PART I

ASP.NET Fundamentals

1 One ASP.NET

WHAT'S IN THIS CHAPTER?

- Introducing One ASP.NET
- Exploring and simplifying the complex web ecosystem
- Benefiting from One ASP.NET

ASP.NET has been a prominent web development framework that has supported .NET developers in building web applications for years. As the Internet has matured and progressed, so have the ASP.NET framework and the ecosystem around it. Thousands of products, services, and open source projects call ASP.NET home.

The Internet has advanced much faster than ASP.NET. The advancements in HTML, CSS, and JavaScript have led to richer experiences for users, which put pressure on ASP.NET to support these emerging technologies. The ASP.NET framework has had to support the growing needs of web developers. Although all this has led to the development of a healthy ecosystem where web developers from all over have come together to make it better, it's also caused core pieces of ASP.NET to show its age.

This chapter looks at how ASP.NET as a framework has matured, and proposes some ways to fix the problems with the ecosystem. How can ASP.NET be a trusted and reliable framework that the enterprise developer can count on, while still meeting the needs of the advanced web developer who demands the latest standards? The goal of the chapter is to give you an idea of some of the changes that have happened in ASP.NET and where the framework is headed.

INTRODUCING ONE ASP.NET

A few years back when ASP.NET MVC was just coming out, a number of people told me they were being discouraged from making "hybrid" applications. They wanted ASP.NET apps with services components, MVC areas, and Web Forms pieces. Some Microsoft employees made them feel that this was an unsupported thing. "Why would you want to do that? That's not a good idea."

But it is. ASP.NET MVC is ASP.NET. Web Forms is ASP.NET. The pipeline set in place more than 10 years ago is with us today, offering many extensibility points that have been exploited not only by ASP.NET but also by alternative frameworks like NancyFx and ServiceStack.

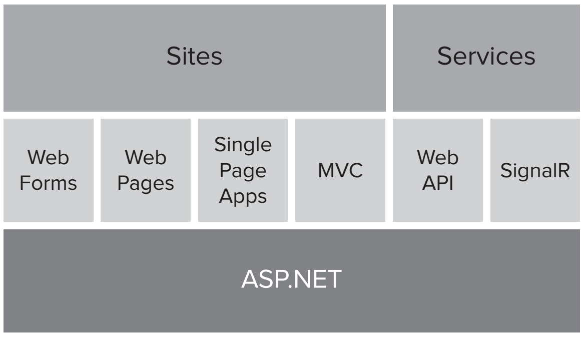


FIGURE 1-1

Why make developers choose from a half dozen different flavors when it's all the same underlying menu? ASP.NET offers core services that you need regardless of your application architecture. Key management, session management, caching, HTTP, authorization, and authentication are all universal web development truths. How you choose to render your angle brackets or curly braces is your preference, but you should feel comfortable using whatever ASP.NET framework or frameworks help you solve your business problem. You should be able to do this not only without guilt, but also in a totally supported and encouraged way. Figure 1-1 shows a breakdown of the different frameworks in ASP.NET and how they all use a common underlying ASP.NET core.

One ASP.NET is all about making it easier for developers to build applications using ASP.NET and Visual Studio. One ASP.NET is a general term and can mean a lot of things. Let's break it down.

In today's world, you have a lot of choices and options when you think about building web applications. Each one has its own merits and drawbacks. When you set out to build applications, how do you know which framework to choose?

If you choose one and then realize it was not a good choice, how easily can you come back and go another route? Was your framework modular enough for you to swap out the offending piece, or did you have to rewrite everything? Are you able to share code between subsystems?

How often should your framework update itself? The Internet moves far faster than most any framework can handle. How can ASP.NET serve two masters? You want HTML5 and CSS3 now, but you can't break an app your business relies on.

One ASP.NET should give developers the confidence that, no matter what they choose, they are still developing on a trusted underlying framework—ASP.NET. To help you understand One ASP.NET, the following section explores how we got here.

Evolution of ASP.NET

Web development for Microsoft started with Classic ASP (Active Server Pages) in the late 90s. A developer was able to write dynamic pages based on the VB scripting language.

ASP.NET 1.0 was introduced in 2002 and was built on top of the .NET Framework. ASP.NET Web Forms simplified developers' transition from Windows application development to web development by enabling them to build pages composed of controls, similar to a Windows user interface. It was unlike anything we'd ever seen! Visual Basic developers who were used to dragging data grids onto Windows Forms could drag data grids onto Web Forms! It was magic.

It was magic because it layered state on top of stateless HTTP. It allowed developers to react to business-level events like `Button.Click` rather than low-level events like HTTP POST, enabling a whole new generation of developers to create great web applications.

ASP.NET 2.0 was released in 2005 along with Visual Studio 2005. This release took the controls metaphor even further with data-focused enhancements like the SqlDataSource and ObjectDataSource. ASP.NET 3.5 followed in 2008 and brought with it DynamicData, which enables you to rapidly generate data-driven applications in minutes.

Shortly thereafter, ASP.NET MVC was released and introduced the popular Model-View-Controller pattern to ASP.NET web development for the first time. Still based on ASP.NET, ASP.NET MVC provided a level of testability and composition as yet unseen on the .NET platform. ASP.NET MVC was developed to meet the growing needs of a developer community that wanted to easily unit-test their applications and have more granular control over how HTML markup was generated. ASP.NET MVC represented a level of absolute control and power that brought a new kind of developer to the .NET Framework, and perhaps also allowed some older, disenfranchised developers to stay on ASP.NET.

A few things were interesting about MVC. ASP.NET MVC was one of the first releases for the ASP.NET framework where a part of the framework was released by itself "out of band" (OOB). It meant that ASP.NET MVC was released as a standalone installer that developers could optionally download and add onto their existing Visual Studio installation. This was a huge step toward making it possible for developers to get new products quickly rather than waiting for 2 to 3 years. The ASP.NET team didn't realize it was happening, but ASP.NET MVC as an out-of-band release arguably marked the beginning of the ASP.NET team's break from the Visual Studio "ship train."

