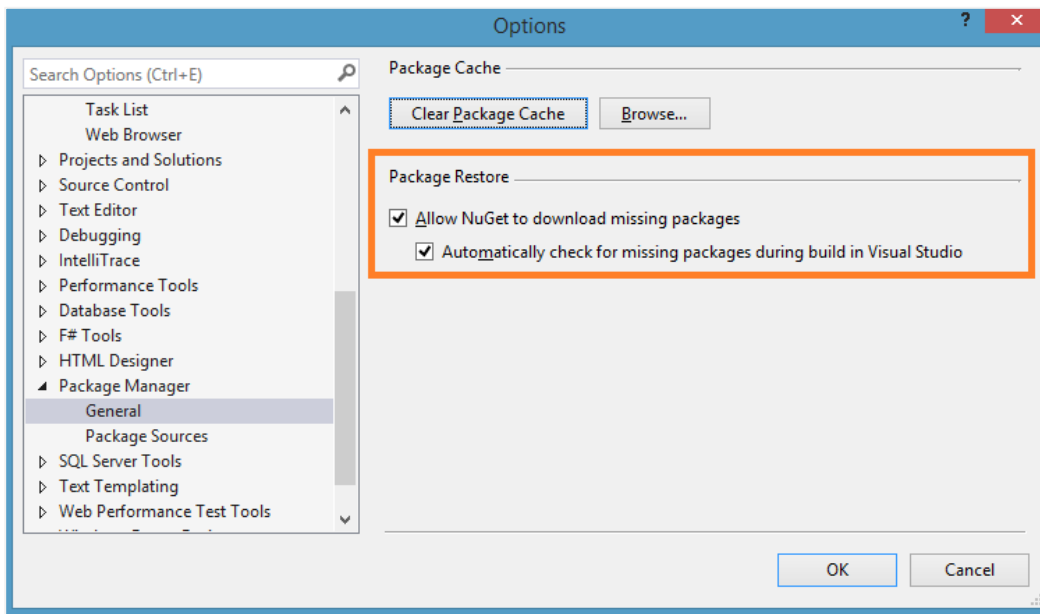


Figura 7. Opciones por defecto de Visual Studio Package Manager Package Restore

Obtener paquetes de NuGet con package restore

Añadir y actualizar paquetes con el gestor de paquetes de NuGet de la misma forma que hicimos antes.

No subas los binarios al control de versiones.

Como antes, sólo un miembro del equipo debe mantener la configuración de paquetes del equipo, y subirlos al control de versiones (y querrás restringir los permisos de escritura al archivo). Cuando cualquier miembro del equipo pulse F5, la primera vez, NuGet se descargará automáticamente los paquetes. Esto ocurrirá también en el servidor de builds.

²⁴ <http://docs.nuget.org/docs/reference/package-restore>

²⁵ <http://docs.nuget.org/docs/reference/package-restore-with-team-build>

²⁶ <http://docs.nuget.org/docs/workflows/migrating-to-automatic-package-restore>

Pros y Contras del package restore de NuGet

Como cualquier otra funcionalidad, siempre hay ventajas y desventajas, y lo mismo ocurre con el package restore de NuGet²⁷. Ten en cuenta que aunque el package restore de NuGet depende de una conexión de red, NuGet puede basarse en escenarios con cache de paquetes para trabajar offline.

Usando NuGet Package restore

Ventajas

- Usarás menos espacio en TFS Source Control ya que no tendrás los binarios en el repositorio.
- Menos datos en TFS son menos datos para hacer backups en escenarios con TFS DR
- Las operaciones de Get estarán optimizadas ya que estamos consultando menos datos

Desventajas

- Los desarrolladores se pueden bloquear si alguno de los repositorios de NuGet no está disponible para obtener un paquete.
- TFS Build fallará o se pospondrá si un repositorio no está disponible (especialmente cuando se limpia el espacio de trabajo antes de compilar) si no hay una versión cacheada disponible.

