Creating Azure Functions in F#

Monday, Jan 13, 2020 3 minute read Tags: fsharp serverless azure-functions

Last year I wrote a blog post on getting started with Azure Functions using F#. Sadly, it was a bit cumbersome as you needed to create a C# project and then convert it, and that was mainly so you got the right properties in the config file.

Thankfully though this has been improved as there are now F# templates for Azure Functions! Let's have a quick look at how to get started with them.

Getting The Templates

Before getting started you'll want to make sure you have the latest templates installed, so you have the latest NuGet packages referenced. To do that install the templates packages from NuGet, Microsoft.Azure.WebJobs.ProjectTemplates and Microsoft.Azure.WebJobs.ItemTemplates:

```
$> dotnet new --install Microsoft.Azure.WebJobs.ItemTemplates
$> dotnet new --install Microsoft.Azure.WebJobs.ProjectTemplates
```

Installing these templates will add a bunch of new options to `dotnet new` for both C# and F#:

```
Templates                                         Short Name              Language          Tags
----------------------------------------------------------------------------------------------------------------------------------------
DurableFunctionsOrchestration                     durable                 [C#]              Azure Function/Durable Functions Orchestration
SendGrid                                          sendgrid                [C#]              Azure Function/Ouput/SendGrid
BlobTrigger                                       blob                    [C#], F#          Azure Function/Trigger/Blob
CosmosDBTrigger                                   cosmos                  [C#], F#          Azure Function/Trigger/Cosmos DB
EventGridTrigger                                  eventgrid               [C#]              Azure Function/Trigger/EventGrid
EventHubTrigger                                   eventhub                [C#], F#          Azure Function/Trigger/EventHub
HttpTrigger                                       http                    [C#], F#          Azure Function/Trigger/Http
IotHubTrigger                                     iothub                  [C#]              Azure Function/Trigger/IotHub
ServiceBusQueueTrigger                            squeue                  [C#]              Azure Function/Trigger/Service Bus/Queue
ServiceBusTopicTrigger                            stopic                  [C#]              Azure Function/Trigger/Service Bus/Topic
QueueTrigger                                      queue                   [C#]              Azure Function/Trigger/Storage Queue
TimerTrigger                                      timer                   [C#], F#          Azure Function/Trigger/Timer
Azure Functions                                   func                    [C#], F#          Azure Functions/Solution
```

Not all the triggers have an F# template provided, but there's a number of good ones to get started with.

Creating Our Solution

With the templates installed we can create them from the CLI just like any other .NET project. Let's start by creating a Functions solution:

```
$> dotnet new func --language F# --name FunctionsInFSharp
```

You'll receive a success message and if we look on disk the files will be like so:

```
$> ls
FunctionsInFSharp.fsproj  host.json  local.settings.json
```

Woo, we have our `fsproj` and ready to go with the right NuGet packages referenced.

Creating a Function

Finally, we want to create our Function itself, and again that's something we can do from the .NET CLI:

```
$> dotnet new http --language F# --name HttpTrigger
```

This will create us a new file called `HttpTrigger.fs` alongside the project file using the `http` template (for a HttpTrigger function). Since F# needs the files to include in compilation to be in the `fsproj` file, make sure you pop open the `fsproj` file and include it within an `<ItemGroup>`:

```
<Compile Include="HttpTrigger.fs" />
```

Now if you open this in VS Code it'll be detected as a Azure Functions project and prompt you to setup the VS Code Extension and artifacts, then it's a matter of hitting **F5** to launch!

Conclusion

There we have it folks, a much simpler way to create an F# Azure Function using the provided templates. No more remembering what NuGet packages to reference, renaming of `csproj` files or working out what additional properties are needed in the project file to make one from scratch.

Happy F#'ing!