Over the next 2 years, two more versions of ASP.MVC were released. More significantly, ASP.NET MVC was released under the Microsoft Public License (MS-PL), which allows developers to see the source of their favorite framework. Whispers began that perhaps ASP.NET MVC might be released as proper open source, but Microsoft would never do that, right?

ASP.NET 4 was released along with VS2010. At the same time ASP.NET MVC 4 was released, ASP.NET Web Pages was introduced with a fantastic new syntax for dynamic page creation called Razor. ASP.NET Web Pages and the new Razor syntax provide a fast, approachable, and lightweight way to combine server code with HTML to create dynamic web content. Razor became the new view engine behind ASP.NET MVC. ASP.NET Web Pages brought Razor to a simple new programming model suitable for smaller sites or folks just getting started.

ASP.NET 4.5 was released along with VS2012 in 2012. This release included ASP.NET MVC 4.5 and a new member into the ASP.NET family called ASP.NET Web API, which made it easier to write REST-based web services.

Shortly thereafter, Microsoft released the majority of its Web Stack (including ASP.NET MVC, Razor, and ASP.NET Web API) under an open source license (Apache License 2.0). But it also announced it would be taking contributions from the community—real open source on a strategically important and flagship web framework. The big ship was starting to turn.

This was the beginning of "development in the open." An interested community member can actually read the code checkins while the developers are working inside Microsoft. Even better, contributors can fix bugs, add features, and submit pull requests knowing their code could be used by millions of developers.

Open development was initially scary, but doing so enabled a more open development model where everyone is engaged and can provide feedback on code checkins, bug-fixes, new feature development, and build and test the products on a daily basis using the most up-to-date versions of the source code and tests. This was a gigantic step in getting the ASP.NET ecosystem to come together in a way that developers could work together toward building a better framework.

At this point in the story, we have a large part of ASP.NET and its runtime components shipping out of band as open source. Many features are optional, or easily added on, like Web Optimization, Universal Membership Providers, and test harnesses. How can the tooling be opened up to support a simultaneously more uniform but modular ASP.NET?

The Web Evolves and We Evolve with It

HTML5 isn't done, and unless this chapter is published in 2017, it still won't be done at the time of this writing. However, HTML5, along with its siblings CSS3 and JavaScript, have won the Internet. Now that a reliable and complete version of JavaScript is available on every modern browser, combined with a cohesive document object model (DOM), applications have changed their architecture dramatically.

JavaScript has moved beyond simple `alert()` statements and input type validation and graduated to being a near-complete virtual machine in the browser. HTML5 is very simple as a spec, and CSS3 makes it shine. Take all these ingredients and combine them with a pocket super computer, and you've got a mobile web revolution.

Today, customers are using different form factors, ranging from smartphones to tablets to personal computers. This means that they can access a web application using a phone or tablet, or using a mobile browser, or using the traditional desktop browser. The web is going more social as well. Users are interacting with each other through social means such as Twitter, Facebook, and so on, and they want to carry their social identities in all web applications with which they interact. This means that they want to log in using their social credentials, so that when they log in to the application they can interact with their friends as well. It's a world of connected devices and services for the web developer.

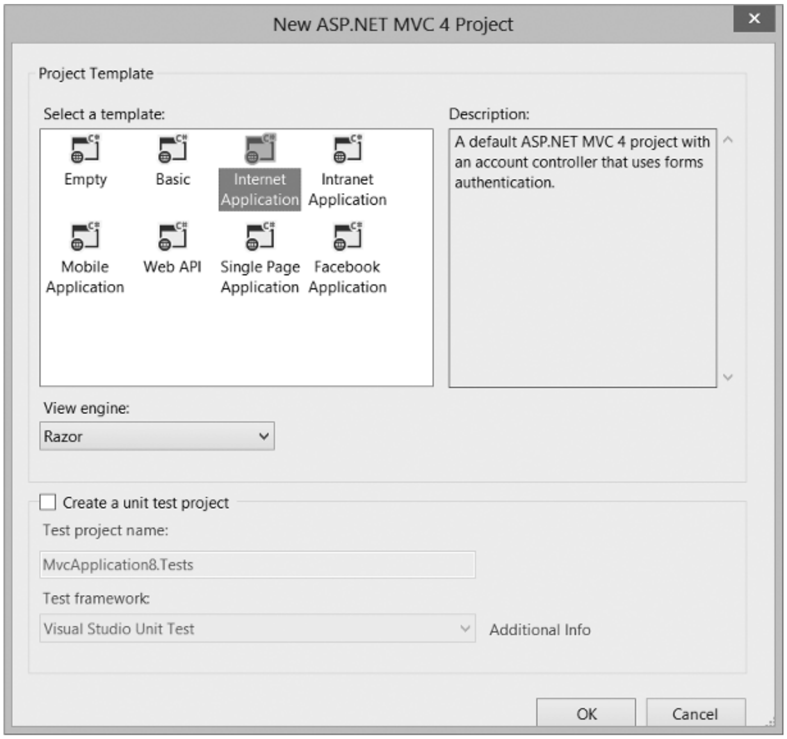
As the web becomes more ubiquitous and web standards continue to evolve, it is crucial that the ASP.NET framework rev at a frequency that matches the advancements in web development.

SIMPLIFYING A COMPLEX ECOSYSTEM

File ➪ New Project is too scary. It forces an artificial choice and makes developers feel that they've proceeded down a fork in the road that may not be reversible. Perhaps, rather than a fork in the road forcing an unnatural choice, we instead consider a unified face to ASP.NET with some choice to select possible subsystems, libraries, or alternative application frameworks.

So you make your choice. How do you now discover and integrate libraries to easily write services in your application? What if you take the safe choice and choose Empty Web Application? Does that make it easier to add services? If you choose ASP.NET Web Application, how do you add DynamicData functionality to this application?

What about social? You can bring in a social support library, but can you easily integrate it into both MVC and Web Forms? From a developer's perspective, this is a fairly complicated story that leaves you hanging, trying to figure out not just the best way to get started, but also how to extend your application. Figure 1-2 shows the choices you have when creating a new project.



