Source: C# Corner (www.c-sharpcorner.com) PRINT

Article

Using Application Insight SDK With Azure Websites

By **Abdul Rasheed Feroz Khan** on **Jun 24 2016**

Introduction

Here, we will add the power of application insights to your application with a customized telemetry, captured as per your business needs instead of the standard data.

- Web Client Telemetry.
- Custom telemetry.

Follow the below steps to add application insights to your application,

Step 1

Open the project in Visual Studio.

Step 2

Right click on your Website and click “Add Application Insights Telemetry”.

In the “Add Application Insights to Project” Window, select the option to add “New Application Insights Resource”.

After Visual Studio finishes adding the Nuget Packages for “Application insights," if we run the project, it will start logging the telemetry data for the Web Application.

Adding Web Client Telemetry

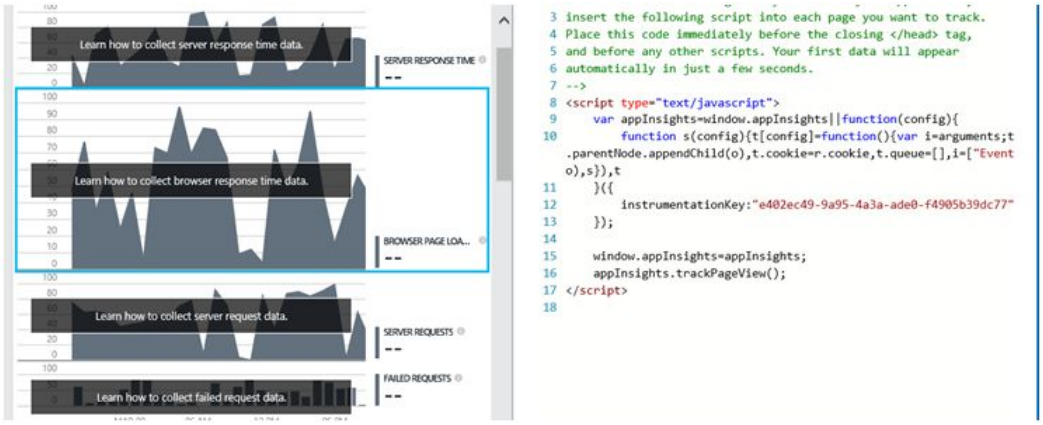
To add web client analytics, we will need to add some JavaScript code in the pages we want to track.

Step 1

Open the Default.aspx page, switch to “Source” view and add the following script after line 3.

```
<script type="text/javascript">
var appInsights = window.appInsights || function(config)
{
    function s(config)
    {
        t[config] = function()
        {
            var i = arguments;
            t.queue.push(function()
            {
                t[config].apply(t, i)
            })
        }
    }
    var t = {
            config: config
        },
        r = document,
        f = window,
        e = "script",
        o = r.createElement(e),
        i, u;
    for (o.src = config.url || "//az416426.vo.msecnd.net/scripts/a/ai.0.js", r.getElementsByTagName(e)
[0].parentNode.appendChild(o), t.cookie = r.cookie, t.queue = [], i = ["Event", "Exception", "Metric", "PageView", "Trace"]; i.length;) s("track" + i.pop
    return config.disableExceptionTracking || (i = "onerror", s("_" + i), u = f[i], f[i] = function(config, r, f, e, o)
    {
        var s = u && u(config, r, f, e, o);
        return s !== !0 && t["_" + i](config, r, f, e, o), s
    }), t
}(
{
    instrumentationKey: "KeyID for your Application Insight Resource"
});
window.appInsights = appInsights;
appInsights.trackPageView();
```

You can get this script from “Microsoft Azure Portal”->Application Insight resource page, shown below:



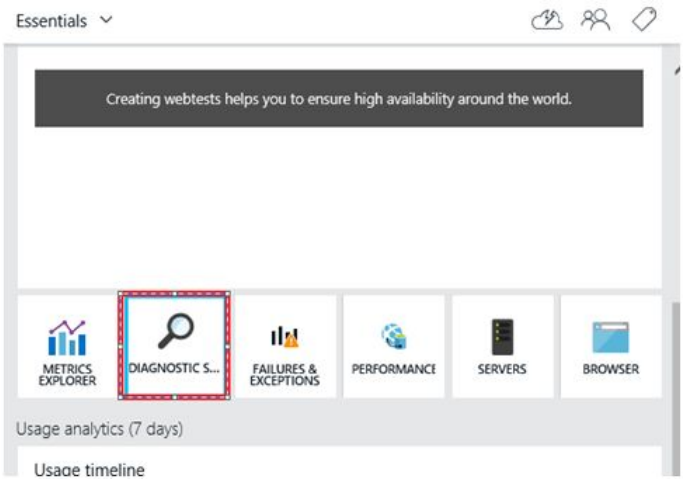
Add the script to all the pages in the project at the same place.

Now, publish the project to the same “Azure Web sites”.

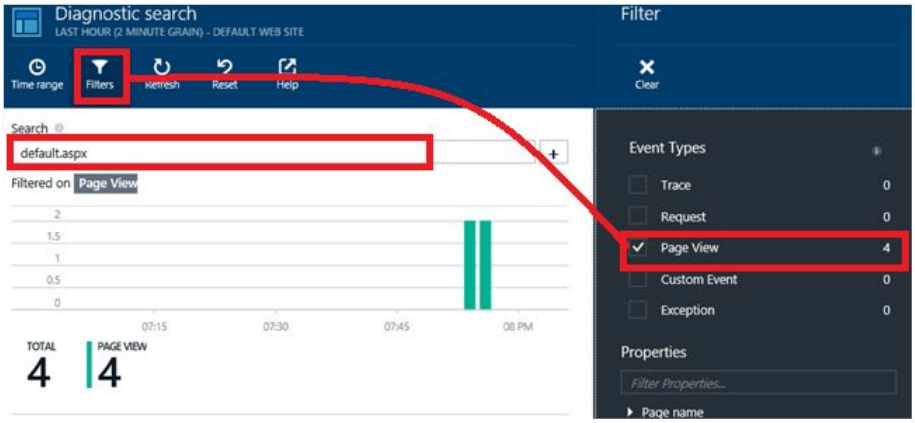
Browse the Website and hit all the pages. Keep the “Default.aspx” running for some time.

Now open the “Application Insights” resource from Visual Studio or from Azure Portal.

Click on “Diagnostic search” to open the Diagnostics blade:



In the search option, enter “Default.aspx”. Click “Filters” button, check “Page View” and uncheck all other options.



Now close the filter blade and come back to the Diagnostics blade.

It will list “PageView” Web client data for your Web Application. You can select the individual pages to see the details of the request

Adding Custom Telemetry

Open “CustomEvents.aspx.cs” file from the project.

Add reference to the Application Insights namespace in the code,

```
1. using Microsoft.ApplicationInsights;
```

