.NET Framework: Developing Modern Web Apps with ASP.NET MVC – Workshop*PLUS*

Module 1: Overview

Student Lab Manual

Instructor Edition (Book Title Hidden Style)

Version 1.0

**Conditions and Terms of Use**

**Microsoft Confidential**

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2017 Microsoft Corporation. All rights reserved.

**Copyright and Trademarks**

© 2017 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at  
[*http*://www.microsoft.com/about/legal/permissions/](http://www.microsoft.com/about/legal/permissions/)

Internet Explorer, Microsoft, Visual Studio, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

[Lab 1: Visual Studio and ASP.NET MVC 5 Overview 1](#_Toc480803417)

[Exercise 1: Create a New ASP.NET MVC 5 Application 3](#_Toc480803418)

[Exercise 2: Middleware 6](#_Toc480803419)

# Lab 1: Visual Studio and ASP.NET MVC 5 Overview

#### Introduction

ASP.NET is a free web framework for building great websites and web applications using HTML, CSS, and JavaScript. You can also create Web APIs and use real-time technologies like Web Sockets.

[ASP.NET Core](https://docs.microsoft.com/aspnet/core/) is an alternative to ASP.NET. See the [guidance on how to choose between ASP.NET and ASP.NET Core](https://docs.microsoft.com/aspnet/core/choose-aspnet-framework).

ASP.NET offers three frameworks for creating web applications: Web Forms, ASP.NET MVC, and ASP.NET Web Pages. All three frameworks are stable and mature, and you can create great web applications with any of them. No matter what framework you choose, you will get all the benefits and features of ASP.NET everywhere.

ASP.NET MVC 5 is a framework for building scalable, standards-based web applications using well-established design patterns and the power of ASP.NET and the .NET Framework. ASP.NET MVC 5 includes ASP.NET Web API, a framework for creating HTTP services that can reach a broad range of clients including browsers and mobile devices.

ASP.NET MVC 5 gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup for enjoyable, agile development. ASP.NET MVC includes many features that enable fast, TDD-friendly development for creating sophisticated applications that use the latest web standards.

#### Objectives

This lab will show you how to:

* Create a new ASP.NET MVC 5 application in Visual Studio 2017.
* Create custom middleware.

#### System Requirements

To complete this lab, you need:

* Visual Studio 2017

#### Hosted Lab Credentials

If the lab is exercised in Microsoft cloud environment, use the following user credentials to sign in:

* Username: aspnetuser
* Password: @Cir9hvc6!w

#### Estimated Time to Complete This Lab

30 minutes

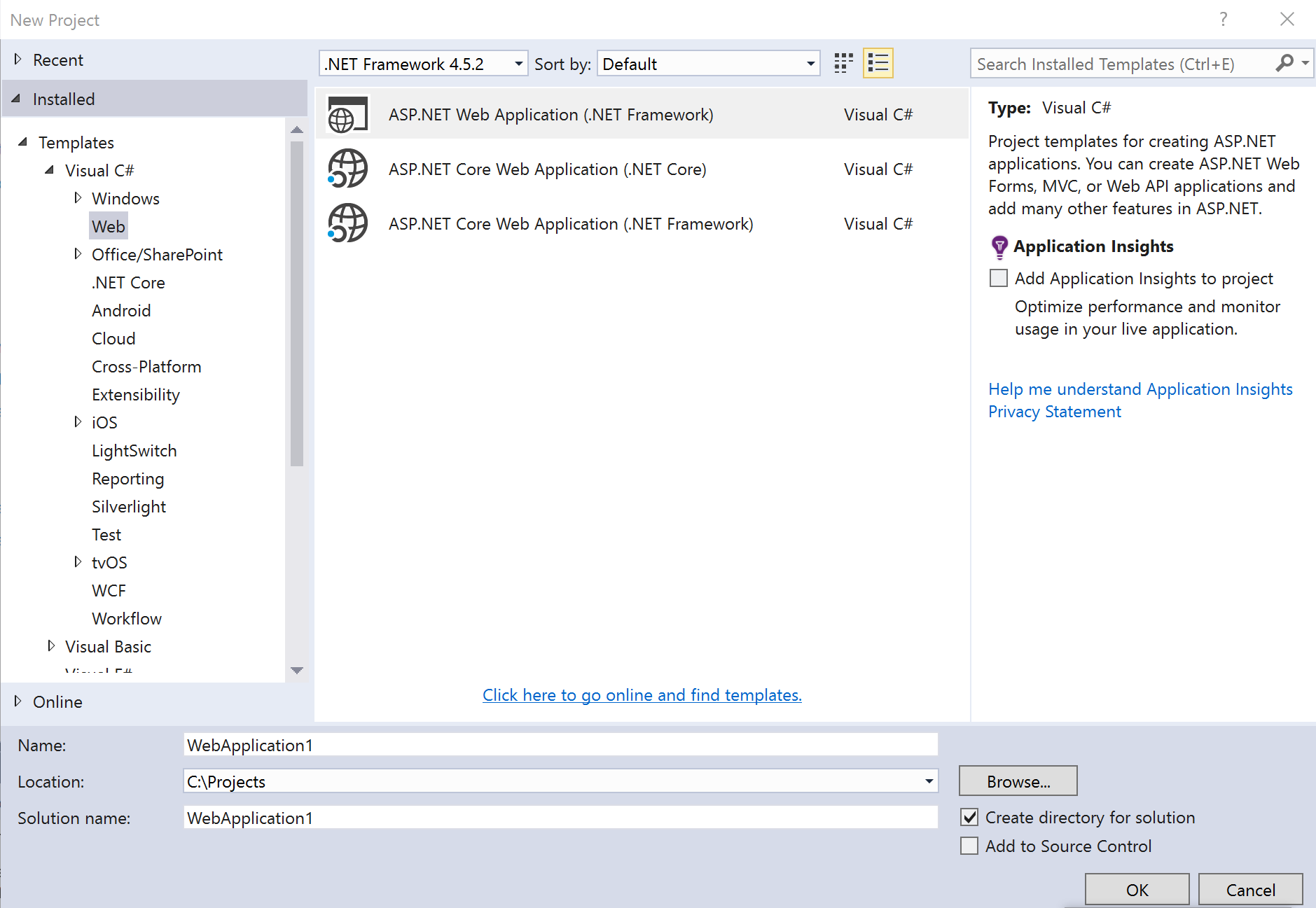
Exercise 1: Create a New ASP.NET MVC 5 Application

