

Capítulo 1

Desarrollo de Aplicaciones Empresariales

Objetivo

Al finalizar el capítulo, el alumno:

- Comprende el ambiente de desarrollo web con .NET y Visual Studio.
- Conoce las herramientas necesarias para el desarrollo web.
- Describe escenarios de aplicación para Web Forms y MVC.
- Reconoce la evolución de ASP.NET MVC.
- Utiliza componentes de terceros mediante NuGet.

Temas

1. Fundamentos de Diseño de Aplicaciones Web.
2. Introducción a Visual Studio 2015.
3. IIS 8/8.5/10.
4. Asp.Net 5.0.
5. Framework Asp.NET MVC 6.0.
6. Evolución de ASP.NET MVC.
7. Análisis del Ciclo de Vida de un Request.
8. Lineamientos para el uso de WebForm y MVC.
9. Herramientas para el desarrollo de Aplicaciones Web.
10. Usando componentes de terceros con NuGet

1. Fundamentos de Diseño de Aplicaciones Web



Existen varias consideraciones que deben tomarse en cuenta para el diseño de aplicaciones web:

- Definición de la arquitectura:

El diseño debe estar basado en función a un framework robusto y escalable, que se encuentre alineado a un patrón sustentado para la construcción de Aplicaciones Web. Esto implica seleccionar qué usar en cada capa de la aplicación y seleccionar el tipo de proyecto web a usar (Web Forms, MVC, etc.). Debe tener una estructura que permita la extensibilidad de la aplicación para la gestión de cambios e implementación de nuevas funcionalidades con el menor esfuerzo posible.

- Identificación y distribución del contenido:

Determinar qué tipo de web se hará, qué recursos se van a incluir y cómo se manejará la navegabilidad y accesibilidad.

Debe tener secciones completamente diferenciadas entre el contenido dinámico y estático para la respectiva localización de componentes y crecimiento horizontal.

- SEO (Search Engine Optimization):

Consiste en la elaboración y distribución del contenido de las páginas web de manera que los buscadores puedan asignar un buen ranking a nuestra aplicación.

- Personalización:

Si se desea que la aplicación pueda ser accedida y visualizada en diferentes idiomas y respetando un formato establecido, se deben tener en cuenta los siguientes criterios:

- Localización: es el proceso de personalización de la aplicación para un idioma dado. Consiste principalmente en la traducción de la interfaz de usuario mediante el uso de archivos de recurso (.resx).
- Globalización: es el proceso de personalización de la aplicación para que las fechas, números, moneda, etc., se muestren en un formato específico según se configure.

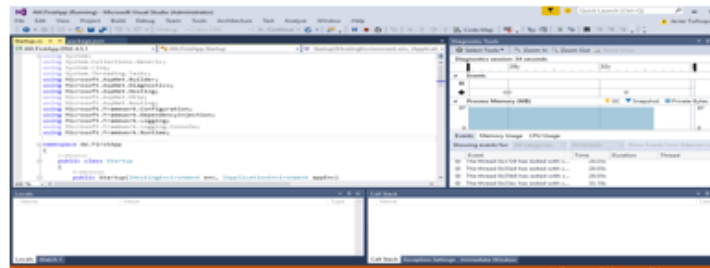
- Configuración:

Un aspecto importante en toda aplicación es su despliegue en diferentes ambientes, y que esto sea de manera configurable. Los ambientes pueden ser diferentes servidores como por ejemplo, pruebas, desarrollo, producción, entre otros. Para esto se puede hacer uso por ejemplo del XDT (XML Data Transform) que permite generar diferentes web.configs para diferentes ambientes.

2. Introducción a Visual Studio 2015

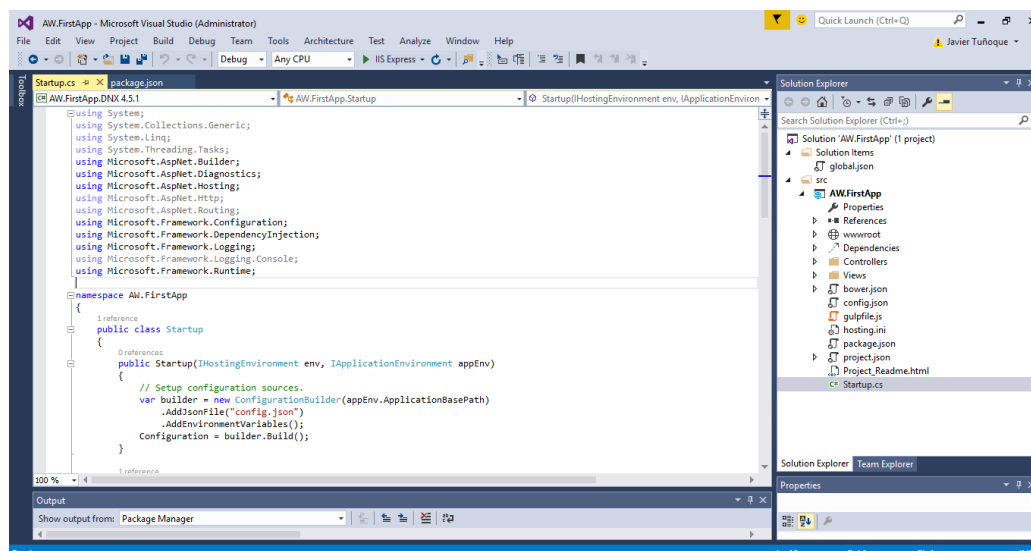
Introducción al Visual Studio 2015

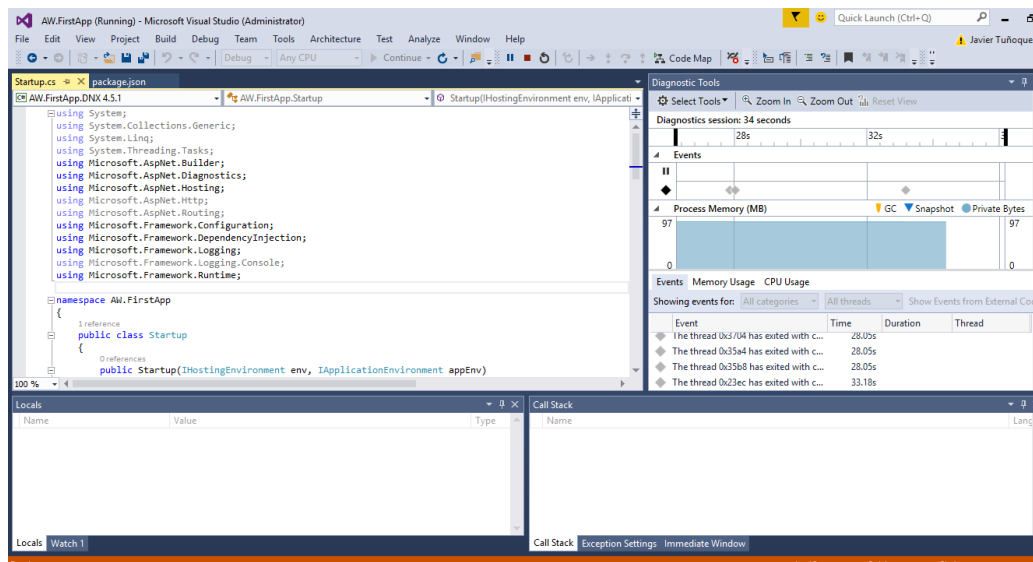
- Algunas características nuevas:
 - El desarrollo móvil de plataforma cruzada para Windows, iOS y Android.
 - La integración de la nube.
 - Desarrollo fácil, más rápido manejo de datos a través de Internet, Windows, Windows Phone, y Windows Store utilizando Entity Framework 7.
 - Soporte IDE mejorado para la construcción de soluciones de JavaScript con TypeScript language.
 - Código abierto de muchos elementos incluyendo el compilador .NET, .NET Core, TypeScript, ASP.NET.



1 - 5

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.





Visual Studio 2015 y la última versión de .NET Framework introducen nuevas características como mobil-first / cloud-first, desarrollo de plataforma cruzada, la adopción de estándares abiertos y la transparencia a través de código abierto.


Esta última versión también mejorará las experiencias en los desarrolladores al escribir código para la web, Windows, Office, base de datos y las aplicaciones móviles. El producto 2015 permite a los desarrolladores aumentar realmente su rango en la construcción de aplicaciones modernas conforme la demanda actual de los usuarios.

- El desarrollo móvil de plataforma cruzada para Windows, iOS y Android. El desarrollo moderno, unificando web con ASP.NET5.
- La integración de la nube lista para facilitar el desarrollo y despliegue. Integración del nuevo compilador de código abierto, "Roslyn" para VB, C #, y TypeScript.
- Desarrollo fácil, más rápido manejo de datos a través de Internet, Windows, Windows Phone, y Windows Store utilizando Entity Framework 7.
- Proyectos compartidos para C# y JavaScript para hacer el intercambio de código entre aplicaciones más fáciles.
- Versión rediseñada de Blend para la creación de hermosas interfaces de usuario (UI) con XAML.
- Soporte IDE mejorado para la construcción de soluciones de JavaScript con TypeScript language (un súper conjunto de librerías del mismo JavaScript).
- Código abierto de muchos elementos incluyendo el compilador .NET, .NET Core, TypeScript, ASP.NET, y mucho más.

Ediciones de Visual Studio:


- Visual Studio Community
- Visual Studio Professional
- Visual Studio Enterprise
- Visual Studio Test Professional.

