Copyright © Todos los Derechos Reservados - Cibertec Perú SAC



|  |  |  |
| --- | --- | --- |
| **Programa** | **:** | Visual Studio 2015 Web Developer – C# |
| **Curso** | **:** | Visual Studio 2015 Web Developer – C# |
| **Duración** | **:** | 60 minutos |

# OBJETIVO

Desarrollar una aplicación básica con Visual Studio.

# REQUISITOS

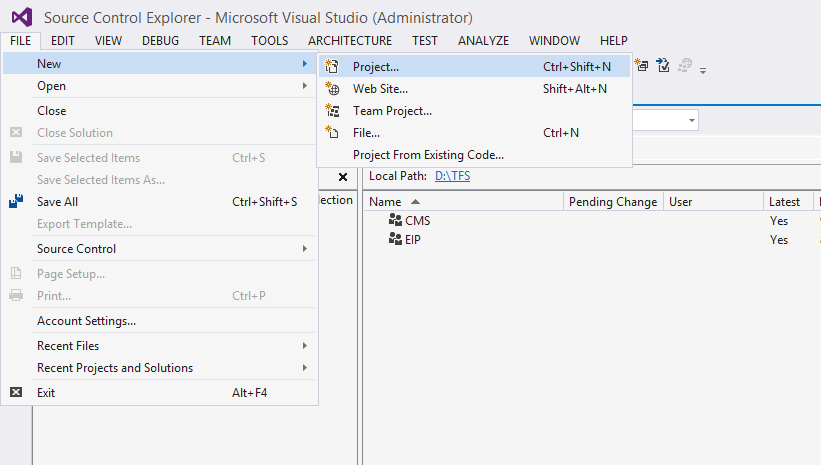
Los siguientes elementos de software son necesarios para la realización del laboratorio:

* Windows 10 (como mínimo Windows 8)
* Visual Studio 2015 (como mínimo Visual Studio 2013)

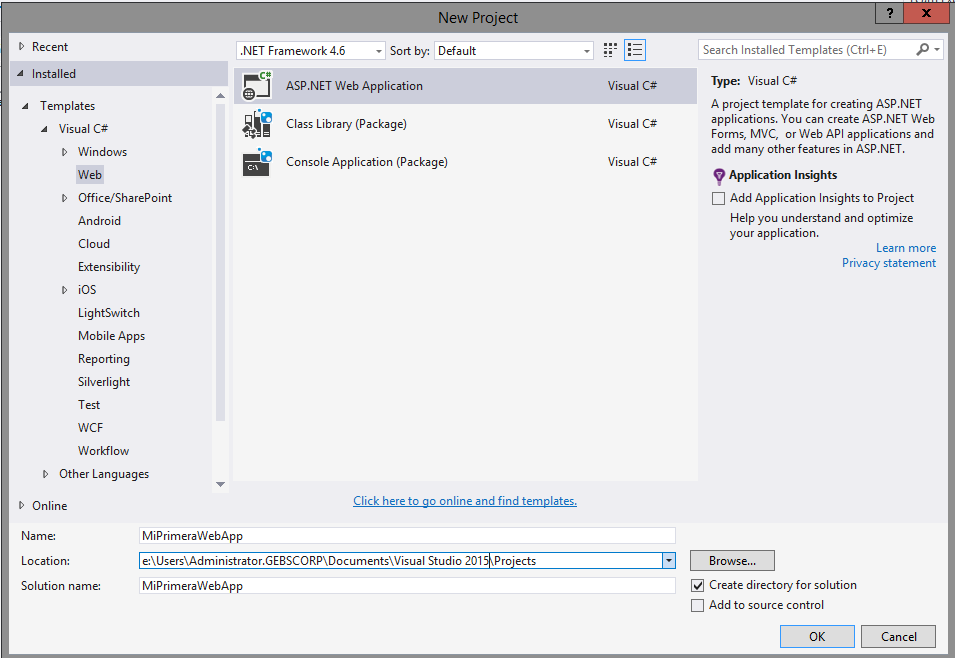
# EJECUCIÓN DEL LABORATORIO

* **Ejercicio 1.1: Crear una aplicación Asp.NET MVC usando la plantilla de Visual Studio 2015.**

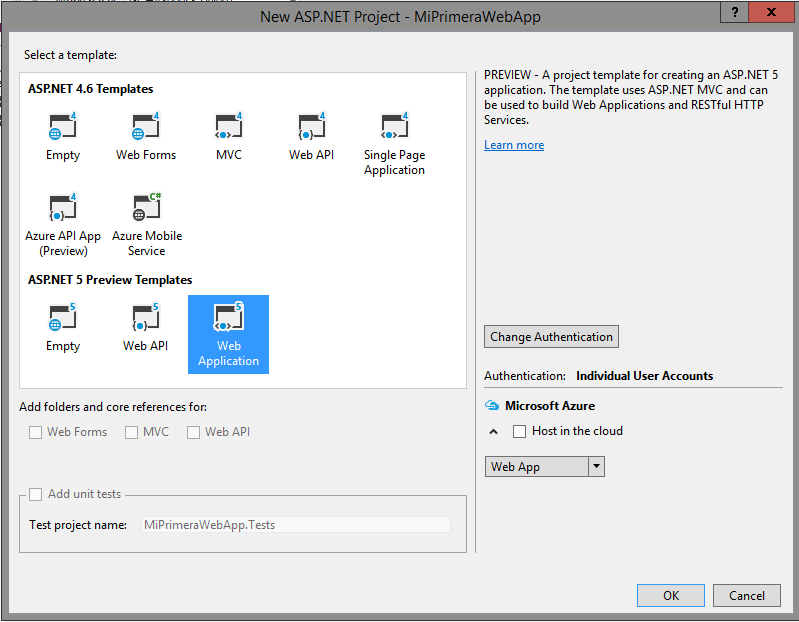
1. Ingresar a Visual Studio 2015.
2. Abrir la ventana de **New Project** (Menú **File > New > Project...**).