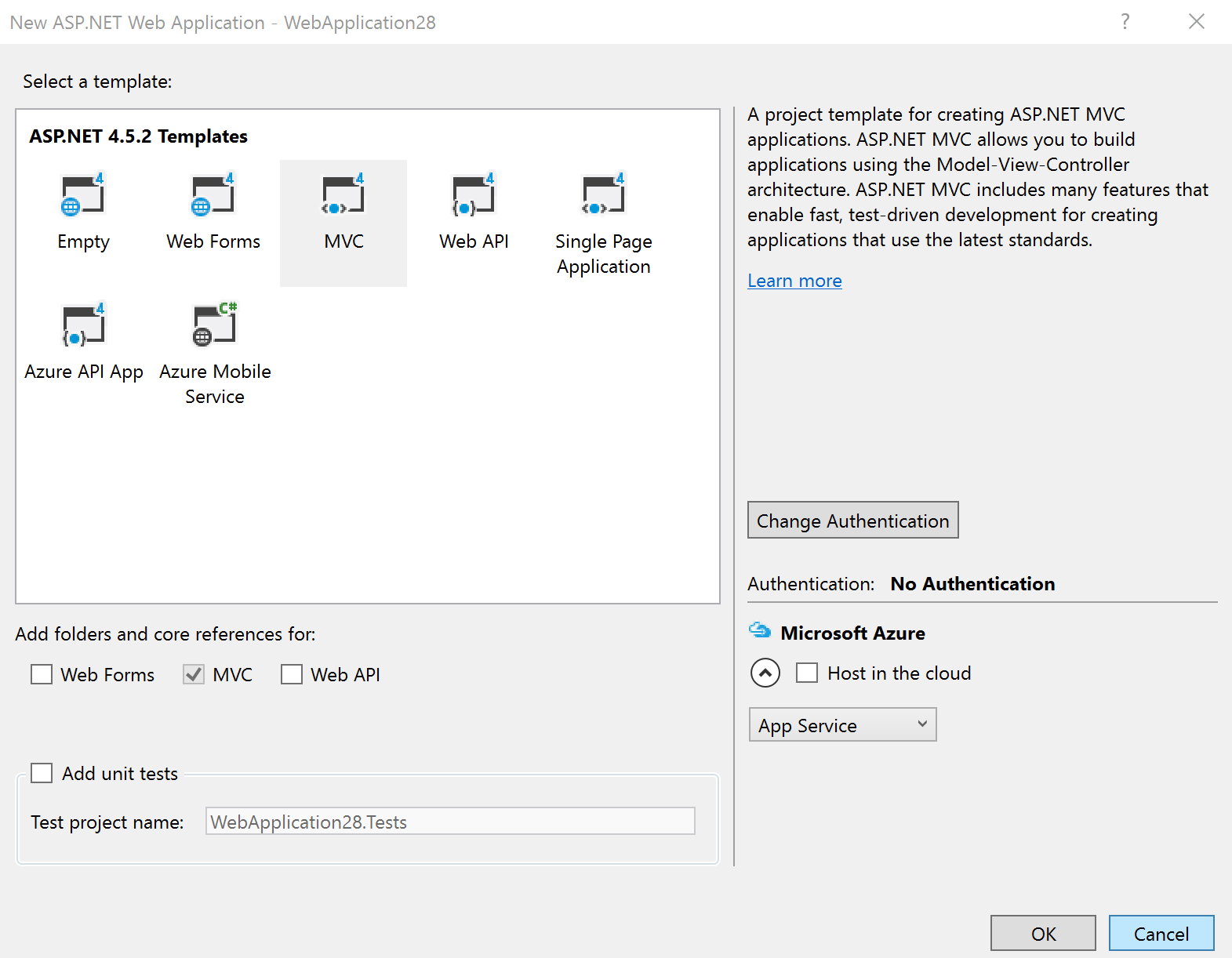
#### Objectives

In this exercise, you will:

* Create a new ASP.NET MVC 5 application in Visual Studio 2017.

Task 1: Create the Visual Studio solution

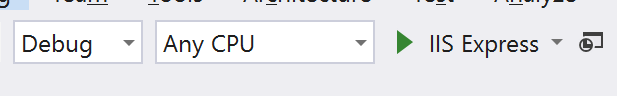
1. Open Visual Studio 2017.
2. Create a new ASP.NET MVC application project by going to **File** > **New Project**.
3. Under templates, go to Visual C# > Web, and then choose ASP.NET Web Application (.NET Framework). 
4. Name the project *HelloWorld*, and change the location as per your own preferences. Leave the check box selected for **Create directory for solution**. Click **OK**.
5. Choose **MVC** under ASP.NET Templates. Use the **Change Authentication** button to select “No authentication” and leave the Host in the cloud option cleared.



1. Click **OK**.
2. Visual Studio will take a few seconds to restore NuGet packages and setup a project.
3. Build the application to ensure everything compiles well.

Task 2: Run the Application on Different Browsers

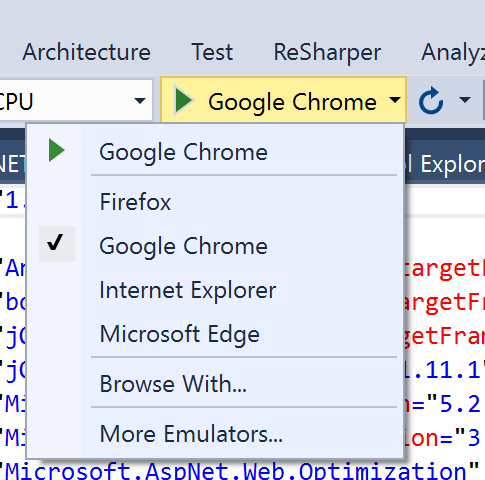
1. Run the application using IIS Express. Go to different application views.



1. Take some time to review the following aspects of the project:

* Global.asax
* packages.config
* Client-side dependencies – Note bootstrap, JQuery
* Controllers, Models, Views, and Content folders
* BundleConfig.cs
* FilterConfig.cs
* RouteConfig.cs

1. Now that you are a bit more familiar with ASP.NET MVC 5 project template and files involved, let us try to run the application using different browsers.
2. Choose “IIS Express” (IIS Express) as the hosting server in Visual Studio and run the application.



1. Try with different browsers.

Exercise 2: Middleware

