**AppInsightsMVCFilters**

This project provides nifty action and exception filters for MVC and MVC Web API projects (either on cloud or on-premise) which will easily capture the telemetry of your Action/Api methods and integrate with Application Insights, without major code changes.

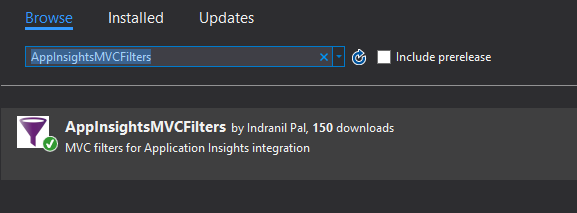
This also provides a Telemetry helper class for you to do custom logging or events, traces, requests etc. [Target .Net framework – 4.6.1 and above]

[Code location – [GitHub](https://github.com/ip28/AppInsightsMVCFilters)]

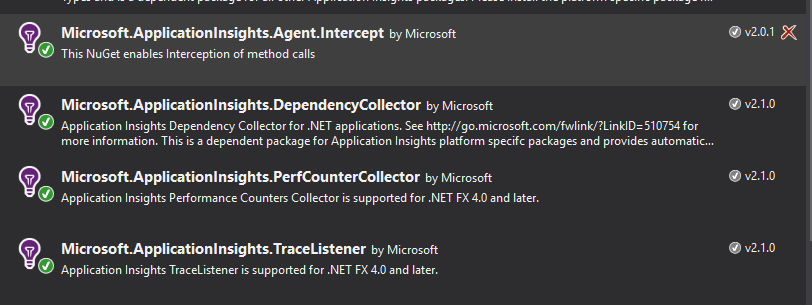
[Package location – [NuGet](https://www.nuget.org/packages/AppInsightsMVCFilters/)]

How to use the package: -

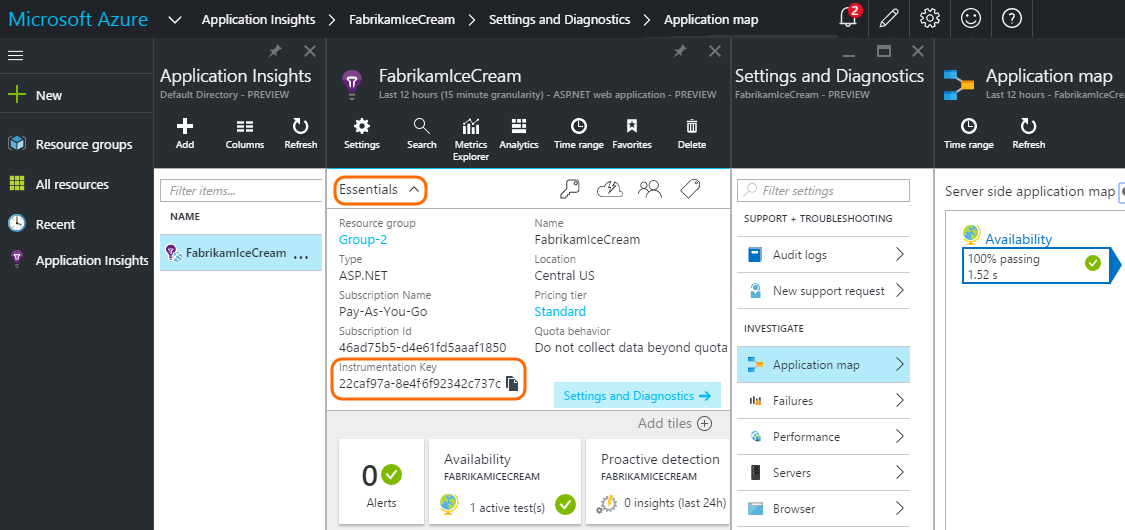
1. Add the nuget package to your MVC/WebApi project by doing one of the following
2. By searching “AppInsightsMVCFilters” in NuGet manager

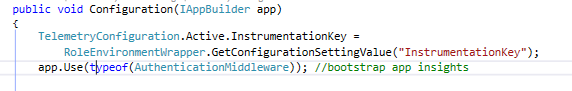


1. By nuget manager console command **PM> Install-Package AppInsightsMVCFilters**
2. The package will automatically install Microsoft.ApplicationInsights nuget package to your project, additionally you can install below packages for advanced logging and tracing capture.

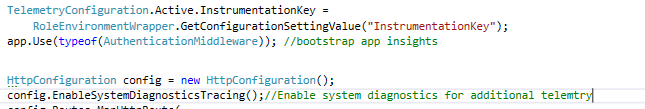


Note: If you already have your service/app is hooked to application insights then move to step 5 for using the AppInsightsMVCFilters

1. Create Application Insights resource from azure portal and copy the instrumentationKey from the portal.
2. Find the Application Insights documentation [here](https://azure.microsoft.com/en-in/documentation/articles/app-insights-overview/)
3. Get the instrumentation key from the portal as shown below
4. Configure App Insights in your application
5. If your application is on-premise add a key in appsettings of your app/web.config file which will contain the key, you copied in the previous step
6. If your application is a cloud service/web app add it to the cloud config of your respective role (cloud host)
7. In the startup file of your WebRole (WebApi, Website etc.) bootstrap AppInsights like below: -



1. Additionally, it is recommended to enable system diagnostics so that you can get more information like stack traces in case of unhandled



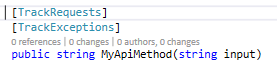
Now your application is hooked up to application insights and events will be logged in the portal.

1. Using AppInsightsMVCFilters

With AppInsightsMVCFilters you will get below utilities: -

1. MVC action filter which can track/log your Action/Api methods pre and post execution
2. MVC exception filter which can track/log your unhandled exception

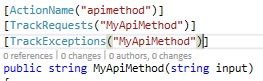
You can decorate your Action/Api method like below (this will log the telemetry by the Action name “MyApiMethod” since there is no specific ActionName mentiond)



[Note: If you want to attach TrackExceptions for Global level exceptions you can add it as a global filter in HttpConfiguration object in your startup class]

1. Log the telemetry with custom method name

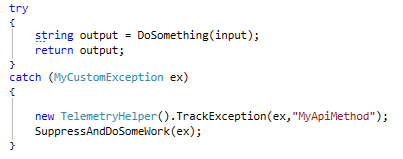
Often we mention ActionName on top of our Api/Action methods but it might make more sense if we log the telemetry by the actual method name, as it makes easier for debugging. Log telemetry by custom method name like below:-



This will log the telemetry by the name of “MyApiMethod”, thus even if you have same action names for multiple method the logs will make sense.

1. Log handled exceptions

All unhandled exceptions are automatically handled by “TrackException” filter, but if you want some exception to be suppressed and to be logged (with complete stack trace and other details related to the exception) use “TelemetryHelper” class like below



Note: In case of HttpResponseExceptions the exception filter will not be invoked, please use the above explicit way to log exceptions for HttpResponseExceptions.