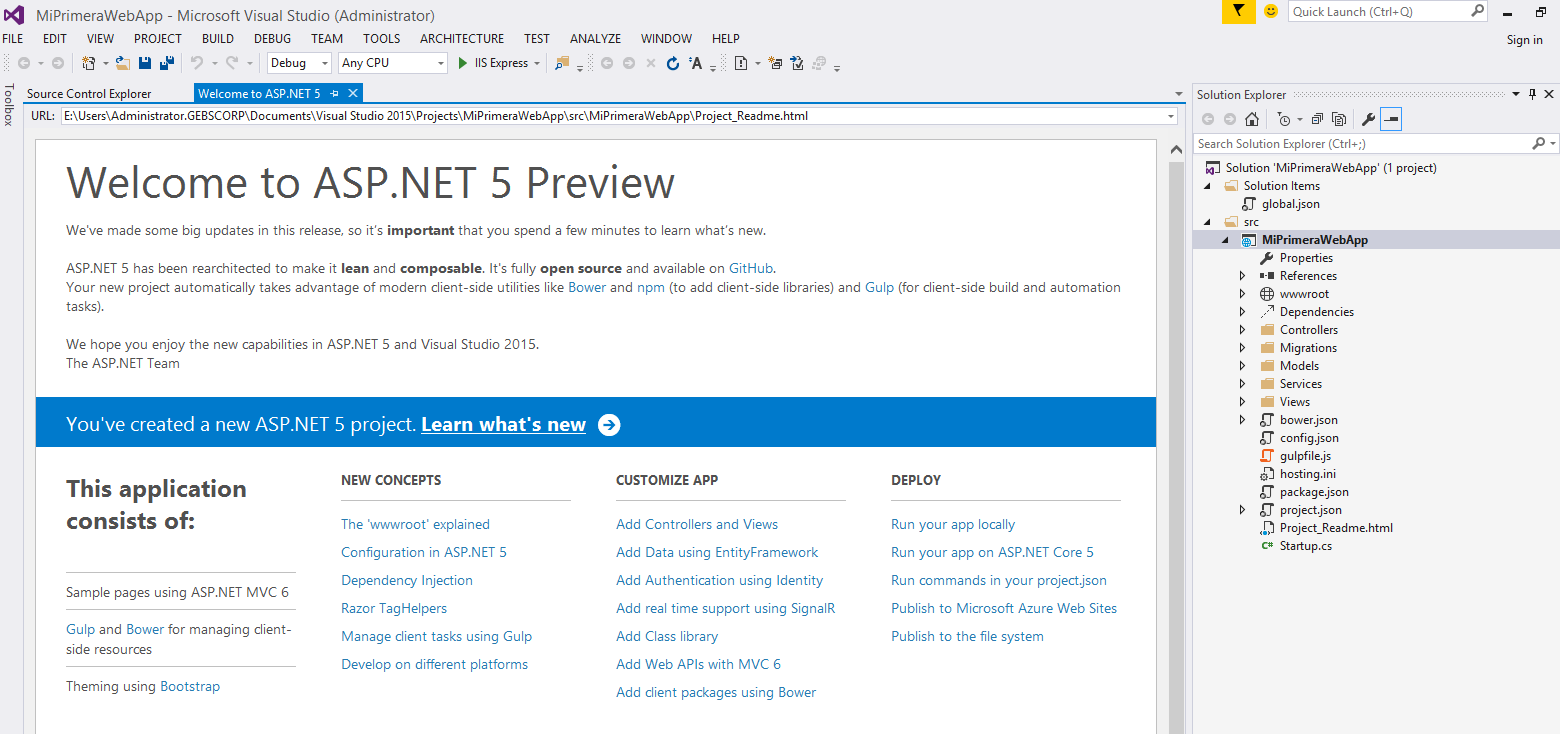
1. En la ventana de **New Project** seleccionar en el panel izquierdo: **Installed >** **Templates** > **Visual C#**, seleccionar **Web**, nombrar a la aplicación **MiPrimeraWebApp** y hacer clic en **OK**.



1. La siguiente pantalla que aparece a continuación da la opción de seleccionar una nueva plantilla de proyecto ASP.NET 5. En este caso se debe seleccionar Web Application, el cual crea una plantilla para un sitio web MVC y Web API. Para finalizar dar clic en el botón OK.

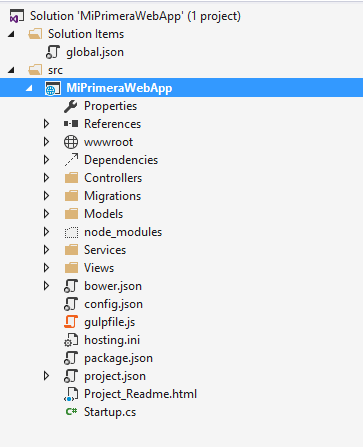
****

1. Al terminarse de aprovisionar el proyecto es probable que el nodo “References” aparezca de esta forma References (Restoring) por unos segundos mientras se terminan de descargar y aprovisionar las DLLs necesarias. Al finalizar se deberá ver una estructura de folders y archivos similar a la siguiente imagen.

