ADVANCED ENTERPRISE DEVELOPMENT

Assignment 2

ABSTRACT

This is the documentation detailing the completed brewery API that was developed as part of assignment 2.

Cian Deely – K00215002

Software Development

Uses Swagger UI

Link to Swagger UI for API Documentation: http://localhost:8888/swagger-ui.html

Self-Evaluation

Strengths

The Brewery API is very robust with some impressive features implemented, such as the ability to generate a QR code that adds a contact to a mobile device for a specified brewery, the ability to generate a PDF brochure for any given beer, the ability to retrieve a map showing the location of a given brewery, and the ability to return either a thumbnail or full-sized image of a given beer.

The API uses bean validation to ensure that submitted values match predefined constraints such as not being empty, being a positive value and matching a pattern such as a phone number or email address.

The API incorporates Swagger-UI which allows for documentation to be accessed both by developers of the API and by users who wishes to incorporate the API in their application and make calls to it. This is useful as it will allow new users to access documentation that will inform them about how the API functions.

Hard crashes due to invalid inputs or calls being made are often avoided using proper validation, this is a positive as it is unprofessional to reveal a hard crash of the server to the user.

The APIs usage of JSON allows for the object it works with to be received and handled across a large and diverse landscape of technologies and programming languages. This allows for the API to be flexible and useful in many different environments, rather than say only being useful in Java based environments.

Maven is used to manage the projects dependencies; this is good as it allows for version control of dependencies.

Weaknesses

The Brewery API mostly deals with only GET requests, with little functionality available for other types of requests such as PUT, POST and DELETE. This means that the API is mostly useful for serving data to be displayed and read on another application, rather than being a database that can be updated or added to.

The API has a lot of imported third party libraries that it depends on to function. This is a disadvantage as if one library is updated it has the potential to cause a conflict or breakdown in the API that may take the server down until it is resolved.

The functionality to return a map showing the location of a specified brewery on it has two weaknesses. Firstly, it is depending on Google's URL to return the map, so if Google were to change the formatting of the URL to request a map object, then the APIs backend code would have to be changed to reflect this. Secondly the map is being requested using the breweries full address, not the breweries coordinates. This means if a brewery doesn't have an address, then the API will be unable to return a map for the brewery.

The API has limited sanitization of input, which leaves the possibility for SQLi in the source code.

Benchmarking & Enhancements

Creating an API for use with a project helps by allowing a large amount of code to be removed from the project and shifted over to an API instead. This allows a project to be more light weight. Additionally, it leaves possibility for an API to be developed in a different language and environment than the project is developed in, the project can then make calls to the API to make use of the functionality in the API.

Additionally, creating the API allows for the functionality it contains to be reused in future applications. This allows us to get more value from the software we create.

I believe the brewery API that was created could be enhanced by adding greater security measures to it to prevent against SQLi and CSRF. This would allow for greater confidence in its security.

Additionally, I believe the API could greatly benefit from added functionality using other methods such as POST, PUT and DELETE. This would transform the API to a service that mostly returns data to be displayed and read into a service that is more dynamic and allows for updating and adding to the database. The addition of an API key to the service with adequate management of that key's distribution may increase the ability to allow for secure use of the API, such as the ability to limit the number of requests any one API key could make to reduce the possibility for spam requests being made.