

Project Documentation: InfoTrack Web Scraper

ASP.NET MVC Web Application: .Net Framework: 4.7.2 (Razor, Bootstrap, ASP.NET MVC 5.2)

<https://infotrackwebscraper.azurewebsites.net/>

The screenshot shows the web application interface. At the top is a navigation bar with links: InfoTrack, Home, Web Scraper, and Contact. Below this is a section titled "Web Scraper Statistics". It contains two input fields: "Keywords" with the placeholder text "Enter search keywords" and "Domain Url" with the placeholder text "https://www.infotrack.com". A blue "Search" button is positioned below these fields. Underneath the search button, it says "Recent search history: 2 record(s)". A sample record is displayed in a green box, showing "Keywords: test (Sample record)" and "Domain Url: https://www.facebook.com". Below this, a table shows a "Score: 5" and a "Search Result: Sample search results".

Azure Function App - .Net Core 3.1

<https://pagerankfunctionapp.azurewebsites.net/api/pagerank?code=23AJANrswmjb446QbLtyTpmSNbaRfrRNIMUzWsGzyf30qjw037AiuQ==>

The screenshot displays the REST client interface for the Azure Function App. The top bar shows a "POST" request to the URL "https://pagerankfunctionapp.azurewebsites.net/api/pagerank?code=23AJANrswmjb446QbLtyTpmSNbaRfrRNIMUzWsG...". Below this, the "Body" tab is selected, showing a JSON request body:

```
{  "Keywords": "efiling integration",  "DomainUrl": "https://www.infotrack.com"}
```

. The bottom section shows the "Body" tab of the response, displaying a JSON object:

```
{  "keywords": "efiling integration",  "domainUrl": "https://www.infotrack.com",  "score": 30,  "searchResult": "No. of times your domain occurred in the search results: 3; Positions: 2, 3, 4",  "recordTimestamp": "2020-12-12T18:26:00.2291015Z"}
```

Key Technical features

Microservice Architecture

1. Designed a serverless microservice architecture using Azure functions and deployed it on Azure Cloud Platform. The MVC app is deployed on a free app service plan, the initial load will be slow due to cold start, but it can be configured to avoid this if moved to paid version. The function app is deployed on a consumption plan and it can scale up based on the web traffic. The core logic to compute the search results is written in the function app so that the back-end logic can be substituted as we implement a more complex logic using Google or third party APIs without the need for the front end to change.

Scope for improvements:

- a. Create deployment slots for different development environments (dev, staging and production).
- b. We can implement CI/CD pipeline for continuous delivery and testing.
- c. Avoid cold starts by moving to a paid production environment.
- d. Setup custom domains (DNS zones can be setup on 1 & 1 or Cloudflare) and secure the web channel using SSL certificates.
- e. Secure the MVC app by using third party platforms like Cloudflare, this will avoid features like Bot Management and also help us with caching.
- f. The function app provides a basic http end point security using function keys but can be more robust by adding the function behind an Azure API Management service + JWT token based restriction.

The screenshot shows the Azure App Service portal for the application 'INFOTRACKWEBSCRAPER'. The interface includes a left-hand navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Events (preview), Deployment, Quickstart, and Deployment slots. The main content area displays the 'Essentials' section, which provides key information about the app service: Resource group (AMTEST), Status (Running), Location (East US), Subscription (IMC-Pharos), and Subscription ID. It also lists the URL, App Service Plan (INFOTRACKWEBSCRAPERHOST (F1: Free)), FTP/deployment username, FTP hostname, and FTPS hostname. At the bottom, there are three tiles: 'Diagnose and solve problems', 'Application Insights', and 'App Service Advisor'.

Home > INFOTRACKWEBSCRAPER ✕

App Service

Search (Ctrl+J) << Browse Stop Swap Restart Delete Refresh Get publish profile Reset publish profile Send us your feedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Security

Events (preview)

Deployment

Quickstart

Deployment slots

Essentials

Resource group (change) : AMTEST

Status : Running

Location : East US

Subscription (change) : IMC-Pharos

Subscription ID : f058d431-629d-4a1c-8e97-ce17202eebc5

Tags (change) : Click here to add tags

URL : <https://infotrackwebscraper.azurewebsites.net>

App Service Plan : INFOTRACKWEBSCRAPERHOST (F1: Free)

FTP/deployment username : No FTP/deployment user set

FTP hostname : <ftp://waws-prod-blu-091.ft.azurewebsites.windows.net/site/www...>

FTPS hostname : <ftps://waws-prod-blu-091.ft.azurewebsites.windows.net/site/www...>

Diagnose and solve problems

Our self-service diagnostic and troubleshooting experience helps you identify and resolve issues with your web app.

