Azure Developer Immersions

Application Insights

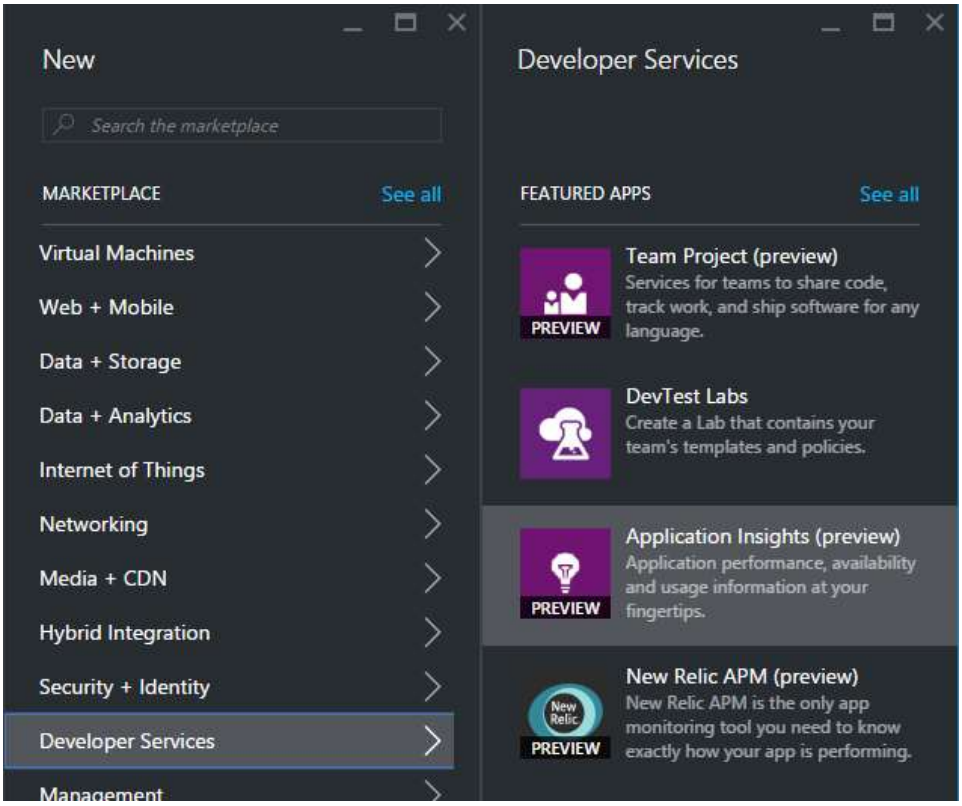
Application Insights provides live monitoring of your applications. You can detect and diagnose faults and performance issues, as well as discover how users are using your app. In this lab, you will create a new Application Insights instance.

There is one exercise in this walkthrough:

1. Configure Application Insights for your Web App

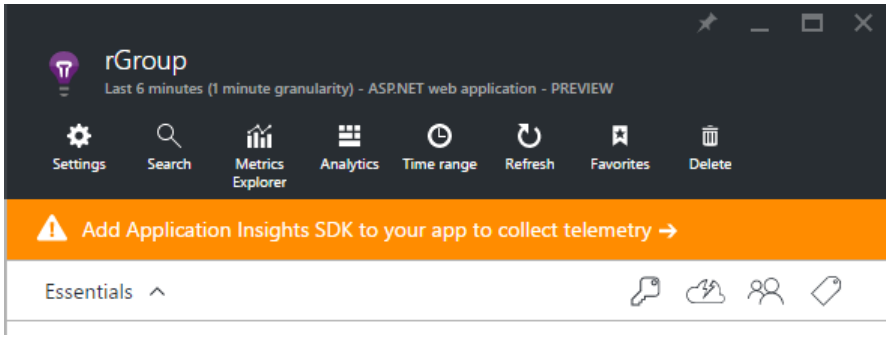
Exercise 1: Configure Application Insights for your Web App

1. In the Azure portal, click **+ New**, select **Developer Services,** and find and select **Application Insights (preview)** in the list that appears.



