Project Documentation: InfoTrack Web Scrapper

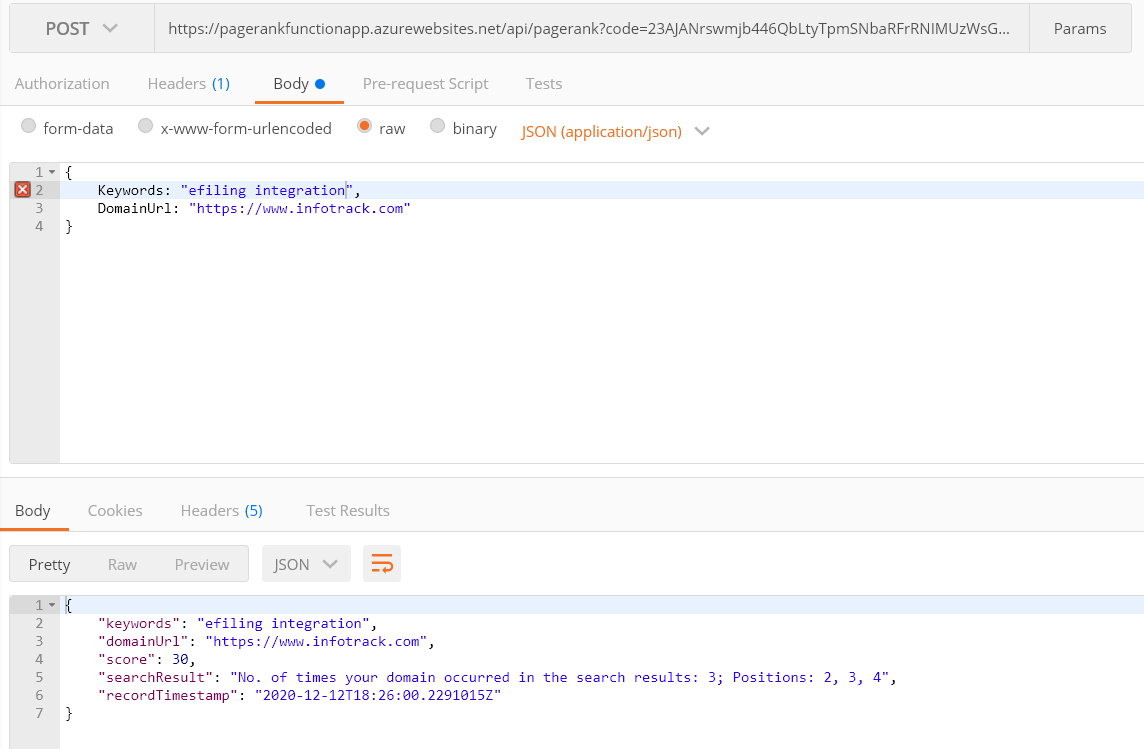
ASP.NET MVC Web Application: .Net Framework: 4.7.2 (Razor, Bootstrap, ASP.NET MVC 5.2)

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



Azure Function App - .Net Core 3.1

<https://pagerankfunctionapp.azurewebsites.net/api/pagerank?code=23AJANrswmjb446QbLtyTpmSNbaRFrRNIMUzWsGzyf30qjwO37AiuQ==>