Application Insights

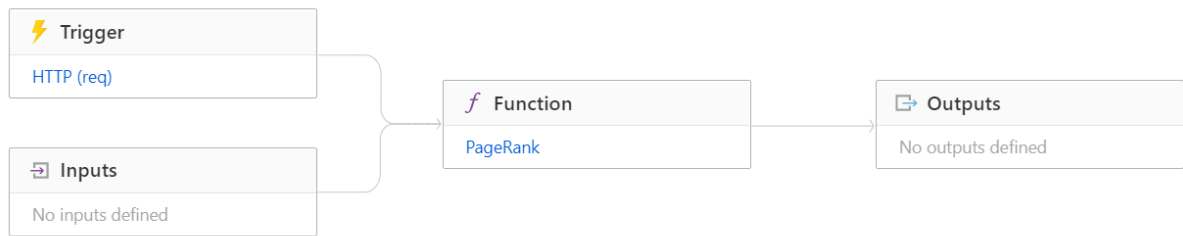
Application Insights helps you detect and diagnose quality issues in your apps, and helps you understand what your users actually do with it.

App Service Advisor

App Service Advisor provides insights for improving app experience on the App Service platform. Recommendations are sorted by freshness, priority and impact to your app.

Integration

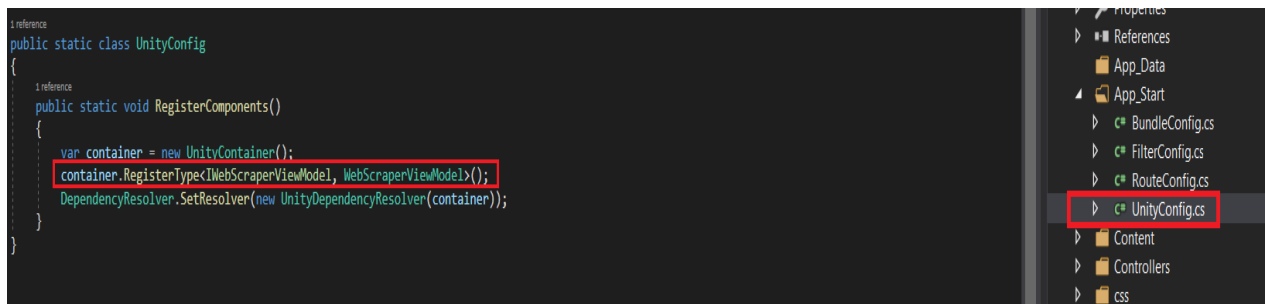
Edit the trigger and choose from a selection of inputs and outputs for your function, including Azure Blob Storage, Cosmos DB and others.



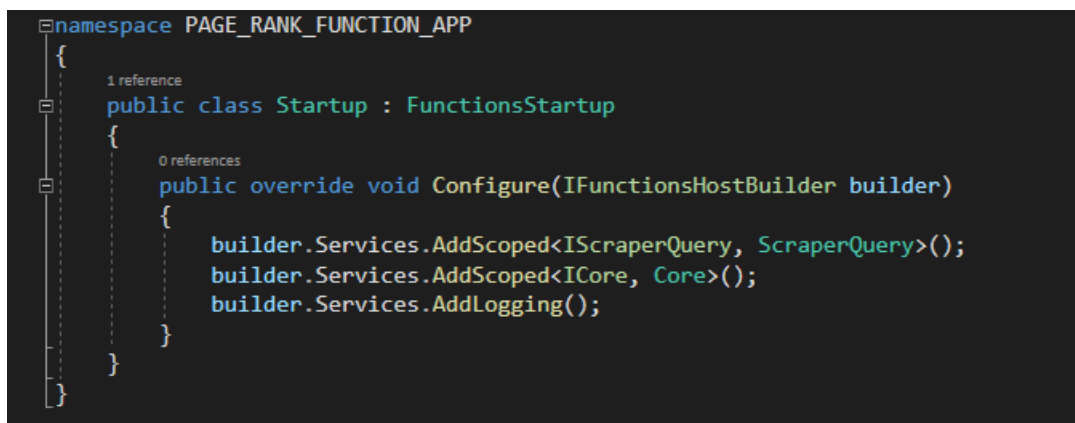
Dependency Injection

Implemented Unity container which is an open source IoC container for .NET applications supported by Microsoft.