3. IIS 8 / 8.5 / 10



IIS 8 / 8.5 / 10

- IIS: Internet Information Services: Servidor web para ASP.NET.
- IIS 8: Application Initialization, CPU Throttling, Dynamic IP Restrictions, Centralized SSL Certificate Support, WebSocket Protocol Support.
- Mejoras de registro – Manejabilidad.
- Escalabilidad – Activación de sitio dinámico.
- Soporte HTTP 2.0.
- Soporte para cabeceras Wildcard host.
- Módulo de PowerShell IISAdministration.
- Variables de entorno para Application Pools.
- Soporte estatus HTTP 308: Utilizando el módulo HTTP Redirect.

1 - 6Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.

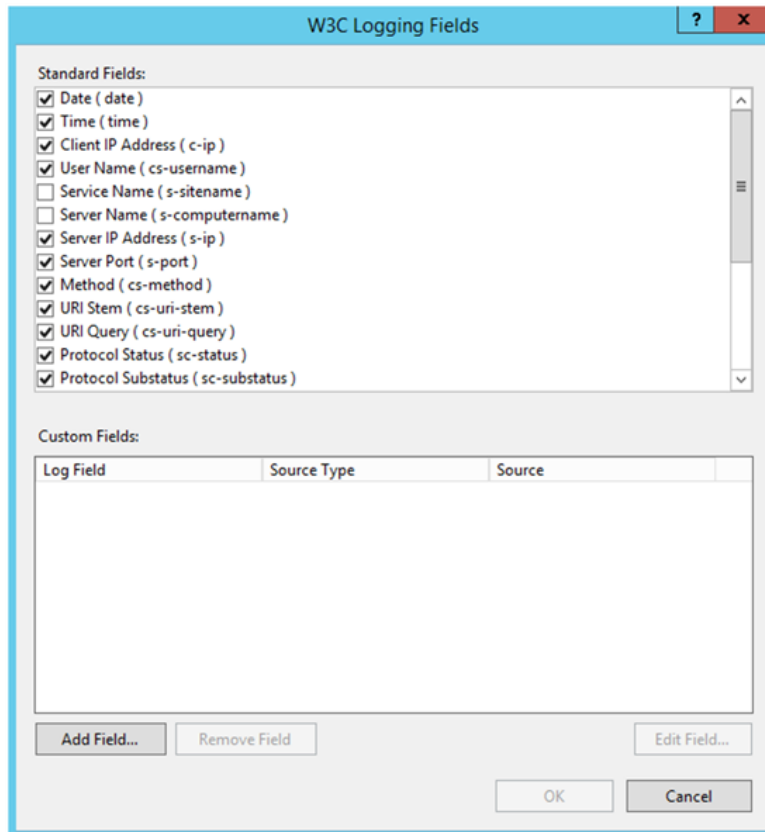
IIS 8 y 8.5

IIS 8.0 tiene una serie de nuevas características y mejoras, algunas de las cuales han sido liberadas para IIS 7.5 actualizaciones out-of-band www.iis.net. Muchas de estas nuevas características se deben a los cambios dentro del nuevo sistema operativo y no son compatibles con versiones anteriores de IIS.

Módulo de aplicación Warm-Up, por ejemplo, fue lanzado para IIS 7.5 y está integrada en IIS 8.0, pero un almacén central para los certificados SSL requiere Windows Server 2012 y solo está disponible en IIS 8.0.

IIS 8.5, con el lanzamiento de la nueva versión de Windows Server 2012 R2 viene una nueva versión de IIS, IIS 8.5. Las nuevas características se pueden dividir en 2 categorías, escalabilidad y facilidad de gestión:

- Mejoras de registro – Manejabilidad.
- Manejabilidad - Eventos ETW.



W3C Logging Fields

Standard Fields:

- ☒ Date (date)
- ☒ Time (time)
- ☒ Client IP Address (c-ip)
- ☒ User Name (cs-username)
- ☐ Service Name (s-sitename)
- ☐ Server Name (s-computername)
- ☒ Server IP Address (s-ip)
- ☒ Server Port (s-port)
- ☒ Method (cs-method)
- ☒ URI Stem (cs-uri-stem)
- ☒ URI Query (cs-uri-query)
- ☒ Protocol Status (sc-status)
- ☒ Protocol Substatus (sc-substatus)

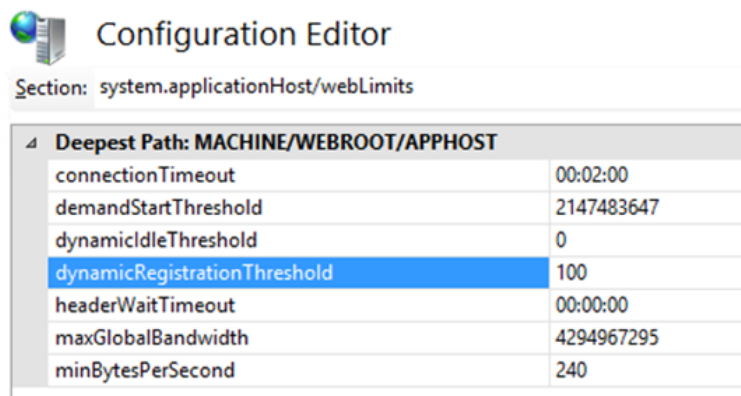
Custom Fields:

Log Field	Source Type	Source
-----------	-------------	--------

Add Field... Remove Field Edit Field...

OK Cancel

- Escalabilidad – Activación de sitio dinámico.



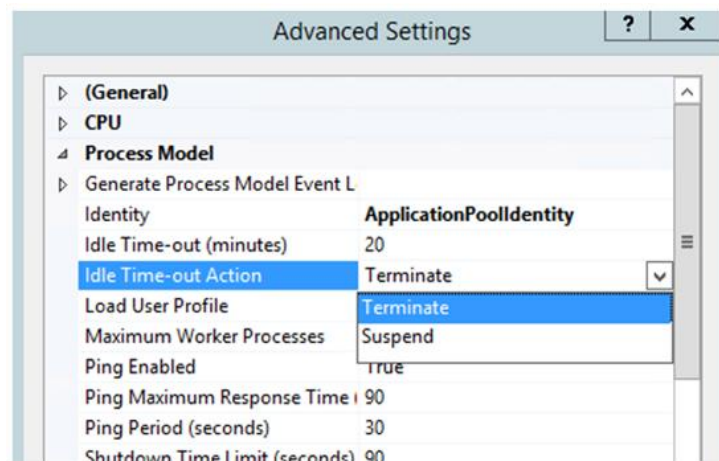
Configuration Editor

Section: system.applicationHost/webLimits

Deepest Path: MACHINE/WEBROOT/APPHOST

connectionTimeout	00:02:00
demandStartThreshold	2147483647
dynamicIdleThreshold	0
dynamicRegistrationThreshold	100
headerWaitTimeout	00:00:00
maxGlobalBandwidth	4294967295
minBytesPerSecond	240

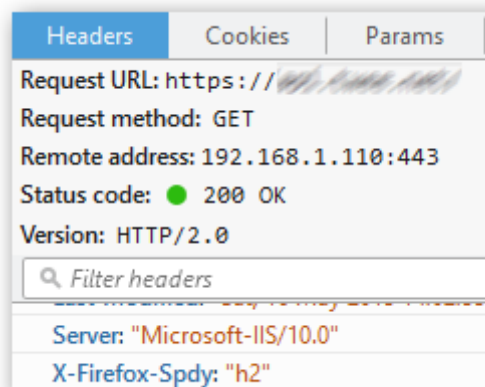
- Escalabilidad - Idle Work Process Page-out.



IIS 10

Una nueva versión de IIS se incluye en Windows 10 y Windows Server 2016, que fue lanzado recientemente. Y entre sus características más resaltantes tenemos:

- Soporte HTTP 2.0.



- Soporte para cabeceras Wildcard host.

```
New-WebBinding -Name "Default Web Site" -IPAddress "*" -Port 80 -HostHeader "*.foo.bar"
```

```
ls iis:\sites
```

Name	ID	State	Physical Path	Bindings
Default Web Site	1	Started	%SystemDrive%\inetpub\wwwroot	http *:80: https *:443: sslFlags=0 http *:80:*.foo.bar

- Módulo de PowerShell IISAdministration: Mientras que el módulo de PowerShell existente (WebAdministration) apenas ha cambiado, el equipo IIS añadió un segundo módulo con acceso directo al objeto "Microsoft.Web.Administration.ServerManager".

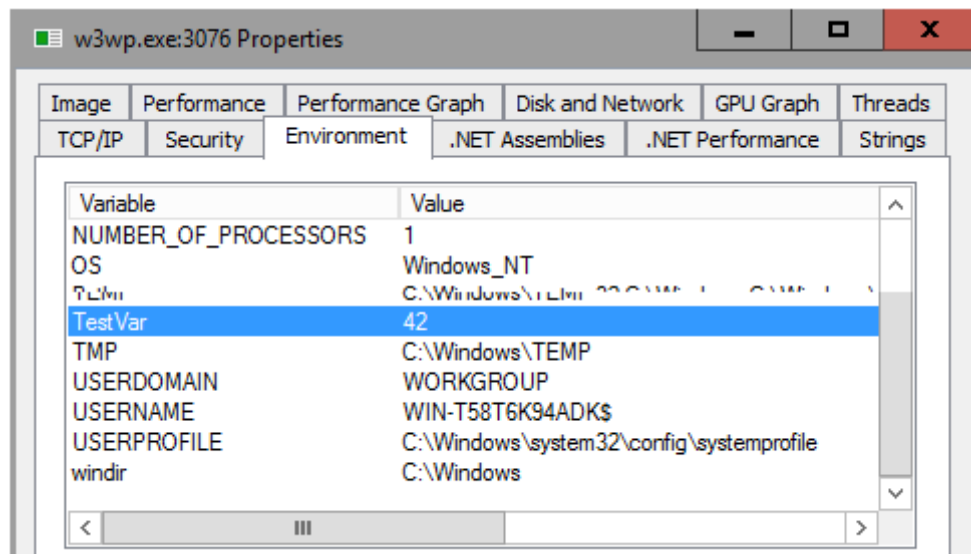
- Lo más importante aquí es el mejor soporte para PowerShell.

```
Get-command -Module IISAdministration | Select Name
```

```
Clear-IISConfigCollection
Get-IISAppPool
Get-IISConfigAttributeValue
Get-IISConfigCollection
Get-IISConfigCollectionElement
Get-IISConfigElement
Get-IISConfigSection
Get-IISServerManager
Get-IISSite
New-IISConfigCollectionElement
New-IISSite
Remove-IISConfigAttribute
Remove-IISConfigCollectionElement
Remove-IISConfigElement
Remove-IISSite
Reset-IISServerManager
Set-IISConfigAttributeValue
Start-IISCommitDelay
Start-IISSite
Stop-IISCommitDelay
Stop-IISSite
```

