# The Next Generation of .NET – ASP.NET vNext

Today at TechEd North America, we announced the latest set of innovations that are part of the next generation of .NET. The biggest of those is ASP.NET vNext, which is an updated version of ASP.NET that been optimized for cloud Web development. We’ve continued to improve the core .NET technologies that we shared at Build last month, specifically the [.NET Native](http://blogs.msdn.com/b/dotnet/archive/2014/04/02/announcing-net-native-preview.aspx) ahead-of-time compiler and the [.NET Next Generation JIT (“RyuJIT”)](http://blogs.msdn.com/b/dotnet/archive/tags/ryujit/). Both have new releases you can try out. We also have a set of smaller announcements to share.

At Build last month, we announced the .NET Foundation. We are currently talking to over 25 community-based .NET projects and organizations about joining the foundation. The interest in the foundation has exceeded our expectations and is off to a great start.

We also announced the [.NET Compiler Platform ("Roslyn")](http://roslyn.codeplex.com/) at Build. It includes new C# and VB compilers and a preview of new language features being considered for C# 6. The project is open source on codeplex and has accepted its first pull requests from the community.

The final release of [Visual Studio 2013 Update 2](http://go.microsoft.com/fwlink/p/?LinkID=390465) is now available. Update 2 brings dozens of [significant new features](http://blogs.msdn.com/b/somasegar/archive/2014/02/25/visual-studio-2013-update-2-ctp2.aspx) to Visual Studio developers, including [tools for Windows Phone 8.1 and universal Windows apps](http://blogs.msdn.com/b/somasegar/archive/2014/04/02/visual-studio-2013-update-2-rc-universal-projects-for-windows-and-windows-phone.aspx).

While we’re looking forward, it’s also good to look at one of the current strengths of .NET. There are currently 1.8 billion active installs of .NET. That’s a very large number by any measure and a great base of desktop and server machines on which to run your apps.

# TechEd .NET Announcements

Here’s a quick rundown of the .NET announcements that we shared at TechEd.

* [.NET vNext](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_.NET_vNext)
* [ASP.NET vNext (MVC, Web API and Web Pages 6; EF 7; SignalR 3)](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_ASP.NET_vNext)
* [.NET Framework 4.5.2](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_Additional_.NET_Framework)
* [.NET Native Developer preview 2 – x86 support](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_Additional_.NET_Framework)
* [.NET Next Generation JIT CTP 4 – supported on Windows 7](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_Additional_.NET_Framework)
* [Better support for Xamarin in .NET PCL NuGet Packages](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_Targeting_Multiple_Platforms)
* [ApiPort API Portablity Analyzer](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_Targeting_Multiple_Platforms)
* [Client Libraries for Office 365 REST APIs](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#_Client_Libraries_for)

# 

# .NET vNext

TechEd is the first time we’re talking about .NET vNext, as the next major release of the .NET Framework. At Build and TechEd, we’ve shared many of the features and components that you can expect in the next release. You will be able to compile C# 6 and VB with the [Roslyn compilers](http://blogs.msdn.com/b/csharpfaq/archive/tags/roslyn/), host ASP.NET vNext apps on the server or cloud, compile your Windows Store apps with the [.NET Native](http://blogs.msdn.com/b/dotnet/archive/tags/dotnetnative/) ahead of time compiler, and enjoy faster desktop and server apps with the [Next Generation JIT](http://blogs.msdn.com/b/dotnet/archive/tags/ryujit/).

We’ve optimized .NET for the mobile-first and cloud-first development options that have become more common today. Device and cloud apps come with significant user expectations around performance, and also run in more specialized hardware/virtual environments. For Windows Store apps, we built the .NET native ahead of time compiler. For cloud apps, we’ve developed a cloud optimized mode.

.NET vNext will have a cloud optimized mode that enables you to deploy your apps with a copy of the .NET Framework libraries they need. Since the runtime and framework libraries are deployed on an app-basis, each app can run different versions of .NET vNext side-by-side and upgrade separately, all on the same machine. These libraries have been slimmed down significantly to reduce the footprint of the framework, and will be distributed via NuGet. Also, libraries such as WPF and Windows Forms have been removed from this mode.

We're developing this with cross-platform in mind, including an active collaboration with Xamarin to ensure that cloud-optimized .NET applications can run on Mac or Linux on top of the Mono runtime. The great productivity of .NET and ASP.NET can be available to teams working in mixed development environments.