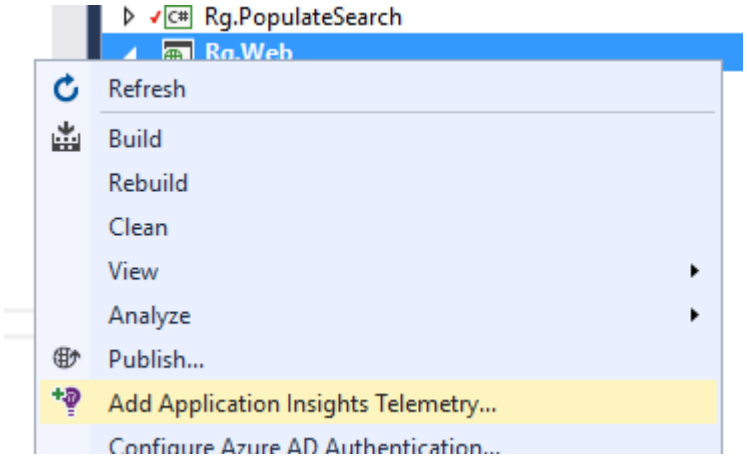
2. Set the **Name** to **rGroup**.
3. Set the **Resource group** to **rGroup**.
4. Note that you might not be able to set the location to be the same as the one you've been using so far in this lab. (At the time of writing, the list only shows a single location: Central US.) This doesn't matter—your web app will send telemetry to Application Insights in the background, so it is not critical to have low latency communication.
5. Click **Create**. Azure will create the Application Insights instance. When this has finished, open the blade for it. (As always, you can just click the relevant icon in your resource group.) At the

top, you should see an orange banner inviting you to add the Application Insights SDK to your app.