Description:
A default ASP.NET MVC 4 project with an account controller that uses forms authentication.

View engine: Razor

Create a unit test project

Test project name: MvcApplication8.Tests

Test framework: Visual Studio Unit Test

Additional Info

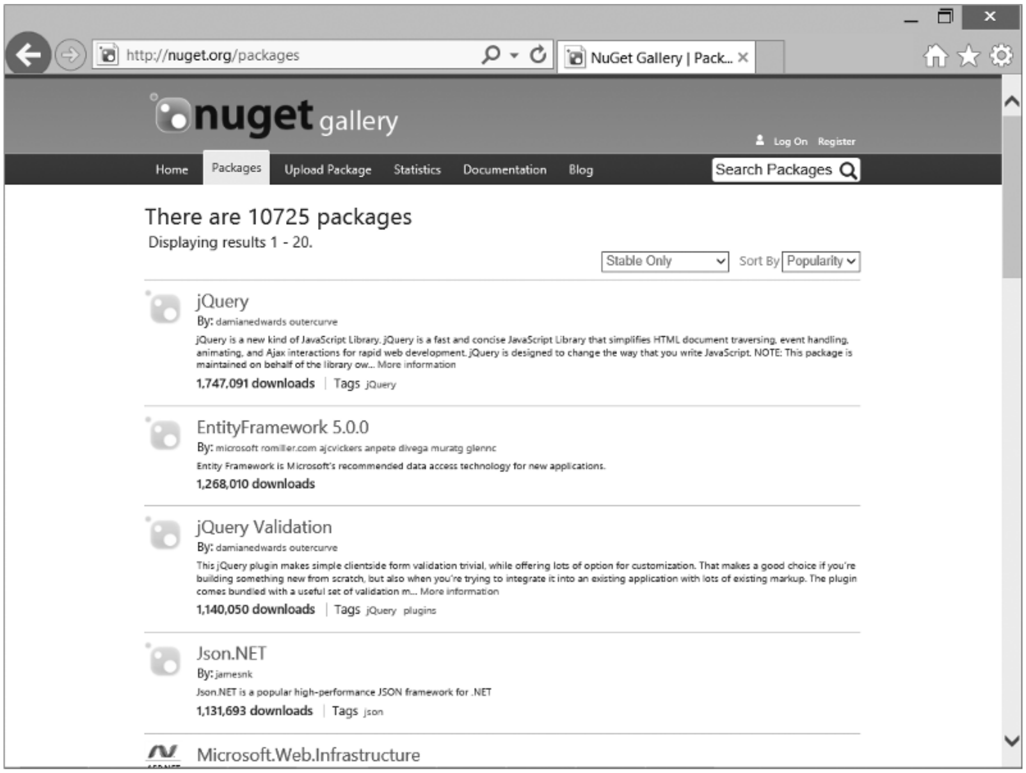
FIGURE 1-2

Web Ecosystem

Web development is an exciting and complicated space to be in. A lot of variables impact the productivity of a web developer. As a web developer, you have to constantly be versed in the latest and greatest in terms of web standards, and once you know what the latest standards are, you then have to worry whether the framework and tools you are using support those standards. Once you know that, how easy is it for you to get the FX, tools, and libraries that you want to use in your application?

A big part of this developer experience is the growing ecosystem. As the web expands and grows, you can find an explosion of libraries that help you develop your applications. These libraries can become popular and widely accepted by the developer community based on the problems they solve. For example, more general libraries such as jQuery are widely popular, and jQuery is the library of choice for many developers who use JavaScript. In addition, lots of specialized libraries are solving unique problems that might not apply to the majority of developers.

We are at a point where lots of such libraries are available, such as Entity Framework, JSON.NET, ELMAH, and so on that are open sourced and can be used in your applications. A big part of your application now comprises a mixture of these open sourced libraries and the ones that come from Microsoft. So the question is that in this growing ecosystem, how easy is it to find such libraries and easily bring them into your applications? Also, once you have these libraries, how do you ensure that they are always updated to the latest versions? Figure 1-3 shows the NuGet gallery with the most popular packages. These packages have been downloaded more than a million times each, and at the time of writing there have been total of 50 million downloads of packages from the NuGet gallery.



There are 10725 packages

Displaying results 1 - 20.

Stable Only   Sort By Popularity

jQuery
By: damianedwards outercurve
jQuery is a new kind of JavaScript Library. jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript. NOTE: This package is maintained on behalf of the library ow... More information
1,747,091 downloads | Tags jQuery

EntityFramework 5.0.0
By: microsoft romiller.com ajcvickers anpete divega muratg glennc
Entity Framework is Microsoft's recommended data access technology for new applications.
1,268,010 downloads

jQuery Validation
By: damianedwards outercurve
This jQuery plugin makes simple clientside form validation trivial, while offering lots of option for customization. That makes a good choice if you're building something new from scratch, but also when you're trying to integrate it into an existing application with lots of existing markup. The plugin comes bundled with a useful set of validation m... More information
1,140,050 downloads | Tags jQuery plugins

Json.NET
By: jamesnk
Json.NET is a popular high-performance JSON framework for .NET
1,131,693 downloads | Tags json