Referencias adicionales

- [How to use NuGet with TFS Version Control](#) ²⁸
- [Introducing NuGet Package Management for .NET](#) ²⁹
- [Manage Project Libraries with NuGet](#) ³⁰
- [NuGet Documentation](#) ³¹
- [NuGet FAQ](#) ³²
- [NuGet Gallery](#) ³³
- [NuGet In Depth: Empowering Open Source on the .NET Platform](#) ³⁴
- [NuGet Videos](#) ³⁵
- [Packages in Visual Studio Templates](#) ³⁶
- [NuGet Package Restore with Team Foundation Build](#) ³⁷

²⁷ <http://docs.nuget.org/docs/workflows/using-nuget-without-committing-packages>

²⁸ <http://loicbaumann.fr/en/2012/02/07/how-to-use-nuget-with-the-tfs-version-control/>

²⁹ <http://www.hanselman.com/blog/IntroducingNuGetPackageManagementForNETAnotherPieceOfTheWebStack.aspx>

³⁰ <http://msdn.microsoft.com/en-us/magazine/hh547106.aspx>

³¹ <http://docs.nuget.org/>

³² <http://docs.nuget.org/docs/start-here/nuget-faq>

³³ <http://www.nuget.org/>

³⁴ <http://channel9.msdn.com/Events/MIX/MIX11/FRM0>

³⁵ <http://docs.nuget.org/docs/start-here/videos>

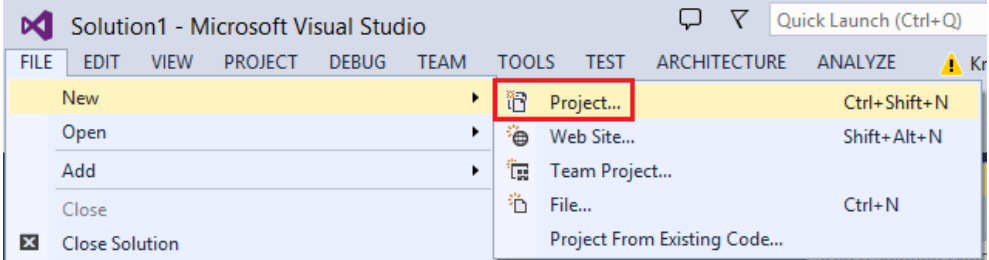
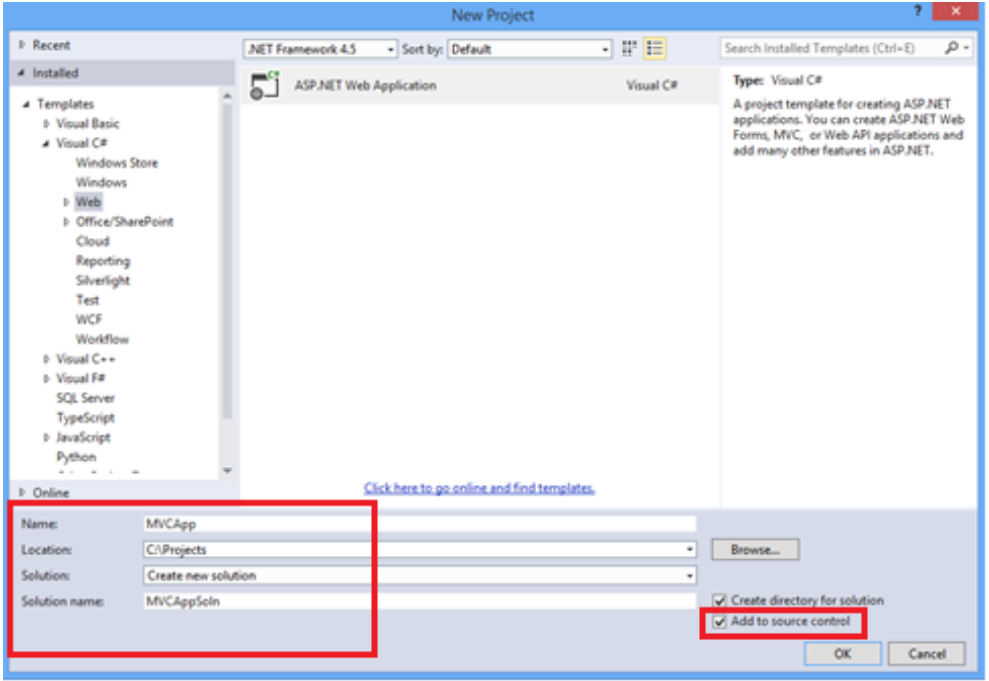
³⁶ <http://docs.nuget.org/docs/reference/packages-in-visual-studio-templates>