#### Introduction

Middleware are components that are assembled into an application pipeline to handle requests and responses. Each component can choose whether to pass the request on to the next component in the pipeline, and can perform certain actions before and after the next component in the pipeline. Request delegates are used to build this request pipeline, which are then used to handle each incoming HTTP request to your application.

Request delegates are configured using Run, Map, and Use extension methods on the IAppBuilder type that is passed into the Configure method in the Startup class. An individual request delegate can be specified in-line as an anonymous method, or it can be defined in a reusable class. These reusable classes are middleware, or middleware components. Each middleware component in the request pipeline is responsible for invoking the next component in the chain, or can opt to short-circuit the chain if appropriate.

#### Objectives

In this exercise, you will:

* Create a custom middleware
* Use extension methods for logging Requests

Task 1: Create a custom middleware

1. Open Visual Studio 2017.
2. Create a new ASP.NET MVC application project by going to **File** > **New Project**.
3. Under templates, go to **Visual C#** > **Web**, and then choose **ASP.NET Web Application (.NET Framework)**.
4. Name the project MiddlewareDemo.
5. Choose **MVC** under ASP.NET Templates. Use the **Change Authentication** button to select “No authentication” and leave the Host in the cloud option cleared.
6. Right-click the project and choose **Add** > **Class**, and name it **RequestLoggerMiddleware**.**cs**
7. Replace the code inside class with the following:

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using Microsoft.Owin.Logging;

using Owin;

namespace MiddlewareDemo

{

using AppFunc = Func<IDictionary<string, object>, Task>;

public class RequestLoggerMiddleware

{

private readonly ILogger \_logger;