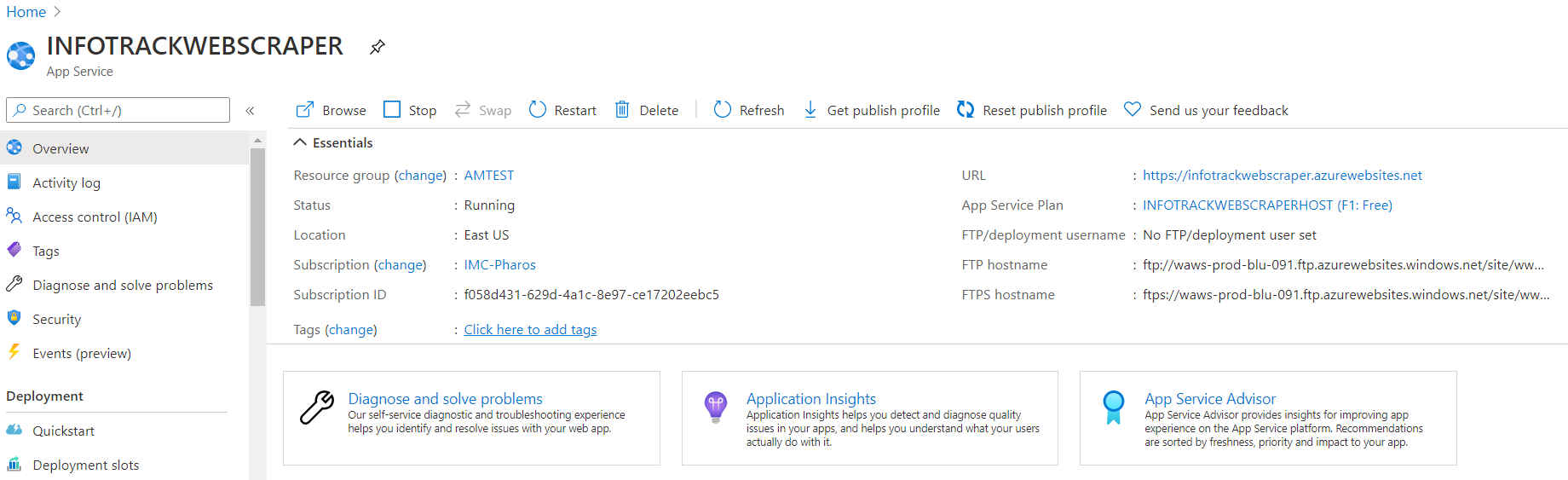
Key Technical features

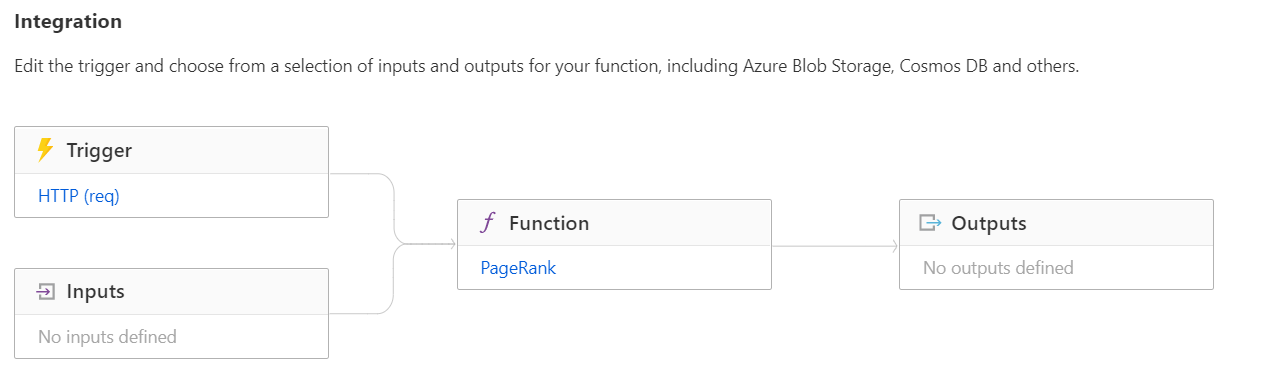
Microservice Architecture

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:

1. Create deployment slots for different development environments (dev, staging and production).
2. We can implement CI/CD pipeline for continuous delivery and testing.
3. Avoid cold starts by moving to a paid production environment.
4. Setup custom domains (DNS zones can be setup on 1 & 1 or Cloudflare) and secure the web channel using SSL certificates.
5. Secure the MVC app by using third party platforms like Cloudflare, this will avoid features like Bot Management and also help us with caching.
6. 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.





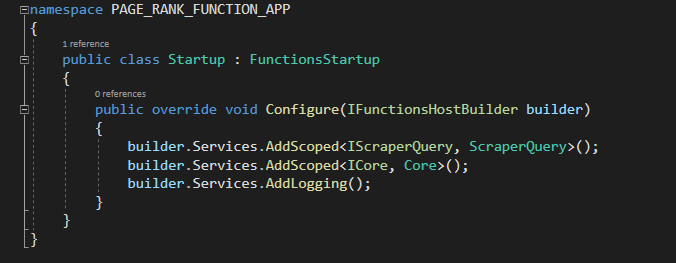
Dependency Injection

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

MVC App:

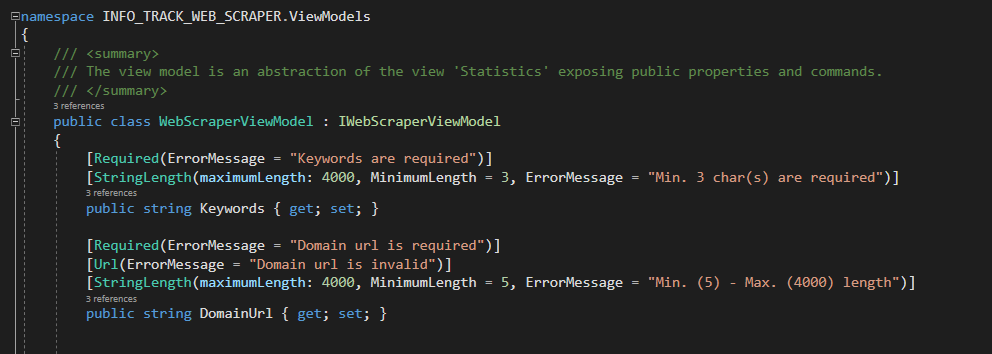


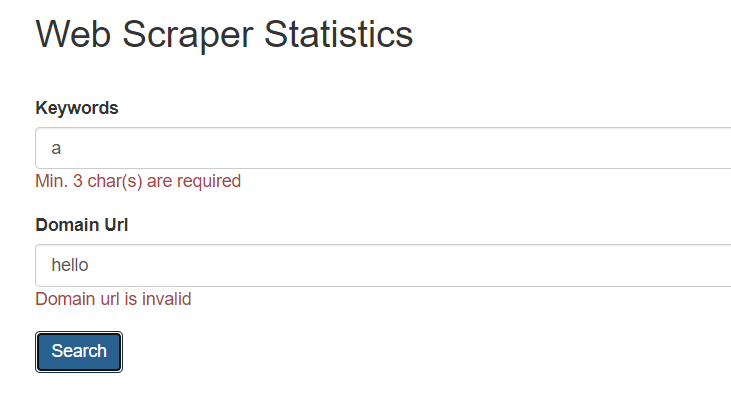
Function App: In-built IOC provided by .Net Core