Microsoft.Web.Infrastructure

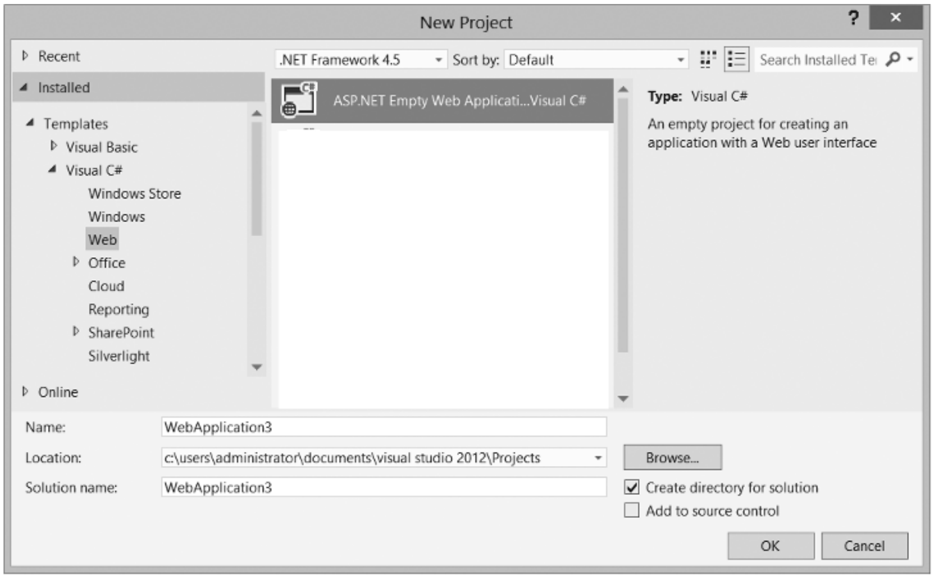
FIGURE 1-3

Web development has definitely evolved over the years, and it is an exciting time to be a web developer. Web developers no longer have to spend time adding basic building blocks to their applications. Features such as logging are so common that developers can reuse a plethora of logging libraries in their applications. How easy is it for you to start with something really small and add these so-called Lego pieces to your application to get the job done more quickly? And how can you now focus on writing code that is central to your applications and bring in these different building blocks as and when required?

The next few sections look at how ASP.NET framework and Visual Studio tooling can help you solve these problems and improve your productivity.

Getting Started Is Easy

In Visual Studio 2012 when you choose File ➪ New Project, you get lots of choices on what type of project you want to create. You can choose from creating a Single Page Application (SPA) or have an application that uses Social Login, or have an application that uses Windows Authentication. This becomes a problem when you want to create an application where you want to mix these scenarios. There is no easy way to get started built into Visual Studio. The templates represent a static list of options, but things would be simpler with a single One ASP.NET template, as shown in Figure 1-4.



.NET Framework 4.5 Sort by: Default

ASP.NET Empty Web Applicati... Visual C#

Type: Visual C#

An empty project for creating an application with a Web user interface

Name: WebApplication3

Location: c:\users\administrator\documents\visual studio 2012\Projects

Solution name: WebApplication3

☑ Create directory for solution

☐ Add to source control

FIGURE 1-4

NOTE *This is just a conceptual idea at the time of writing this book.*

Once you create an ASP.NET project, Visual Studio can guide you through a series of steps that help you get started with an SPA framework of your choice, and then enable you to choose how you want users to log in to your application.

The next section looks at how you can get more Lego pieces, or update the ones you have, once you start with such an application.

Integrating the Ecosystem Is Easy

In the previous section you looked at how easy it would be to create an SPA application with all the required libraries, such as Knockout, jQuery, ASP.NET Web API, and so on. In Visual Studio 2012, all the libraries are installed as NuGet packages. You can see what libraries are installed in your project by right-clicking the project and clicking Manage NuGet Packages. Figure 1-5 shows this option.

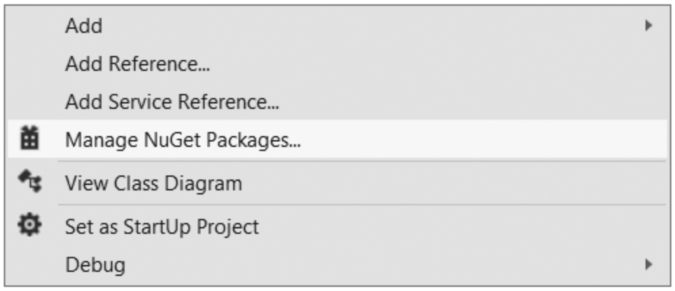
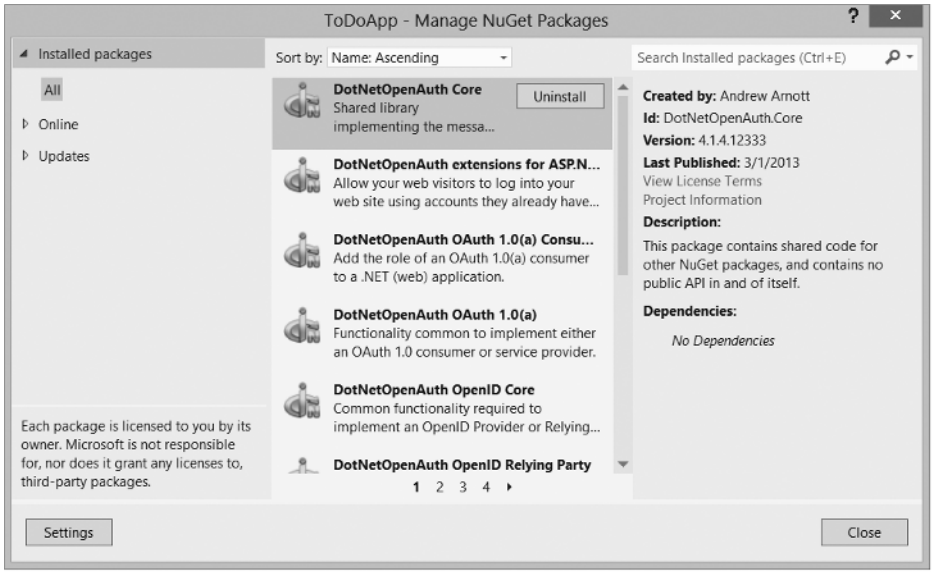


FIGURE 1-5

This launches the Manage NuGet Packages window, where you can choose the Installed packages tab and see all the packages that are installed in your project. Figure 1-6 shows all the packages installed in a project.



DotNetOpenAuth Core
Shared library implementing the messa...

DotNetOpenAuth extensions for ASP.N...
Allow your web visitors to log into your web site using accounts they already have...

DotNetOpenAuth OAuth 1.0(a) Consu...
Add the role of an OAuth 1.0(a) consumer to a .NET (web) application.

DotNetOpenAuth OAuth 1.0(a)
Functionality common to implement either an OAuth 1.0 consumer or service provider.

