Screenshot Service

Peda Dizdarevic

November 2019

Stockholm

Overview

Screenshot Service is a server-side application for taking and storing screenshots of web pages. It is an ASP .Net Core Web application, providing a REST API supporting the GET and POST http methods for retrieving and taking screenshots. The screenshots are persisted in a SQL Server database.

In addition to the various parts of .Net Core, created by Microsoft, the application utilizes the following third party frameworks and solutions.

- Selenium: A browser automation tool used for the taking of screenshots.
- Swagger: Provides a simple GUI from which the API can be conveniently accessed.
- DbUp: A script runner used to automatically update the database if necessary upon application start.

Scalability

The next step in the development of this application should involve the separation of the API and the business logic, and the introduction of a message queue for queuing work.

Currently, a user makes a POST request to create a number of screenshots, and all of these screenshots have to be taken before a response is returned. This means that a connection needs to be maintained for a non-trivial amount of time, reducing the capacity for handling many user in a short amount of time.

If large numbers of users are expected, a new service should be created for the actual taking of the screenshots. The endpoints accessed by the user would in this case publish a message to a queue (such as RabbitMQ) and then return the code 202 (accepted) to the user immediately. The second service would, at whatever pace it can handle, poll the queue for messages and perform the queued work (i.e take screenshots and save them to the database).

Furthermore, multiple replicas of this second service can be run simultaneously on separate servers, with more added if demand increases. This can be handled conveniently by containerizing the services with Docker and spinning up multiple replicas with Docker Swarm or Kubernetes. Likewise, the API facing the user can run on multiple parallel servers, joined by and hidden behind a load balancer.

Some notes on Selenium and performance

The decision to use Selenium for taking screenshots can perhaps be questioned. Selenium was chosen for reasons of familiarity and so as to quickly get something that worked in place (considering the imminent deadline). But, it is a rather heavyweight tool, as it provides lots of functionality that is of no use for this application and as it uses a browser which introduces some overhead.

It is possible that there is a more efficient way of creating the screenshots, a way that involves getting the HTML from the site and rendering it in something lighter than a full web browser. Depending on how the screenshots are to be presented in the future, some sort of client side rendering can also be a possibility, which of course scales well with the number of users.