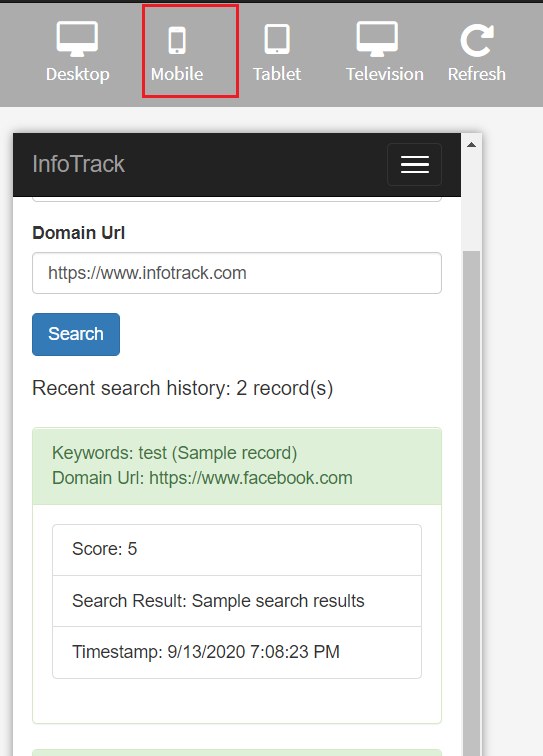
Client-Side Validation

Implemented model validation Using Data Annotations in ASP.NET MVC



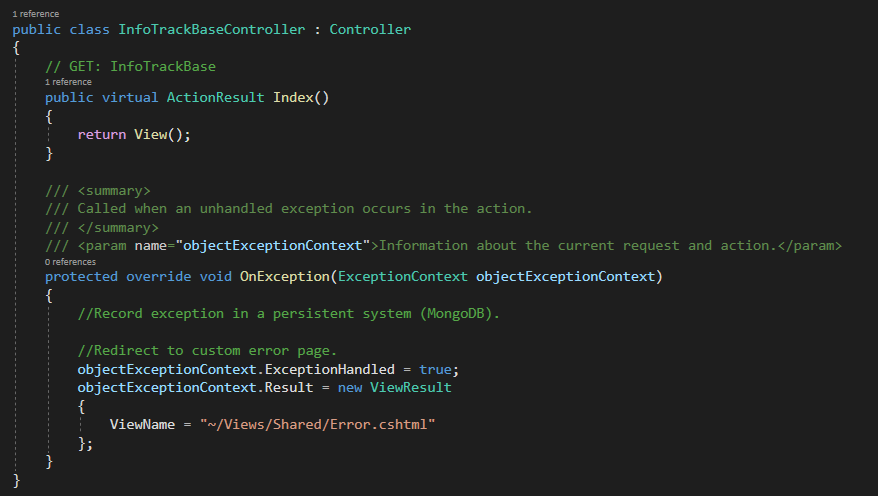


Responsive web design using Bootstrap

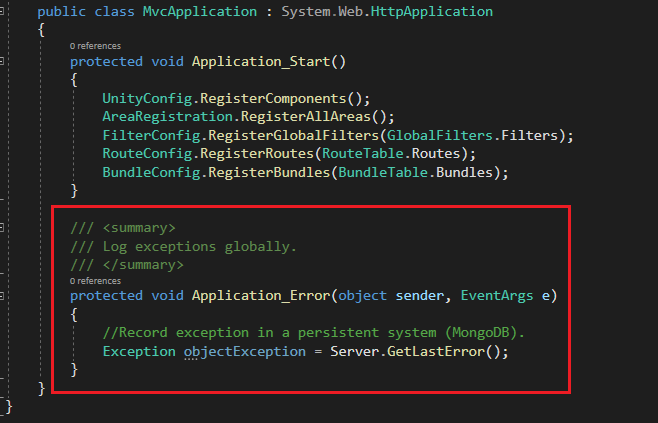


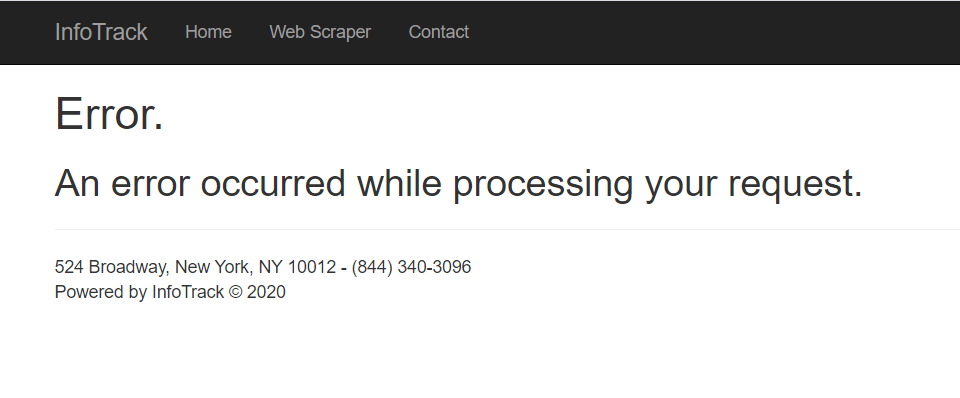
Error handling

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



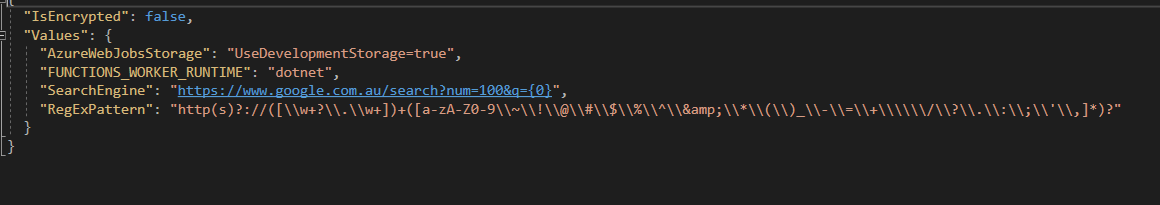
We can handle errors globally





Core Logic

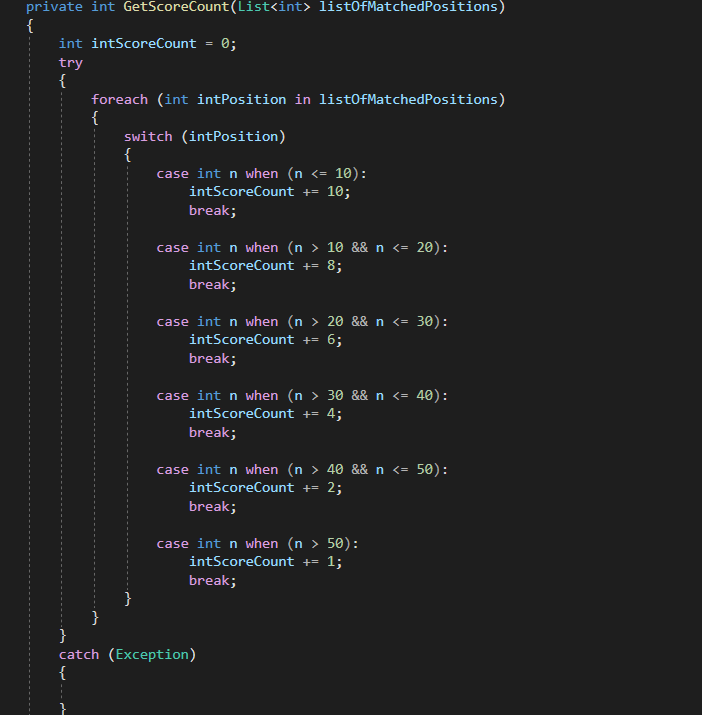
The function app uses [https://www.google.com.au/search?num=100&q={0}](https://www.google.com.au/search?num=100&q=%7b0%7d) as the primary search engine to lookup the keywords and the domain given domain url for matches based on the below regular expression pattern. I have also implemented a simple scoring model to award extra points if the given domain appears at the top of the list.