MVC App:



Function App: In-built IOC provided by .Net Core



Client-Side Validation

Implemented model validation Using Data Annotations in ASP.NET MVC

```
namespace INFO_TRACK_WEB_SCRAPER.ViewModels
{
    /// <summary>
    /// The view model is an abstraction of the view 'Statistics' exposing public properties and commands.
    /// </summary>
    3 references
    public class WebScraperViewModel : IWebScraperViewModel
    {
        [Required(ErrorMessage = "Keywords are required")]
        [StringLength(maximumLength: 4000, MinimumLength = 3, ErrorMessage = "Min. 3 char(s) are required")]
        3 references
        public string Keywords { get; set; }

        [Required(ErrorMessage = "Domain url is required")]
        [Url(ErrorMessage = "Domain url is invalid")]
        [StringLength(maximumLength: 4000, MinimumLength = 5, ErrorMessage = "Min. (5) - Max. (4000) length")]
        3 references
        public string DomainUrl { get; set; }
    }
}
```

Web Scraper Statistics

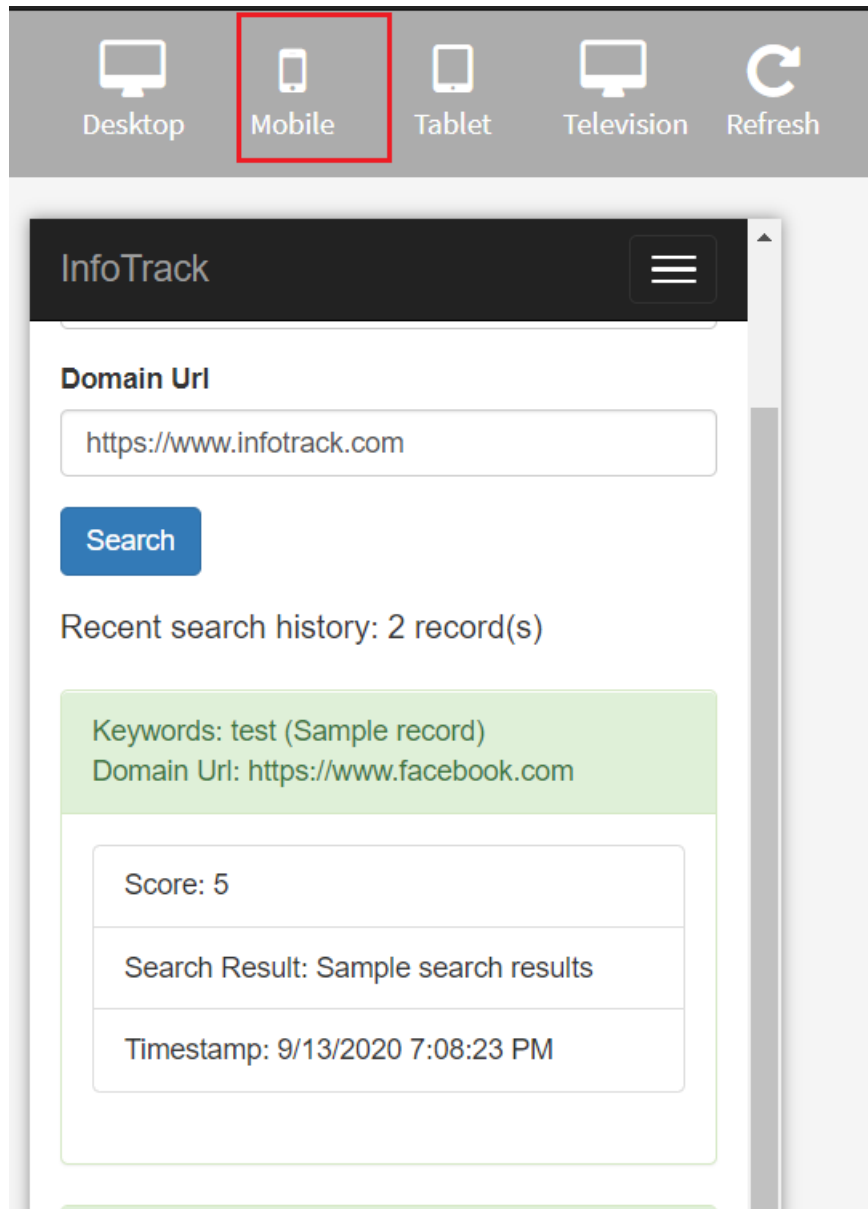
Keywords

Min. 3 char(s) are required

Domain Url

Domain url is invalid

Responsive web design using Bootstrap



Error handling

MVC App: Implemented a generic base controller for handling errors and redirecting the user to a custom error page.

```
1 reference
public class InfoTrackBaseController : Controller
{
    // GET: InfoTrackBase
    1 reference
    public virtual ActionResult Index()
    {
        return View();
    }

    /// <summary>
    /// Called when an unhandled exception occurs in the action.
    /// </summary>
    /// <param name="objectExceptionContext">Information about the current request and action.</param>
    0 references
    protected override void OnException(ExceptionContext objectExceptionContext)
    {
        //Record exception in a persistent system (MongoDB).

        //Redirect to custom error page.
        objectExceptionContext.ExceptionHandled = true;
        objectExceptionContext.Result = new ViewResult
        {
            ViewName = "~/Views/Shared/Error.cshtml"
        };
    }
}
```

We can handle errors globally

```
public class MvcApplication : System.Web.HttpApplication
{
    0 references
    protected void Application_Start()
    {
        UnityConfig.RegisterComponents();
        AreaRegistration.RegisterAllAreas();
        FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);
    }

    /// <summary>
    /// Log exceptions globally.
    /// </summary>
    0 references
    protected void Application_Error(object sender, EventArgs e)
    {
        //Record exception in a persistent system (MongoDB).
        Exception objectException = Server.GetLastError();
    }
}
```

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet",