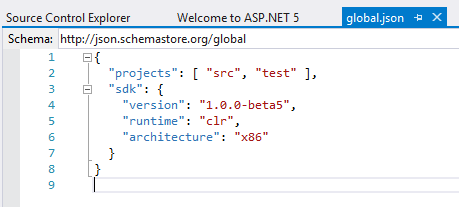
****

* **Ejercicio 1.2: Revisar la estructura de un aplicación MVC.**

1. En esta nueva versión de .NET y Visual Studio, NuGet toma mucho protagonismo, ya que se busca que con .NET Core solo se traiga lo mínimo necesario y lo demás que se ocupe sea descargado a través de NuGet. Esta nueva estructura es muy similar a las de los proyectos en Github o Codeplex. Los dos principales directorios son “Solutions Items” y “src”

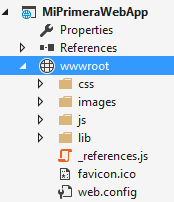


1. El primer archivo global.json dentro del folder Solution Items es el archivo que contiene la información del proyecto para Visual Studio u otras herramientas, en este caso el archivo está casi vacío, solo contiene información de las carpetas y versión del SDK. Para los que vienen de versiones anteriores de .NET ver archivos configurativos con extensión JSON es algo totalmente nuevo, por otro lado para los que vienen de otras plataformas como Node.JS es algo muy común y natural.



1. En el siguiente folder “src” donde se ven los archivos propios del proyecto, como ya se han conocido anteriormente. Dentro del folder “src” parecen los proyectos de la solución, en este ejemplo solo existe el proyecto de Interfaz de Usuario “MiPrimeraWebApp”.

En este nuevo proyecto de MVC 6 aparece un folder llamado “wwwroot” (nuevo en esta versión), el cual contiene archivos que no deben ser compilados, tales como hojas de estilo (CSS), imágenes y archivos de JavaScript.



1. A nivel de raíz de proyecto hay otros archivos extensión .json, tales como bower.json, config.json, package.json y project.json. ¡Sí! No hay web.config por defecto, sin embargo ASP.NET 5 soporta archivos configurativos en formato XML, INI o JSON. Para tratar de entender todo este cambio configurativo se hará un pequeño overview de cada archivo.

**bower.json**

El archivo bower.json, contiene, como su nombre lo dice, configuraciones propias de bower.



¿Qué es bower? Bower, al igual que NPM son administradores de paquetes, algo así como NuGet pero se comparten entre diferentes tecnologías ya que usualmente no dependen de la tecnología backend. En este caso Bower permite manejar dependencias de lado cliente, tales como jquery, bootstrap, etc. config.json, package.json y project.json

Entre estos archivos prácticamente se distribuyen las configuraciones que anteriormente existían en el archivo web.config. Si eres nuevo en desarrollo web sobre .NET, el web.config era un archivo XML que contenía configuraciones propias del proyecto.

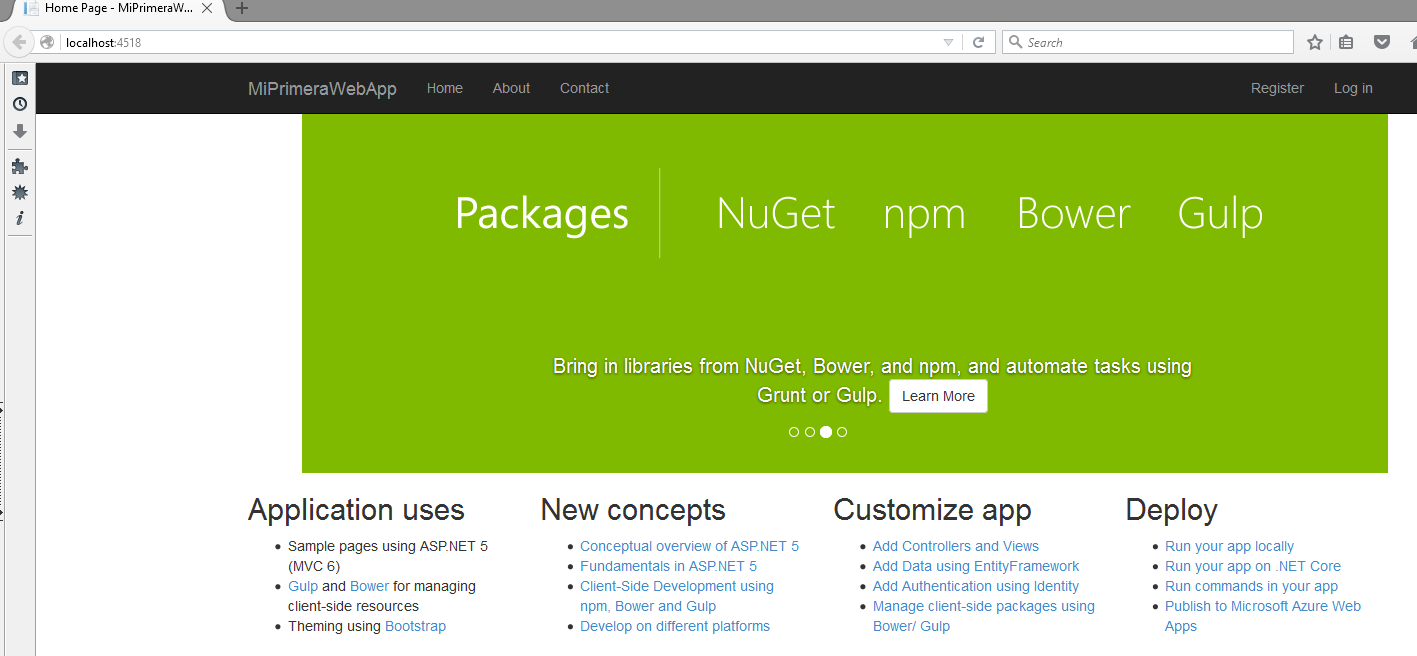
Durante este libro no se va a ahondar tantísimo en cada detalle nuevo, más bien, se intentará que sea algo más funcional y directo a entender lo mínimo necesario de ASP.NET 5.