DotNetOpenAuth OpenID Core
Common functionality required to implement an OpenID Provider or Relying...

DotNetOpenAuth OpenID Relying Party

Created by: Andrew Arnott
Id: DotNetOpenAuth.Core
Version: 4.1.4.12333
Last Published: 3/1/2013
View License Terms
Project Information
Description:
This package contains shared code for other NuGet packages, and contains no public API in and of itself.
Dependencies:
No Dependencies

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

FIGURE 1-6

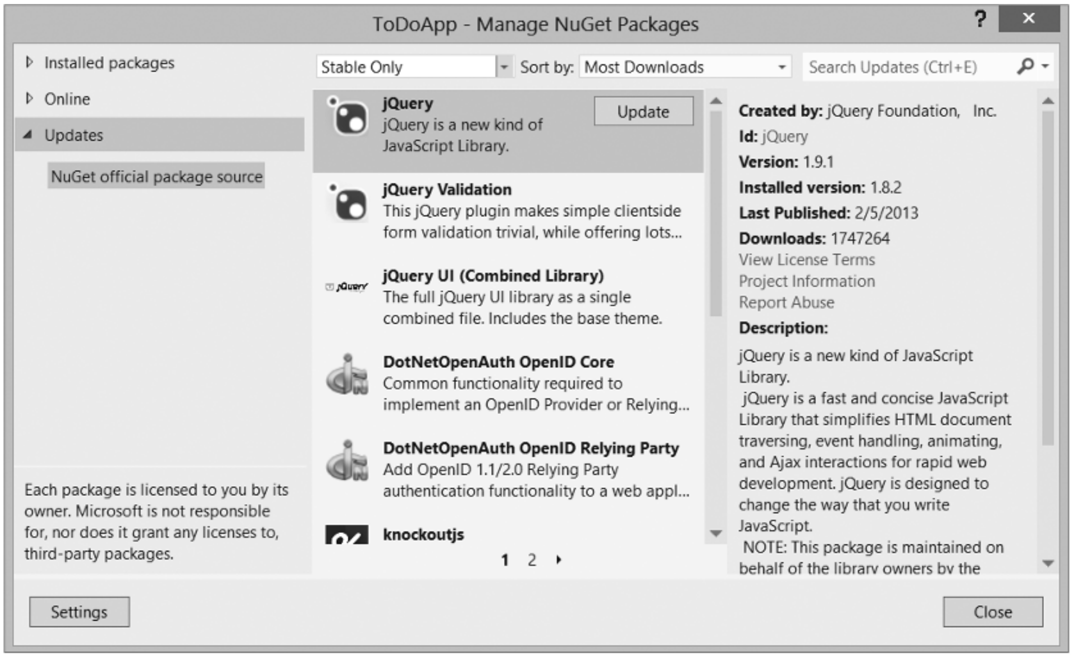
This window shows you the list of packages that were installed when you created the application. This application has packages that are a mix of Microsoft owned and open sourced libraries as well. This section discusses how NuGet helps you to integrate the ecosystem easily into your project.

NuGet is a Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects. NuGet works on both Visual Studio 2010 and Visual Studio 2012, and you can download it from the Visual Studio Extension Gallery. If you develop a library that you want to share with other developers, you can create a NuGet package and upload the package to the NuGet gallery. If you want to use a library or tool that someone else has developed, you can download the package from the gallery and install it in your Visual Studio project. Simply put, NuGet is changing the .NET developer ecosystem by making it easy to redistribute and install packages into a project.

Imagine that you had to install ELMAH into your application. Without NuGet, you would have to download an installer and then manually add references to the ELMAH library from the install location. Beyond this, you would have to read the documentation on the ELMAH website to make all the necessary changes to the `web.config` to configure ELMAH. If a new version of ELMAH came out during the course of your development, you would have no way of knowing that the version you have in your project is outdated and that you need to download a new version.

NuGet makes this entire process of finding, installing, configuring, and updating libraries very easy. It is a one-stop solution that can solve all of these problems.

Figure 1-6 showed the packages that were installed in an SPA application. Imagine that you were working on this project for a while and wanted to check for updates to any of the libraries that you were using. To do this, you would simply launch the Manage NuGet Packages window and click the Updates tab. Figure 1-7 shows the updates available for the packages.



jQuery
jQuery is a new kind of JavaScript Library.

jQuery Validation
This jQuery plugin makes simple clientside form validation trivial, while offering lots...

jQuery UI (Combined Library)
The full jQuery UI library as a single combined file. Includes the base theme.

DotNetOpenAuth OpenID Core
Common functionality required to implement an OpenID Provider or Relying...

DotNetOpenAuth OpenID Relying Party
Add OpenID 1.1/2.0 Relying Party authentication functionality to a web appl...

knockoutjs

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Created by: jQuery Foundation, Inc.
Id: jQuery
Version: 1.9.1
Installed version: 1.8.2
Last Published: 2/5/2013
Downloads: 1747264
View License Terms
Project Information
Report Abuse
Description:
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.
NOTE: This package is maintained on behalf of the library owners by the

FIGURE 1-7

You click Update on the package that you want to update, and NuGet downloads a new version of that package and updates the references as well. You can easily update any of the packages quickly.

NOTE *When you create any new project in Visual Studio 2012, the project already has NuGet packages installed so you can get these benefits easily.*