    "SearchEngine": "\"https://www.google.com.au/search?num=100&q={0}\"",
    "RegexPattern": "\"http(s)?://([\\w+?\\.\\w+])+(.[a-zA-Z-0-9~\\/!@#\\$%&'()*_+-=:\\/\\\\|'\" ,]*)?"
```

```

string stringSearchEngine = Environment.GetEnvironmentVariable("SearchEngine");
string stringKeywords = string.Format(stringSearchEngine, HttpUtility.UrlEncode(objectOfScraperQuery.Keywords));
Uri objectOfUri = new Uri(objectOfScraperQuery.DomainUrl);

int intOccurrenceCount = 0;
List<int> listOfMatchedPositions = new List<int>();

HttpRequest objectOfHttpRequest = (HttpRequest)WebRequest.Create(stringKeywords);
using (HttpWebResponse objectOfHttpWebResponse = (HttpWebResponse)objectOfHttpRequest.GetResponse())
{
    using (StreamReader objectOfStreamReader = new StreamReader(objectOfHttpWebResponse.GetResponseStream(), Encoding.ASCII))
    {
        string stringHtmlResponse = objectOfStreamReader.ReadToEnd();
        string stringRegex = Environment.GetEnvironmentVariable("RegexPattern");

        MatchCollection objectOfMatchCollection = Regex.Matches(stringHtmlResponse, stringRegex);
        for (int intLoopCounter = 0; intLoopCounter < objectOfMatchCollection.Count; intLoopCounter++)
        {
            try
            {
                string match = objectOfMatchCollection[intLoopCounter].Groups[0].Value;
                if (match.Contains(objectOfUri.Host))
                {
                    intOccurrenceCount++;
                    listOfMatchedPositions.Add(intLoopCounter + 1);
                }
            }
            catch (Exception)
            {
            }
        }
    }
}

objectOfScraperQuery.SearchResult = GetFormattedSearchResults(intOccurrenceCount, listOfMatchedPositions);
objectOfScraperQuery.Score = GetScoreCount(listOfMatchedPositions);

```

Property Validations Using Getter and Setters

```

private List<ScraperQuery> _listOfScraperQueries;
4 references
public List<ScraperQuery> ListOfScraperQueries
{
    get
    {
        if (this._listOfScraperQueries is null || _listOfScraperQueries.Count == 0)
        {
            this._listOfScraperQueries = GetDefaultScraperQueries();
        }
        return this._listOfScraperQueries;
    }

    set
    {
        if (value != null)
        {
            foreach (ScraperQuery objectCurrentScraperQuery in value)
            {
                if (!string.IsNullOrEmpty(objectCurrentScraperQuery.Keywords) && !string.IsNullOrEmpty(objectCurrentScraperQuery.DomainUrl))
                {
                    this._listOfScraperQueries.Add(objectCurrentScraperQuery);
                }
            }
        }

        if (this._listOfScraperQueries is null || this._listOfScraperQueries.Count == 0)
        {
            this._listOfScraperQueries = GetDefaultScraperQueries();
        }
    }
}

```