AppFunc \_next;

public RequestLoggerMiddleware(AppFunc next, IAppBuilder app)

{

\_next = next;

\_logger = app.CreateLogger<RequestLoggerMiddleware>();

}

public async Task Invoke(IDictionary<string, object> environment)

{

\_logger.WriteInformation($"Handling request: {environment["owin.RequestPath"]}");

await \_next(environment);

\_logger.WriteInformation("Finished handling request.");

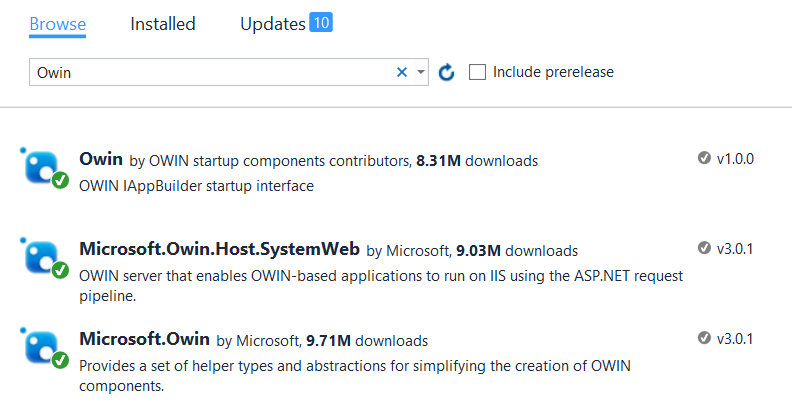
}

}

}

1. Now you need to fix references by adding the required Nuget Packages.
2. Right-click **References** and choose **Manage Nugget Packages.**
3. Click **Browse** and search for **Owin,** and then install following packages:

* Owin
* Microsoft.Owin
* Microsoft.Owin.Host.SystemWeb



1. Note the **AppFun** definition before class definition, you are using this to simplify Func<IDictionary<string, object>, Task> so you don’t have to keep retyping it.

Also note how **next** middleware and application builder **app** is passed through the constructor.

1. Now note the **Invoke** method, which is responsible for the actual work and responsible for calling the next middleware inside the pipeline.
2. Now you need to create an extension method, so you can use it to attach the middleware to the pipeline.
3. Create a new class by right-clicking the project and choose **Add** > **Class**, and name it **RequestLoggerExtension.cs**
4. Replace the class code with the following:

using Owin;

namespace MiddlewareDemo

{

public static class RequestLoggerExtension

{

public static IAppBuilder UseRequestLogger(this IAppBuilder builder)

{

return builder.Use<RequestLoggerMiddleware>(builder);

}

}

}

1. Every OWIN Application has a startup class where you specify components for the application pipeline. There are different ways you can connect your startup class with the runtime, depending on the hosting model you choose. The startup class can be used in every hosting application. You connect the startup class with the hosting runtime using one of the these approaches:
   * **Naming Convention**: Katana looks for a class named Startup in namespace matching the assembly name or the global namespace.
   * **OwinStartup Attribute**: This is the approach most developers will take to specify the startup class. The following attribute will set the startup class to the **Startup** class in the **MiddlewareDemo** namespace.

[assembly: OwinStartup(typeof(MiddlewareDemo.Startup))]

1. You will use the second option so let’s create a new startup class by right-clicking the project and choose **Add** > **Class**, and name it **Startup.cs**
2. Replace the **Startup** class code with the following:

using Microsoft.Owin;

using Owin;

[assembly: OwinStartup(typeof(MiddlewareDemo.Startup))]

namespace MiddlewareDemo

{

public static class Startup

{

public static void Configuration(IAppBuilder app)

{

app.UseRequestLogger();

}

}

}

**Note:** You could also use app.Use<RequestLoggerMiddleware>(app) in the **Configuration** method to have the same result without creating an extension method.

1. Note the **OwinStartup** attribute and how you are using our created extension method to use our custom middleware inside the **AppBuilder**.

**Note:** You configure the HTTP pipeline using the extensions **Run, Map**, and **Use**. By convention, the Run method is simply a shorthand way of adding middleware to the pipeline that does not call any other middleware (that is, it will not call a next request delegate). Thus, Run should only be called at the end of your pipeline. Run is a convention, and some middleware components may expose their own Run[Middleware] methods that should only run at the end of the pipeline.

1. One last configuration remaining is setting up trace level. In **web.config** file, insert following lines just after </system.web> tag:

<system.diagnostics>

<switches>

<add name="Microsoft.Owin" value="Information" />

</switches>

</system.diagnostics>

1. Run the application now and navigate to the about page.
2. Go back to Visual Studio and click the **Output** window, note the logs.