El archivo Startup.cs es el que invoca los .json e inyecta dependencias necesarias (más adelante se conversará un poco más al respecto).

**Gulp**

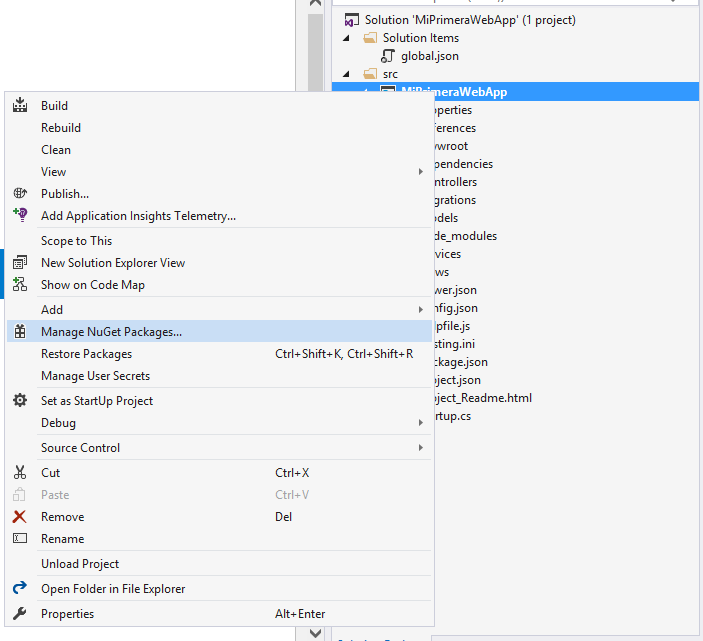
El desarrollo web moderno utiliza herramientas que permiten crear mejores aplicaciones (más veloces, optimizadas y seguras). Anteriomente se mencionaron Grunt y Gulp por encima, voy a tratar de ahondar un poco más en Gulp que es el task runner incluido por defecto.

1. Entrando ya en temática de propia ya de MVC, se pueden observar los folders Controllers, Views y Models. Para correr inicialmente la plantilla se debe presionar F5 y se podrá observar la nueva plantilla corriendo. Este template ya tiene incluidas las librerías **jQuery** y **Bootstrap**, además de otras más. Por lo cual ya trae un diseño inicial y unas páginas iniciales que responden solicitudes a través de controladores.