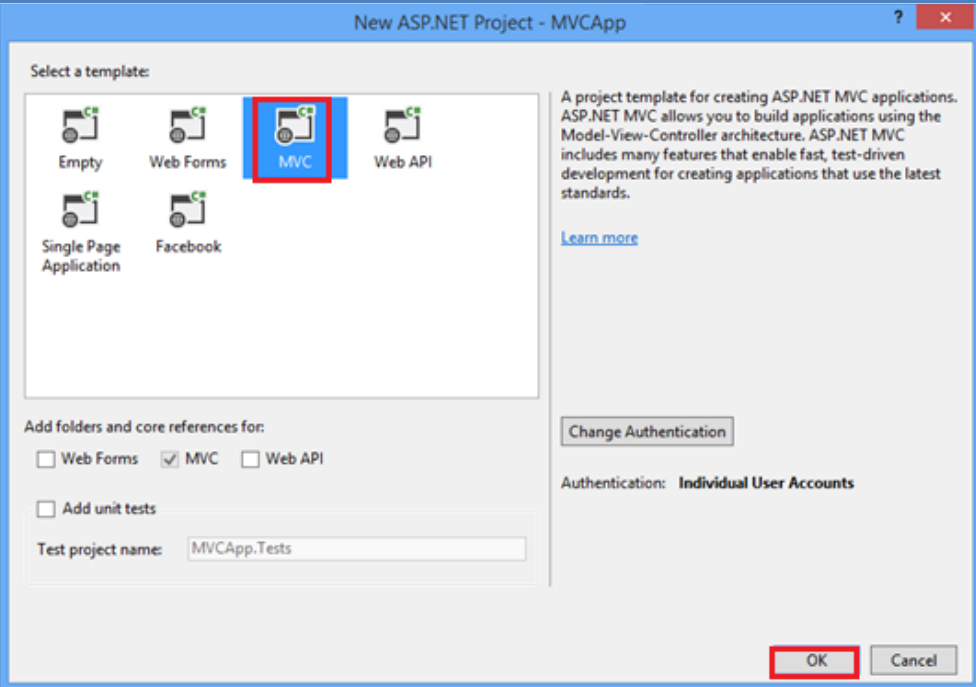
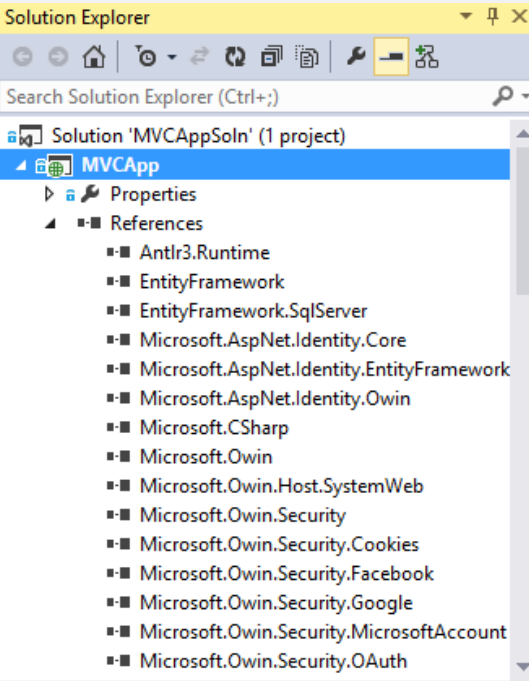
³⁷ <http://docs.nuget.org/docs/reference/package-restore-with-team-build>