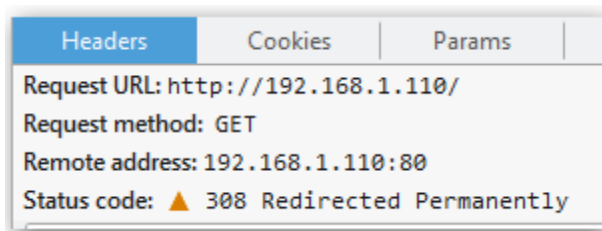
- Variables de entorno para Application Pools: Podemos establecer variables de entorno específicas por application pool. Debido a que no hay interfaz de usuario para esto todavía.

```
Add-WebConfigurationProperty -value @{name='TestVar';value='42'} -filter "system.applicationHost/applicationPools"
```

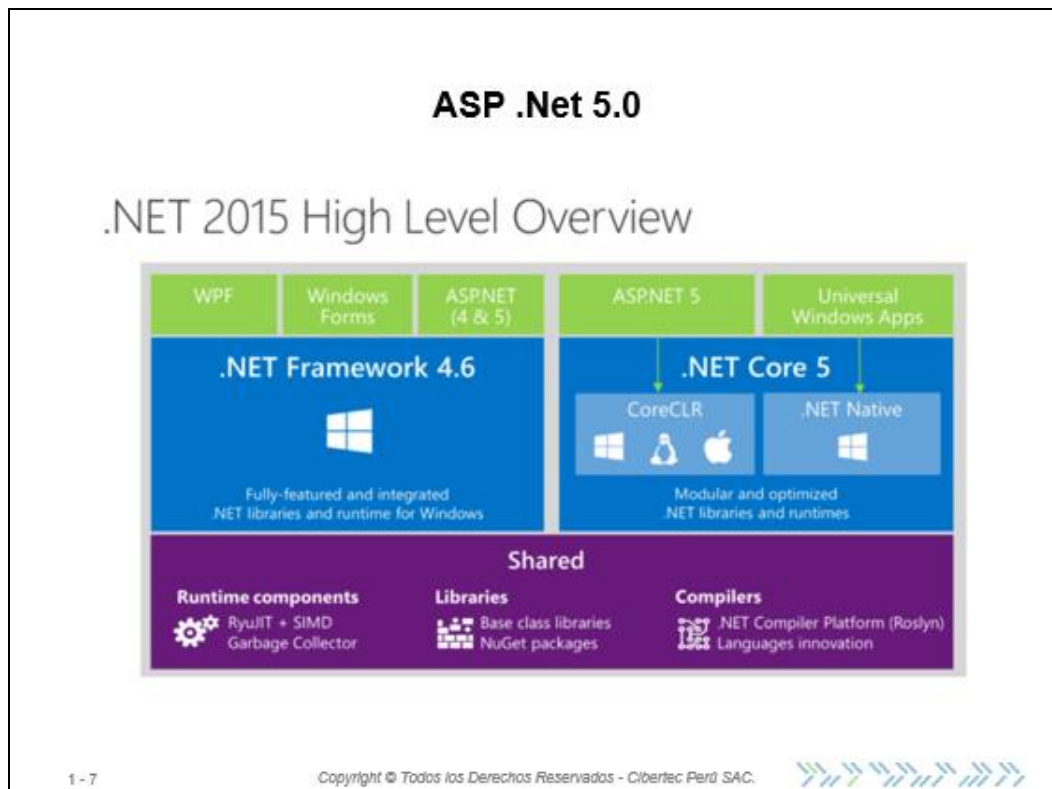


- Soporte estatus HTTP 308: Utilizando el módulo HTTP Redirect.

```
Install-WindowsFeature Web-Http-Redirect
```



4. ASP .Net 5.0

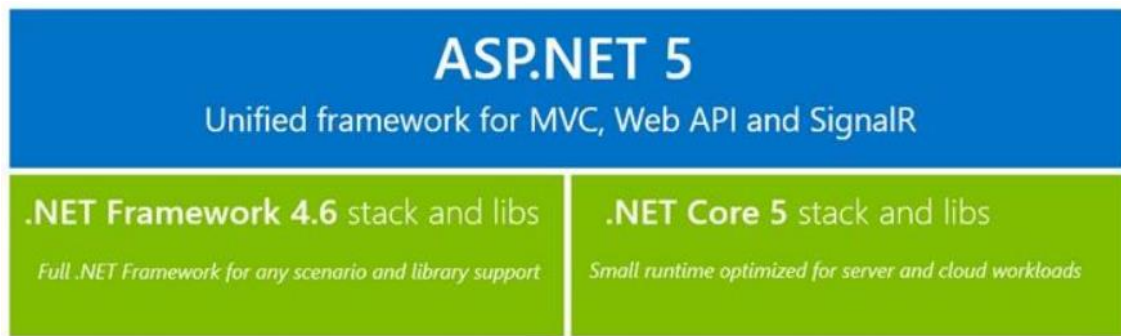


Antes de entrar en materia, es importante entender los cambios que .NET Framework está teniendo para conocer cómo impacta esto a ASP.NET.

Lo más notorio de este diagrama es la separación de .NET Framework en dos bloques: .NET Framework 4.6 y .NET Core 5. .NET Framework 4.6: Continúa el trabajo que trae .NET 4.5.2 con muchísimas mejoras alrededor del framework. Se puede ubicar como el framework que viene incluido en el sistema operativo, en este caso en Windows 10 y el cual recibe actualizaciones a través del conocido Windows Update.

Es importante notar que sobre .NET Framework 4.6 están las tecnologías WPF, Windows Forms y ASP.NET versiones 4 y 5. .NET Core 5: Microsoft lo describe como un “framework modular”, el cual llega a nosotros como una versión de software abierto, el cual puede ser desplegado de manera modular y local, además de ser mucho más ligero. Al ser modular busca también ser multiplataforma, corriendo en Windows, Linux y OSX. A diferencia de .NET Framework 4.6, .NET Core 5 permite correr aplicaciones ASP.NET solamente en la versión 5 y Universal Windows Apps con .NET Native.

De esta forma ASP.NET se ubica de la siguiente manera dentro del universo .NET.

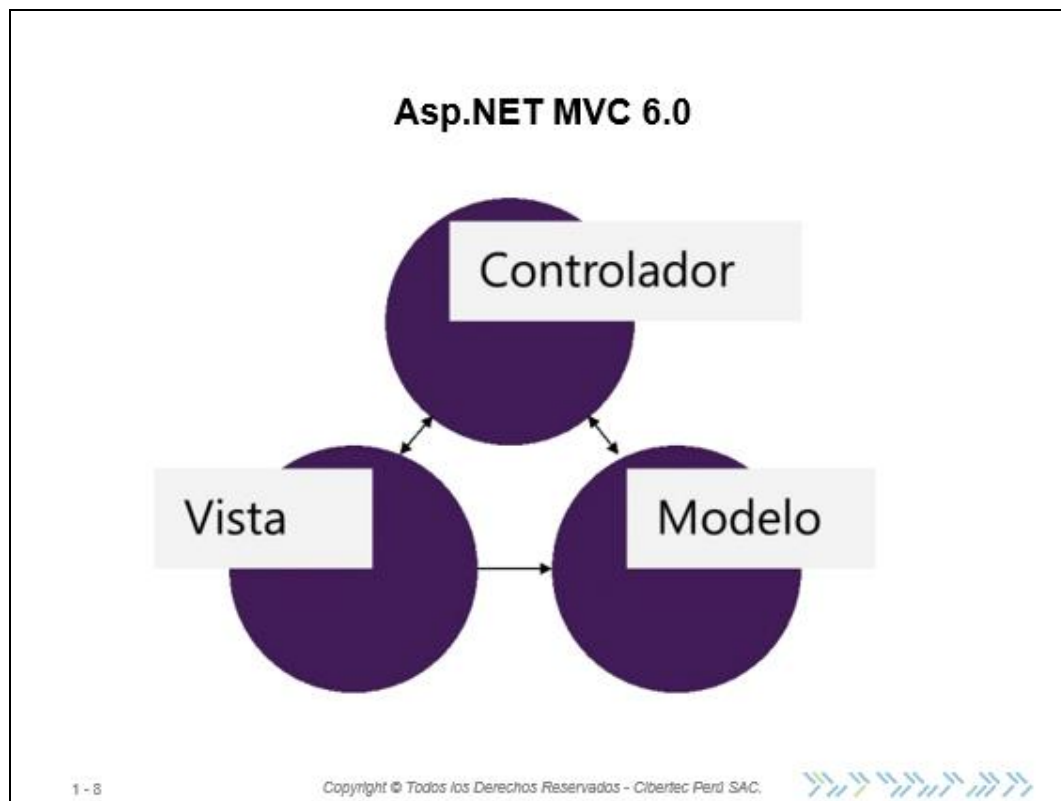


Con ASP.NET 5 corriendo en ambas “ediciones” del framework se logra desplegar y desarrollar aplicaciones web tanto en Windows (sobre .NET Framework 4.6 o corriendo lado a lado sobre .NET Core 5 con otra versión de .NET Framework instalada en el servidor) como en Linux y OSX.