# ASP.NET vNext

ASP.NET vNext is our big announcement at TechEd. We’ve updated many aspects of ASP.NET to make ASP.NET apps easier to build and perform significantly better. We’ve considered sites and services that get infrequent traffic and those that get bombarded by visitors all day long. We’ve also opened up new scenarios that were just not possible with ASP.NET before.

We’ve designed ASP.NET with a key set of design principles in mind:

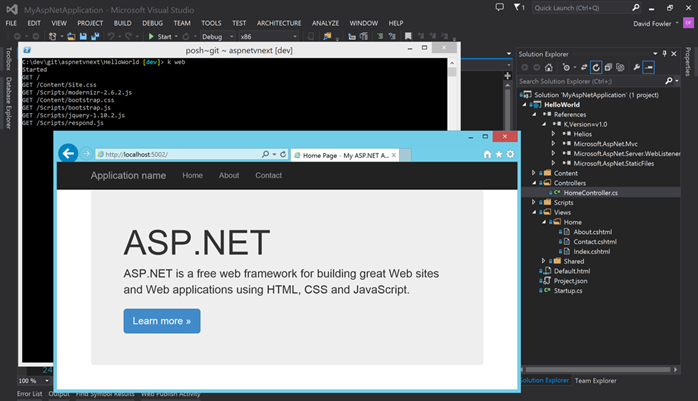
* Cloud-ready out of the box
* A single programming model for Web sites and services
* Low-latency developer experience
* Make high-performance and high-productivity APIs and patterns available – enable them both to be used and compose together within a single app
* Fine-grained control available via command-line tools and standard file formats
* Delivered via NuGet
* Release as [open source](https://github.com/aspnet/Home) via the [.NET Foundation](http://www.dotnetfoundation.org/)
* Can run on Mono, on Mac and Linux

ASP.NET vNext includes updated versions of MVC, Web API, Web Pages, SignalR and EF. The key improvement with these frameworks is that MVC, Web API and Web Pages have been merged into a single programming model. For example, there’s now unified controller and routing concepts between all three. You can now have a single controller that returns both MVC views and formatted Web API responses, on the same HTTP verb.

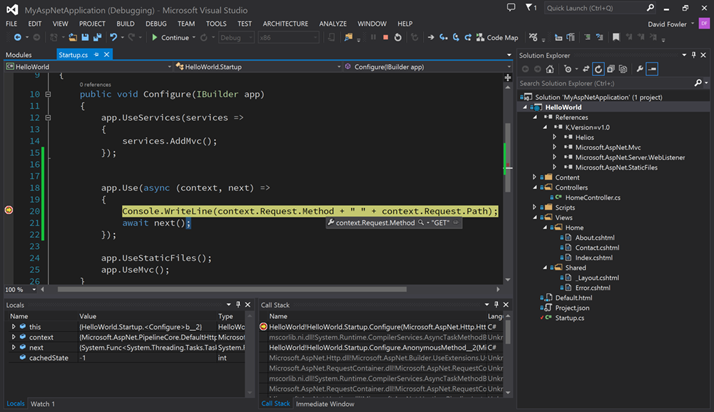
ASP.NET vNext apps are cloud ready by design. Services such as session state and caching adjust their behavior depending on whether the app is running in the cloud or in a traditional hosting environment, while providing a consistent API. We use dependency injection behind the scenes to provide your app with the correct implementation for these services. Using this approach, it is really easy to move your app from on-premises to the cloud, since our code changes, not yours.

You will be able to make changes to your web applications and see the results after a browser refresh, with no separate build step needed. This significant productivity enhancement is based on improvements to load times in the underlying CLR, as well as use of the new .NET Compiler Platform ("Roslyn").

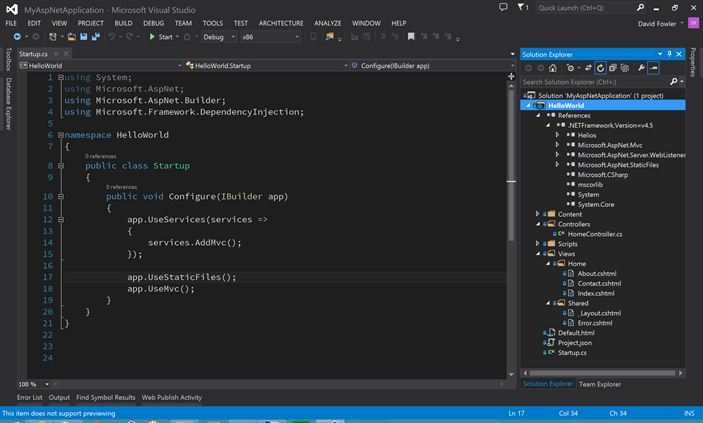
You can see the ASP.NET vNext in action in the images below. The first image shows an ASP.NET vNext Hello World app hosted from the command-line and drive from browser. Any edits in Visual Studio will be compiled and executed when the browser is next refreshed. This app is using .NET vNext cloud optimized mode.