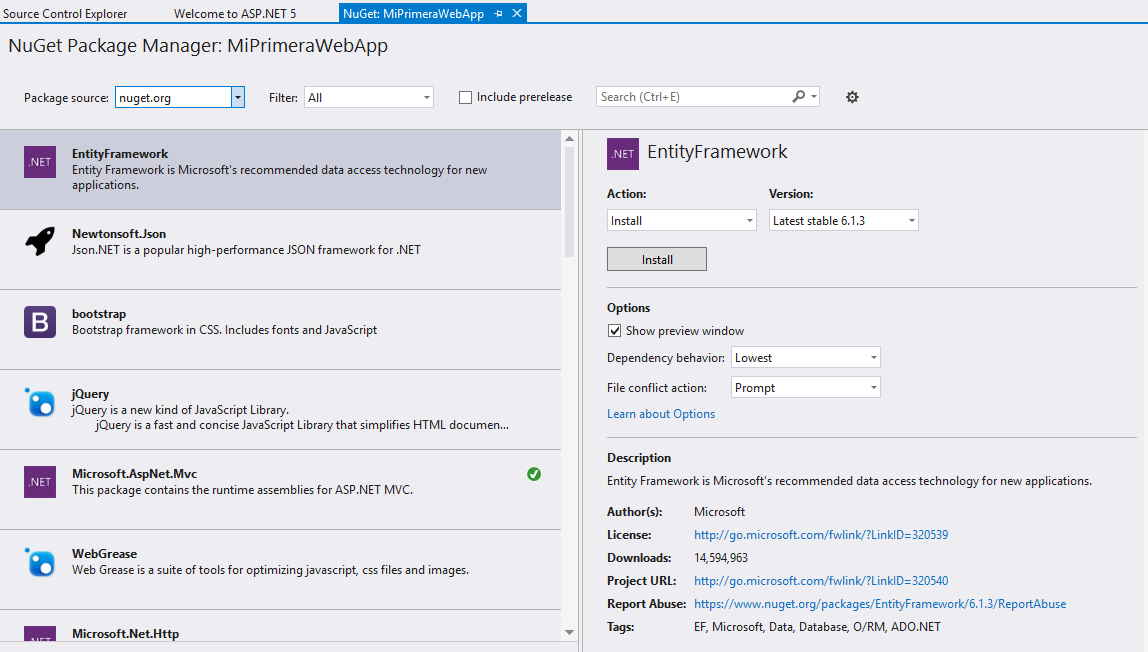


* **Ejercicio 1.3: Agregar la librería Log4Net usando la consola de NuGet.**

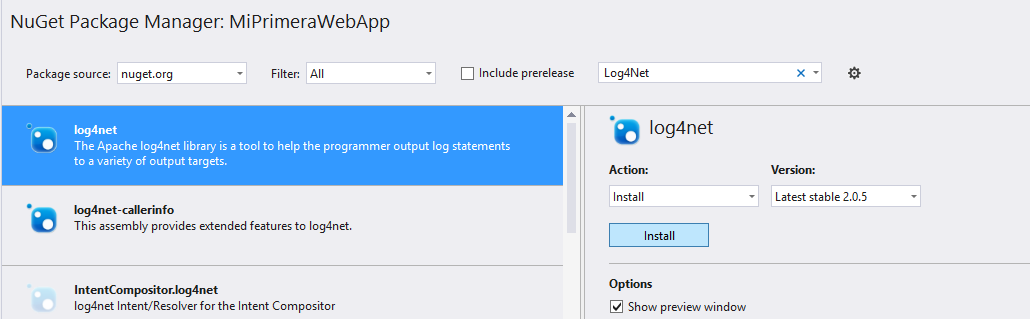
1. Clic derecho en el proyecto MiPrimerWebApp y seleccionar “Manage NuGet Packages…”.



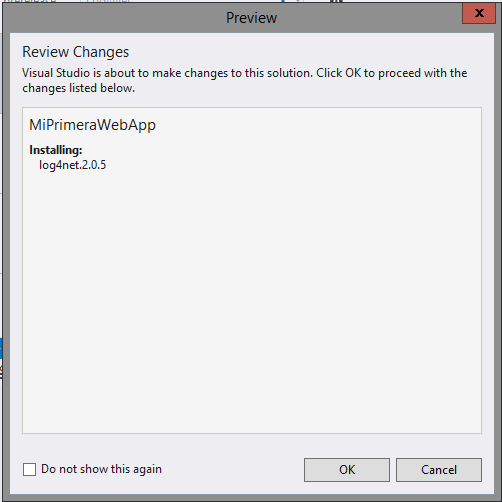
1. Se abrirá la siguiente ventana:



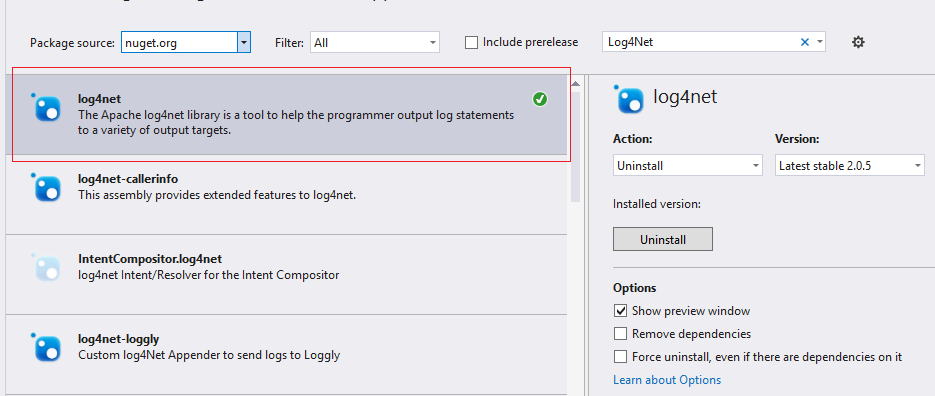
1. En el textbox “Search” escribir Log4Net y presionar Enter para iniciar la búsqueda. Posteriormente en la grilla de resultados de la parte inferior izquierda, seleccionar la opción “Log4Net”, luego dar clic en el botón “Install” ubicado en la parte inferior derecha, tal como se muestra en la siguiente figura.



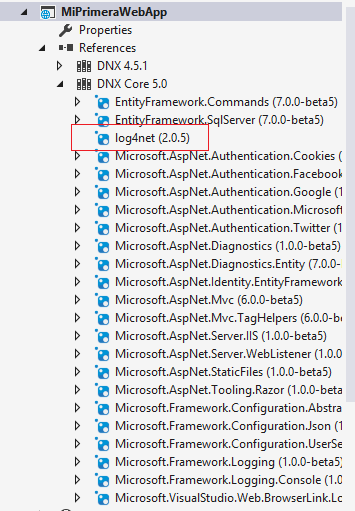
1. Confirmaremos la siguiente pantalla.