A este punto .NET Core 5 soporta solamente ASP.NET MVC en C#, es decir, no Web Forms ni VB.NET. Esto no significa que no pueda ser que Microsoft incluya soporte a Web Forms o VB.NET en el futuro. Por otro lado, .NET Framework 4.6 sí continúa soportando el modelo de desarrollo en Web Forms y el lenguaje de programación VB.NET.

Algunas de las características más notables de ASP.NET 5 es la unificación de MVC, Web API y Web Pages en un solo modelo conocido como MVC 6. Otra importante adición es la integración con herramientas populares de desarrollo web como Bower, Grunt y Gulp, los cuales ya se podían utilizar con otros frameworks de desarrollo como PHP, Node.JS y Ruby.

5. Framework Asp.NET MVC 6.0



Modelo Vista Controlador, es lo que las siglas MVC representan, y su objetivo principal es lograr la separación entre estos tres “personajes”, lo cual permite, a través de este patrón de diseño, crear aplicaciones robustas tomando en consideración buenas prácticas aprendidas de otras plataformas de desarrollo y del propio Microsoft.

La separación de conceptos busca precisamente que cada “bloque” de la aplicación realice solo el trabajo que le corresponde, así logra obtener otras ventajas tales como mantenibilidad del sistema, extensibilidad y crecimiento ordenado.

Así como otros patrones ya buscan lograr esto, tales como programación en N-Capas, MVC puede seguir estos lineamientos como separación de proyectos para distribuir componentes comunes. Todo esto es porque MVC como tal es un patrón, no una tecnología.

- Vista: La vista es el conjunto de markup de la aplicación, es decir el HTML. En ASP.NET MVC este markup es pre-renderizado a través de una tecnología llamada Razor, la cual permite, entre otras cosas, ejecutar lógica previa a renderizar el HTML que será enviado al cliente. Con Razor esta compilación o renderización ocurre del lado servidor. No debería incluir ningún tipo de lógica de negocio, sino más bien lógica propia de interacción

con el usuario, como iterar una lista (List) y presentar una lista ordenada (ul) de ítems al usuario.

- Modelo: Son los objetos de dominio o negocio de la aplicación, más adelante, veremos que no es necesario que esté en el mismo proyecto que la aplicación web.
- Controlador: El controlador se encarga de recibir solicitudes y ser el punto de entrada de la aplicación, a través de lo que se conoce como acciones (Actions). Se basa en solicitudes HTTP y puede responder con datos o con contenido, como HTML.

Para comenzar a explicar de forma general este modelo con la ayuda de Visual Studio 2015, es importante entender que una plantilla de Visual Studio, no es más que una serie de DLLs que funcionan para un fin específico incluidas en un proyecto + una base de código configurativa de la tecnología + uno o varios ejemplos de código. Es decir, si uno quisiera hasta podría empezar un proyecto sin necesidad de seleccionar un template de Visual Studio.

La forma de “unificar” archivos en forma de proyecto es a través de archivos con extensión .csproj o .vbproj, los cuales a su vez se pueden agrupar en una solución con archivos extensión .sln. Para entender un poco más de archivos de proyecto y su funcionalidad.

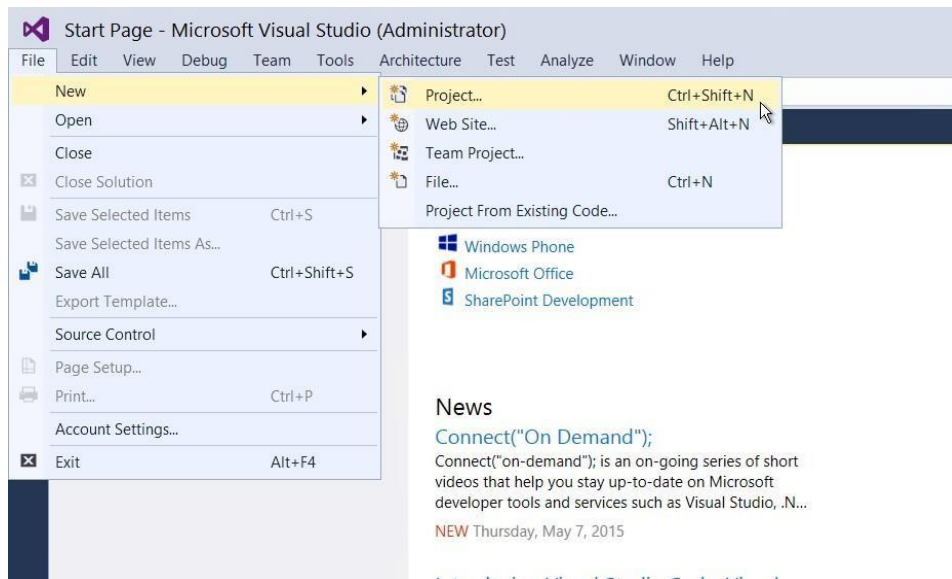


Estructura de solución

- Archivos no específicos de usuario: archivos propios del software, que pueden ser controlados por un manejador de versiones tal como TFS o Github.
 - -sln
 - .csproj o .vbproj
- Archivos específicos de usuario: archivos propios del software desarrollado para el usuario de sistema actual. No son controlados por controladores de versiones.
 - -suo
 - .csproj.user o .vbproj.user

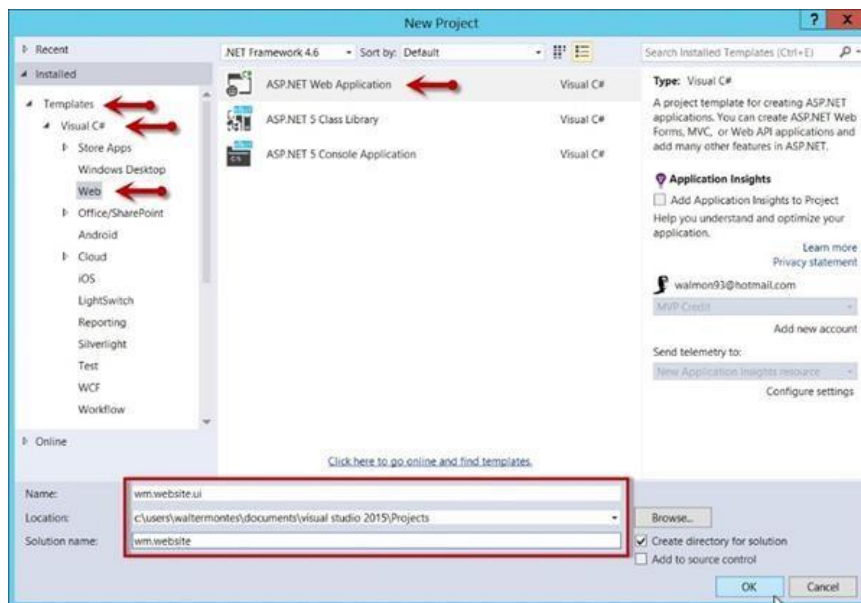
Volviendo al tema de ASP.NET MVC el proceso de creación de un proyecto basado en una plantilla es sencillo, en Visual Studio

> File/Archivo > New/Nuevo> Project/Proyecto



Nuevo Proyecto

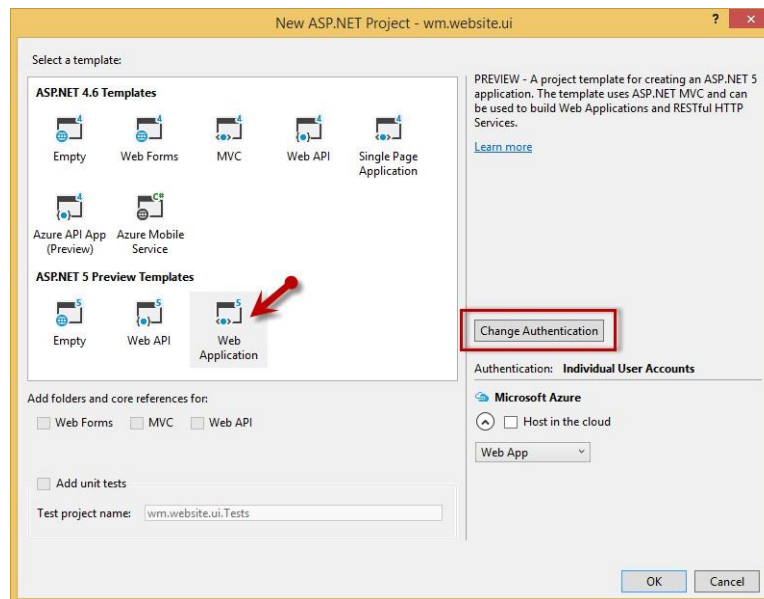
El siguiente paso es seleccionar la plantilla de MVC 6, debe fijarse que seleccione .NET Framework 4.6 y colocar un nombre de proyecto y de solución acorde a sus necesidades ya que este será el nombre de espacio que seguirá su proyecto (orden interno del proyecto). Usualmente y por estándar de Microsoft se coloca el nombre de espacio de la siguiente forma:



Definición de la plantilla MVC 6

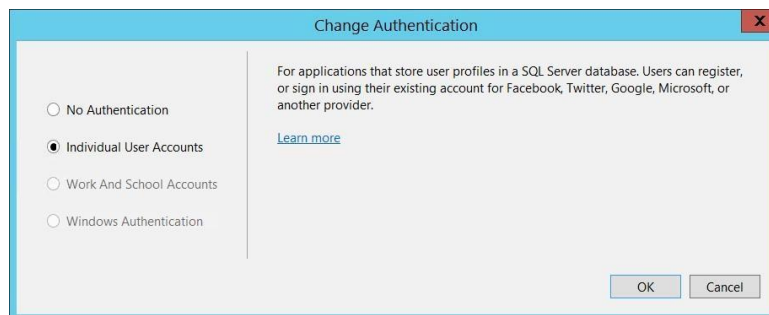
Para los que vienen desde versiones anteriores previo a Visual Studio 2012, han visto un par de cosas nuevas, entre ellas, que no tuvieron que seleccionar el tipo de proyecto (Web Forms, MVC o Web Pages), sino que se selecciona un proyecto de tipo ASP.NET, esto es por un cambio que se dio en el 2013, en el cual aparece el término ONE ASP.NET, el cual busca que se puedan crear proyectos con tecnología Web Forms y MVC/Web API en un solo proyecto de Visual Studio, a diferencia de cómo anteriormente solo se podía seleccionar un tipo.

