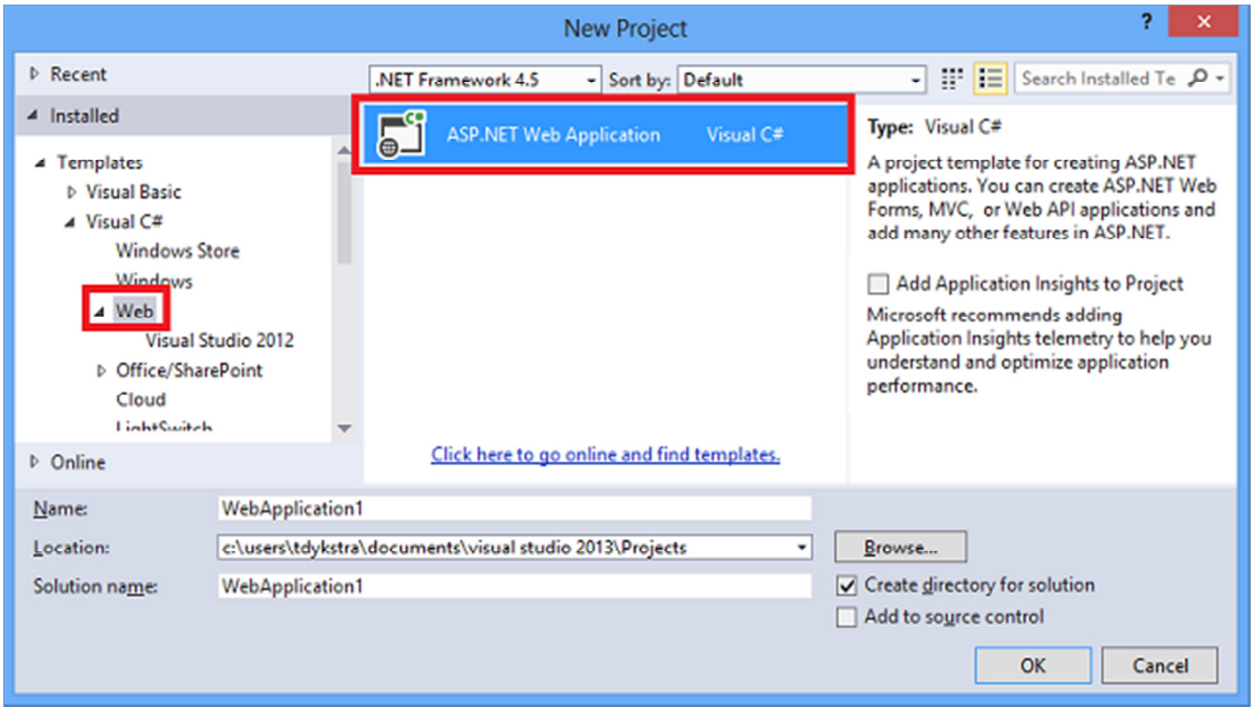
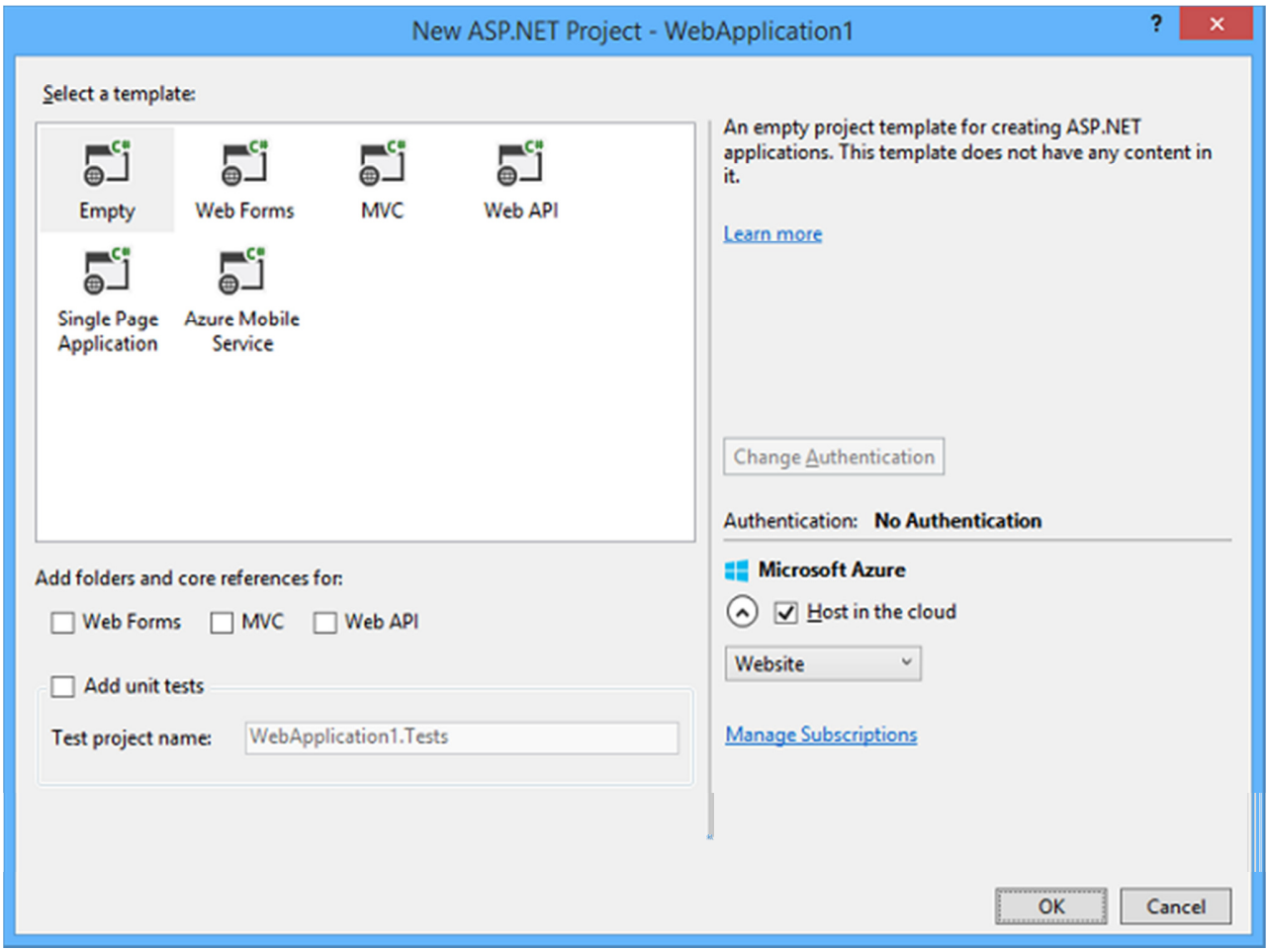
Overview of Web Application Project Creation

The following steps show how to create a web project:

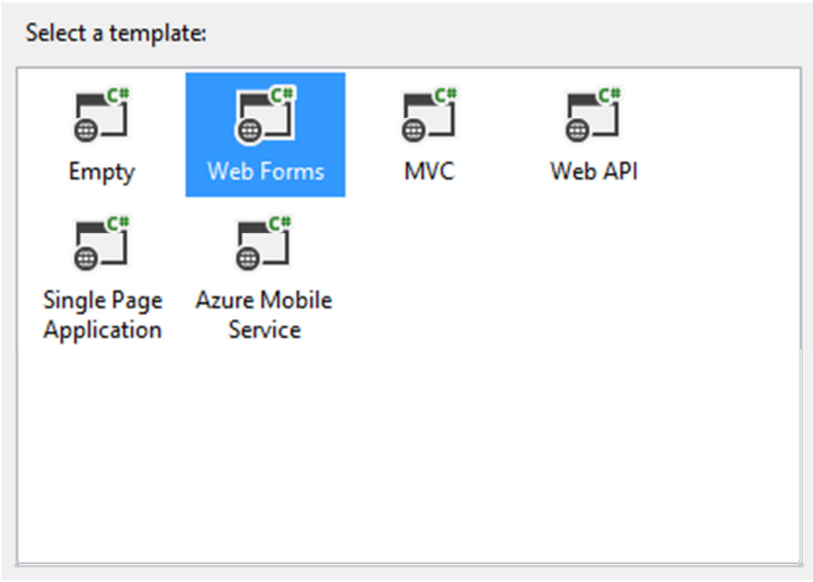
1. Click **New Project** in the **Start** page or in the **File** menu.
2. In the **New Project** dialog, click **Web** in the left pane and **ASP.NET Web Application** in the middle pane.



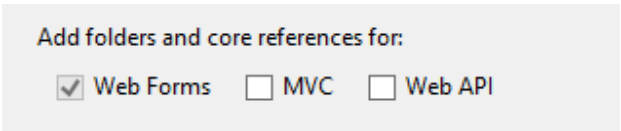
   You can choose **Cloud** in the left pane to create an Azure Cloud Service, Azure Mobile Service, or Azure WebJob. This topic doesn't cover those templates.
3. In the right pane, click the **Add Application Insights to Project** check box if you want health and usage monitoring for your application. For more information, see Monitor performance in web applications.
4. Specify project **Name**, **Location**, and other options, and then click **OK**.
   The **New ASP.NET Project** dialog appears.



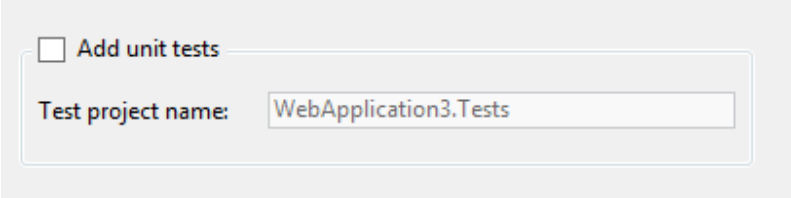
5. Click a template.



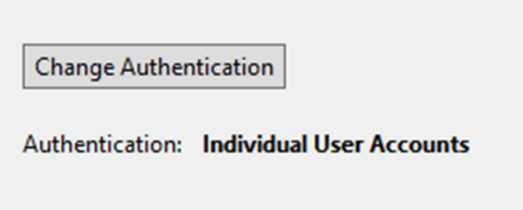
6. If you want to add support for additional frameworks not included in the template, click the appropriate check box. (In the example shown, you could add MVC and/or Web API to a Web Forms project.)