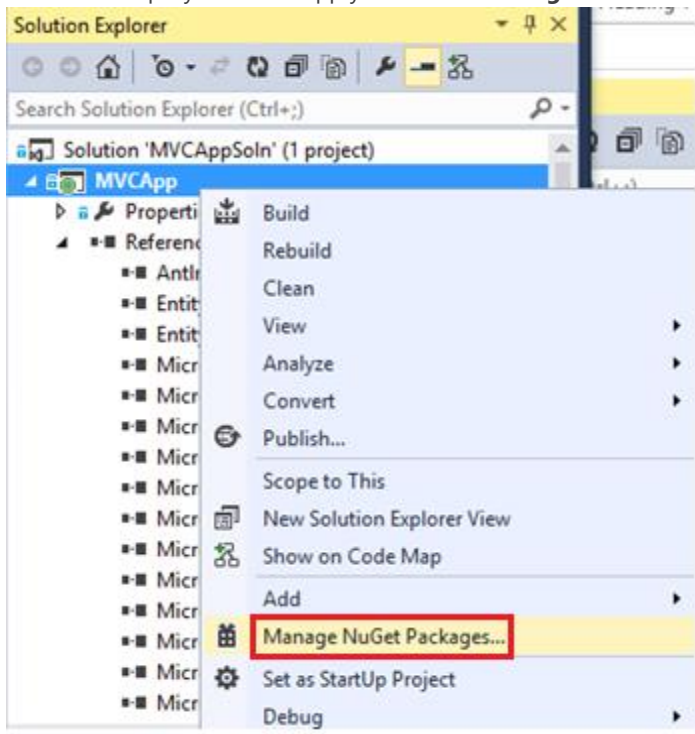
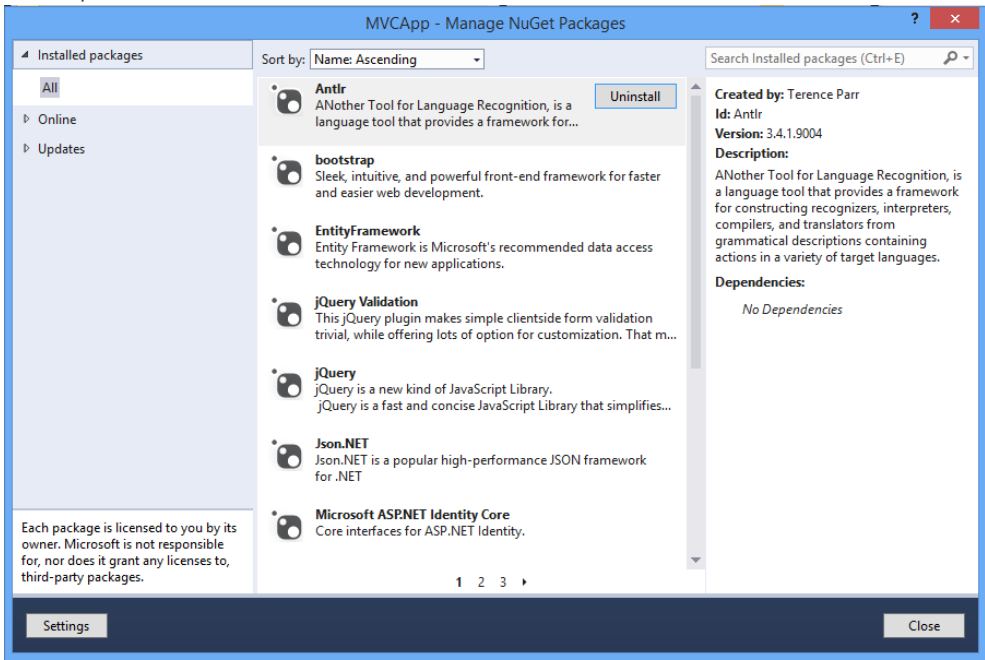
Hands-on Lab (HOL) – Usando NuGet

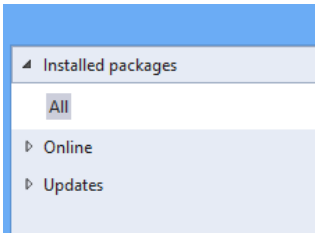
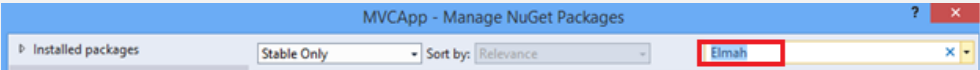
Ejercicio 1 – Buscar y encontrar un paquete NuGet

NuGet es una extensión de Visual Studio que hace muy fácil instalar y actualizar librerías y herramientas en Visual Studio.

Paso	Instrucciones
1 Crea una solución de ejemplo □ - Hecho	<ul style="list-style-type: none">Abre Visual Studio 2013.Selecciona File New Project.  <ul style="list-style-type: none">Selecciona ASP.NET Web Application de la plantilla Web. Asegúrate de marcar Add to source control para añadir la solución a TFS.  <ul style="list-style-type: none">Selecciona la plantilla MVC del diálogo New ASP.NET Project. Haz clic en OK.