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 Site, el cual crea una plantilla para un sitio web MVC y Web API.



Tipo de Proyecto ASP.NET

En la imagen anterior se ha resaltado la opción “Change Authentication” para dar un poco de énfasis a esta opción. Si se le da clic a este botón, aparece la siguiente pantalla.



Cambiar autenticación

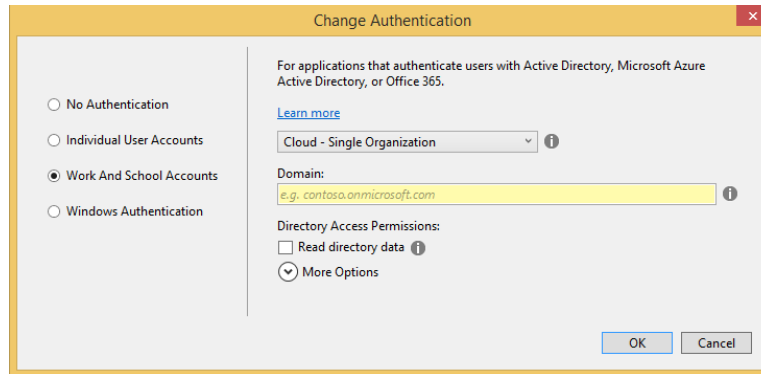
Esta ventana da la opción de seleccionar la estrategia de autenticación, por lo cual se debe tener de antemano una idea de la estrategia de autenticación y autorización del sistema. Esto porque más adelante se puede cambiar, sin embargo puede ser un poco complicado si no se conoce bien a detalle cómo se implementa cada modelo, ya que al seleccionar una opción se crean las clases y configuraciones de ASP.NET y/o IIS para procesar un modelo de autenticación.

¿Cómo determinar el modelo de autenticación?

- No Authentication: en caso de que no se requiera autenticación o se desee implementar un tipo especial de autenticación
- Individual User Accounts: en caso de que se quiera contar con usuarios propios para el sistema, es decir, con una base de datos SQL Server que almacene usuarios y contraseñas de manera segura y con buenas prácticas, y/o cuando

se quiere autenticar usuarios con una tercera parte, tal como Facebook, Twitter, Google o Microsoft.

- **Work And School Accounts:** permite autenticar con proveedores de autenticación empresariales, tal como Azure Active Directory o Active Directory Federation Services.



- **Windows Authentication:** en ambientes corporativos cuando las computadoras clientes están en un dominio en Active Directory Domain Services, de esta forma no se solicita al usuario autenticarse, sino que se utilizan los credenciales de Windows.

Para la demostración se va a dejar marcada la opción de Individual User Accounts, el cual utiliza el Simple Membership Provider para obtener autenticación de usuarios en una base de datos SQL Server en LocalDB (más adelante se ahondará un poco más en este tema).

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 de la imagen de la derecha.

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.



Estructura del Proyecto

Los primeros niveles son:

- Solutions Items
- src

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.

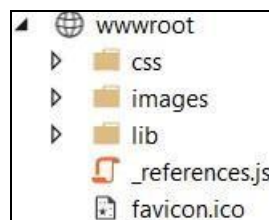
```
{
  "projects": [ "src", "test" ],
  "sdk": {
    "version": "1.0.0-beta5",
    "runtime": "clr",
    "architecture": "x86"
  }
}
```

Archivo global.json

En el siguiente folder src es donde se ven los archivos propios del proyecto, como ya se han conocido anteriormente.

Dentro del folder src aparecen los proyectos de la solución, en este ejemplo solo existe el proyecto de Interfaz de Usuario “wm.website.ui”.

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.

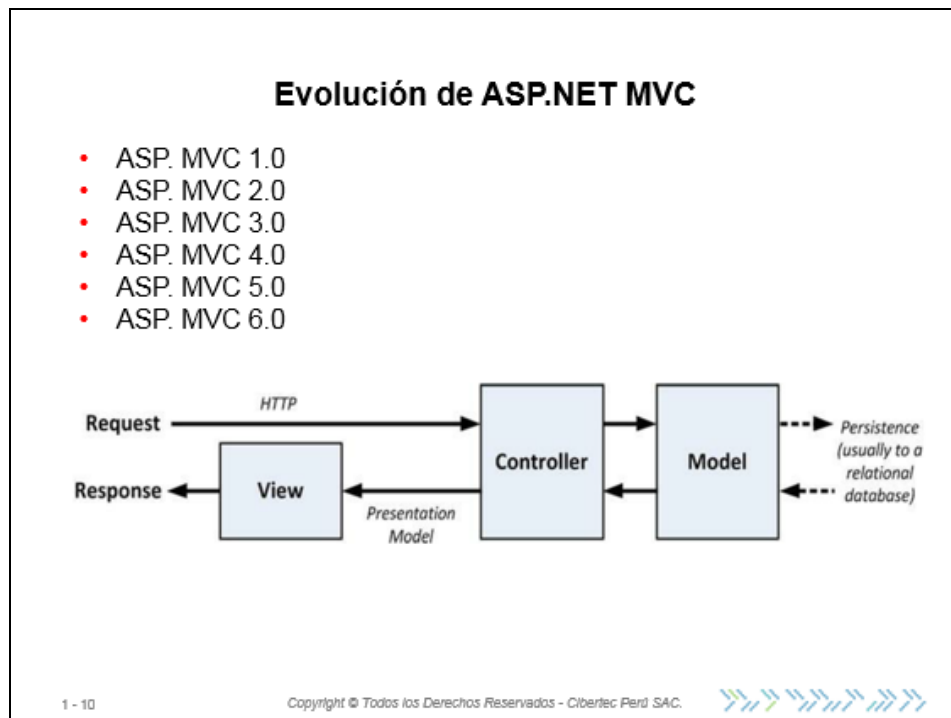


Folder wwwroot

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.

6. Evolución de ASP.NET MVC



El patrón de desarrollo ASP.NET MVC es la idea que inicialmente fue desarrollada por Scott Guthrie en 2007. Él es el arquitecto clave de este patrón de diseño. Por lo que entendemos, siempre se designa a Scott Gu, como el padre de .NET. Él es la persona principal que originó la idea de .NET. Desde entonces el equipo de Microsoft o de Scott ha añadido toneladas de mejoras y/o cambios en el desarrollo .NET, haciendo que los desarrolladores se sientan cómodos escribiendo cualquier tipo de proyecto utilizando .NET.

ASP.NET MVC 1:

Aquí, en esta sección, aparte del desacoplamiento de módulos, se agregaron pruebas unitarias. Así, mientras escribíamos cualquier proyecto, se podría escribir pruebas unitarias para módulos específicos de lado a lado. MVC 1 fue lanzado el 13 de marzo del 2009.

ASP.NET MVC 2:

Fue liberado justo después de 1 año, específicamente esta versión fue lanzada al mercado en marzo del 2010. La misma que traía consigo algunos cambios interesantes, los cuales nos permitían implementar MVC de una forma más robusta y poderosa. Algunas de sus características fueron:

- UI Helpers con plantillas Scaffolding.
- Modelo de validación Client/Server.

- HTML Helpers fuertemente tipados.
- Mejoras en las herramientas de Visual Studio.

ASP.NET MVC 3:

Fue lanzado oficialmente en 2011. MVC 3 llegó con importantes mejoras en muchas secciones. Algunas de sus características se enumeran a continuación:

- Inclusión de Razor View Engine.
- .Net 4 Data Annotations.
- Modelo robusto de binding y validación.
- Inclusión de Global Action Filters.
- Soporte Nice JS, jQuery. Además, se incluye validación Unobtrusive JavaScript.
- JSON Binding.
- Integración con NuGet para resolver las dependencias de software en el flujo del proyecto.