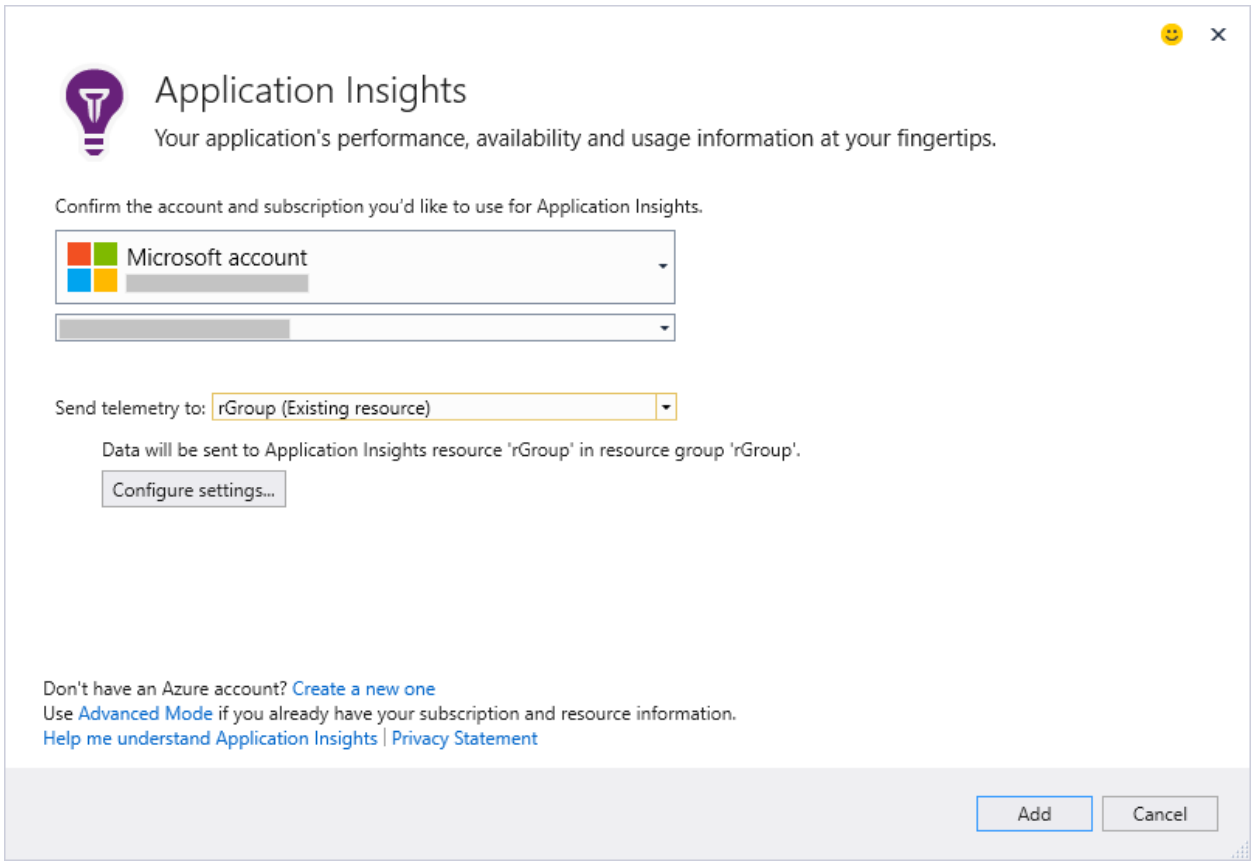


6. If you click on the arrow, you'll see that it simply tells you to get Visual Studio to do the work for you.

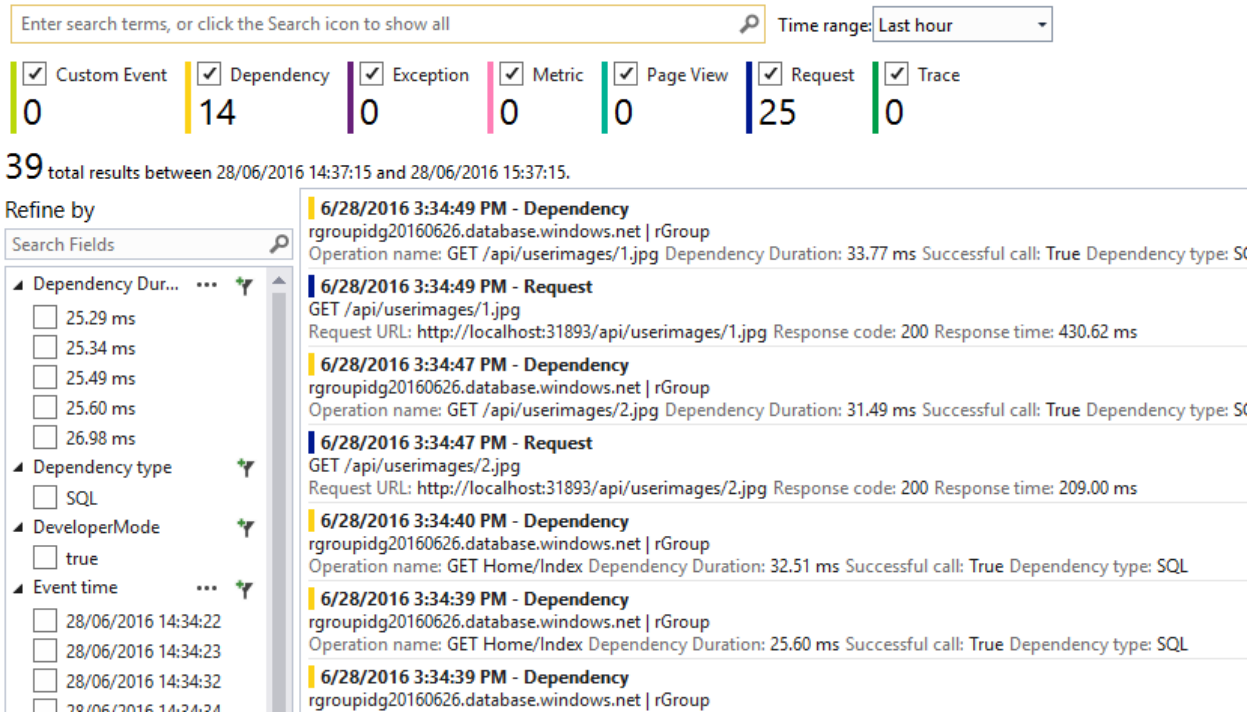
7. In Visual Studio, right-click on the **Rg.Web** project and select **Add Application Insights Telemetry**.



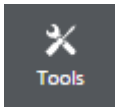
8. In the **Application Insights** dialog that opens, the **Send telemetry to** field is a drop-down list that will show every Application Insights instance in your Azure Subscription. Select **rGroup**.