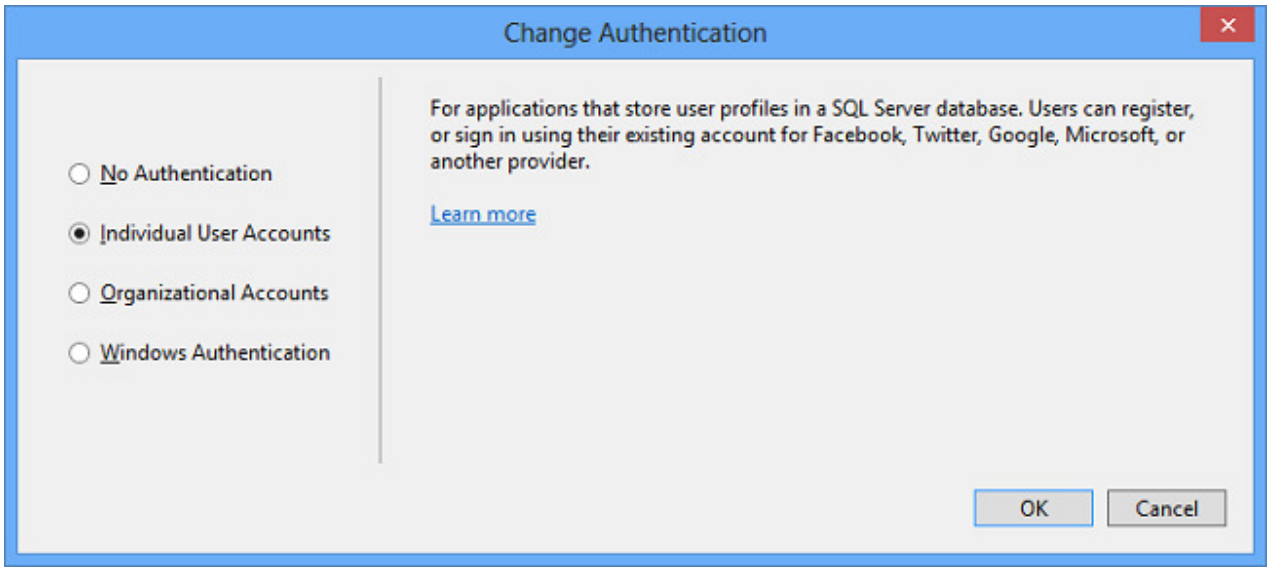


7. If you want to add a unit test project, click **Add unit tests**.



8. If you want a different authentication method than what the template provides by default, click **Change Authentication**.



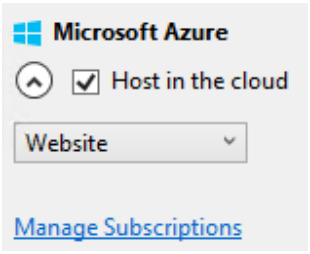


Create a web app or virtual machine in Azure

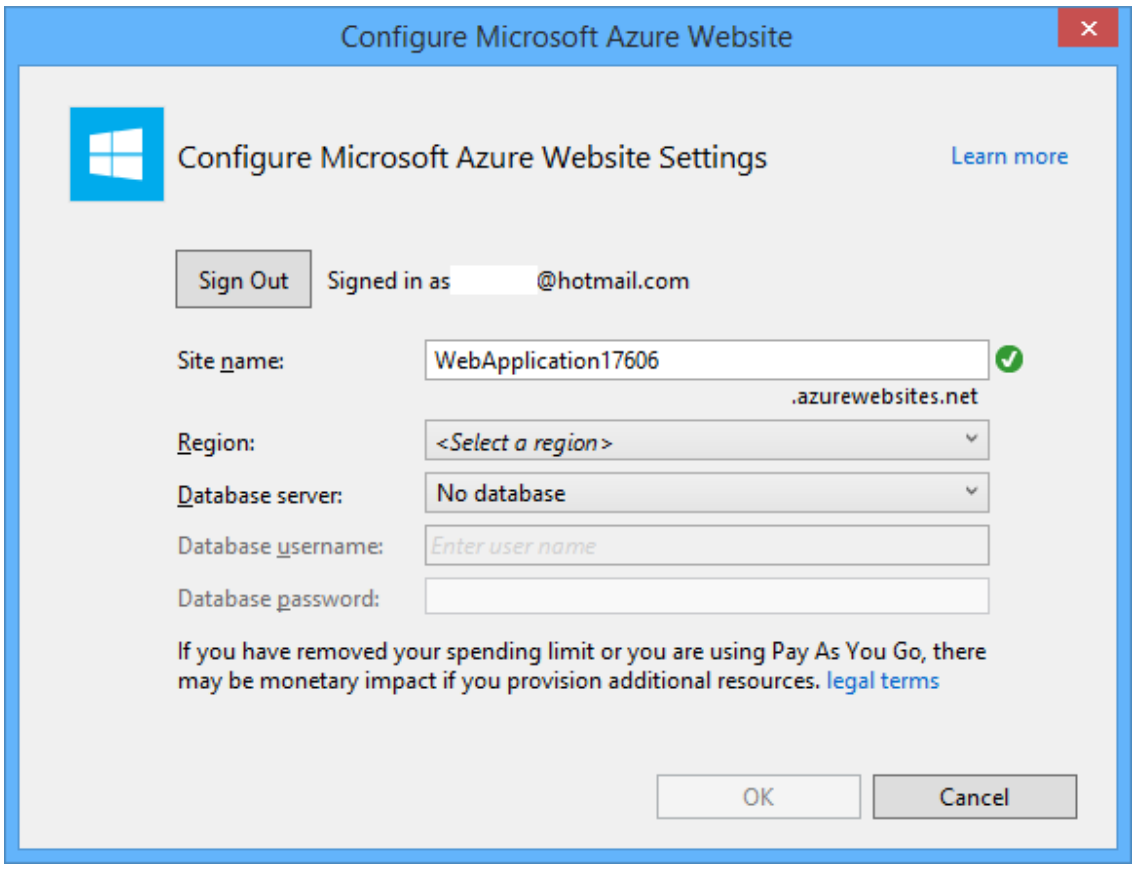
Visual Studio includes features that make it easy to work with Azure services for hosting web applications. For example, you can do all of the following right from the Visual Studio IDE:

- Create and manage web apps or virtual machines that make your application available over the Internet.
- View logs created by the application as it runs in the cloud.
- Run in debug mode remotely while the application runs in the cloud.
- View and manage other Azure services such as SQL databases.

You can create an Azure account that includes basic services such as web apps for free, and if you are an MSDN subscriber you can activate benefits that give you monthly credits toward additional Azure services.
By default the **New ASP.NET Project** dialog box enables you to create a web app or virtual machine for a new web project. If you don't want to create a new web app or virtual machine, clear the **Host in the cloud** check box.



The check box caption might be **Host in the cloud** or **Create remote resources**, and in either case the effect is the same. If you leave the check box selected, Visual Studio creates a web app in Azure App Service by default. You can use the drop-down box to change that to a **Virtual Machine** if you prefer. If you're not already signed in to Azure, you're prompted for Azure credentials. After you sign in, a dialog box enables you to configure the resources that Visual Studio will create for your project. The following illustration shows the dialog for a web app; different options appear if you choose to create a virtual machine.



For more information about how to use this process for creating Azure resources, see Get Started with Azure and ASP.NET and Creating a virtual machine for a web site with Visual Studio.
The remainder of this article provides more information about the available templates and their options. The article also introduces Bootstrap, the layout and theming framework used in the templates.

Visual Studio 2013 Web Project Templates

Visual Studio uses templates to create web projects. A project template can create files and folders in the new project, install NuGet packages, and provide sample code for a rudimentary working application. Templates implement the latest web standards and are intended to demonstrate best practices for how to use ASP.NET technologies as well as give you a jump start on creating your own application.

Visual Studio 2013 provides the following choices for web project templates for projects that target .NET 4.5 or later versions of the .NET framework:

- Empty template
- Web Forms template
- MVC template
- Web API template
- Single Page Application template
- Azure Mobile Service template
- Visual Studio 2012 Templates

You can also install a Visual Studio extension that provides a Facebook template.
For information about how to create projects that target .NET 4, see Visual Studio 2012 Templates later in this topic.
For information about how to create ASP.NET applications for mobile clients, see Mobile Support in ASP.NET.

Empty Template

The Empty template provides the bare minimum folders and files for an ASP.NET web app, such as a project file (*.csproj* or *.vbproj*) and a*Web.config* file. You can add support for Web Forms, MVC, and/or Web API by using the check boxes under the **Add folders and core references for:** label.
For the Empty template no authentication options are available. Authentication functionality is implemented in sample applications, and the Empty template does not create a sample application.

Web Forms Template

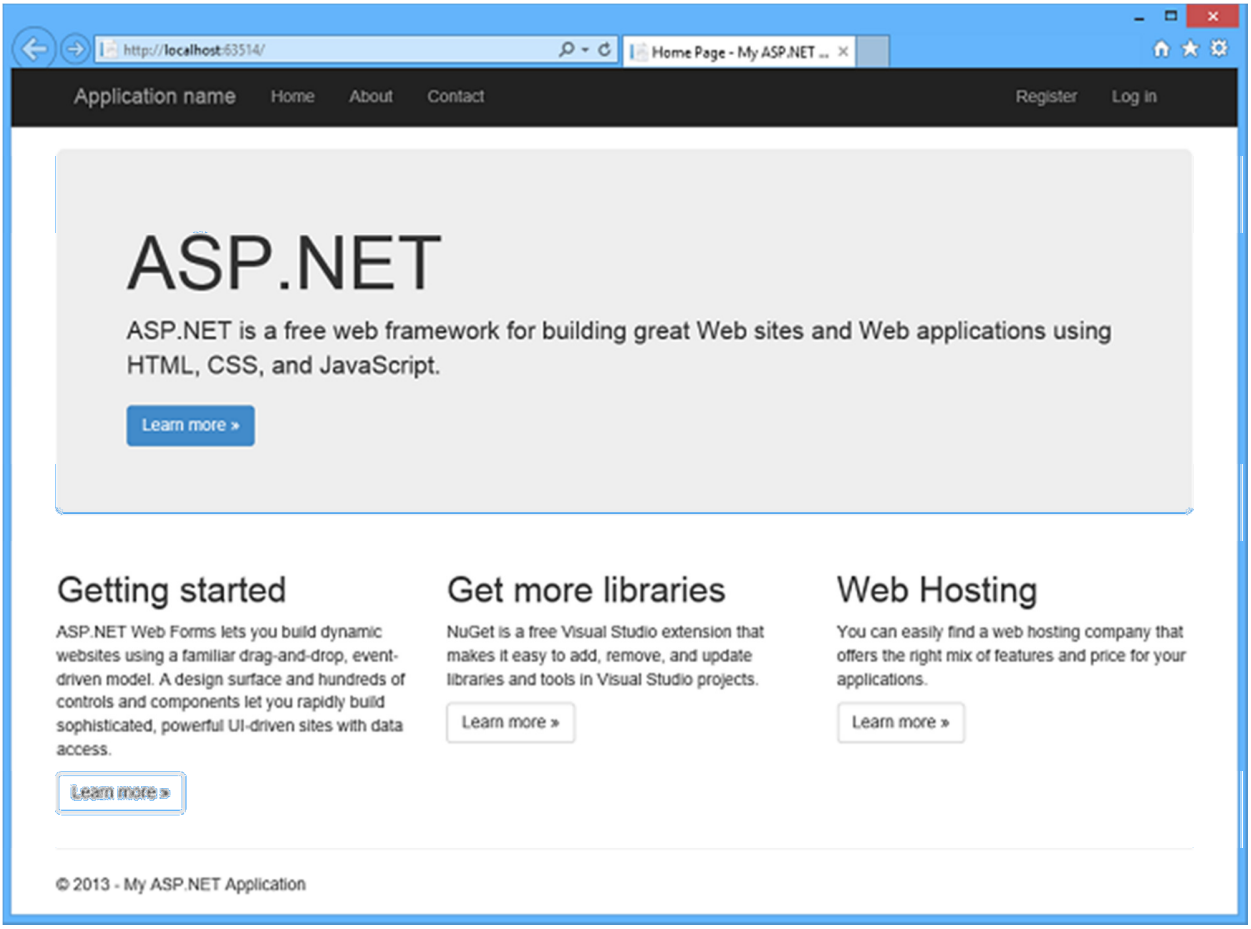
The Web Forms framework provides the following features that let you rapidly build web sites that are rich in UI and data access features:

- A WYSIWYG designer in Visual Studio.
- Server controls that render HTML and that you can customize by setting properties and styles.
- A rich assortment of controls for data access and data display.
- An event model that exposes events which you can program like you would program a client application such as WPF.
- Automatic preservation of state (data) between HTTP requests.