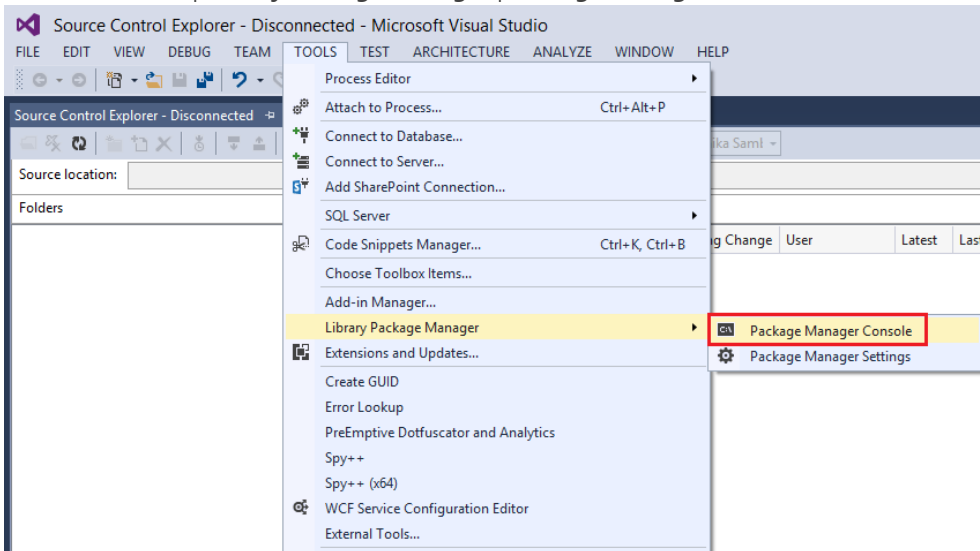
Paso	Instrucciones
	<div data-bbox="402 195 1372 877"></div> <ul style="list-style-type: none">• Haz checkin de los cambios.
2 Añade referencias ☐ - Hecho	<ul style="list-style-type: none">• Abre la solución que acabas de crear en el Solution Explorer.• En el Solution Explorer, expande References.• Como parte de la plantilla MVC se han incluido varios paquetes de NuGet a la solución como el EntityFramework. <div data-bbox="402 1077 928 1753"></div>