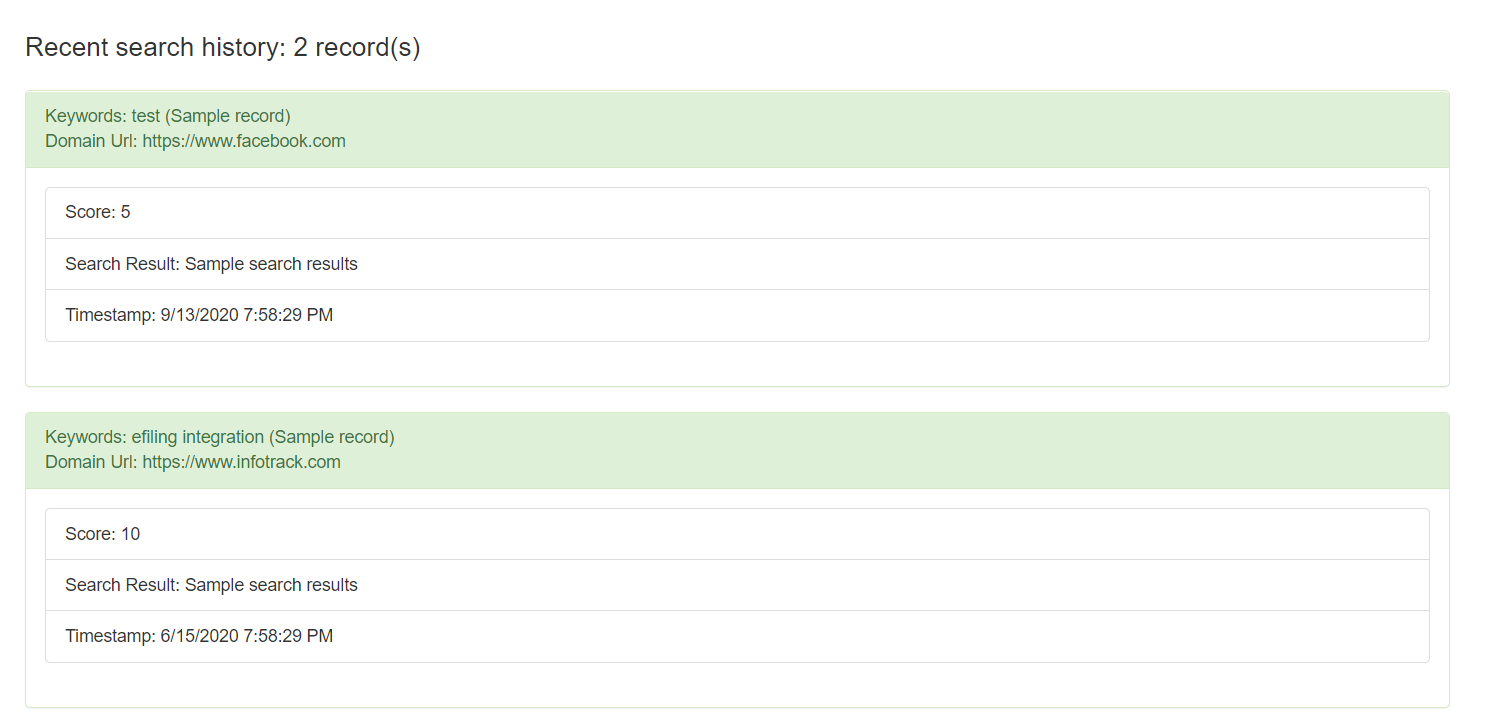
Property Validations Using Getter and Setters