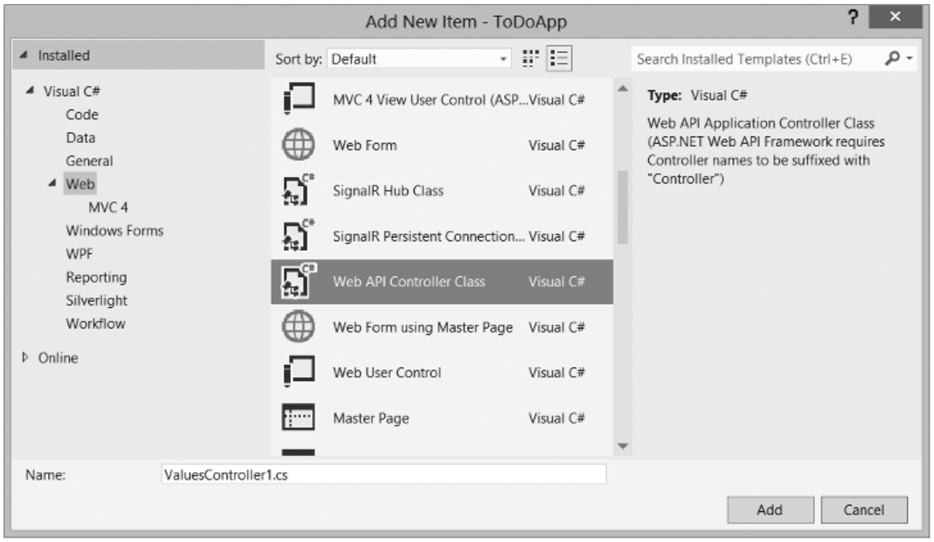
Real-World Example

This section shows how you can apply One ASP.NET in a practical real-world application. The following approach by no means represents any best practice, but it is meant to demonstrate how One ASP.NET and the web ecosystem can come together to help you build an ASP.NET application. Assume that you wanted to create a simple single page to-do list web application with the following requirements:

- A user should be able to log in using Facebook, Twitter, or register as an account on the website.
- A user should be able to create to dos and mark items when done.
- An administrator should be able to manage user accounts via an administrative section.

You can get started by choosing the ASP.NET SPA template and configuring the template to use Facebook, Twitter, and Local login. Once you have this template ready, you can start building your application.

First, implement an ASP.NET Web API so you can have a REST-based service to manage to-do items. You can add a Web API by right-clicking your project and adding a new item template—ASP.NET Web API Controller Class (see Figure 1-8).


Add New Item - ToDoApp
Installed
Visual C#
Code
Data
General
Web
MVC 4
Windows Forms
WPF
Reporting
Silverlight
Workflow
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
MVC 4 View User Control (ASP... Visual C#
Web Form Visual C#
SignalR Hub Class Visual C#
SignalR Persistent Connection... Visual C#
Web API Controller Class Visual C#
Web Form using Master Page Visual C#
Web User Control Visual C#
Master Page Visual C#
Add
Cancel

Type: Visual C#

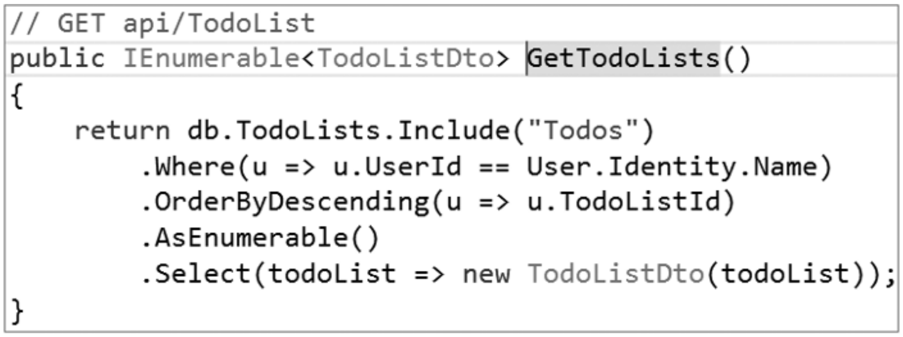
Web API Application Controller Class (ASP.NET Web API Framework requires Controller names to be suffixed with "Controller")

Name: ValuesController1.cs

FIGURE 1-8

When you install this item template, it installs all the NuGet packages needed for running a Web API in ASP.NET.

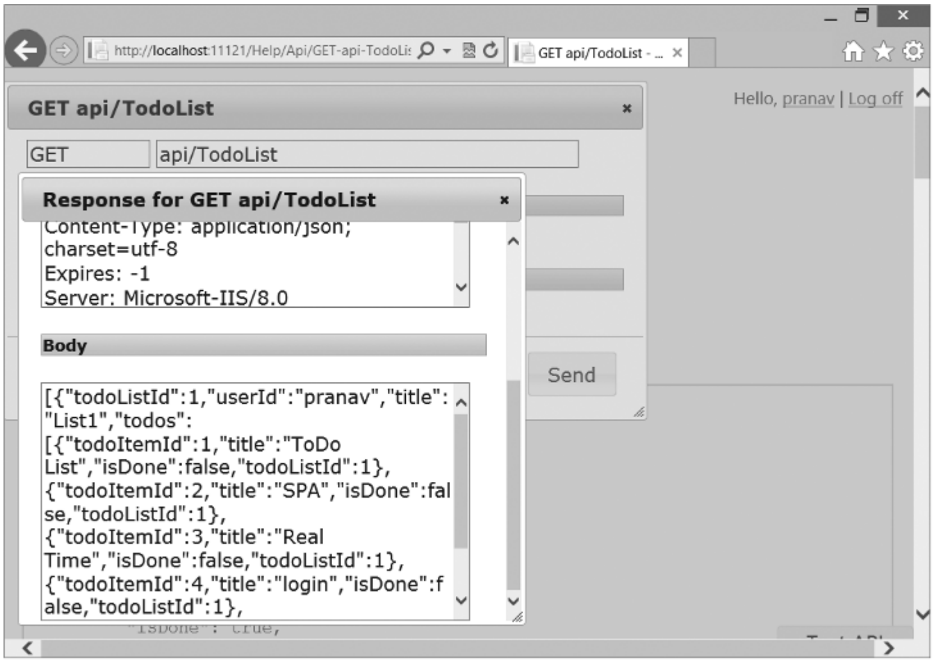
Write some code to implement to-do functionality of your application that includes features such as adding, deleting, modifying, and getting all to dos. Figure 1-9 shows a snippet of code for retrieving a list of to-do items.



```
// GET api/TodoList
public IEnumerable<TodoListDto> GetTodoLists()
{
    return db.TodoLists.Include("Todos")
        .Where(u => u.UserId == User.Identity.Name)
        .OrderByDescending(u => u.TodoListId)
        .AsEnumerable()
        .Select(todoList => new TodoListDto(todoList));
}
```

FIGURE 1-9

While writing this To do Web API, you will want to test out the functionality without running your entire application, because it can become quite cumbersome. You can use NuGet to search for some test clients for testing your To do Web API. A quick search might return a list of packages from which you can choose one based on its rating. For example, you could download the WebApiTestClient NuGet package, which was popular at the time of writing this chapter. Figure 1-10 shows the result when you use this test client to test your Web API.