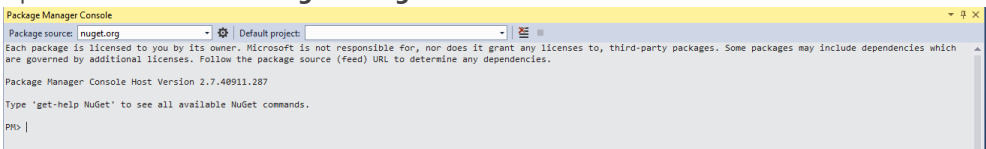
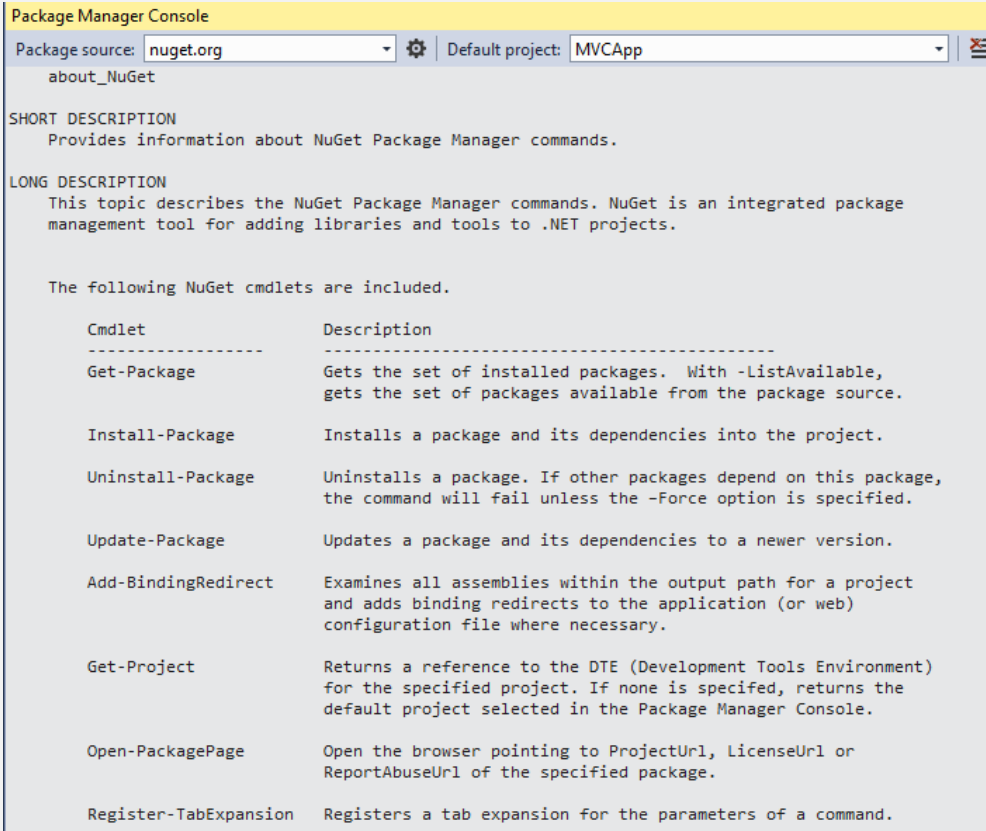
Paso	Instrucciones
3 Gestiona los paquetes NuGet ☐ - Hecho	<ul style="list-style-type: none"> Haz clic derecho en el proyecto MVCApp y selecciona Manage NuGet Packages.  Aparecerá el diálogo Manage NuGet Packages mostrando los paquetes instalados como parte del proyecto MVCApp. También podemos hacer clic derecho en MVCAppSoln y gestionar los paquetes NuGet para toda la solución.  Podemos usar el diálogo Manage NuGet Packages para desinstalar paquetes también. En la parte izquierda del diálogo Manage NuGet Packages, hay tres secciones. <ul style="list-style-type: none"> Installed packages – Muestra los paquetes instalados disponibles en nuestro proyecto o solución.

Paso	Instrucciones
	<ul style="list-style-type: none"> Online – EnNuGet.org o en nuestros feeds privados (si hemos configurado alguno). Podemos buscar por algún paquete NuGet e instalarlo en esta sección. Updates – Muestra una lista de todos los paquetes instalados que tengan alguna actualización. 
4 Añade Elmah ☐ - Hecho	<ul style="list-style-type: none"> Expande Online del panel izquierdo y selecciona nuget.org. usa el campo de búsqueda de arriba a la derecha para buscar Elmah.  <ul style="list-style-type: none"> Elmah (Error Logging Modules and Handlers) es un framework de logging para ASP.NET. Instala el paquete ELMAH haciendo clic en el botón Install. Además, podemos usar la consola Package Manager Console para instalar paquetes NuGet. Vuelve a References para comprobar que hemos instalado Elmah. El framework Elmah ya está listo para usarse.