```
private int GetScoreCount(List<int> listOfMatchedPositions)
{
    int intScoreCount = 0;
    try
    {
        foreach (int intPosition in listOfMatchedPositions)
        {
            switch (intPosition)
            {
                case int n when (n <= 10):
                    intScoreCount += 10;
                    break;

                case int n when (n > 10 && n <= 20):
                    intScoreCount += 8;
                    break;

                case int n when (n > 20 && n <= 30):
                    intScoreCount += 6;
                    break;

                case int n when (n > 30 && n <= 40):
                    intScoreCount += 4;
                    break;

                case int n when (n > 40 && n <= 50):
                    intScoreCount += 2;
                    break;

                case int n when (n > 50):
                    intScoreCount += 1;
                    break;
            }
        }
    }
    catch (Exception)
    {
    }
}
```

Sample Data: Maintains recent search history

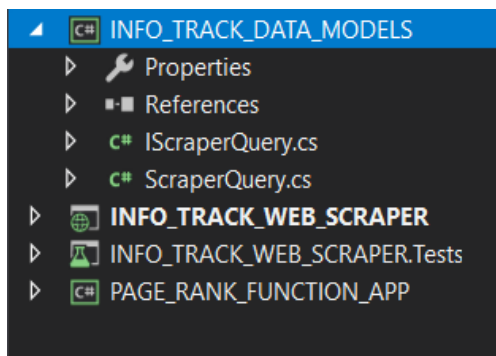
Sorted by the most recent search in descending order.

Recent search history: 2 record(s)

Keywords: test (Sample record) Domain Url: https://www.facebook.com
Score: 5
Search Result: Sample search results
Timestamp: 9/13/2020 7:58:29 PM

Keywords: efilling integration (Sample record) Domain Url: https://www.infotrack.com
Score: 10
Search Result: Sample search results
Timestamp: 6/15/2020 7:58:29 PM

Separate reusable Data Models class library to share objects between the MVC and Function app.



Bundling JavaScript and CSS files to prevent multiple get requests to the server.

```
1 reference
public class BundleConfig
{
    // For more information on bundling, visit https://go.microsoft.com/fwlink/?LinkId=301862
    1 reference
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
            "~/Scripts/jquery-{version}.js"));

        bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
            "~/Scripts/jquery.validate*"));

        // Use the development version of Modernizr to develop with and learn from. Then, when you're
        // ready for production, use the build tool at https://modernizr.com to pick only the tests you need.
        bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
            "~/Scripts/modernizr-*"));

        bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
            "~/Scripts/bootstrap.js"));

        bundles.Add(new StyleBundle("~/Content/css").Include(
            "~/Content/bootstrap.css",
            "~/Content/Site.css"));
    }
}
```

Verified Function app on the Azure cloud platform as well as Postman

The screenshot displays the Azure Functions portal for a function app named "PageRank". The left sidebar shows the "Code + Test" view selected. The main area shows the "function.json" file with the following configuration:

```
{
  "generatedBy": "Microsoft.NET.Sdk.Functions-3.0.11",
  "configurationSource": "attributes",
  "bindings": [
    {
      "type": "httpTrigger",
      "direction": "in",
      "methods": [
        "get",
        "post"
      ],
      "authLevel": "function"
    }
  ]
}
```

The right sidebar shows the "Output" tab with the following HTTP response content:

```
{
  "keywords": "efiling integration",
  "domainUrl": "https://www.infotrack.com",
  "score": 30,
  "searchResult": "No. of times your domain occurred in the search results: 3; Positions: 2, 3, 4",
  "recordTimestamp": "2020-12-12T20:06:03.0759533Z"
}
```

At the bottom, the "Log Level" dropdown is set to "Info", and the "Run" button is visible.

Easy to switch search engine provider and pattern matching by making it config based