ASP.NET MVC 4:

Hizo una aportación tremenda a su antecesor MVC 3. Se convirtió en un paquete de desarrollo web íntegro en su conjunto, que ofrece suites completas de herramientas para los desarrolladores. Algunas de las características más comunes se enumeran a continuación:

- ASP.NET Web API.
- Mejora de plantillas de proyecto. Agregando algunos nuevos.
- Inclusión de plantillas de proyectos móviles utilizando jQuery Mobile.
- Variedad de modos de visualización.
- Controladores asincrónicos
- Agrupación y Minificación

ASP.NET MVC 5:

De nuevo ASP.NET MVC 5 ha hecho muchas adiciones importantes a MVC 4. Pero, todos estos cambios son en torno a sus plantillas scaffolding, técnica de autenticación, Bootstrap y muchos más. Algo de las características más comunes se enumeran a continuación:

- Scaffolding
- ASP.Net Identity
- Bootstrap
- Enrutamiento de atributos

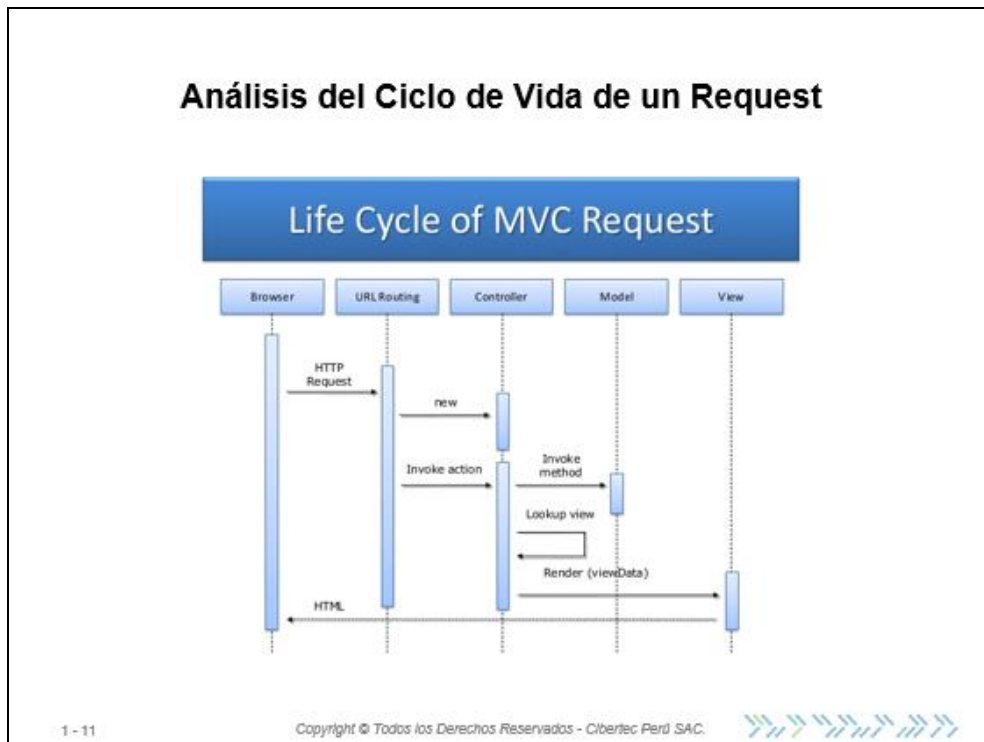
ASP.NET MVC 6:

ASP.NET MVC 6 en muchos términos es un marco único, ya que es un cambio importante en MVC. A continuación, algunas enumeraciones de las características de esta versión:

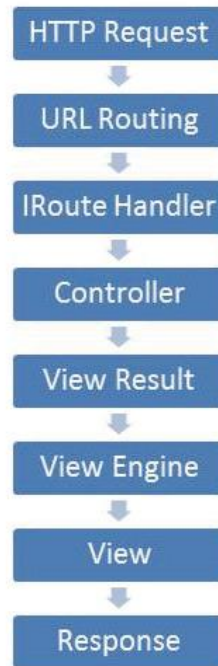
- Common Framework para MVC, Web API y Web Pages.
- Facil migración de Web Pages a MVC.
- Built DI First
- Ejecución de aplicaciones sobre IIS o Self Host.

- Está basado sobre el nuevo Request Pipeline en ASP.NET vNext.
- Ejecución en Cloud optimizada.
- Independencia de Builds.
- Experiencia de desarrollo mejorada.
- Open Source
- Soporte Cross-Platform (Plataforma cruzada).

7. Análisis del Ciclo de Vida de un Request



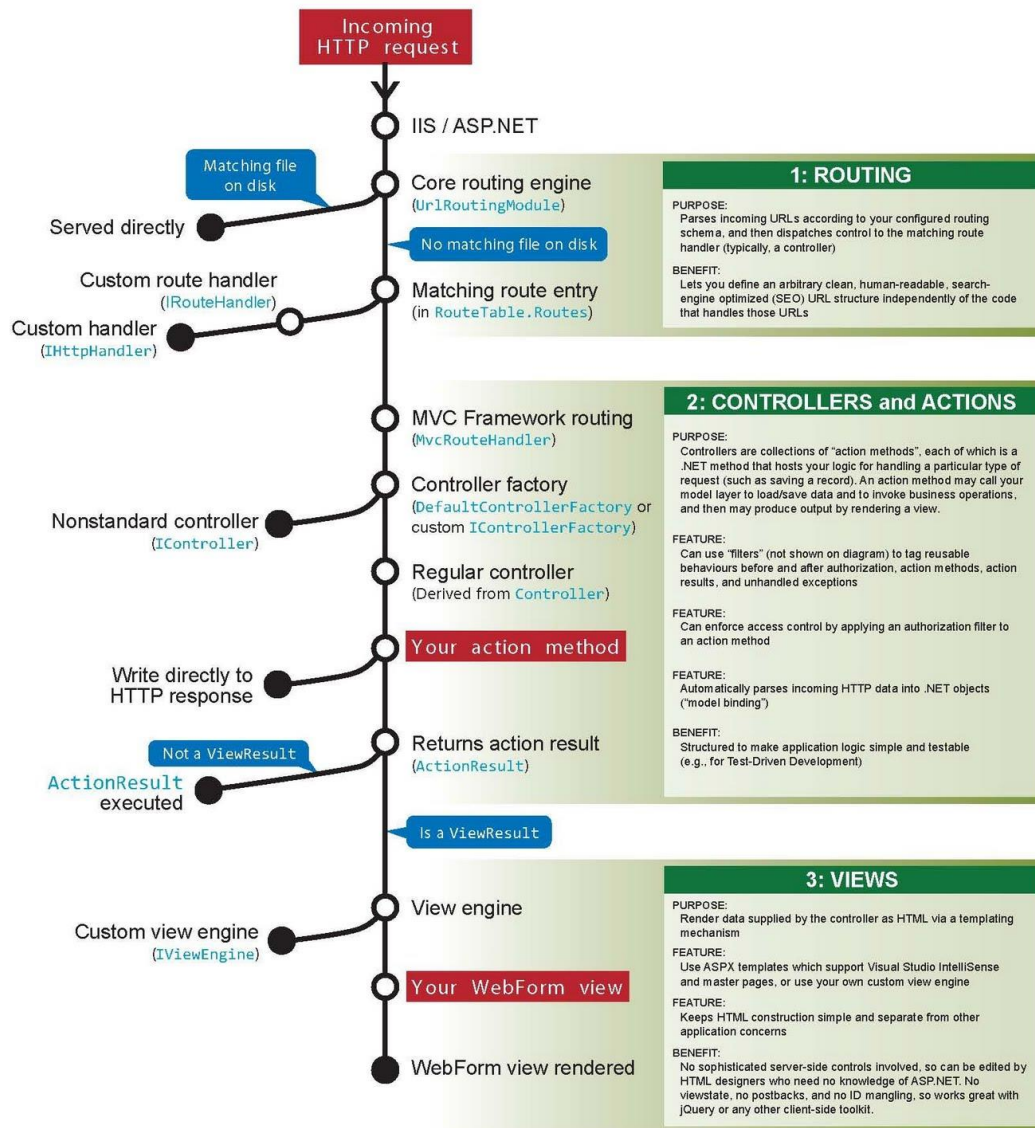
El ciclo de vida de MVC en pocas palabras le dará visión de la arquitectura en general. Aquí, en el siguiente diagrama se llega a saber cómo se comporta la aplicación MVC cuando se invoca desde el navegador. Así que, como puedes ver en el diagrama, tan pronto como la petición viene desde el navegador, es recogida por el motor de enrutamiento. Por lo tanto, ASP.NET Routing es el primer paso en el ciclo de vida MVC.



Básicamente es un sistema de coincidencia de patrones que coincide con el patrón de la petición en contra de los patrones registrados en el route table (tabla de rutas). Cuando existe una coincidencia de patrones se encuentran en la tabla de rutas, el routing engine reenvía la solicitud al correspondiente **IRouteHandler** para esa petición. Si el routing engine no coincide con el patrón, devuelve el código de error HTTP 404.

Entonces, el MVC handler implementa la interface **IHandler** y continúa procesando la solicitud invocando el método **ProcessRequest**. Por tanto, el tercer paso MVCHandler utiliza una instancia de **ControllerFactory** para intentar obtener una instancia de **Controller**. Si hasta este punto todo ha fluido sin ningún error, el método **Execute** es llamado. Ahora, una vez el controlador ha sido instanciado, el método **ActionInvoker** del mismo controlador verificará qué acción deberá invocarse. Seguidamente, el método sobre dicha acción recibirá una entrada del usuario, preparará la apropiada data de respuesta y ejecutará el resultado retornando un tipo de resultado.