In general, creating a Web Forms application requires less programming effort than creating the same application by using the ASP.NET MVC framework. However, Web Forms is not just for rapid application development. There are many complex commercial applications and frameworks built on top of Web Forms.

Because a Web Forms page and the controls on the page automatically generate much of the markup that's sent to the browser, you don't have the kind of fine-grained control over the HTML that ASP.NET MVC offers. The declarative model for configuring pages and controls minimizes the amount of code you have to write but hides some of the behavior of HTML and HTTP. For example, it's not always possible to specify exactly what markup might be generated by a control.

The Web Forms framework doesn't lend itself as readily as ASP.NET MVC to patterns-based development practices such as test-driven development, separation of concerns, inversion of control, and dependency injection. If you want to write code factored that way, you can; it's just not as automatic as it is in the ASP.NET MVC framework. The ASP.NET Web Forms MVP project shows an approach that facilitates separation of concerns and testability while maintaining the rapid development that Web Forms was built to deliver. Microsoft SharePoint is built on Web Forms MVP.
The Web Forms template creates a sample Web Forms application that uses Bootstrap to provide responsive design and theming features. The following illustration shows the home page.



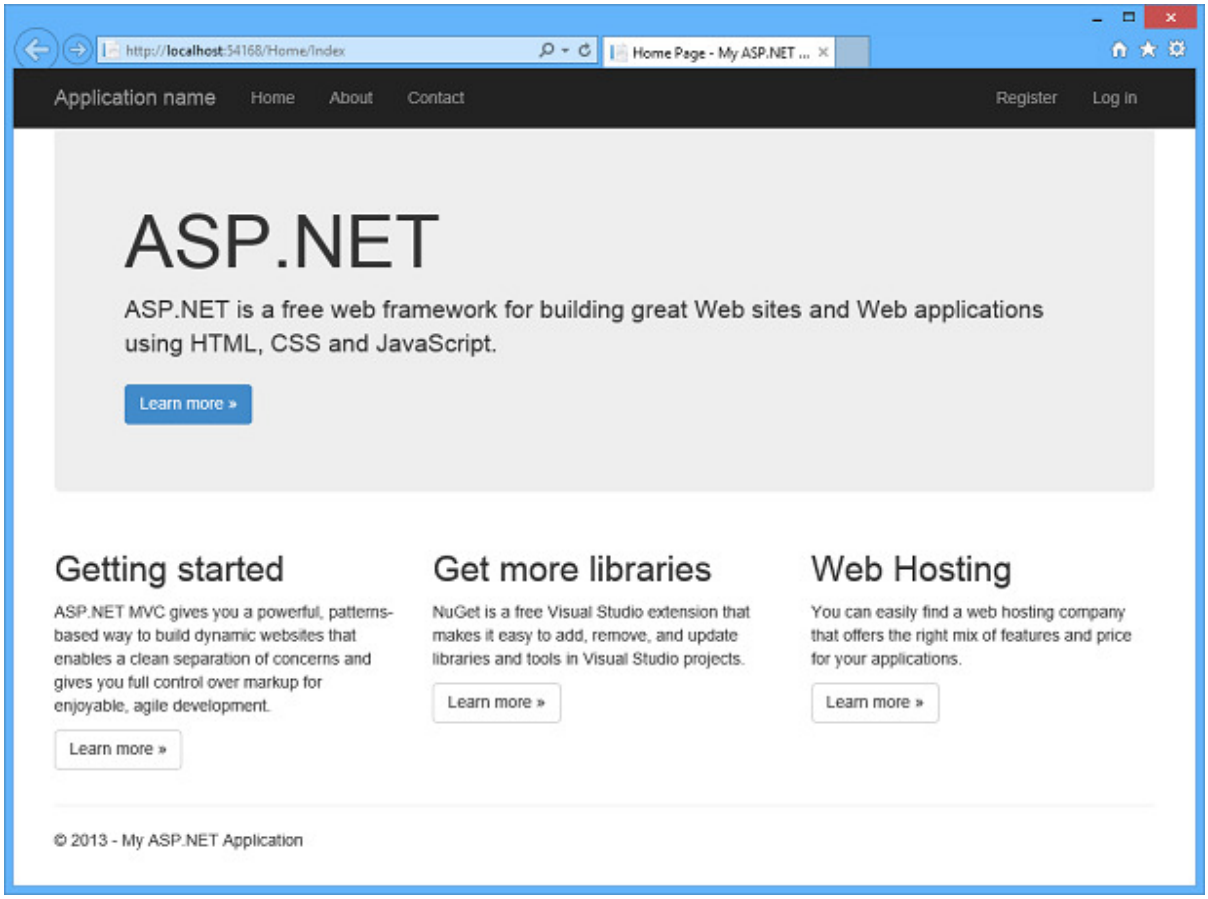
For more information about Web Forms, see ASP.NET Web Forms. For information about what the Web Forms template does for you, see Building a basic Web Forms application using Visual Studio 2013.

MVC Template

ASP.NET MVC was designed to facilitate patterns-based development practices such as test-driven development, separation of concerns,inversion of control, and dependency injection. The framework encourages separating the business logic layer of a web application from its presentation layer. By dividing the application into models (M), views (V), and controllers (C), ASP.NET MVC can make it easier to manage complexity in larger applications.
With ASP.NET MVC, you work more directly with HTML and HTTP than in Web Forms. For example, Web Forms can automatically preserve state between HTTP requests, but you have to code that explicitly in MVC. The advantage of the MVC model is that it enables you to take complete control over exactly what your application is doing and how it behaves in the web environment. The disadvantage is that you have to write more code.

MVC was designed to be extensible, providing power developers the ability to customize the framework for their application needs. In addition, the ASP.NET MVC source code is available under an OSI license.

The MVC template creates a sample MVC 5 application that uses Bootstrap to provide responsive design and theming features. The following illustration shows the home page.



For more information about MVC, see ASP.NET MVC. For information about how to select the MVC 4 template, see Visual Studio 2012 templates later in this article.

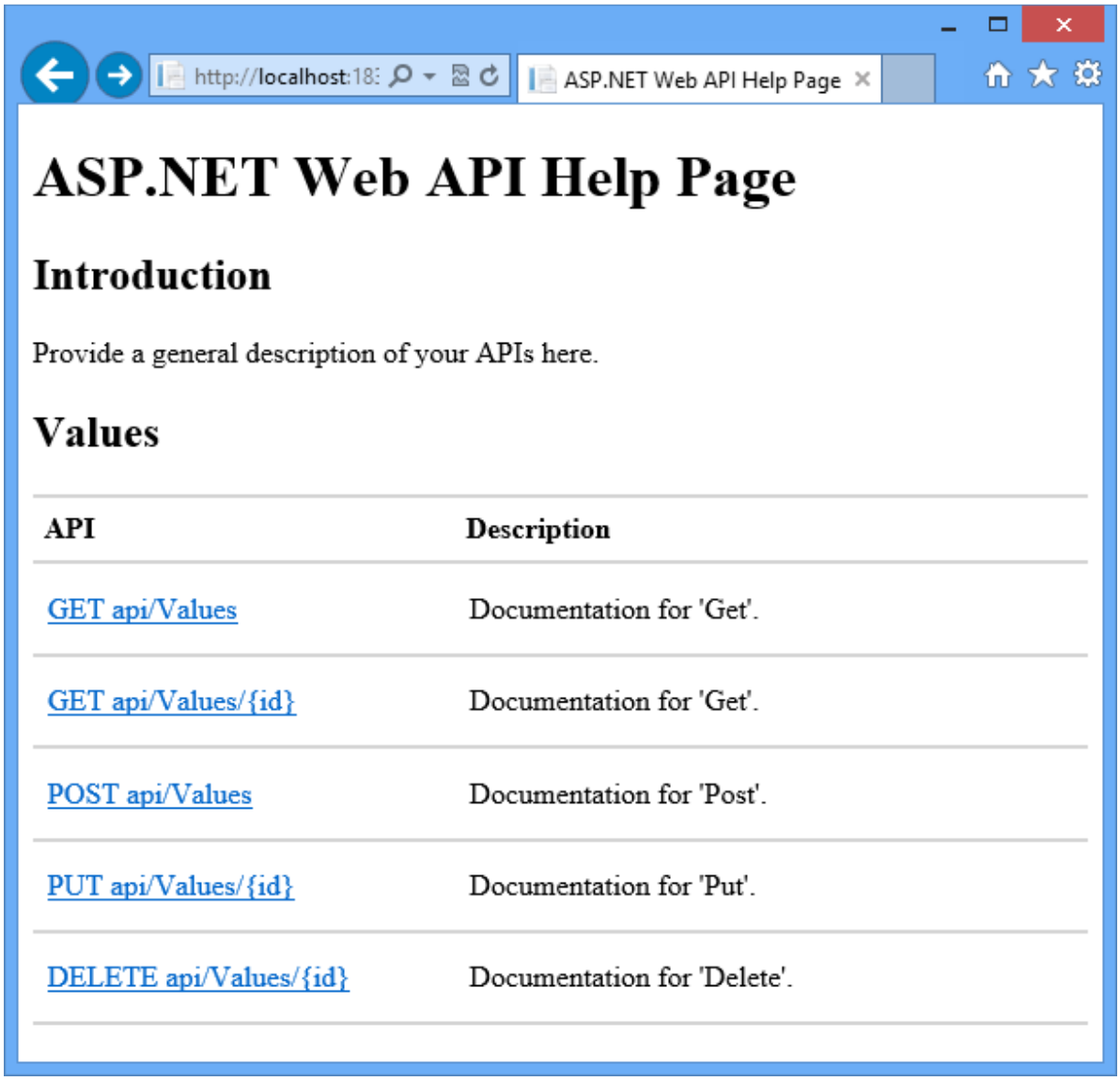
Web API Template

The Web API template creates a sample web service based on Web API, including API help pages based on MVC.

ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building RESTful services on the .NET Framework.

The Web API template creates a sample web service.