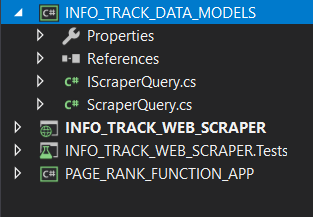


Sample Data: Maintains recent search history

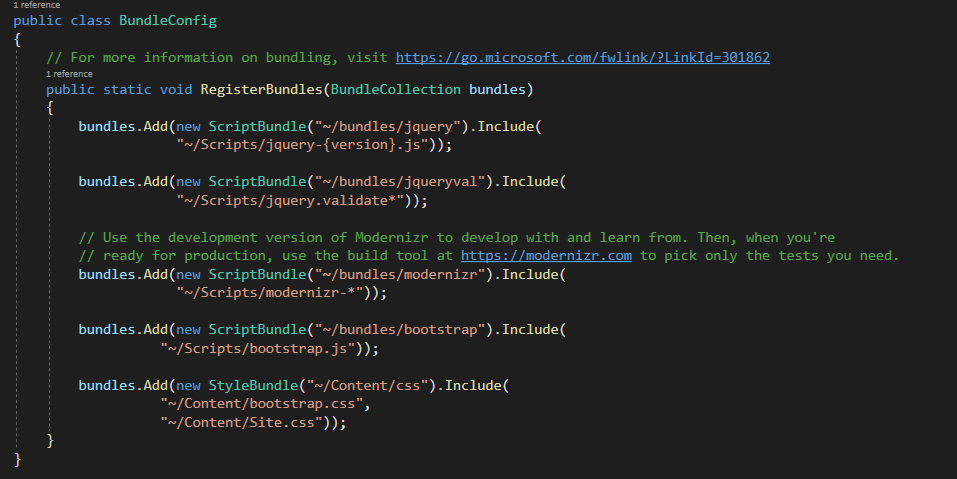
Sorted by the most recent search in descending order.



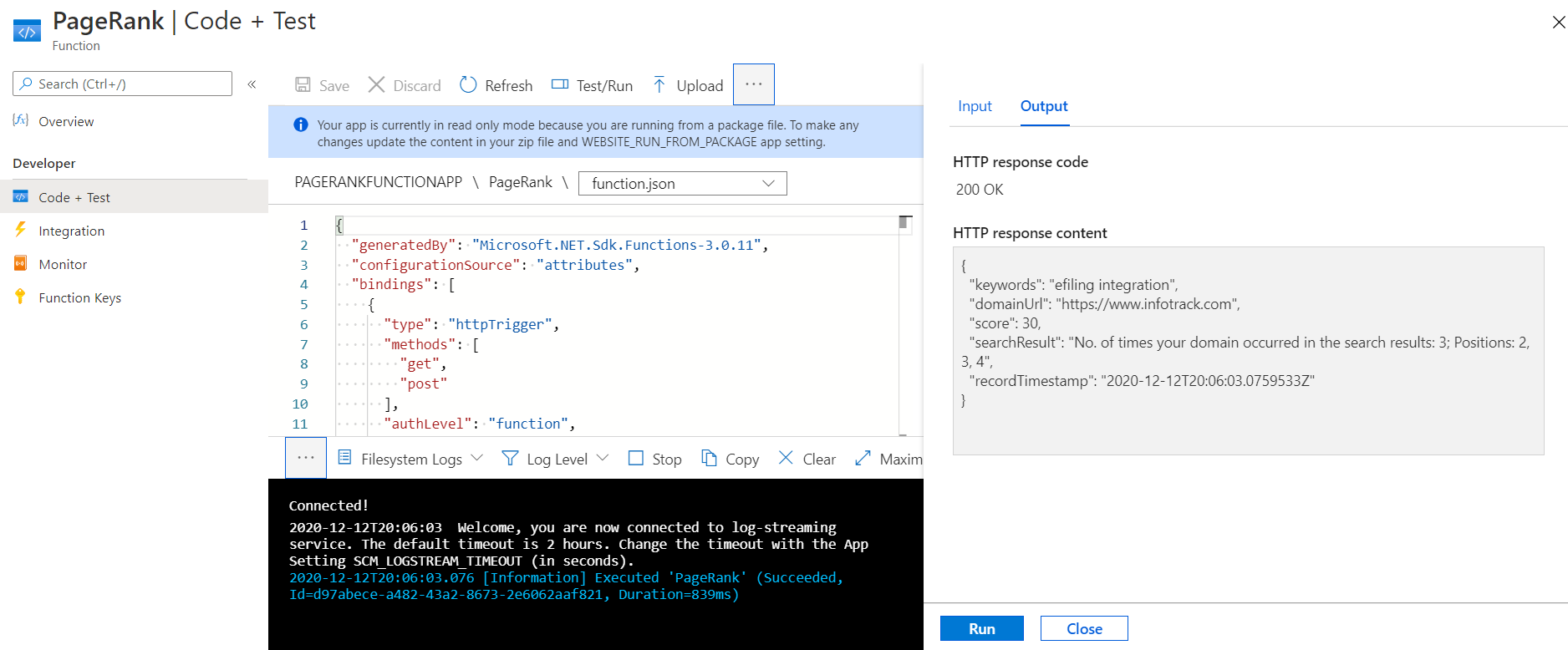
Separate reusable Data Models class library to share objects between the MVC and Function app.