You can also use Visual Studio with a more traditional Visual Studio F5 workflow that automatically starts the webserver and launches the browser. This is the same app as shown in the image above.



In the image below, the app has been re-configured to run on the full .NET vNext framework, instead of the cloud optimized mode. That’s just a setting on the project. The app now has access to all of the APIs in the .NET Framework. The browser refresh feature is available in this configuration, too.



The table below outlines the ASP.NET vNext scenarios we’ve built and where they are available.

|  |  |  |
| --- | --- | --- |
| **ASP.NET vNext Feature** | **On .NET vNext** | **On .NET vNext (Cloud Optimized)** |
| Cloud Ready | \* | \* |
| Modular Design | \* | \* |
| Dependency Injection | \* | \* |
| Consistent Tracing / Debugging | \* | \* |
| Faster Development (browser refresh) | \* | \* |
| Open Source | \* | \* |
| Full Side by Side (runtime and framework  deployed with application) |  | \* |
| Faster startup, Lower memory / Higher throughput (best of class) |  | \* |
| Uses a smaller set of framework libraries |  | \* |
| Enabled on Mono, on Mac and Linux |  | \* |

ASP.NET vNext will be open source and will be contributed to the [.NET Foundation](http://www.dotnetfoundation.org/). This shouldn’t come as a big surprise since the [ASP.NET Web stack](http://aspnetwebstack.codeplex.com/) is already open source. All of ASP.NET vNext will be delivered via NuGet, will be open source and will take contributions. Read [ASP.NET vNext: the future of .NET on the Server](http://blogs.msdn.com/b/webdev/archive/2014/05/13/asp-net-vnext-the-future-of-net-on-the-server.aspx) to learn more.

Our announcement at TechEd is the first stop for .NET vNext and ASP.NET vNext. We’ll share much more in the months to come before we release the final versions. We’re looking forward to shipping pre-release versions in order to get your feedback.

# 

# Additional .NET Framework Updates and Improvements

We recently announced the [.NET Framework 4.5.2](http://blogs.msdn.com/b/dotnet/archive/2014/05/05/announcing-the-net-framework-4-5-2-release.aspx). It including significant improvements in ASP.NET and Windows Forms and other areas of the product. You can start incorporating 4.5.2 features into your apps now.

We released several key improvements to [ASP.NET Web Forms](http://blogs.msdn.com/b/webdev/archive/2014/05/13/improvements-to-asp-net-web-forms.aspx). These include support for ASP.NET Identity 2.0, Entity DataSource control for Entity Framework 6 and Roslyn support.

We have also added new features and scenarios to both [.NET Native](http://blogs.msdn.com/b/dotnet/archive/tags/dotnetnative/) and the [Next Generation JIT](http://blogs.msdn.com/b/dotnet/archive/tags/ryujit/). .NET Native now support x86, in addition to ARM and x64 apps. The Next Generation JIT now supports Windows 7 and later for x64 apps. Both of these technologies are a critical part of our .NET vNext roadmap. Expect to hear more about both of them in the coming months.

* [Get .NET Native Developer Preview 2](http://aka.ms/dotnetnative)
* [Get Next Generation JIT (“RyuJit”) CTP 4](http://aka.ms/rj-ctp4)

# Targeting Multiple Platforms

We’ve been working for several years to make it easier to write code for multiple platforms, both as apps and libraries. We started by [enabling our PCL reference assemblies for Xamarin](http://blogs.msdn.com/b/dotnet/archive/2013/11/13/pcl-and-net-nuget-libraries-are-now-enabled-for-xamarin.aspx), who quickly moved forward with [that change](http://blog.xamarin.com/microsofts-pcl-reference-assemblies/). More recently, we’ve been working closely with Xamarin to make our [.NET NuGet packages](https://twitter.com/DotNet/status/464441664291893249) work better with Xamarin tools, to make it easier to build .NET apps for iOS and Android. There’s still work to do, but the experience has gotten much better and will continue to improve.