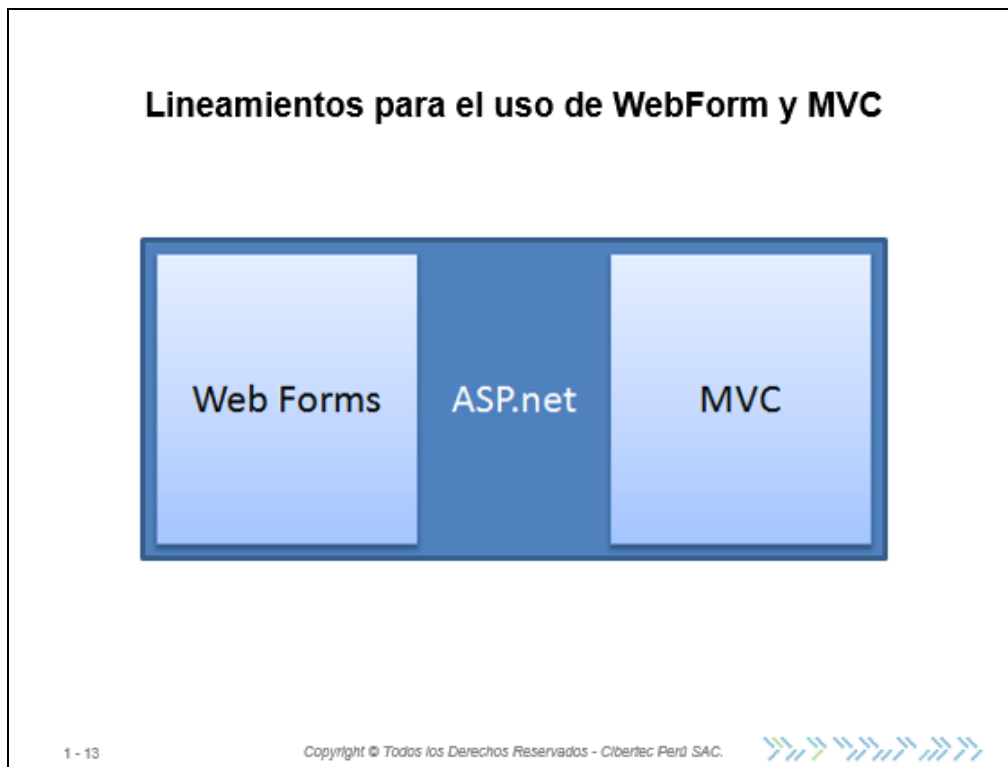
ASP.NET MVC: The Request-Handling Pipeline



Este tipo de resultado puede ser **ViewResult**, **RedirectToRouteResult**, **RedirectResult**, **ContentResult**, **JSONResult**, **FileResult**, and **EmptyResult**. Ahora, el posterior paso es la ejecución del View Result el cual abarca la selección del apropiado View Engine. Esto es básicamente manejado por la interface **IViewEngine** del View Engine.

Para finalizar, el método de la acción retornará a string, binary file o un JSON data. El Action Result más importante es el ViewResult, el cual renderiza y retorna una página HTML para el browser utilizando el view engine actual.

8. Lineamientos para el uso de WebForm y MVC



Hace mucho tiempo, con la ayuda de tecnologías como Visual Basic y Visual C++, Microsoft habilitó la creación de interfaces gráficas de forma rápida, con lo cual, los programadores podían enfocarse más en el negocio que en el diseño de las interfaces de usuario.

Esas tecnologías eran limitadas a las aplicaciones de escritorio, en el aspecto web, Microsoft solo ofrecía ASP.

Cuando hablamos de web y escritorio, hay que considerar dos cosas:

- Cómo se administra el estado?
- Cómo funciona el pedido/respuesta?

Las aplicaciones web funcionan sobre el protocolo HTTP, el cual es un protocolo sin estado. A diferencia del escritorio, no hay variables que se preserven y no hay una programación completa manejada por eventos. Como en el escritorio, la web espera la entrada de un usuario, pero cada interacción actúa como un pedido nuevo al servidor. En base a este escenario, Microsoft presentó ASP.NET Web Forms, manteniendo el desarrollo de aplicaciones rápido y la facilidad de aprendizaje.

ASP.NET es un framework de aplicaciones web creado por Microsoft, construido sobre el CLR (Common Language Runtime). Permite crear aplicaciones web dinámicas utilizando lenguajes como C# o Visual Basic.NET.

Web Forms

Los Web Forms aparecieron para solucionar los problemas que tenía la tecnología ASP clásica. Se creó un nuevo nivel de abstracción sobre la web sin estado, simulando el modelo con estado de los Windows Forms. Se introdujeron conceptos como postback (un envío desde la misma página) y ViewState (un objeto que almacena los valores de los controles durante los postbacks) y se redujo la cantidad de código a escribir.

Ventajas

- Los Web Forms contienen controles de servidor ricos: ASP.NET detecta el navegador del usuario y genera HTML y JavaScript apropiado para él. Además, controles como el GridView o el ListView, contienen funcionalidad para enlazar datos.
- ViewState: Normalmente los controles no retienen los valores entre los pedidos al servidor, pero en los Web Forms esto se logra guardando el último estado conocido en un campo oculto denominado ViewState.
- Programación manejada por eventos: Microsoft introdujo la programación manejada por eventos, con la cual, los desarrolladores no tenían que apoyarse sobre los métodos POST y GET para las interacciones con el servidor. Haciendo doble clic sobre un control, se genera un bloque de código que manejará el evento de dicho control en el servidor; sin que el desarrollador tenga que saber qué es lo que ocurre.
- Desarrollo de aplicaciones rápido: Con los controles de servidor + el modelo manejado por eventos + ViewState, el desarrollador queda abstraído de muchas de las complejidades, con lo cual programa más rápido.
- Curva de aprendizaje pequeña: Es posible hacer aplicaciones con muy poco conocimiento de HTML y JavaScript.

Desventajas

- Unit Testing: El Code Behind (archivo que contiene código a ejecutarse en el servidor) termina conteniendo muchos manejadores de eventos, lo cual hace que el unit testing automático sea una tarea imposible.
- Performance: ViewState se convirtió en una solución para varios problemas de ASP, pero como se guarda en la misma página hace que el tamaño de la misma se incremente y se reduzca la performance general.
- Reutilización: El código incluido en el Code Behind de un Web Form no es fácilmente reutilizable en otro.
- Poco control sobre el HTML: Muchas veces no está muy claro cuál es el HTML que resultará de la utilización de los controles, lo cual hace complicada la integración con frameworks como jQuery.
- SEO: No se generan URLs amigables, lo cual afecta el SEO de la página.
- Trabajo en paralelo: Como cada Web Form está atado a su Code Behind, no es recomendable que dos desarrolladores trabajen al mismo tiempo en estos archivos.

ASP.NET MVC

ASP.NET MVC es un nuevo framework para aplicaciones web creado por Microsoft, diseñado bajo la idea de la separación de responsabilidades y la posibilidad de implementar el testing. No posee ni ViewState, ni controles de servidor.

Ventajas

- Arquitectura de proyecto: Al estar fuertemente implementada la separación de responsabilidades, también tenemos una arquitectura del proyecto ordenada.
- Desarrollo manejado por tests: Los controles son clases separadas, por lo cual, es posible hacer test automáticos.
- Reutilización: Los controles no están atados a una vista, por lo cual pueden ser reutilizados.
- Performance: Las páginas son mucho más livianas comparadas con los Web Forms.
- Control total del HTML: Como no existen los controles de servidor, la única opción es utilizar los controles HTML, por lo que sabemos cómo se terminará renderizando la página. La integración con librerías JavaScript es realmente simple.
- Soporte para trabajo en paralelo: Al estar todo realmente separado, es posible que un desarrollador esté trabajando en una vista, mientras otro esté en el controlador y un tercero esté en el modelo, sin que interfieran entre ellos.
- SEO, URL Routing y REST: Las características de routing permiten utilizar interfaces REST.
- Extensión: Soporte para múltiples motores de vistas como aspx, razor, etc.
- Características preexistentes de ASP.NET: Como está construido sobre el framework ASP.NET, provee características como autenticación, caching, session, etc.

Desventajas

La principal desventaja de ASP.NET MVC es que presenta una curva de aprendizaje mucho más importante, por lo cual, puede ser muy difícil para los desarrolladores que recién están empezando.

Lineamientos de uso

Ambas tecnologías pueden ser las mejores, todo depende de los requerimientos, el problema dado y el conocimiento que tenga el equipo de desarrollo. Lo que es importante saber, es que una tecnología no suplanta a la otra.

Existen unos factores que pueden hacernos decidir por una o por otra:

- Quiero desarrollar la aplicación de forma rápida: **ASP.NET Web Forms**
- Quiero poder hacer Unit Testing: **ASP.NET MVC**
- Mi equipo tiene conocimiento de Windows Forms: Van a aprender mucho más rápido a utilizar **Web Forms** que MVC.
- Mi equipo no tiene conocimientos de las tecnologías Microsoft: Si ya utilizaban JSP, Ruby On Rails, etc. **MVC es la elección.**
- El uso de JavaScript va a ser importante: Si se va a utilizar mucho JavaScript, **MVC es más adecuado.**
- No tengo mucho conocimiento de tecnologías web: **ASP.NET Web Forms**

9. Herramientas para el desarrollo de Aplicaciones Web

Herramientas para el desarrollo de Aplicaciones Web

- Visual Studio 2015
- HTML5
- CSS3
- JavaScript
- JQuery
- Bootstrap
- Herramientas de desarrollo para Chrome, Firefox, IE, etc.



1 - 14

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.

Vamos a ver un resumen de las principales herramientas que ofrece Microsoft para el desarrollo.

- SharePoint Designer
 - Permite crear aplicaciones sobre SharePoint.
 - Viene incluido en el paquete Microsoft Office.
 - Permite crear aplicaciones bastante sofisticadas sin necesidad de programar.
 - No soporta la propagación de entornos.
 - No soporta test unitarios.
 - No soporta software de control de versiones
- Expression Web
 - Editor enfocado en diseñadores y desarrolladores front-end.
 - Permite crear páginas en ASP.NET y PHP.
 - Es compatible con Microsoft Visual Studio y Adobe Photoshop.
 - Permite probar páginas en navegadores no instalados en el sistema.
- Expression Blend
 - Permite crear aplicaciones XAML, Silverlight, WPF y Windows Phone.
 - No permite crear aplicaciones web HTML5, pero si Windows 8.
 - Se utiliza en conjunto con Visual Studio para crear las interfaces gráficas.
 - Viene incluido en Visual Studio 2015.