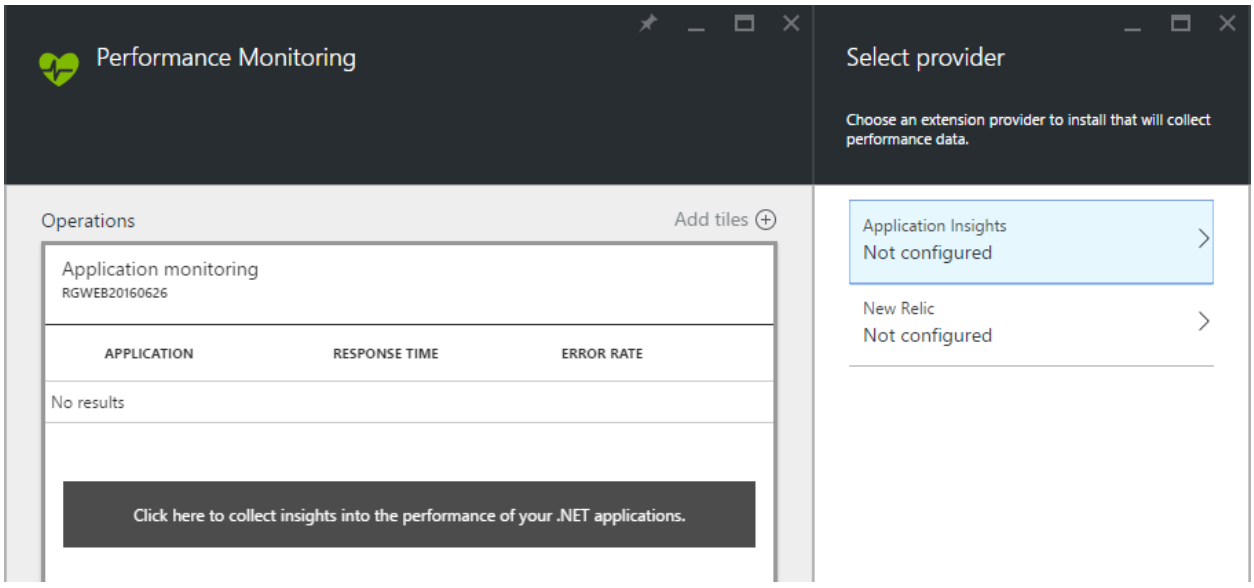
9. Click the **Add** button. This will add the **Microsoft.ApplicationInsights.Web** and **Microsoft.ApplicationInsights.JavaScript** NuGet packages to your project. It also adds entries to the **Web.config** file to ensure Application Insights can get access to information about what your application is doing It also adds a new **ApplicationInsights.config** file configuring how the SDK collects information.
10. Visual Studio is often a little behind the latest version of the Application Insights SDK, and the Azure Portal might complain, so you should right click on the **Rg.Web** project's **References** node and select **Manage NuGet Packages**. Select the **Updates** page. If an update for **Microsoft.ApplicationInsights.Web** is available, select it and click **Update**.
11. Press **F5** to run the application locally. Navigate around the application a little bit.
12. Return to Visual Studio, stop debugging, and then right-click on **Rg.Web** and select **Application Insights | Search Debug Session Telemetry**. This shows all of the events captured by Application Insights.



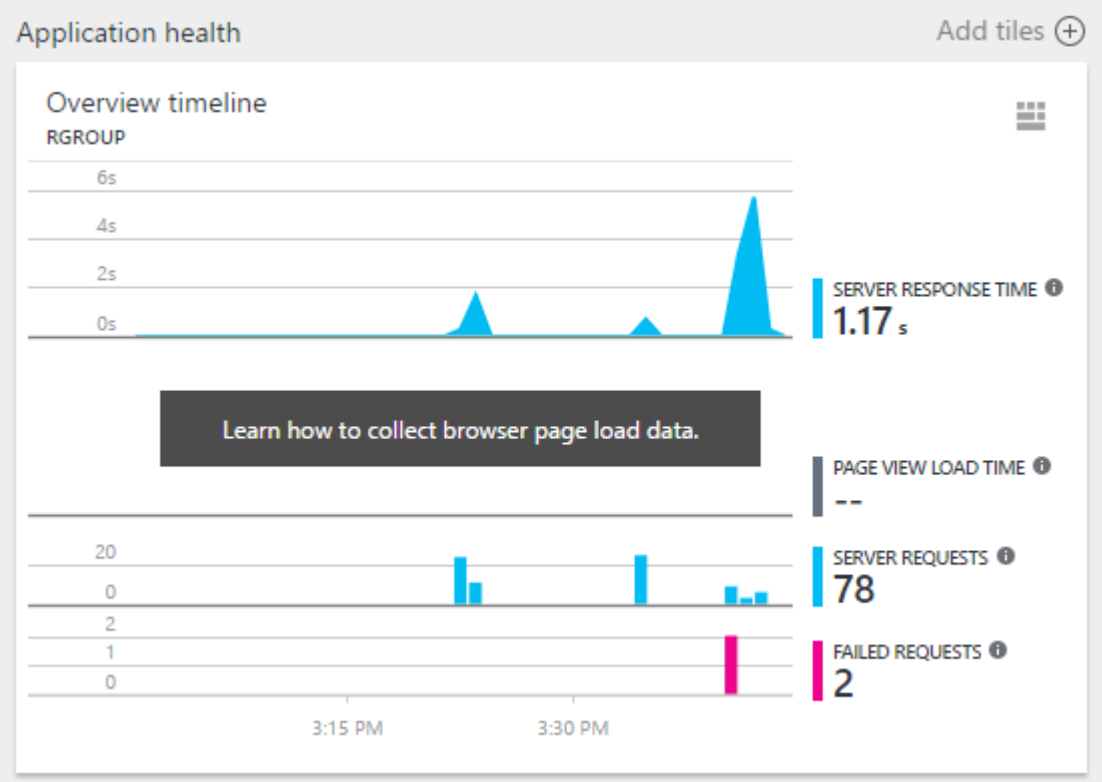
13. This verifies that events are being captured correctly, so now let's deploy to Azure. Right-click on **Rg.Web** and select **Publish**. When the **Publish Web** dialog appears, click the **Publish** button.
14. Once publication is complete, leave the page open in the browser and, in another tab, go to the Azure Portal and find your Web App. Click the **Tools** button on its toolbar