1. Para validar nuestros cambios podemos volver a ingresar a la pantalla “Manage NuGet Packages…” y ver lo siguiente:

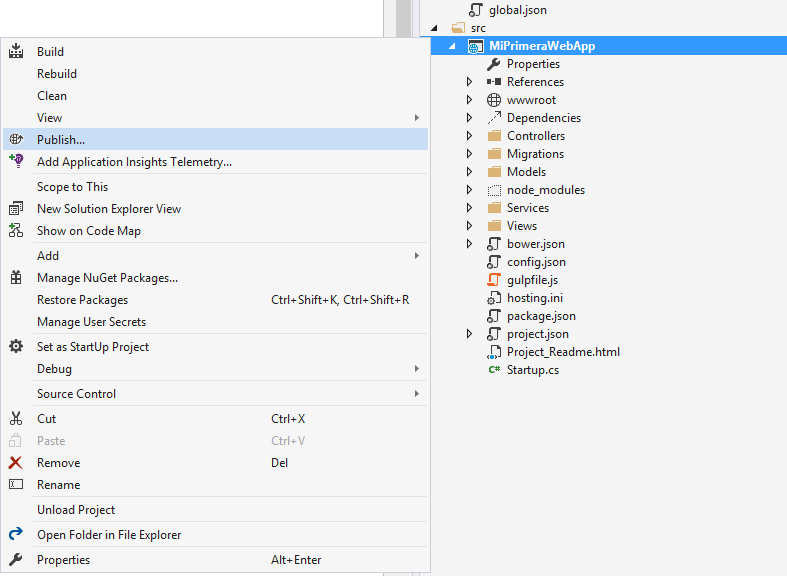


1. Para finalizar podemos ver estos cambios desde el Solution Explorer.

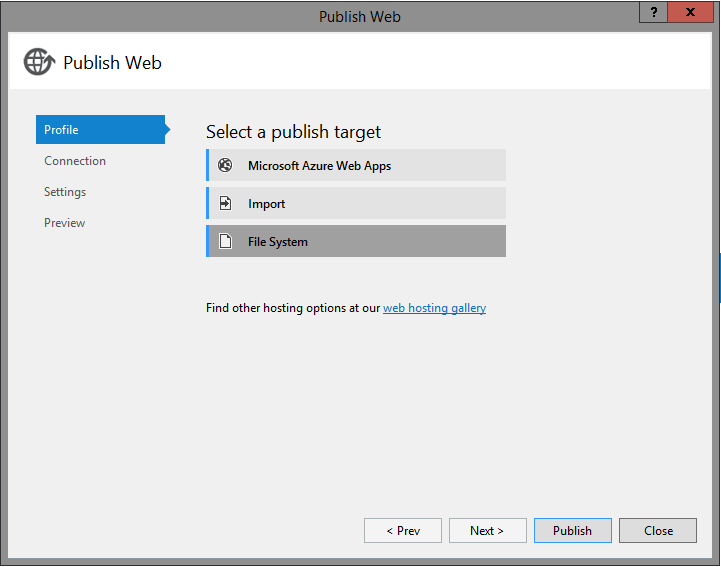


* **Ejercicio 1.4: Publicar la aplicación en IIS.**

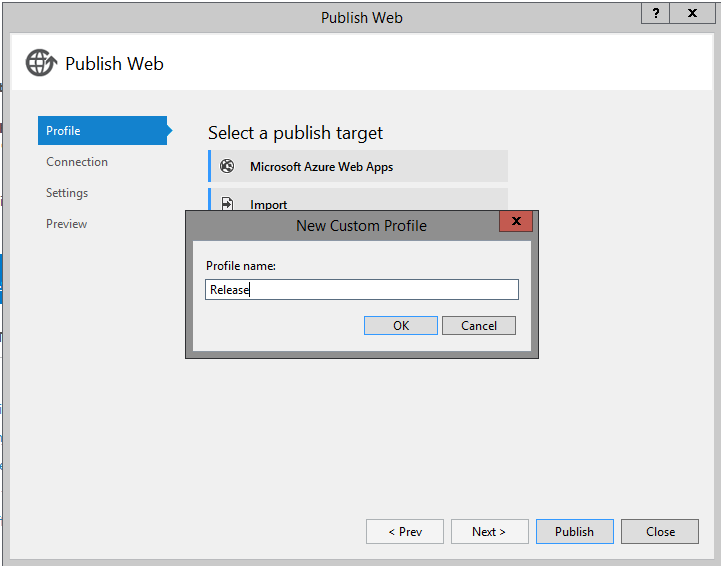
1. Clic derecho sobre el proyecto MiPrimeraWebApp y seleccionar la opción “Publish”.



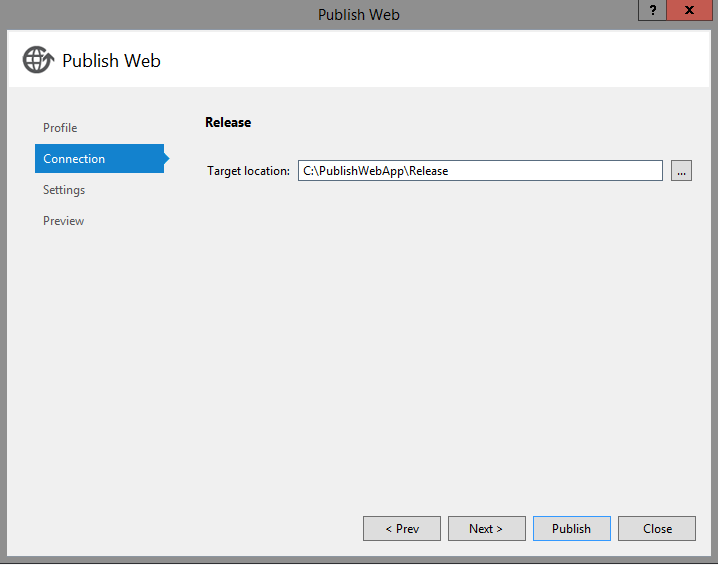
1. Se abrirá la pantalla “Publish Web”, ahí dar clic en la opción “File System”.