At TechEd, we announced a new [portability analysis tool, called ApiPort](http://go.microsoft.com/fwlink/?LinkID=398760). It provides you with two main pieces of data: the platforms that you can easily/reasonably target with your code, and the dependencies that are preventing you from targeting additional platforms.

The command line tool generates an Excel report that provides you with two views of its portability analysis. It provides a high-level color-coded view for a given set of platforms. It also provides a very detailed list of all the types and members used within your code, and whether they are supported, per platform. Given that the report is in Excel, it is very easy to filter the list, build pivot tables and do whatever else you want to perform further analysis.

The image below shows a small sample of the high-level portability analysis view. Only one assembly is shown, but there can be multiple. Check out this [sample portability analysis](https://dotnetweb.blob.core.windows.net/platforms/ApiPortAnalysis.xlsx) to get a first-hand view.

clip_image008

The tool has another function, too. All of the dependency data (not the assemblies) are uploaded to an Azure service that the .NET team maintains. The data that the tool uploads is the list of assemblies and APIs that your code relies on. We do not record where the data came from or by whom. We do not upload any of your actual code or binaries. We want to know which functionality we need to bring to each platform to make it easier to target all platforms.

If you are finding it difficult to target a particular platform, please “vote” for the APIs you want added to a particular platform by running the tool on your app and libraries. It’s really easy to run the tool on a whole directory.

This first release of tool is missing a few features that we are in the process of adding. The Xamarin/Mono platforms are currently missing from the tool. It also doesn’t yet take into consideration NuGet packages that make .NET Framework APIs available on other platforms, counting them as missing APIs.

* [Get the ApiPort API portability analyzer](http://go.microsoft.com/fwlink/?LinkID=398760)

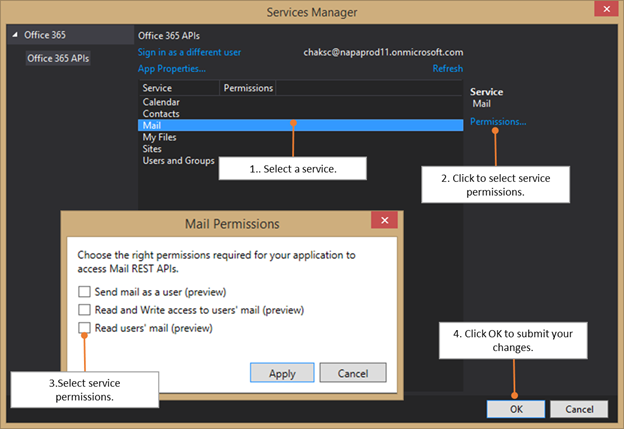
# Client Libraries for Microsoft Services

You’ve probably heard that Microsoft is both a “services first” and “devices first” company. At the nexus of those statements are client libraries that make it easy to target Microsoft services from apps. While [Client libraries for Microsoft services](http://www.nuget.org/packages?q=windowsazureofficial) are not new, we are more recently working to provide support for more platforms. We’ve started with Office 365 services, and intend to add more services over time.

At TechEd, we announced a preview release of a new set of client libraries for Office 365, for .NET and JavaScript. You can read the announcement for the new [Office 365 client libraries](http://aka.ms/ClientLibraries) on the [Office Developer Blog](http://blogs.office.com/dev/). The .NET Client libraries support WPF, Windows Forms, Windows Store, Xamarin.iOS, Xamarin.Android and ASP.NET apps and Portable Class Libraries, and are delivered via NuGet.

* [Microsoft Office 365 Mail, Calendar and Contacts Library for .NET](https://www.nuget.org/packages/Microsoft.Office365.Exchange/)
* [Microsoft Office 365 My Files Library for .NET](https://www.nuget.org/packages/Microsoft.Office365.MyFiles/)
* [Microsoft Office 365 Users and Groups Library for .NET](https://www.nuget.org/packages/Microsoft.Office365.UsersGroups/)
* [Microsoft Office 365 Authentication Library for .NET (Windows Store, Windows Desktop)](https://www.nuget.org/packages/Microsoft.Office365.OAuth.Windows/)
* [Microsoft Office 365 Authentication Library for ASP.NET](https://www.nuget.org/packages/Microsoft.Office365.OAuth.AspNet/)
* [Microsoft Office 365 Authentication Library for .NET (Android and iOS)](https://www.nuget.org/packages/Microsoft.Office365.OAuth.Xamarin/)

We have also provided an integrated experience for adding these libraries to your apps within Visual Studio. Services require app registration, permission selection and a platform-specific user authentication experience. You also need to add the right client libraries to your app. The Visual Studio experience takes care of all of that for you, as part of the Connected Services Manager, displayed below.