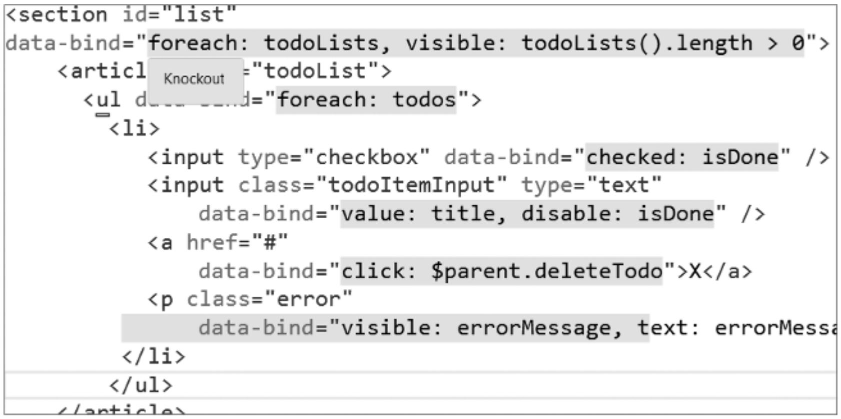
```
Content-Type: application/json;
charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
```

```
[{"todoListId":1,"userId":"pranav","title":
"List1","todos":
[{"todoItemId":1,"title":"ToDo
List","isDone":false,"todoListId":1},
{"todoItemId":2,"title":"SPA","isDone":fal
se,"todoListId":1},
{"todoItemId":3,"title":"Real
Time","isDone":false,"todoListId":1},
{"todoItemId":4,"title":"login","isDone":f
alse,"todoListId":1},
```

FIGURE 1-10

Now that you have the Web API working, you can start adding SPA libraries so that you can write the front end of the application. Again, you can search on NuGet for an SPA library of your choice. For this example, choose Knockout. Once you install the NuGet package, it brings in all the required libraries for using Knockout.

After you install Knockout, you can create web pages where you can call your To do Web API and bind the results in Knockout views. The Visual Studio editor provides rich support for Knockout syntax highlighting and IntelliSense, which makes it easier to use Knockout. Figure 1-11 shows the syntax highlighting support for Knockout in Visual Studio.



```
<section id="list"
data-bind="foreach: todoLists, visible: todoLists().length > 0">
    <articl       ="todoList">
      <ul d     d="foreach: todos">
        <li>
           <input type="checkbox" data-bind="checked: isDone" />
           <input class="todoItemInput" type="text"
               data-bind="value: title, disable: isDone" />
           <a href="#"
               data-bind="click: $parent.deleteTodo">X</a>
           <p class="error"
               data-bind="visible: errorMessage, text: errorMessa
         </li>
        </ul>
```

FIGURE 1-11

Once you have the client-side data binding working with Knockout, you will have a fully functional SPA application. If you want to receive updates about your to-do items, you can get real-time update functionality by installing ASP.NET SignalR. You can do this by downloading the Microsoft.AspNet.SignalR NuGet package. Once you install this package, you can modify your application code to add real-time functionality. Figure 1-12 shows the to-do application that you just developed.

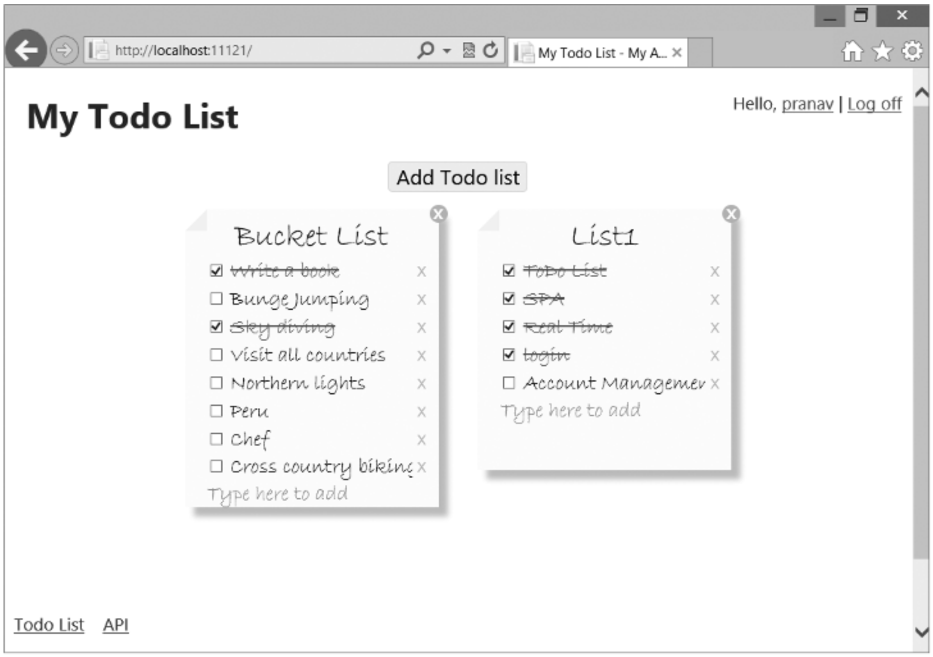


FIGURE 1-12

The final part in creating your application is to build some user account management for administrators. You can quickly build this part using ASP.NET Web Forms and Dynamic Data, which is a good framework for rapidly building data-driven applications. You can also secure this section with ASP.NET authorization using roles so that only administrators can access this section of the website.

Finally, if you look at this application, it is a mixture of ASP.NET frameworks and technologies such as ASP.NET Web Forms, MVC, Web API, and SignalR working in cohesion with open source libraries such as Knockout, jQuery, JSON.NET, and many more. With all these different sets of frameworks and libraries, you are still developing on a trusted underlying framework—ASP.NET.