```
local.settings.json
```

Schema: https://json.schemastore.org/local.settings

```
{  
  "IsEncrypted": false,  
  "Values": {  
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",  
    "FUNCTIONS_WORKER_RUNTIME": "dotnet",  
    "SearchEngine": "https://www.google.com.au/search?num=100&q={0}",  
    "RegexPattern": "http(s)?://([\\w+?\\.\\w+])+(<[a-zA-Z0-9\\-\\.\\/!@#\\$%&'()*~`=^_{}|;:\",'`]+)"  
  }  
}
```

PAGERANKFUNCTIONAPP

Function App

Configuration

[Refresh](#)
[Save](#)
[Discard](#)

Events (preview)

Functions

App keys

App files

Proxies

Deployment

Deployment slots

Deployment Center

Deployment Center (Preview)

Settings

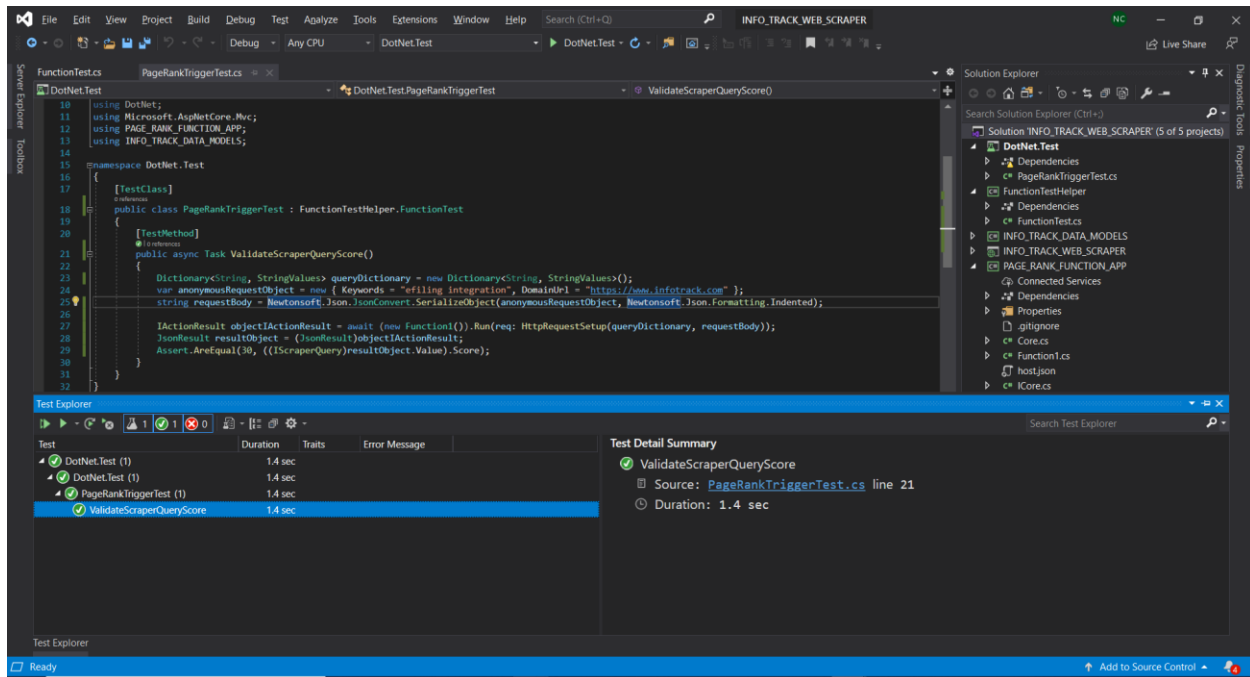
+ New application setting

Hide values

Advanced edit

Name	Value	Source	Deployment slot setting
AzureWebJobsDashboard	DefaultEndpointsProtocol=https;Account	App Config	
AzureWebJobsStorage	DefaultEndpointsProtocol=https;Account	App Config	
FUNCTIONS_EXTENSION_VERSION	~3	App Config	
FUNCTIONS_WORKER_RUNTIME	dotnet	App Config	
RegExPattern	http(s)?://([\\w+?\\. \\w+])+([a-zA-Z0-9\\	App Config	
SearchEngine	https://www.google.com.au/search?num	App Config	
WEBSITE_CONTENTAZUREFILECONNECTIONST	DefaultEndpointsProtocol=https;Account	App Config	
WEBSITE_CONTENTSHARE	pagerankfunctionapp	App Config	

Unit Tests



Mocked http request to debug the Azure function