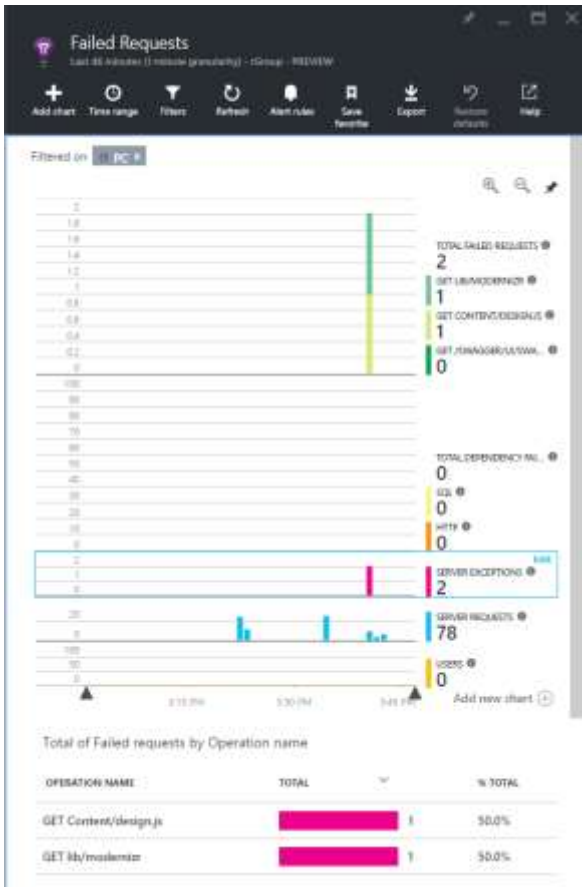
15. In the **Tools** panel, click **Performance monitoring**. In the **Performance monitoring** blade, click on **Click here to collect insights into the performance of your .NET applications**. Then click **Application Insights**.



16. The **Application Insights** pane that opens lists all available Application Insights instances. Select your **rGroup** instance. Click **OK** in this pane and again in the next pane. This configures the Azure Web App to supply Application Insights with additional information that is not available through the SDK alone. You need to enable Application Insights in both ways. Neither the SDK nor this setting on its own can provide full information.
17. Return to the browser showing your application and navigate around the site a little more to generate some more telemetry. Make sure that the last thing you do is return to the Home page (or refresh it if you're already there) so that you can follow the next steps in this walkthrough.
18. Back in the Azure Portal, open the **rGroup** Application Insights instance. You should see some overview information about your application's health:

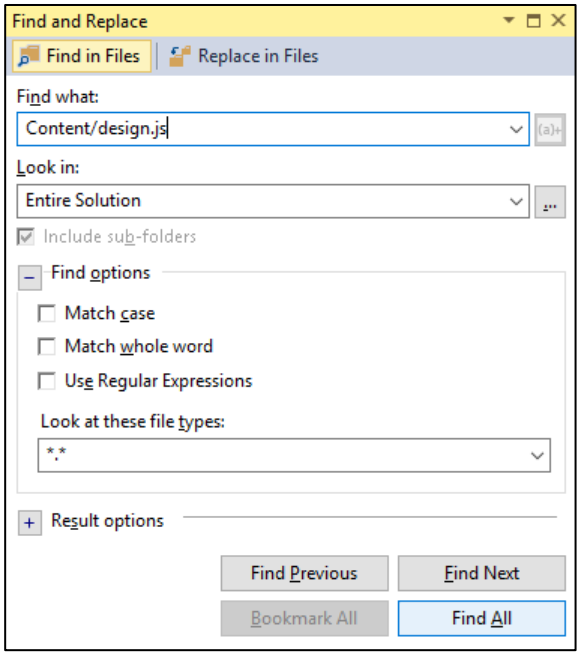


19. Notice that we seem to have some failed requests. Click on the **FAILED REQUESTS** section of this overview. This shows more detail about the failed requests:



The graphs show when various things happened. The horizontal position indicates the time. The failures of interest are all recent so they're over on the right. The top graph shows failed requests broken down by the operation that failed. We can see from that (as well as again in the **Total of Failed requests by Operation name** list further down) that the failures are GET requests attempting to get **Content/design.js** and **lib/modernizr**.

20. Let's see if we can fix that. Back in Visual Studio, select the **Edit | Find and Replace | Find in Files** (Ctrl+Shift+F) item. In **Find what** type **Content/design.js** and then click **Find All**.



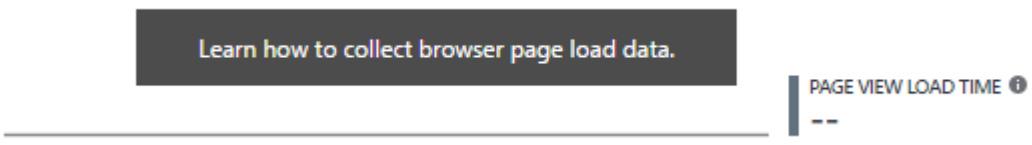
21. This indicates that the **_LayoutNoUser.cshtml** file in the **Views\Shared** folder has a **<script>** tag referencing this item on line 60. Open the file.
22. That's the wrong location for the script file—design.js lives in the **Scripts** folder, not **Content**. You need to adjust the URL, replacing **Content** with **Scripts**.
23. The second problem is also on this page. At line 11, you will find this:

    ```
    <script src="~/lib/modernizr/modernizr.js"></script>
    ```

    That's the wrong folder. Although there is a **/lib/modernizr/modernizr.js** file in the project, it does not get deployed to the server; the **lib** folder effectively contains source code from which the script files ultimately uploaded get generated. This should refer to the same scripts folder as the tag you just fixed further down in the file. It should look more like this:

    ```
    <script src="https://YOURCDNENDPOINT.azureedge.net/Scripts/modernizr.js"></script>
    ```

    You will have to replace YOURCDNENDPOINT with the endpoint you set up in an earlier walkthrough. If you didn't do the CDN walkthrough, just use **~/Scripts/modernizr.js**.
24. Before we publish the updated web site, there's another issue. If you look at the Application Insights pane in the Azure Portal, you will see in the **Application health** overview that in the **PAGE VIEW LOAD TIME** section that there's no data, just a message:

25. Click on that **Learn how to collect browser page load data** label. This opens a blade with the title **Client application monitoring and diagnosis**. It includes a code snippet you can add to web pages. This is necessary for Application Insights to be able to measure performance inside users' web browsers.

26. Copy that code. In Visual Studio, in the **Rg.Web** project's **Views\Shared** folder, open **_Layout.cshtml** and paste in the code snippet just after the closing **</head>** tag on line 14.
27. Publish the Web App again. Navigate around a bit more in the app.
28. Go back to the Azure portal. If you still have the **rGroup** Application Insights blade open, click the **Refresh** button.

    If you don't have it open, just open it again (no need to refresh in that case).
29. You should now find that the **PAGE VIEW LOAD TIME** section is showing data.

Let's review what you have done. You created a new Application Insights instance to collect information about your Web App, and you configured your Web App to send diagnostic data to that instance. Virtually instantly, this revealed a problem in the app—it showed that we had a couple of bad URLs leading to failed requests. You now have an overview of how the application is performing, with response time information both from the server's perspective and the client's.

Last Updated: August 10, 2016.