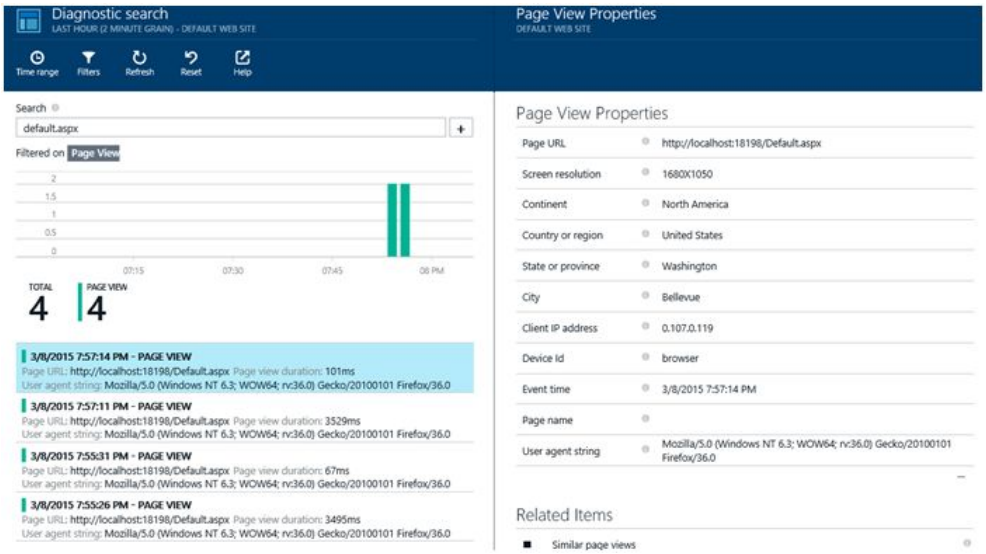
Define a telemetry instance of TelemetryClient in page load, shown below:

```
1. private TelemetryClient telemetry;
2. protected void Page_Load(object sender, EventArgs e)
3. {
4.    telemetry = new TelemetryClient();
5. }
```

Now to track the page views from the code along with the user and session data, add the following code in Btn_LogView, as shown below:

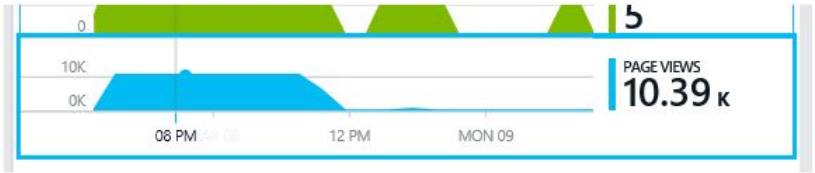
```
1. protected void Btn_LogView_Click(object sender, EventArgs e)
2. {
3.    telemetry.TrackPageView("MyPageTracker");
4. }
```

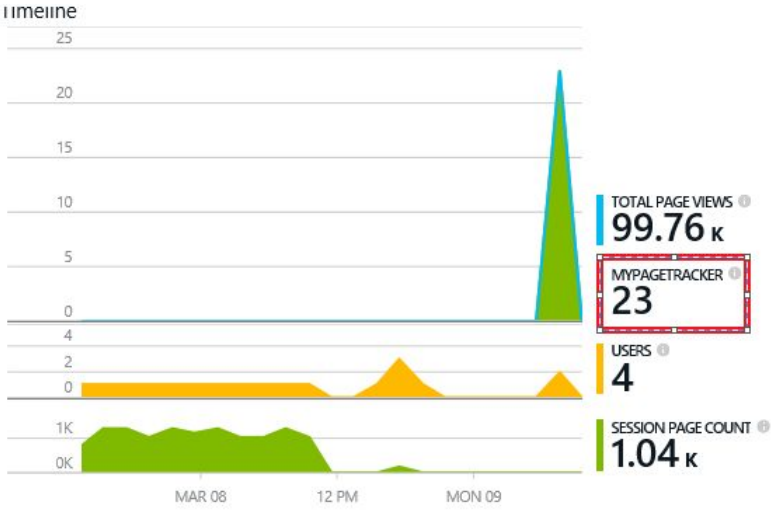
Run the project and click the button a couple of times.



Switch to Microsoft Azure Portal and check refresh the entries in the application insight usage lens.

Click “Page View” chart, observe that there is a new entry for “MyPageTracker” and its view count.





Activity by page name

PAGE NAME	PAGE VIEWS	USERS	SESSIONS
<no_name>	100 K	4	55
MyPageTracker	23	1	1

Now, switch back to Visual Studio and stop the project.

A similar approach can be used to track the events, which can be displayed on the portal as an aggregate count.