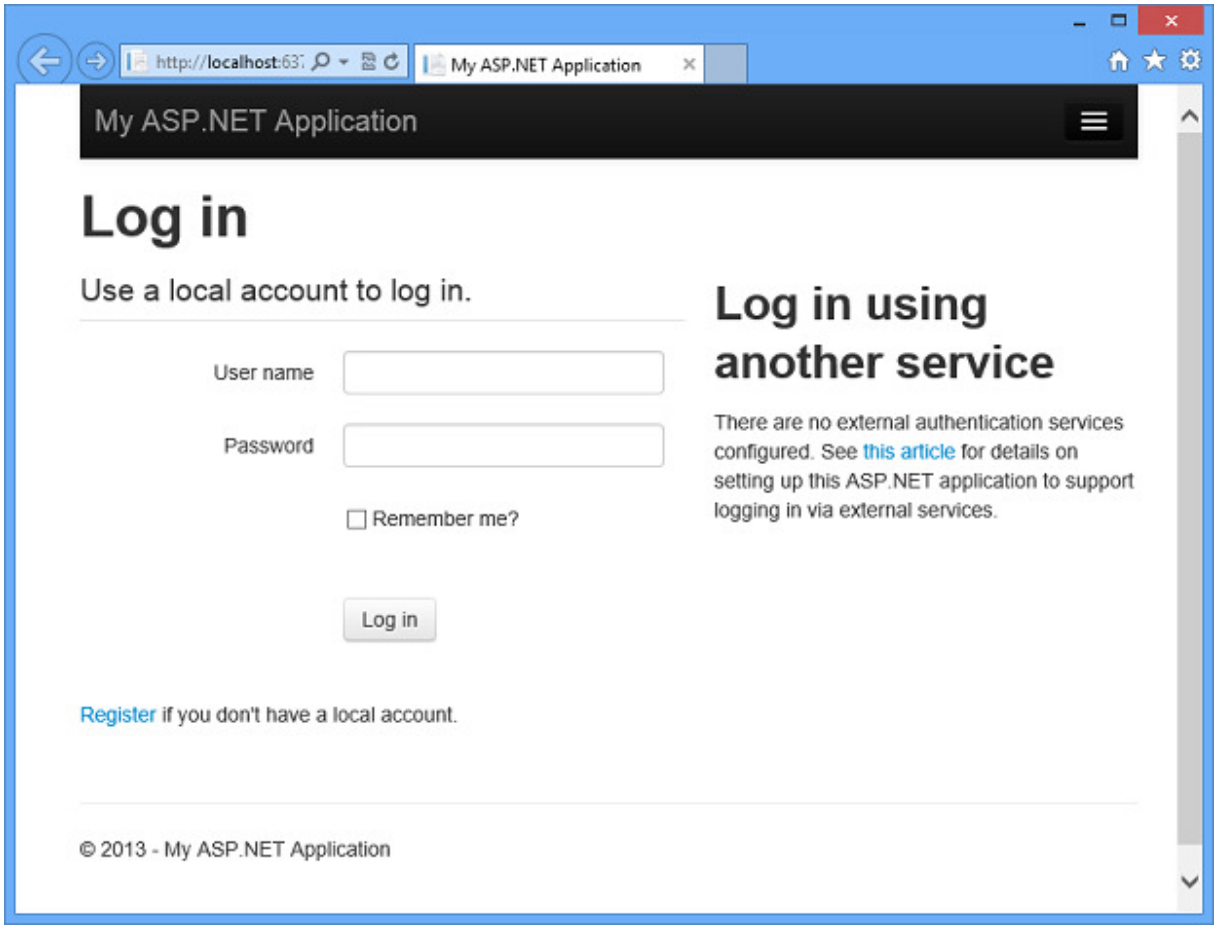
The following illustrations show sample help pages.



For more information about Web API, see ASP.NET Web API.

Single Page Application Template

The Single Page Application (SPA) template creates a sample application that uses JavaScript, HTML 5, and KnockoutJS on the client, and ASP.NET Web API on the server.
The only authentication option for the SPA template is Individual User Accounts.
The following illustration shows the initial state of the sample application that the SPA template builds.



For information about how to create an application by using the SPA template, see Web API - External Authentication Services.
For more information about ASP.NET Single Page Applications, and about additional SPA templates that use JavaScript frameworks other than KnockoutJS, see the following resources:

- ASP.NET Single Page Application.
- Understanding Security Features in the SPA Template for VS2013 RC
- Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET

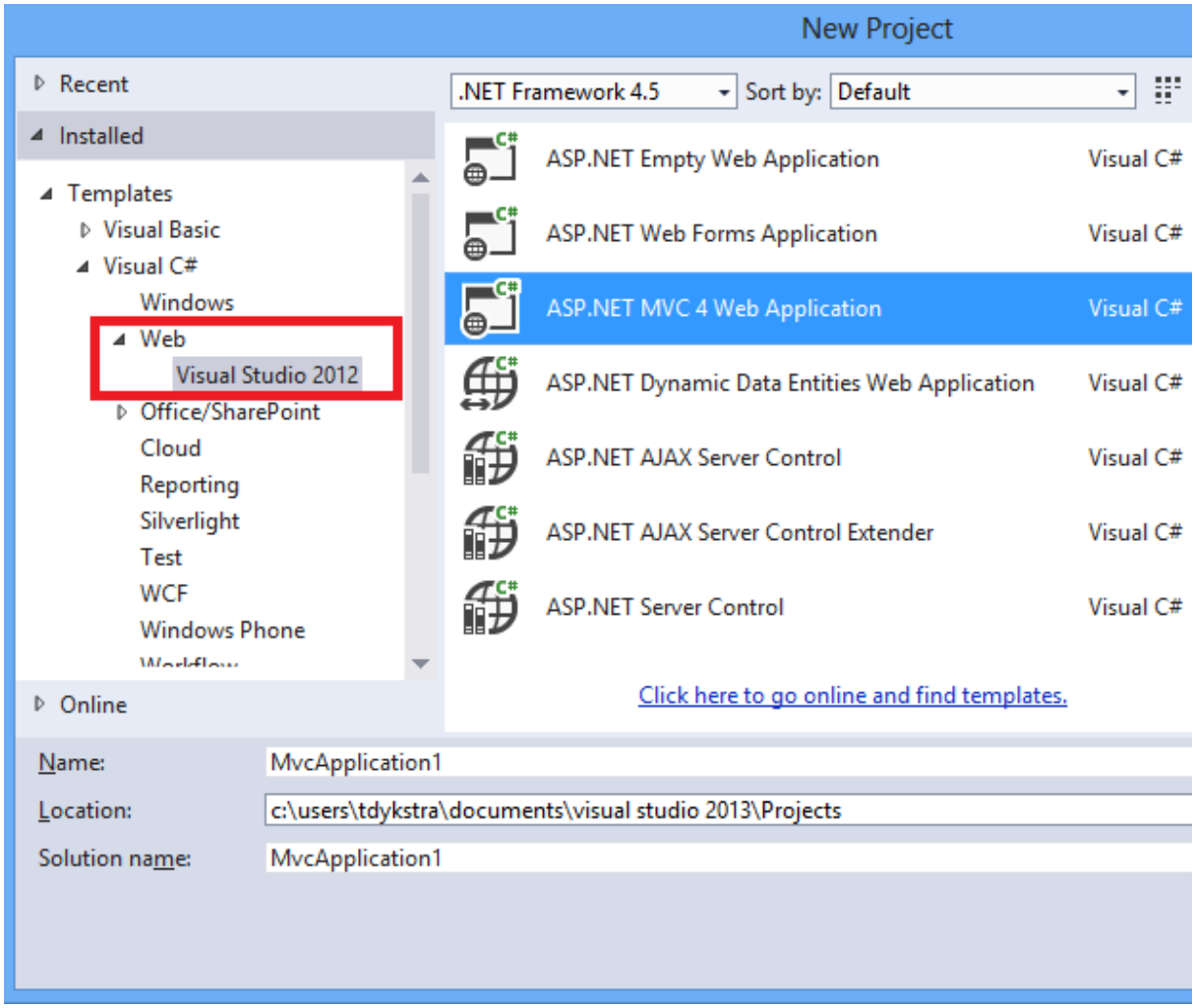
Facebook Template

You can install a Visual Studio extension that provides a Facebook template. This template creates a sample application that is designed to run inside the Facebook web site. It is based on ASP.NET MVC and uses Web API for real-time update functionality.
No authentication options are available for the Facebook template because Facebook applications run within the Facebook site and rely on Facebook's authentication.

For more information about ASP.NET Facebook applications, see Updating the MVC Facebook API and ASP.NET MVC Facebook Birthday App.

Visual Studio 2012 Templates

The Visual Studio 2013 web project creation dialog does not provide access to some templates that were available in Visual Studio 2012. If you want to use one of these templates, you can click the Visual Studio 2012 node in the left pane of the Visual Studio New Project dialog box.

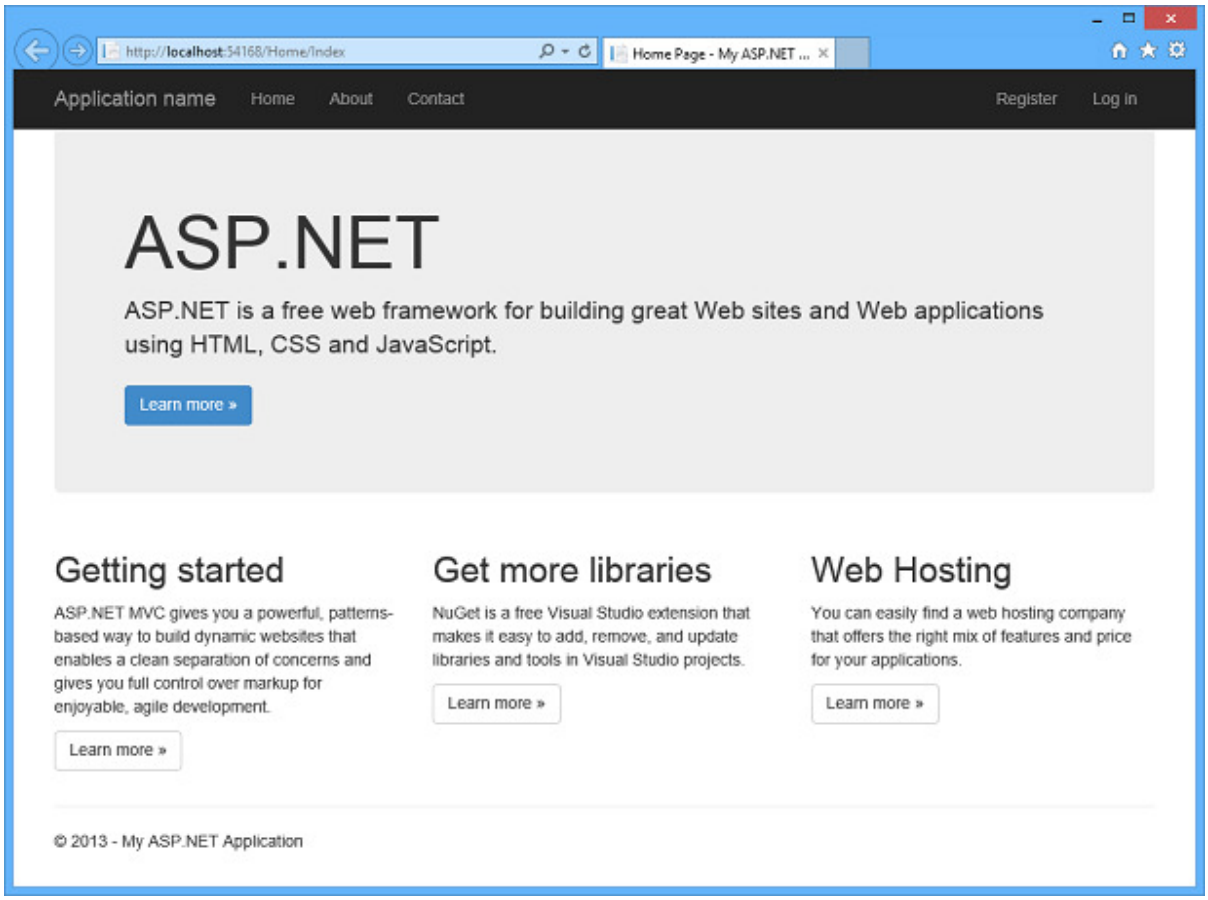


The **Visual Studio 2012** node lets you choose the following web templates that you don't have access to in the default list of templates for Visual Studio 2013:

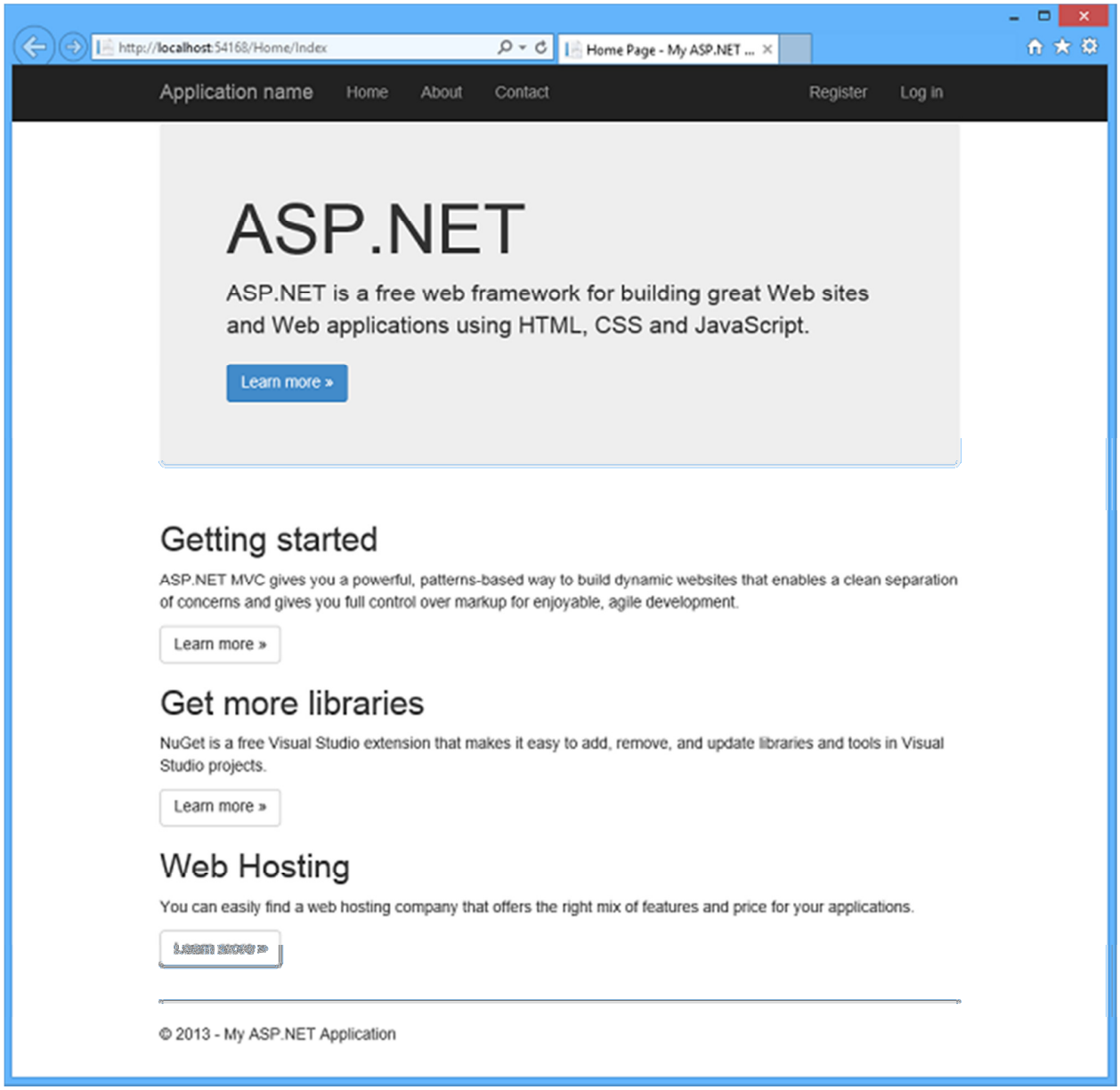
- ASP.NET MVC 4 Web Application
- ASP.NET Dynamic Data Entities Web Application
- ASP.NET AJAX Server Control
- ASP.NET AJAX Server Control Extender
- ASP.NET Server Control

Bootstrap in the Visual Studio 2013 web project templates

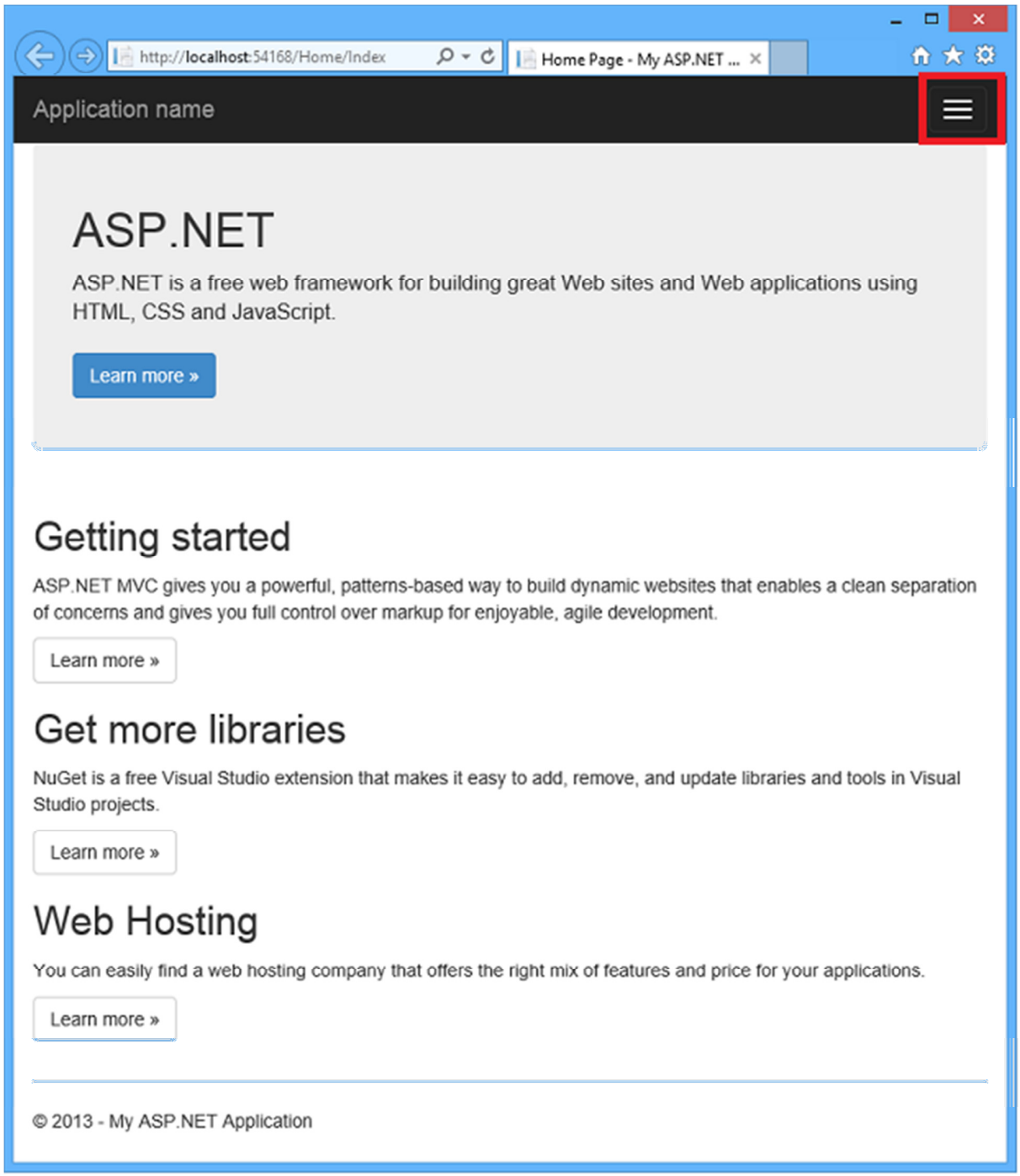
The Visual Studio 2013 project templates use Bootstrap, a layout and theming framework created by Twitter. Bootstrap uses CSS3 to provide responsive design, which means layouts can dynamically adapt to different browser window sizes. For example, in a wide browser window the home page created by the Web Forms template looks like the following illustration:

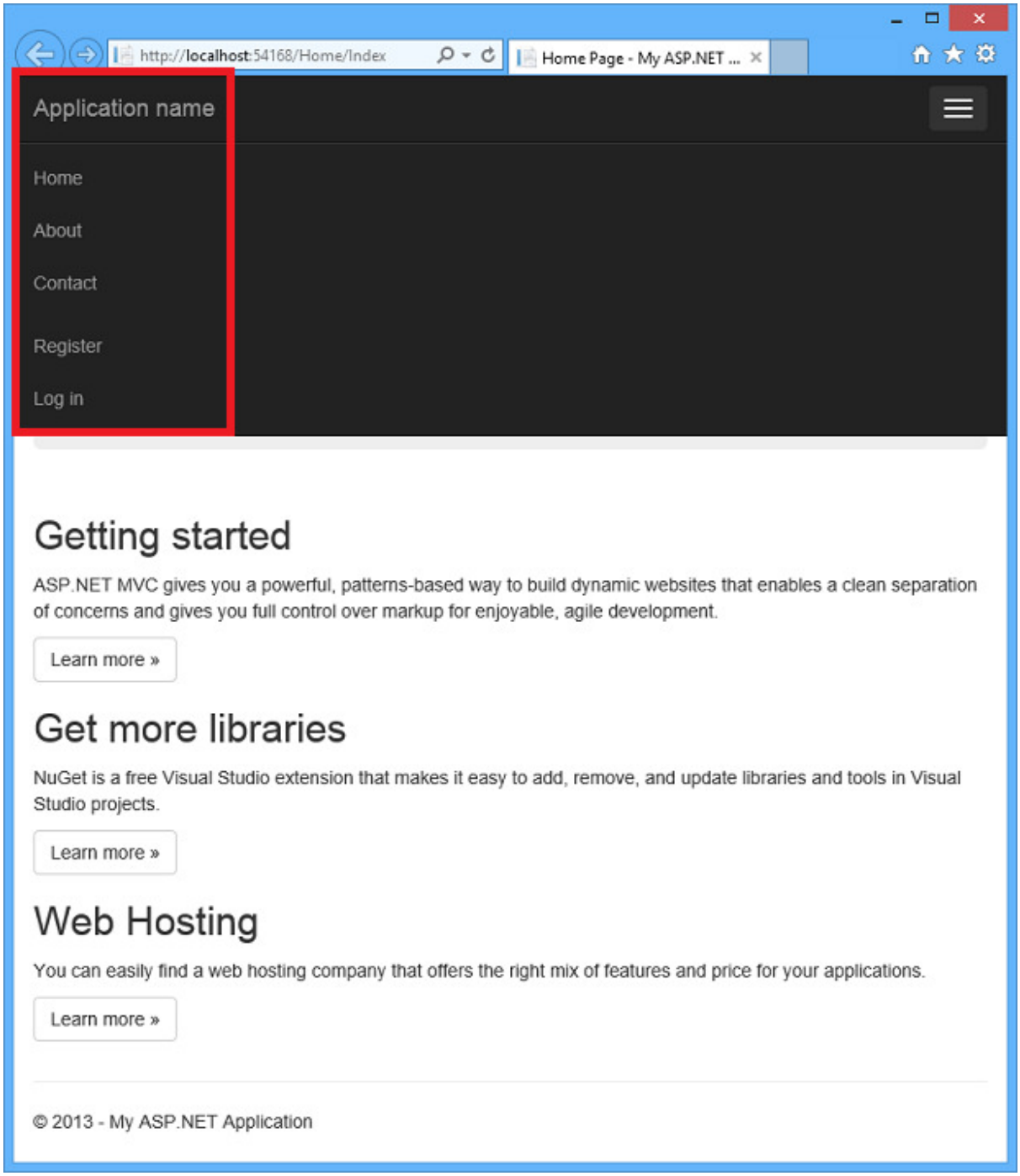


Make the window narrower, and the horizontally arranged columns move into vertical arrangement:



Narrow the window a little more, and the horizontal top menu turns into an icon that you can click to expand into a vertically oriented menu:

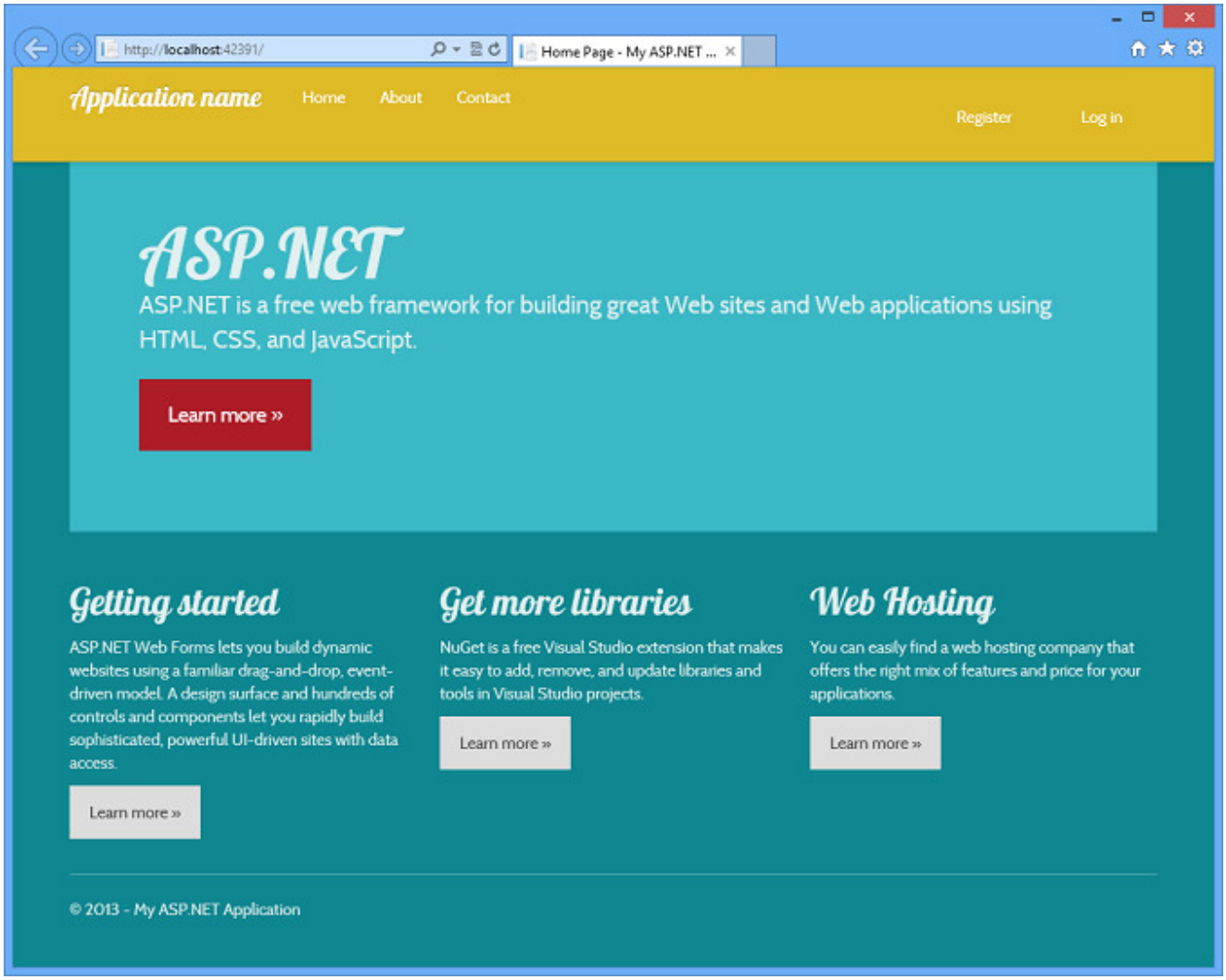




You can also use Bootstrap's theming feature to easily effect a change in the application's look and feel. For example, you can do the following steps to change the theme.

1. In your browser, go to http://Bootswatch.com, choose a theme, and then click **Download**. (This downloads *bootstrap.min.css* by default; if you want to examine the CSS code, get *bootstrap.css* instead of the minified version.)
2. Copy the contents of the downloaded CSS file.
3. In Visual Studio, create a new **Style Sheet** file named *bootstrap-theme.css* in the *Content* folder and paste the downloaded CSS code into it.
4. Open *App_Start/Bundle.config* and change *bootstrap.css* to *bootstrap-theme.css*.

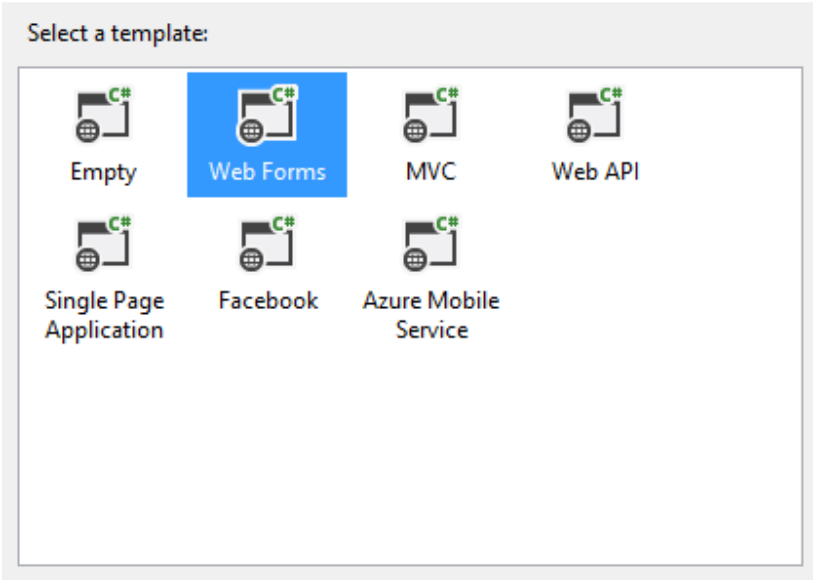
Run the project again, and the application has a new look. The following illustration shows the effect of the Amelia theme:

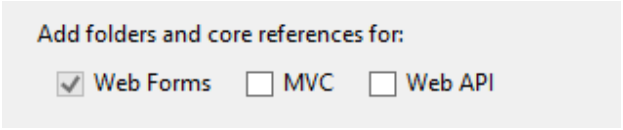


Many Bootstrap themes are available, both free and premium versions. Bootstrap also offers a wide variety of UI components, such as drop-downs, button groups, and icons. For more information about Bootstrap, see the Bootstrap site.
If you use the Web Forms designer in Visual Studio, note that the designer doesn't support CSS3, so it doesn't accurately show all the effects of Bootstrap themes or responsive layout changes. However, the Web Forms pages will display correctly when viewed with a browser.

Adding Support for Additional Frameworks

When you select a template, the check box for the framework(s) used by the template is automatically selected. For example, if you select the **Web Forms** template, the **Web Forms** check box is selected and you can't clear it.





You can select the check box for a framework that isn't included in the template in order to add support for that framework when the project is created. For example, to enable the use of Web Forms *.aspx* pages when you've selected the MVC template, select the **Web Forms** check box. Or to enable MVC when you're using the Web Forms template, click the **MVC** check box. Adding a framework enables design-time as well as run-time support. For example, if you add MVC support to a Web Forms project, you will be able to scaffold controllers and views.

If you combine Web Forms and MVC in a project and enable Friendly URLs in Web Forms, there might be unexpected routing problems where one URL has multiple possible targets. The routes that are defined first will take precedence. For example, if you have a `Home` controller and a *Home.aspx* page, the `http://contoso.com/home` URL will go to *Home.aspx* if you call the `EnableFriendlyUrls` method before you call the`MapRoute` method in *RouteConfig.cs*, or the same URL will go to the default view for your `Home` controller if you call `MapRoute` before`EnableFriendlyUrls`.

Adding a framework does not add any sample application functionality. For example, if you add Web Forms support when you've selected the MVC template, no *Default.aspx* home page file is created. Only the folders, files, and references required to support the framework are added. For that reason, adding frameworks doesn't change authentication options, which are implemented by code in sample applications created by the templates. For example, if you select the Empty template and add Web Forms or MVC support, the **Configure Authentication** button will still be disabled.

The following sections describe briefly the effect of each check box.