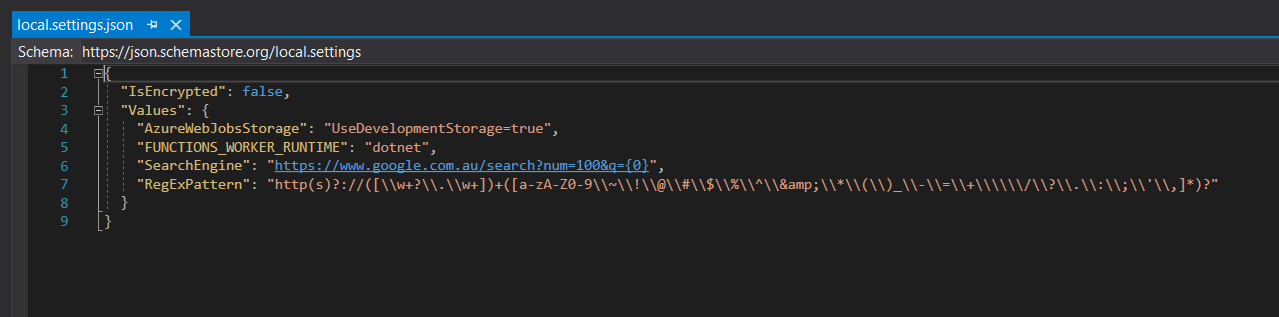
Bundling JavaScript and CSS files to prevent multiple get requests to the server.