Let's say you want to categorize your uses per age and want the statistics. You can log an event with the age as property to the Application insight and get the chart details over time. To test this, copy the following code in the button click event of "*Btn_LogEvent*",

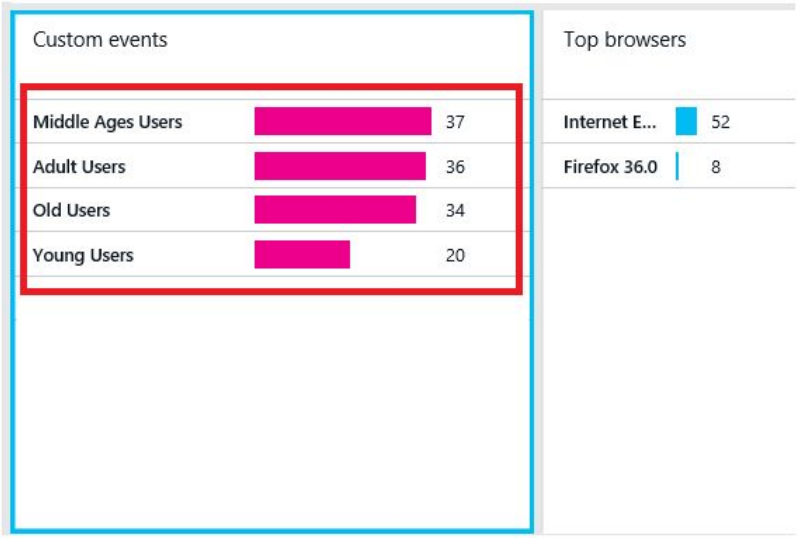
```
protected void btn_LogEvent_Click(object sender, EventArgs e)
{
    Random r = new Random(10);

    for (int i = 0; i <= 100; i++)
    {
        string EventName="";

        int num = r.Next(10, 60);
        if (num < 18)
            EventName = "Young Users";
        if(num>=18 && num<30)
            EventName = "Adult Users";
        if (num >= 30 && num <45 )
            EventName = "Middle Ages Users";
        if (num >= 45)
            EventName = "Old Users";

        var properties = new Dictionary <string,string>  {{"Age Category", EventName}};
        var metrics = new Dictionary <string, double>   {{"Age", num }};

        telemetry.TrackEvent(EventName, properties,metrics);

        Console.WriteLine(EventName + num.ToString());

    }

    lbl_message.Text = "Custom Event logged";

}
```

Now, run the page and click the button.

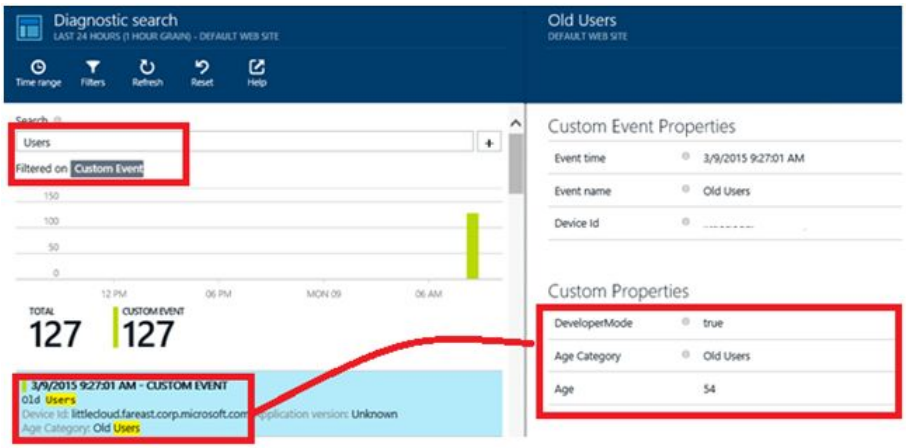
The code will generate some random numbers between 10-60 and will log the event accordingly.

Now, switch to Azure Portal and scroll down to the "CustomEvents" lens and observe the data logged.

You will see the individual events are logged for each age category and its count, as shown below:



You can also check the individual occurrence of the event along with the “Metrics” and “Properties” added through the code, as shown below:



Thank you for using C# Corner