Adobe & AEM Engineering Test

AEM Engineering follows an open development model with an engineering team of hundreds that spans the globe. Every engineer is empowered to contribute to each part of the codebase and engage in asynchronous communication with a diverse team.

To assess your programming skills, communication, and web technology knowledge, we ask you to build us a web service that takes in a number and outputs a Roman numeral.

Requirements

Treat this project as you would treat any serious engineering effort for your employer.

Your HTTP Endpoint must accept a URI in this format:

```
http://localhost:8080/romannumeral?query={integer}
```

`{integer}` must be any integer value in the range 1-255 unless you are asked for Extension 1 (see below). Errors can be returned in plain text format, but success responses must include a JSON payload with two string values, an ‘input’ value and an ‘output’ value. For example, if the input is 1, the response must be a JSON document equivalent to:

```
{
   “input” : “1”,
   “output” : “I”
}
```

Rules

For the number to Roman numeral conversion itself, do not use existing code or libraries. We want to see your development methodology in action.

Use Java or JavaScript as the programming language. If you wish to work in another language, please ask first in response to the email accompanying this test.

Use a public Github repository for your work. Include the link to the Github repository in your response. You may use publicly available technologies for the HTTP server, test framework, etc. Please identify any such dependencies you use.

Tips for a good score

- Find *and reference* a specification for Roman numerals online. Wikipedia is acceptable.
- Provide clear and concise documentation:
  - README.md with:
    - How to build and run your project.
    - Your engineering and testing methodology
    - Your packaging layout
    - Dependency attribution
  - Inline documentation in your source code
- Tests
- Error Handling

© Copyright 2021, Adobe, Inc. All Rights Reserved.

Extensions

You are not required to implement these extensions unless you are asked to do so in the email accompanying this test.

Extension 1: Expand the range of numbers to 1-3999.

Extension 2: Add support for range queries. In addition to the standard integer conversion query described above, you must also support a query of this format:

```
http://localhost:8080/romannumeral?min={integer}&max={integer}
```

`{integer}` must be an integer value for both the min and max parameters. Both min and max must be provided. Min must be less than max. Both must be in the supported range (1-255, or 1-3999 if you were asked to do Extension 1).

Use multithreading (Java) or async processing (JavaScript) to compute the values in the range in parallel. Your code must then assemble the results and return them in ascending order from the minimum to the maximum value.

Successful responses to these queries must include a JSON payload with an array named **conversions**, where each element of the array is an input/output pair of strings, in ascending order. For example, a JSON document equivalent to the following is expected for an input with a minimum of 1 and a maximum of 3:

```
{
  “conversions”: [
    { “input” : “1”, “output” : “I” },
    { “input” : “2”, “output” : “II” },
    { “input” : “3”, “output” : “III” }
  ]
}
```

Extension 3: Include additional DevOps capabilities in your project to represent how you would prepare your project for ease of operation in a production environment (e.g. metrics, monitoring, logging, etc.). Add tooling to build a runnable Docker container for your service if you are familiar with Docker.

© Copyright 2021, Adobe, Inc. All Rights Reserved.