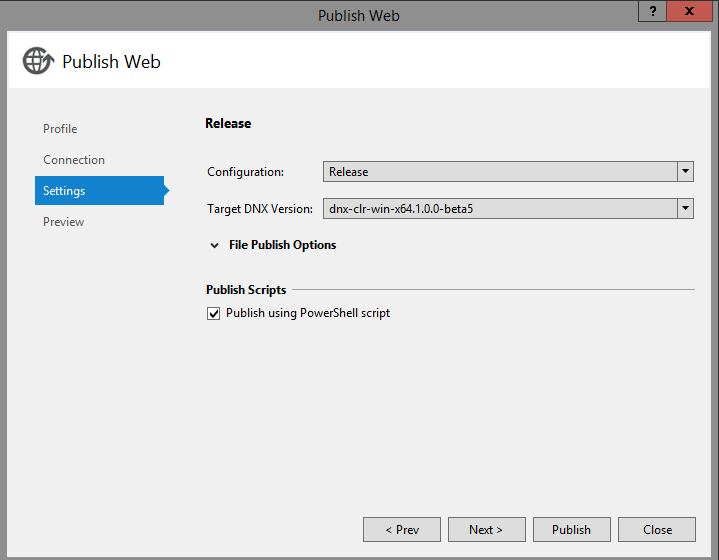
1. Posteriormente usted indicara un nombre para cual será asignado a un tipo de perfil. Para nuestro ejemplo colocaremos el perfil “Release” en el cuadro de texto “Profile Name”.



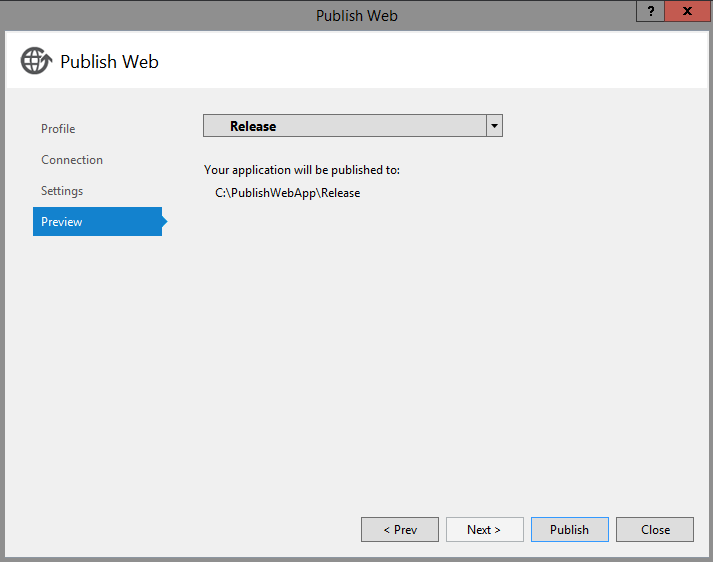
1. En la siguiente pantalla “Connection”, dar clic en el botón “Next”.



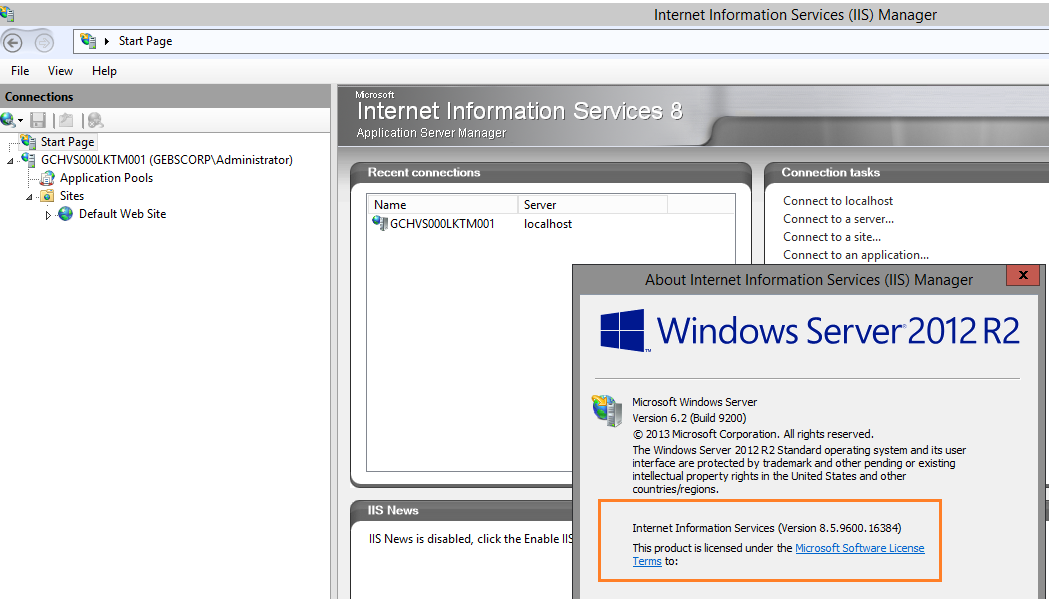
1. Posteriormente en la pantalla “Setting” dar clic en el botón “Next” nuevamente.



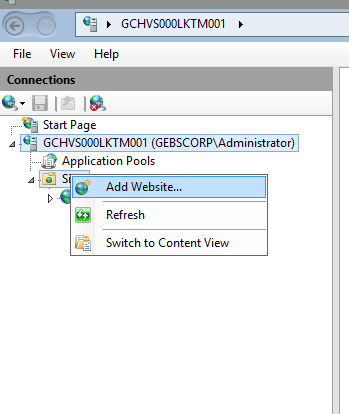
1. En la pantalla “Preview”, dar clic en el botón “Publish”.