- WebMatrix
 - Es un IDE liviano alternativo a Visual Studio.
 - Integrado con el Web Platform Installer.
 - Permite elegir un desarrollo open-source desde una galería y usarlo como punto de inicio.
 - También permite desarrollar aplicaciones ASP.NET, Node.js y PHP.
- LightSwitch
 - Permite crear aplicaciones en Silverlight y HTML5.
 - Sirve para crear ABMs rápidos y casi sin código.
 - Las aplicaciones pueden ejecutarse en el navegador o como aplicaciones de escritorio.
 - Pensado para aplicaciones pequeñas, ya que posee bastantes limitaciones.
- Visual Studio Express for Web
 - Versión reducida de Visual Studio.
 - Es gratuito.
 - Diferencias con la versión de pago:
 - Muchos instaladores en vez de una sola aplicación: for Web, for Windows 8, for Desktop y for Windows Phone.
 - No tiene explorador de servidores.
 - No tiene soporte integrado para software de control de versiones.
 - Soporte para reportes limitado.
 - Soporte limitado para deploy.
 - Sin soporte para dispositivos móviles.
- Visual Studio 2015: Como vimos anteriormente existen 4 versiones
 - Visual Studio Community
 - Visual Studio Professional
 - Visual Studio Enterprise
 - Visual Studio Test Professional.

Y como agregado a lo que anteriormente hemos mencionado como características de este producto podemos decir también lo siguiente:

- **ASP.NET Core 5.0 framework** puede instalarse con su aplicación y lograr ejecutarse sobre múltiples dispositivos y plataformas. (Mac, Linux, Windows y tampoco no necesariamente en un web server). Los desarrolladores pueden también ahora construir sobre todos estos dispositivos utilizando otras herramientas tal como por ejemplo **Sublime Text**.
- **Plantillas de proyecto unificadas** para construir aplicaciones web.
- Nuevas y mejoras en **herramientas para Cloud** tanto desde Visual Studio como de Azure para desarrollar, trazabilidad, depuración, and actualizaciones en Cloud.
- **Compilación automática de cambios**, ahorrando mucho tiempo cada vez que cambiamos código y necesitamos ver el resultado en el browser.
- Mejora de las herramientas de desarrollo en **Internet Explorer**.
- Soporte de desarrollo para múltiples formas de renderizar las páginas para desktops, tablets, y phones.
- **Integración de web API** como un único modelo de proyecto para construir Web back ends.

- Soporte enriquecido para los frameworks tanto del lado del servidor como del cliente, tales como **jQuery**, **Ember.js**, y **AngularJS**.
- Soporte para herramientas como **Grunt** y **Bower** que se conectan directamente con Visual Studio.
- Mejora en el soporte de **NuGet** a través del diálogo de referencias (no más referenciar DLLs en nuestros proyectos), incluyendo **IntelliTrace** para paquetes NuGet.

10. Usando componentes de terceros con NuGet

Usando componentes de terceros con NuGet

- Permite gestionar los componentes de terceros (dll, js, css, etc.) que son usados en las aplicaciones.

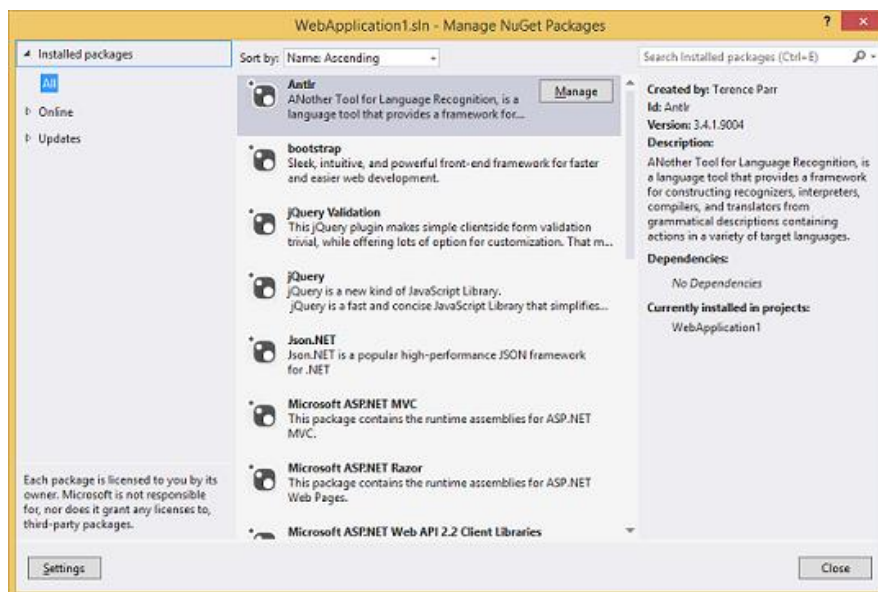


1 - 15

Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.



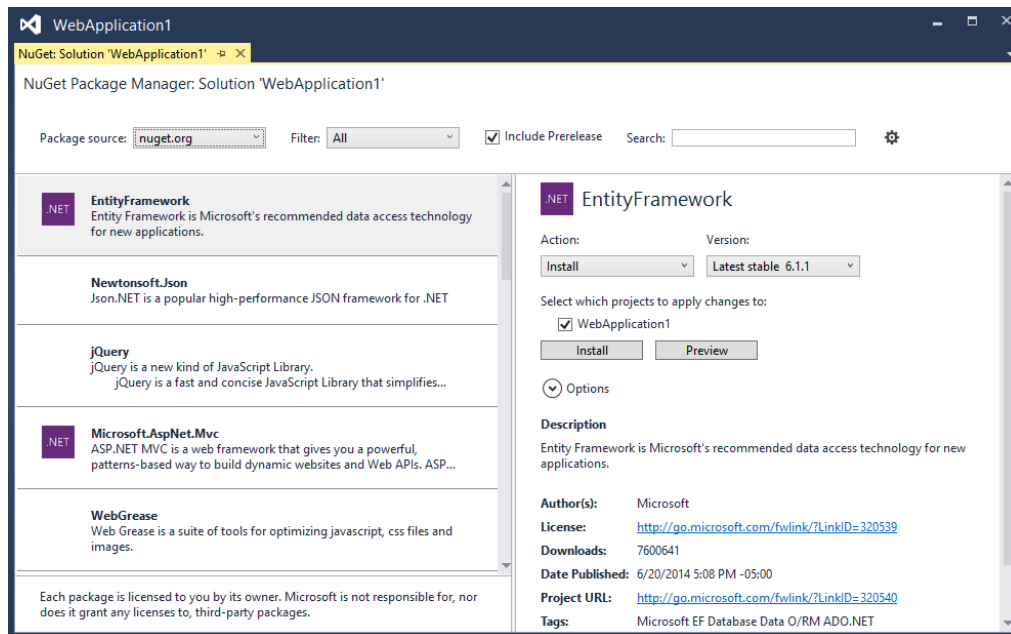
Uno de los cambios que vienen con Visual Studio 2015 es como se manejan los paquetes en NuGet. En Visual Studio 2013 teníamos algo como esto en la gestión de paquetes para una solución.



Esta interfaz de usuario proporciona información útil como.

- Qué paquetes están instalados y en qué proyectos
- Qué paquetes tienen actualizaciones
- Qué paquetes están disponibles

En Visual Studio 2015 se obtiene una interfaz de usuario completamente diferente.



Una de las primeras cosas a destacar es que ya no es una ventana modal, sino de una ventana de documento normal. Cómo moverse en la interfaz de usuario puede ser un poco confuso al principio así que vamos a echar un vistazo a cómo acceder a toda la información que teníamos en las versiones anteriores.

Instalando Paquetes

Se realiza desde esta misma ventana, ya sea seleccionando un paquete común de la izquierda o por la búsqueda de la parte superior. Tenga en cuenta que las fuentes de paquetes configurados se encuentran en la parte superior ahora, en vez de desglosarse en una estructura de árbol.

Una vez que haya encontrado el paquete que desea instalar es necesario asegurarse de la acción, el lado derecho debe estar establecido en Instalar (por defecto). A diferencia de Visual Studio 2013 también puede seleccionar la versión específica de instalar. Con Visual Studio 2013 si quería instalar versiones previas a la última tenía que hacerlo por la consola. Ahora puede seleccionar la versión adecuada de todo dentro de la misma interfaz de usuario. Por último, puede seleccionar el proyecto(s) donde el paquete se instalará. Actualmente, se muestran todos los proyectos de la solución y no proporciona ninguna manera de seleccionar o anular la selección de todos los proyectos. Además, la lista de proyectos se muestra de forma desordenada lo que puede ser tedioso cuando tengamos mayor cantidad de proyectos en nuestra solución.

Cuando esté listo para instalar, haga clic en el botón Instalar y se ejecutará a través del proceso de instalación. Si desea ver una vista previa de los cambios a continuación, utilizar Vista previa en su lugar.

Dependencias

Una de las nuevas características es cómo se instalan las dependencias. Esto se encuentra en el menú Options. Actualmente, las opciones incluyen obtener la versión más alta o baja de un paquete o conseguir la versión secundaria más alta del paquete dependiente. Esto es útil cuando se quiere usar los últimos paquetes, pero no necesariamente a una mayor, y posiblemente con versiones incompatibles. Usted también tiene la opción de determinar lo que ocurre si se producen conflictos entre archivos.

Administrando Paquetes

Una de las cosas confusas que encontraremos la primera vez que trabajemos con la nueva interfaz de usuario es cómo determinar qué paquetes ya están instalados. Esto es útil cuando usted está tratando de diagnosticar problemas. Para llegar al paquete instalado cambie el combo de filtro en la parte superior como **Installed**. Ahora solo verá los paquetes que están instalados.

Las opciones de acción han cambiado también. Ahora puede actualizar a una versión diferente o desinstalar el paquete de uno o más proyectos.

Actualizando Paquetes

Mientras que usted puede seleccionar cada paquete debajo de Installed y proceder a actualizarlos manualmente, esto es un proceso engorroso y lento. En lugar de ello, considere cambiar el filtro a **Update Available** para obtener la lista de paquetes que tengan alguna actualización pendiente.