Add Web Forms Support

Creates empty *App_Data* and *Models* folders and a *Global.asax* file. These are already created by all templates other than the Empty template, so selecting the Web Forms check box makes no difference for other templates.

The Web Forms template enables Friendly URLs by default, but when you add Web Forms support to other templates by selecting the Web Forms check box Friendly URLs are not automatically enabled.

Add MVC Support

Installs MVC, Razor, and WebPages NuGet packages, creates empty *App_Data*, *Controllers*, *Models*, and *Views* folders, creates *App_Start* folder with *RouteConfig.cs* file, and creates *Global.asax* file.

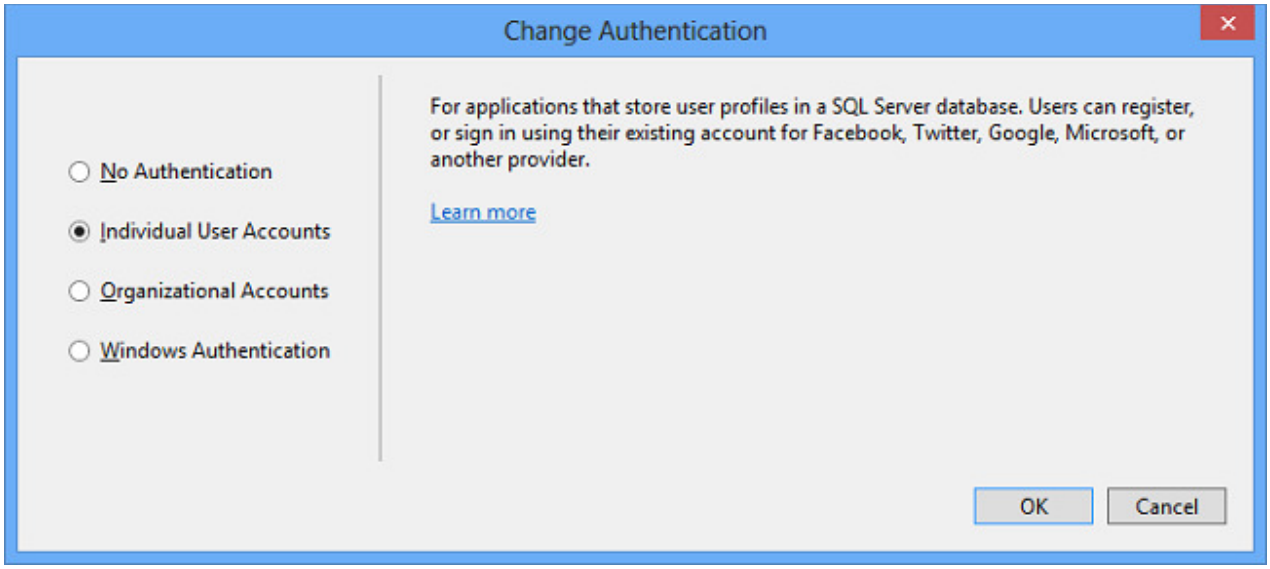
Add Web API Support

Installs WebApi and Newtonsoft.Json NuGet packages, creates empty *App_Data*, *Controllers*, and *Models* folders, creates *App_Start* folder with*WebApiConfig.cs* file, and creates *Global.asax* file.

Authentication Methods

Visual Studio 2013 offers several authentication options for the Web Forms, MVC, and Web API templates:

- No Authentication
- Individual User Accounts (ASP.NET Identity, formerly known as ASP.NET membership)
- Organizational Accounts (Windows Server Active Directory or Azure Active Directory)
- Windows Authentication (Intranet)



No Authentication

If you select **No Authentication**, the sample application will contain no web pages for logging in, no UI indicating who is logged in, no entity classes for a membership database, and no connection string for a membership database.

Individual User Accounts

If you select **Individual User Accounts**, the sample application will be configured to use ASP.NET Identity (formerly known as ASP.NET membership) for user authentication. ASP.NET Identity enables a user to register an account, by creating a username and password on the site or by signing in with social providers such as Facebook, Google, Microsoft Account, or Twitter. The default data store for user profiles in ASP.NET Identity is a SQL Server LocalDB database, which you can deploy to SQL Server or Azure SQL Database for the production site.
In Visual Studio 2013 these features are the same as in Visual Studio 2012, but the underlying code for the ASP.NET membership system has been rewritten. Advantages of the new code base include the following:

- The new membership system is based on OWIN rather than the ASP.NET Forms Authentication module. This means that you can use the same authentication mechanism whether you're using Web Forms or MVC in IIS, or you're self-hosting Web API or SignalR.
- The new membership database is managed by Entity Framework Code First, and all of the tables are represented by entity classes that you can modify. This means that you can easily customize the database schema and profile-related web UI to fit your own needs, and you can easily deploy your updates using Code First Migrations.

The new membership system is implemented automatically in the new templates, and it can be implemented manually in any project that targets .NET 4.5 or later.

ASP.NET Identity is a good choice if you are creating an Internet web site which is mainly for external customers. If your organization uses Active Directory or Office 365 and you want to create a project that enables single-sign-on for employees and business partners, the **Organizational Accounts** option might be a better choice.
For more information about the Individual User Accounts option, see the following resources:

- www.asp.net/identity. Documentation about ASP.NET Identity on the ASP.NET web site.
- Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on. Also shows how to customize user profile data.
- Web API - External Authentication Services
- Adding External Logins to your ASP.NET application in Visual Studio 2013

Organizational Accounts

If you select **Organizational Accounts**, the sample application will be configured to use Windows Identity Foundation (WIF) for authentication based on user accounts in Azure Active Directory (Azure AD, which includes Office 365) or Windows Server Active Directory. For more information, see Organizational account authentication options later in this topic.

Windows Authentication

If you select **Windows Authentication**, the sample application will be configured to use the Windows Authentication IIS module for authentication. The application will display the domain and user ID of the Active directory or local machine account that is logged into Windows but won't include user registration or log-in UI. This option is intended for Intranet web sites.
Alternatively, you can create an Intranet site that uses AD authentication by choosing the On-Premises option under Organizational Accounts. The On-Premises option uses Windows Identity Foundation (WIF) instead of the Windows Authentication module. Some additional steps are required in order to set up the On-Premises option, but WIF enables features that aren't available with the Windows Authentication module. For example, with WIF you can configure application access in Active Directory and query directory data.

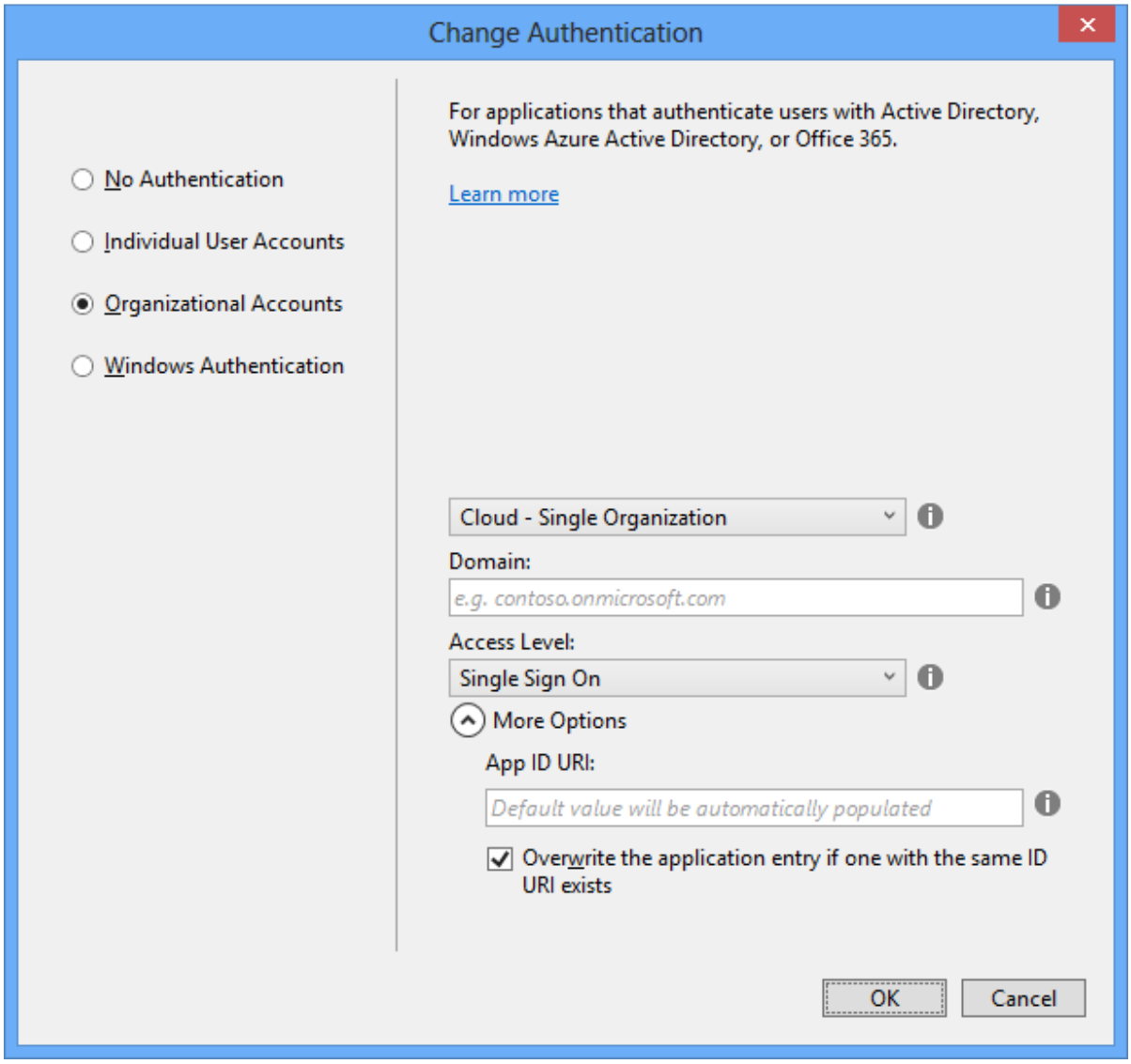
Organizational account authentication options

The **Configure Authentication** dialog gives you several options for Azure Active Directory (Azure AD, which includes Office 365) or Windows Server Active Directory (AD) account authentication:

- Cloud - Single Organization (Azure AD, or AD using directory integration with Azure AD)
- Cloud - Multi Organization (Azure AD, or AD using directory integration with Azure AD)
- On-Premises (AD)

If you want to try one of the Azure AD options but don't have an account yet, click here to sign up for a Azure AD account.
Note: If you choose one of the Azure AD options, your project requires a database and you have to sign in to a global administrator account for your Azure AD tenant. Enter the name and password for an organizational account (for example, admin@contoso.onmicrosoft.com) that has administrative permissions for your Azure AD tenant.
Don’t enter credentials for a Microsoft account (for example, contoso@hotmail.com) in the sign-in dialog box.

Cloud - Single Organization Authentication



Choose this option if you want to enable authentication for user accounts that are defined in one Azure AD tenant. For example, the site is contoso.com and it will be made available to employees of the Contoso Company who are in the contoso.onmicrosoft.com tenant. You won't be able to configure Azure AD to allow users from other tenants to access the application.