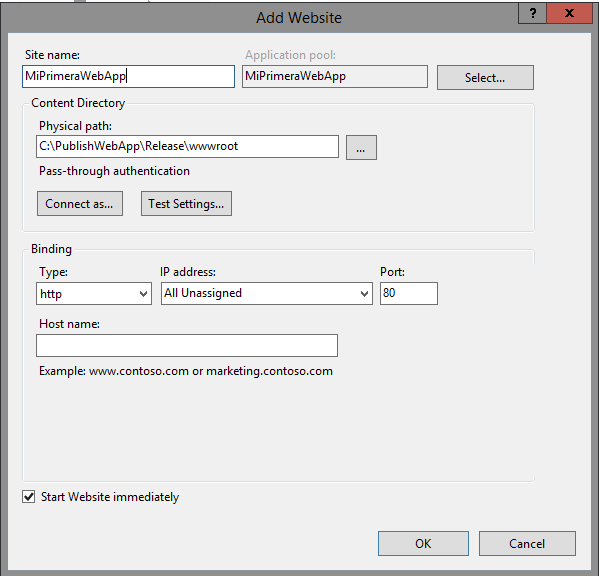
1. Ahora abra IIS Manager.



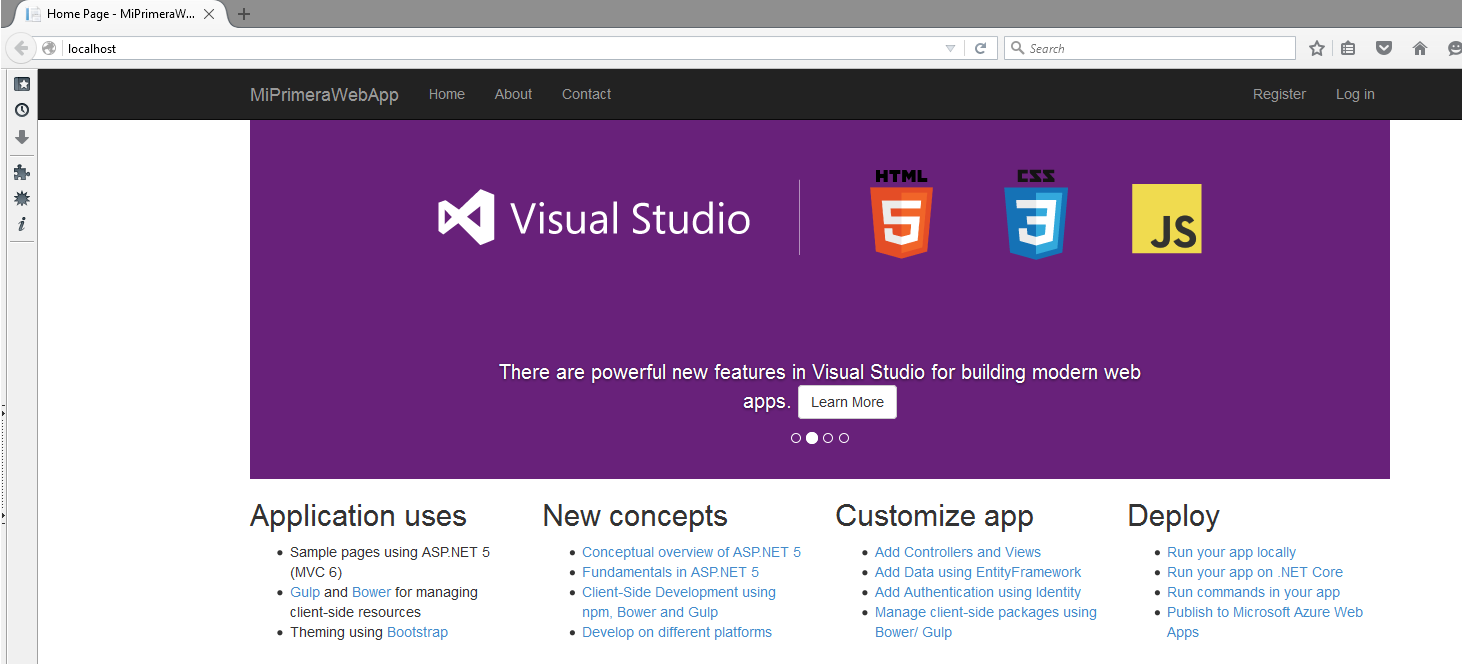
1. Crear un nuevo WebSite con el nombre “MiPrimeraWebApp” dentro de IIS.



1. Seleccionamos el directorio wwwrooot como ruta física del WebSite. En nuestro caso esta se encontraría dentro del directorio: **C:\PublishWebApp\Release**



1. Verificamos que el WebSite esté trabajando correctamente. Si ustedes tuvieron algún error por favor revisar cuidadosamente todos los pasos.



# EVALUACIÓN

1. Que significa definir la arquitectura de una aplicación web.

Determinar los componentes necesarios para construir una aplicación web, para ellos debemos elegir un framework robusto y escalable que nos facilite la construcción de ello

1. Visual Studio 2015 es un framework o una herramienta.

Es una herramienta que permite la construcción de aplicaciones basadas en .NET Framework.

1. Definir Asp.NET.

Asp.Net es un framework que permite la construcción de aplicaciones web basadas en .NET.

1. Que es ASP.NET MVC.

Asp.NET MVC es un framework cuyo core es Asp.NET pero que permite la construcción de aplicaciones web basadas en el patrón MVC.

1. Para que sirve IIS.

Sirve para publicar aplicaciones web basadas en .NET.

1. Para que sirve NuGet.

Permite gestionar los paquetes de terceros que se puedan usar en las aplicaciones, estos paquetes pueden ser .dll, .js, .css, etc.