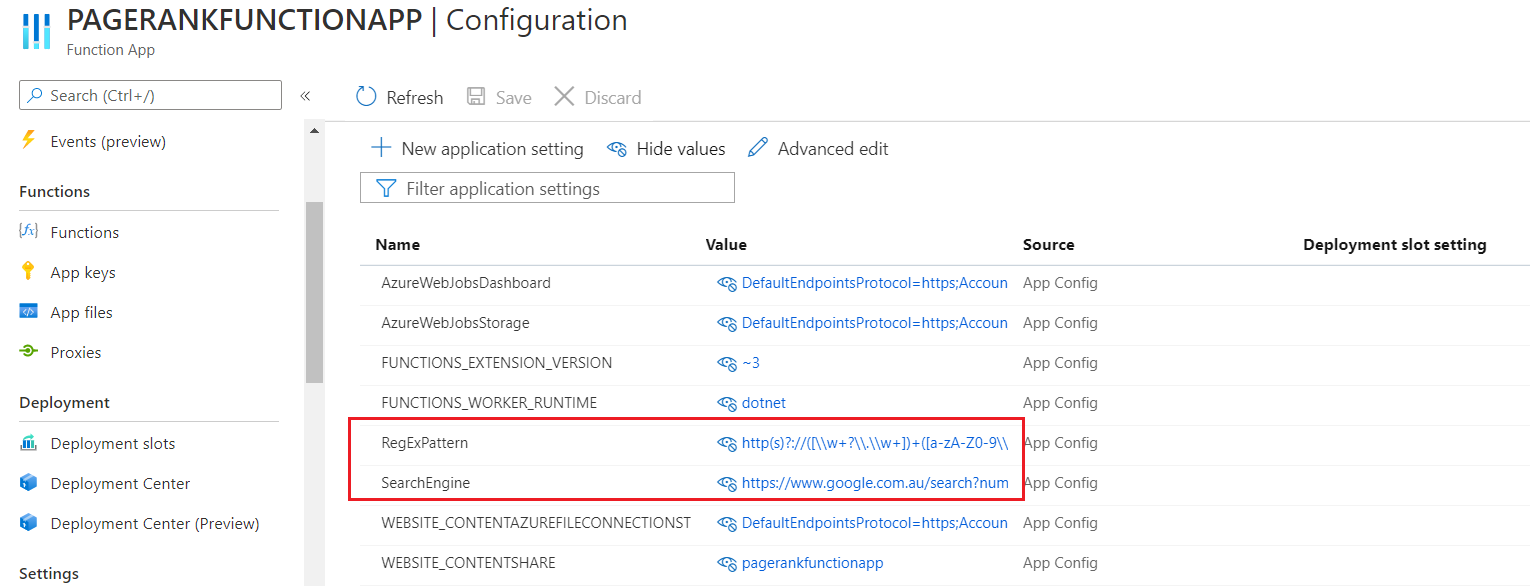


Verified Function app on the Azure cloud platform as well as Postman

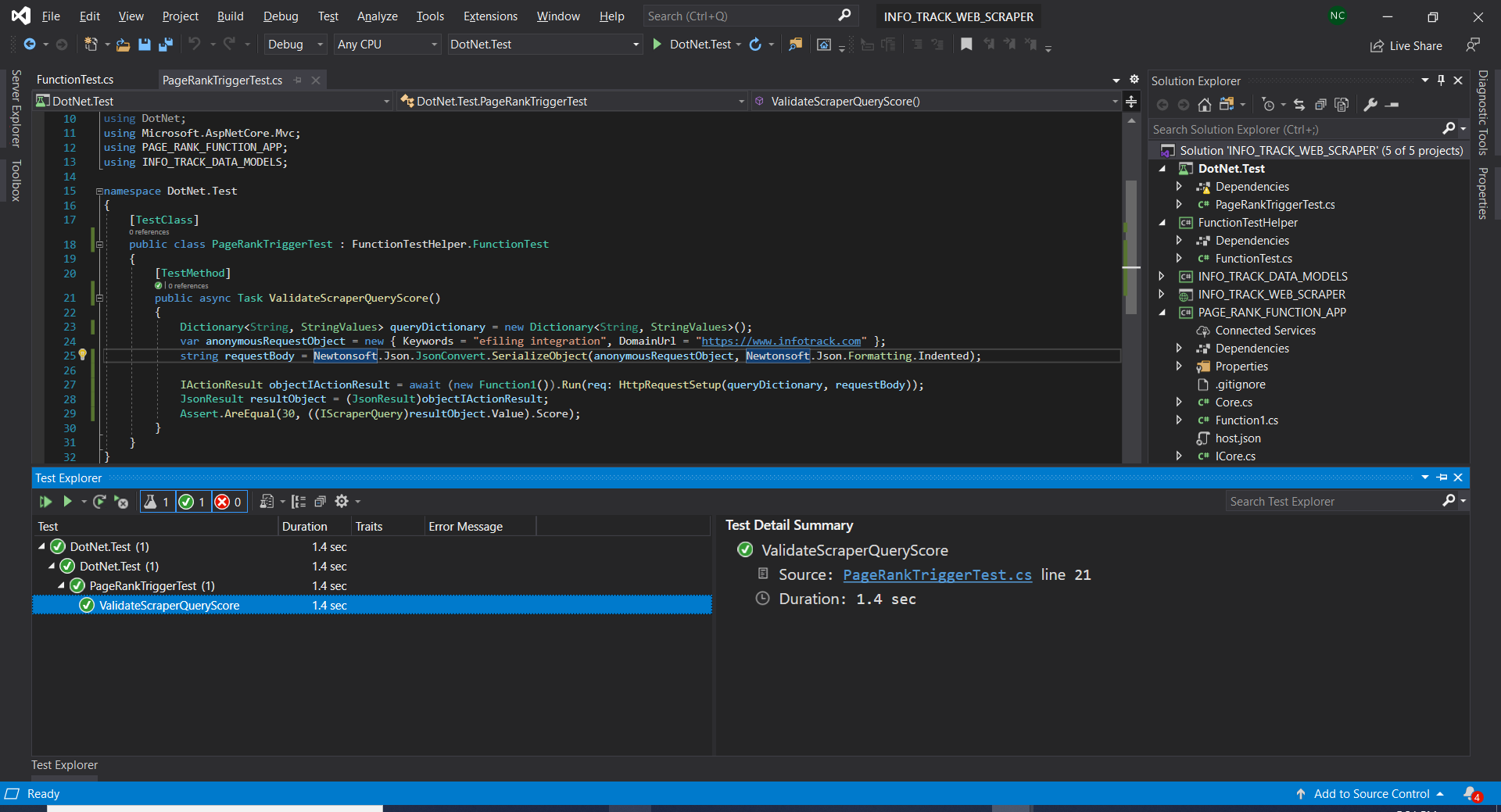


Easy to switch search engine provider and pattern matching by making it config based





Unit Tests



Mocked http request to debug the Azure function