Domain

Enter the Azure AD domain that you want to set up the application in, for example: `contoso.onmicrosoft.com`. If you have a custom domain, such as `contoso.com` instead of `contoso.onmicrosoft.com`, you can enter that here.

Access Level

If the application needs to query or update directory information by using the Graph API, choose **Single Sign-On, Read Directory Data** or**Single Sign-On, Read and Write Directory Data**. Otherwise, choose **Single Sign-On**. For more information, see Application Access Levelsand Using the Graph API to Query Azure AD.

Application ID URI

By default, the template creates an application ID URI for you by appending the project name to the Azure AD domain. For example, if the project name is `Example` and the domain is `contoso.onmicrosoft.com`, the application ID URI becomes`https://contoso.onmicrosoft.com/Example`. If you want to manually specify the application ID URI, expand the **More Options** section and enter the application ID URI in the text box. The application ID URI must begin with `https://`.
By default, if an application that is already provisioned in Azure AD has the same application ID URI as the one that Visual Studio is using for the project, the project will be connected to the existing application instead of provisioning a new one. If you want a new application to be provisioned in that case, clear the **Overwrite the application entry if one with the same ID already exists** check box.
If the **Overwrite** check box is cleared, and Visual Studio finds an existing application with the same application ID URI, it creates a new URI by appending a number to the URI it was going to use. For example, suppose the project name is `Example`, you leave the text box blank, you clear the **Overwrite** check box, and the Azure AD tenant already has an application with the URI `https://contoso.onmicrosoft.com/Example`. In that case, a new application will be provisioned with an application ID URI like`https://contoso.onmicrosoft.com/Example_20130619330903`.

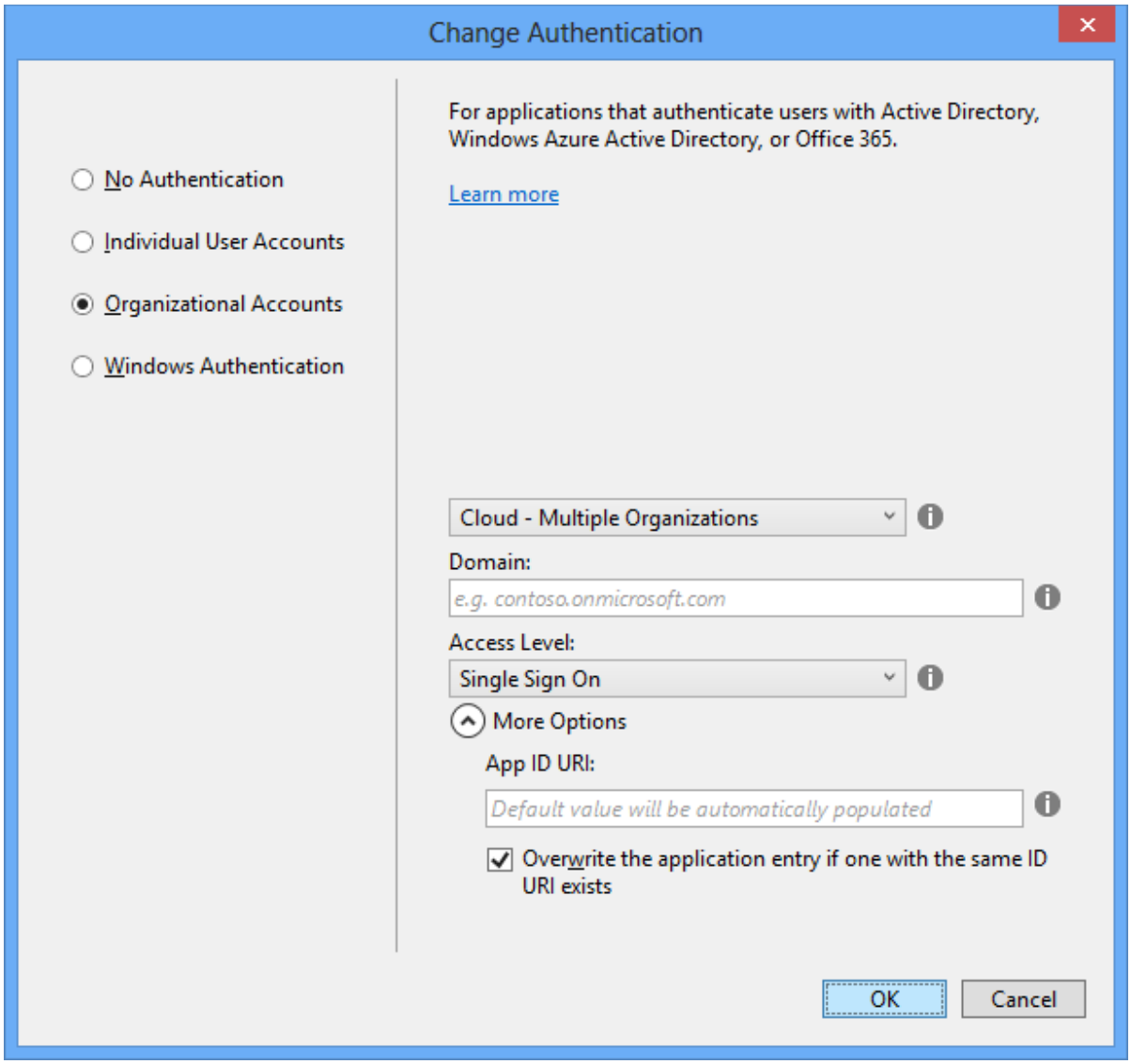
Provisioning the application in Azure AD

In order to provision the application in Azure AD or connect the project to an existing application, Visual Studio needs the credentials of a global administrator for the domain. When you click **OK** in the **Configure Authentication** dialog box, you are prompted for the user name and password of a global administrator for the domain you specified. Later, when you click **Create Project** in the **New ASP.NET Project**dialog, Visual Studio provisions the application in Azure AD. Note that as part of this process Visual Studio embeds client secret values in the Web.config file that expire one year after creation.
For information about how to create applications that use **Cloud - Single Organization** authentication, see the following resources:

- Azure Authentication
- Adding Sign-On to Your Web Application Using Azure AD
- Developing ASP.NET Apps with Azure Active Directory
- Secure ASP.NET Web API with Azure AD and Microsoft OWIN Components

The tutorials have not yet been updated for Visual Studio 2013; some of what the tutorials direct you to do manually is automated in Visual Studio 2013.

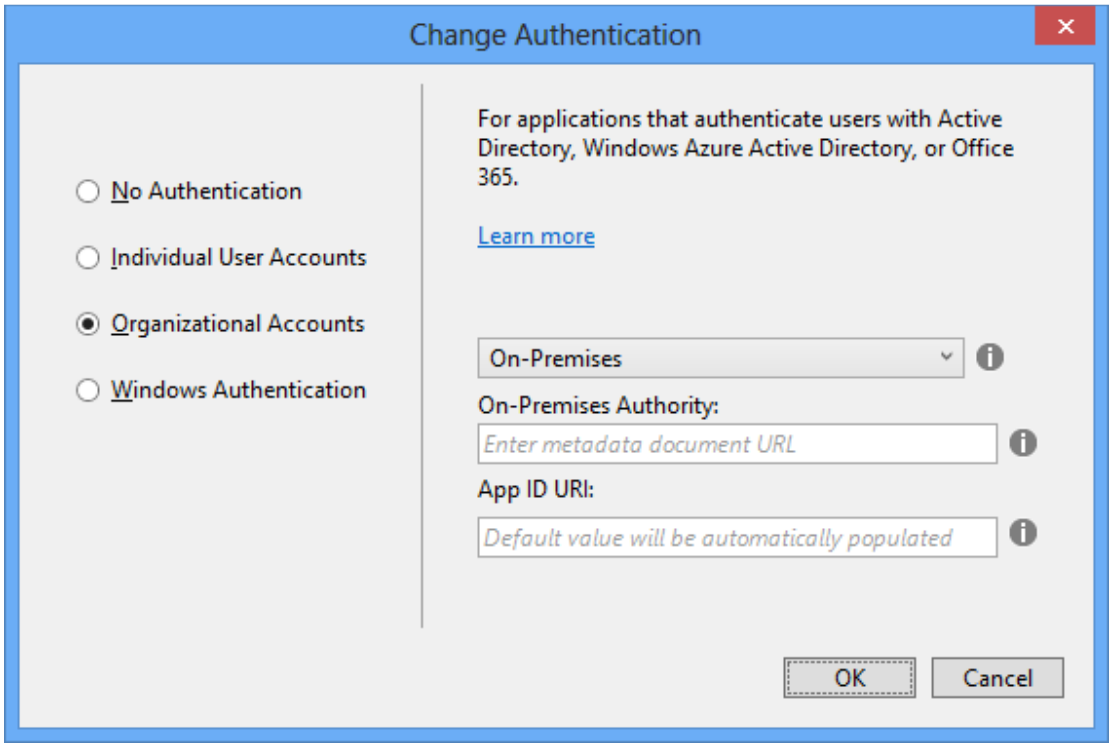
Cloud - Multi Organization Authentication



Choose this option if you want to enable authentication for user accounts that are defined in multiple Azure AD tenants. For example, the site is contoso.com and it will be made available to employees of the Contoso Company who are in the contoso.onmicrosoft.com tenant, and employees of the Fabrikam Company who are in the fabrikam.onmicrosoft.com tenant. The settings that you enter and the application provisioning step are similar to single organization authentication.
For information about how to create applications that use **Cloud - Multi Organization** authentication, see the following resources:

- Easy Web App Integration with Azure Active Directory, ASP.NET & Visual Studio on the Active Directory Team blog.
- Developing Multi-Tenant Web Applications with Azure AD tutorial. The tutorial hasn't yet been updated for Visual Studio 2013; some of what the tutorial directs you to do manually is automated in Visual Studio 2013.
- You Have to Sign Up With Your Own Multiple Organizations ASP.NET App Before You Can Sign In. Blog by Vittorio Bertocci that explains how to resolve a common problem people encounter when creating a project that uses multi-organization authentication.

On-Premises Organizational Authentication



Choose this option if you want to enable authentication for user accounts that are defined in Windows Server Active Directory (AD), and you don't want to use Azure AD. You can use this option to create an Intranet site or an Internet site. For an Internet site, use Active Directory Federation Services (ADFS) to provide access to AD. For more information, see Use the On-Premises Organizational Authentication Option (ADFS) With ASP.NET in Visual Studio 2013.
For an Intranet site, as an alternative you can choose Windows Authentication instead of this option. For the Windows Authentication option you don't have to provide a metadata document URL. However, Windows Authentication does not give you the ability to control application access in Active Directory or to query directory data.

On-Premises Authority

Enter a URL that points to the metadata document. The metadata document contains the coordinates of the authority. The application will use those coordinates to drive the web sign-on flow.

Application ID URI

Provide a unique URI that AD can use to identify this application, or leave blank to let Visual Studio create one.