Ejercicio 2 – Usando la consola Package Manager Console

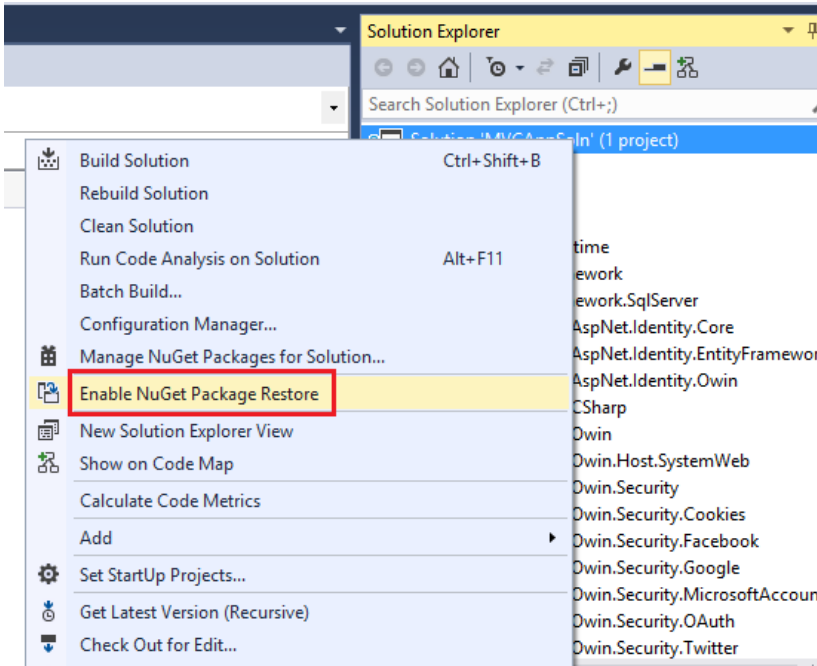
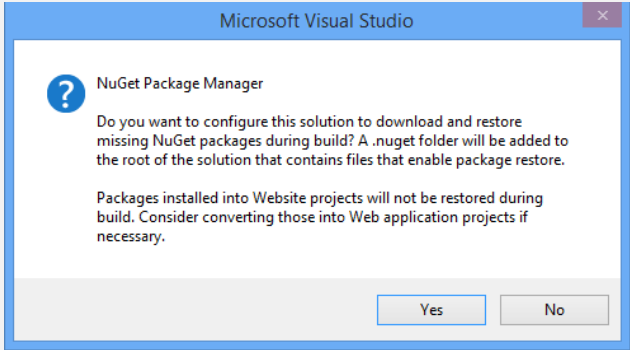
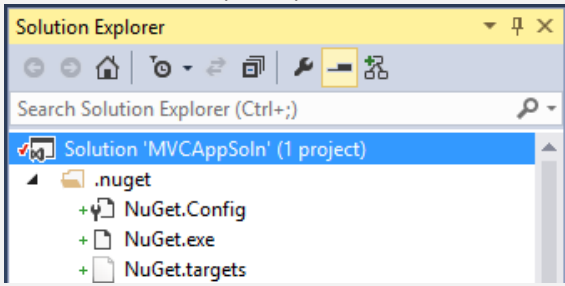
Package Manager Console es una ventana de PowerShell en Visual Studio. Los comandos de PowerShell nos permiten instalar paquetes cuando no tenemos la solución abierta. También necesitaremos usarlo para paquetes que crean comandos que requieren PowerShell.

Paso	Instrucciones
1 Abre Package Manager Console ☐ - Hecho	<ul style="list-style-type: none"> Abre Visual Studio 2013 Haz clic en Tools Library Package Manager Package Manager Console 

Paso	Instrucciones
	<ul style="list-style-type: none"> • Aparecerá la ventana Package Manager Console. 
2 NuGet PowerShell cmdlets <input type="checkbox"/> - Hecho	<ul style="list-style-type: none"> • Package Manager Console es una ventana de PowerShell. Podemos usar comandos de NuGet específicos para PowerShell para realizar varias operaciones. • Para ver una lista de los comandos NuGet disponibles, escribe <i>Get-Help NuGet</i> en la Package Manager Console  <ul style="list-style-type: none"> • Escribe <i>Get-Package</i> para ver la lista de paquetes instalados. • Escribe <i>Uninstall-Package Elmah -Force</i> para desinstalar Elmah y sus dependencias. • Comprueba que NuGet ha desinstalado Elmah y sus dependencias.

Ejercicio 3 – NuGet Package Manager

NuGet es capaz de restaurar paquetes durante la compilación descargándose todos los paquetes y configurarlos, pero no los encontrarás en el disco duro. Esto no cambia el hecho de que NuGet necesita instalar los paquetes del proyecto e instala las actualizaciones de la misma manera. NuGet habilita el Package Restore a nivel de solución.

Paso	Instrucciones
1 Habilita NuGet Package Restore - Hecho	<ul style="list-style-type: none"> Haz clic derecho en MVCAppSoln y selecciona Enable NuGet Package Restore. 
2 Restaura los paquetes NuGet que faltan - Hecho	<ul style="list-style-type: none"> Haz clic en Yes.  <ul style="list-style-type: none"> Vuelve al Solution Explorer para ver el nuevo directorio .NuGet que se ha añadido. 

Conclusión

Aquí termina nuestra aventura usando NuGet. Hemos visto la teoría básica de NuGet, gestionado recursos compartidos, gestionado dependencias, y concluido con un HOL práctico.

¡Esperamos que esta guía te haya resultado útil!

Atentamente

The Microsoft Visual Studio ALM Rangers