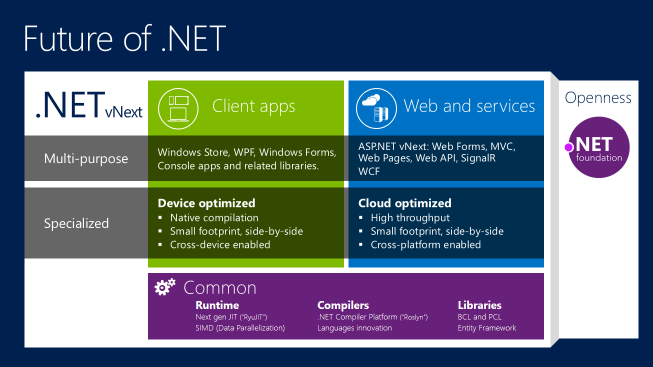
You can learn how to start using this experience from the [Office Developer blog](http://aka.ms/ClientLibraries). We are using [StackOverflow](http://stackoverflow.com/questions/tagged/Office365APIs) as the community forum for this project, under the [Office365APIs tag](http://stackoverflow.com/questions/tagged/Office365APIs). Please tell us what you think about the client libraries and the new Visual Studio integration. Do note that these libraries, and the Office services they are targeting, are in Preview and don’t yet support production apps.

# Summary

On the .NET team, we’re excited to be sharing the Next Generation of .NET. As you can see from this post and the one we [published at Build](http://blogs.msdn.com/b/dotnet/archive/2014/04/03/the-next-generation-of-net.aspx), we are lining up a significant set of technologies, features and scenarios that will be part of .NET vNext, our next major release of the .NET Framework. In this post, we focused on ASP.NET vNext, our next generation Web and services platform.

For .NET vNext, we’ve been talking about major features and asking for feedback before we even announced the larger release. We’ve been actively engaging with experts and enthusiasts in our preview and pre-release programs to validate our ideas and product directions. It’s turned out to be a great approach. The feedback has been incredible. Thanks to everyone that has participated in our CTPs, developer previews, pre-releases and any other program we ran to collect feedback. We’ve also received great feedback in blog comments. It’s all been very helpful. Do expect multiple additional preview releases over the coming months, particularly for ASP.NET vNext. It’s going to be a fun time.

You can see how these technology investments come together into a single .NET Framework release, in the following slide that we showed at TechEd. Open Source is also an important part of our plans, as you can see with our ASP.NET vNext contribution plans with [the .NET Foundation](http://www.dotnetfoundation.org/). The future looks very bright for .NET.



* [56 Comments](http://blogs.msdn.com/b/dotnet/archive/2014/05/12/the-next-generation-of-net-asp-net-vnext.aspx#comments)



|  |  |
| --- | --- |
|  |  |

[clr](http://blogs.msdn.com/b/dotnet/archive/tags/clr/), [releases](http://blogs.msdn.com/b/dotnet/archive/tags/releases/), [performance](http://blogs.msdn.com/b/dotnet/archive/tags/performance/), [announcement](http://blogs.msdn.com/b/dotnet/archive/tags/announcement/), [portable class libraries](http://blogs.msdn.com/b/dotnet/archive/tags/portable+class+libraries/), [conferences](http://blogs.msdn.com/b/dotnet/archive/tags/conferences/), [nuget](http://blogs.msdn.com/b/dotnet/archive/tags/nuget/), [windows store](http://blogs.msdn.com/b/dotnet/archive/tags/windows+store/), [asp.net web api](http://blogs.msdn.com/b/dotnet/archive/tags/asp-net+web+api/), [asp.net](http://blogs.msdn.com/b/dotnet/archive/tags/asp-net/), [visual studio](http://blogs.msdn.com/b/dotnet/archive/tags/visual+studio/), [xamarin](http://blogs.msdn.com/b/dotnet/archive/tags/xamarin/), [ryujit](http://blogs.msdn.com/b/dotnet/archive/tags/ryujit/), [jit](http://blogs.msdn.com/b/dotnet/archive/tags/jit/), [compiler](http://blogs.msdn.com/b/dotnet/archive/tags/compiler/), [azure](http://blogs.msdn.com/b/dotnet/archive/tags/azure/)