HOW DO YOU BENEFIT?

Now that you have read through all of this, you must be thinking that this is all great, but what is in it for you as a developer? How do you benefit from One ASP.NET? This section looks at some of the scenarios where One ASP.NET will help increase your developer productivity. It also looks at the common problems faced by developers today, and how ASP.NET and Visual Studio can help solve these problems.

ASP.NET Makes Getting Started Easy

With ASP.NET and Visual Studio, you should be able to:

- ➤ Start with nothing (an empty project) and add the required pieces of the framework and libraries needed to build your application.
- ➤ Choose the set of frameworks and libraries you need to build your application.

One ASP.NET will make the ASP.NET framework more modular so you can get started with an application and be able to easily add the required set of pieces needed to build your application.

ASP.NET Supports the Web Ecosystem

We all know by now that future web applications will be much richer in terms of the user experience that they will offer. Also, with the growth of the .NET developer community, there has been an explosion of libraries being created and distributed amongst developers. In the .NET ecosystem, developers are now relying more and more on open sourced libraries and are contributing to these libraries to make them even better.

The majority of the Microsoft Web Stack is now open sourced. Products such as ASP.NET MVC, ASP.NET Web Pages, ASP.NET Web API, ASP.NET SignalR, and Entity Framework are open sourced. This means that developers who do not work for Microsoft can make contributions to these projects.

NOTE *Even though these products are open sourced, they continue to be fully supported and staffed by Microsoft.*

Apart from making contributions, you can browse the source code, subscribe to checkin notifications, and browse through the active list of issues that developers on these products are working on. By doing all of this, you, as a developer, have more direct impact to the features being designed and developed in these products, which eventually results in a better product being developed.

ASP.NET Makes Finding, Adding, and Updating Lego Blocks to Your Site Easy

Although the growing ecosystem is a great sign for the web developer community, today there is a problem of discovering and installing all these great libraries in this ecosystem. The installation experience for such libraries has centered around downloading an installer and running through the installation steps. This makes the process of adding, configuring, and updating libraries in the project very hard. In contrast, NuGet is a giant step forward in this space. Libraries/frameworks can be redistributed as NuGet packages, which can be easily installed into your project through Visual Studio.

ASP.NET Helps You Apply Concepts from One Framework to Another

The ASP.NET umbrella has lots of different features that were developed in one framework, such as ASP.NET MVC, and have now made their way into other frameworks, such as ASP.NET Web Forms, and vice versa. For example:

- There was a feature called Data Annotation Attributes that was introduced in ASP.NET Dynamic Data and later introduced in ASP.NET MVC.
- Routing and model binding were some of the highlights of ASP.NET MVC that eventually made their way into ASP.NET Web Forms.
- ASP.NET Web Pages had a simple routing model called Smarty Routes that let you generate cleaner looking URLs without any configuration. This feature made its way into ASP.NET Web Forms and is now called ASP.NET FriendlyURLs.

This means that if you are an ASP.NET developer, you can use these well-known concepts or features and apply them to any framework of your choice. If you like model binding and are a Web Forms developer, now you can continue building Web Forms applications while using new and well-known features such as model binding. You do not have to switch to another framework such as ASP.NET MVC, just for the simple reason that ASP.NET MVC supports model binding.

ASP.NET Moves as Fast as the Web

The web is evolving at a crazy pace. This is caused by many factors. The proliferation of devices and services has caused new kinds of applications to emerge that need to be more closely interconnected. The web is going more social, where users want to collaborate with each other in real time. To support all these changes, the Web Standards Consortium is pushed to make sure the specifications of HTML, JavaScript, and CSS are updated quickly to support these demands. Given all these changes, developers are pushed as well to stay abreast of all of them and demand that the frameworks and tools they use are updated to these latest standards as well.

ASP.NET and Visual Studio have been responding to these changes as well. When ASP.NET 2.0 was released, it was released with VS2005, which was released 3 years after ASP.NET 1.0. This meant that developers had to wait for 3 years to get the latest framework that supported the latest web standards. We are moving to a new world now where parts of ASP.NET can be released more frequently and updated easily without waiting for the next release of Visual Studio.

ASP.NET MVC, Web Pages, Web API, and SignalR are released as NuGet packages on the NuGet gallery so you can update these frameworks by updating the NuGet packages installed in your project. Microsoft ASP.NET and Web Frameworks 2012.2 and Microsoft Web Developer Tools 2012.2 add support for a number of improvements to make it easy to write web applications in VS and publish them to Azure. Visual Studio 2012 shipped with 1.0 version of Microsoft Web Developer Tools, but by the time this book is published this would be 1.2. This means in a matter of a few months as a developer you were able to get the latest tools to increase your productivity while building web applications. The ASP.NET team is moving toward a more frequent release schedule to be able to deliver updated tools and libraries to make web development more relevant for the emerging standards and libraries.

SUMMARY

ASP.NET should be a reliable, fun, modular, powerful, and scalable platform for building web applications. Whether you're using a reporting control in Web Forms from a third-party control builder, or writing your own custom HTML Helper, ASP.NET can be a platform you should feel confident to have underlying your work.

Perhaps your solution is developed using TDD and Agile with SpecFlow and targeting ASP.NET MVC, or perhaps you prefer Ember.js and ASP.NET Web API. Likely you'll mix and match, and as well you should. ASP.NET is certainly becoming a web framework with choices.

Don't be afraid to try new things, swap out components, or explore new open source libraries. ASP.NET and the team will try to solve the big, hard problems of scale, security, and performance. How you architect your application is a personal choice, and you should have no concerns about ASP.NET's ability to rise to the occasion and support your architecture.

With the release of ASP.NET and Web Tools 2012.2, ASP.NET is about halfway to the dream of One ASP.NET. Someday soon perhaps Microsoft project templates will live side by side with community templates, and project templates will be easily shared and distributed as NuGet packages. The most recent version of ASP.NET doesn't make it too hard to imagine that future.