APRIL 1, 2022

ASSIGNMENT TWO: REST API

SD4 - ADVANCED APPLICATION DEVELOPMENT

K00242965: LEWIS BROWNE

Documentation

The accompanying solution has an exported postman workstation, intended to mitigate the time required to setup this project. The collections present here have all been documented, as can be seen by importing this workstation.

The reason for writing the documentation through postman was to export it into this report, however that is not possible with the current version. Instead Postman allows for the publication of documentation of a developed API which could be shared with a team, a link is provided below to the published documentation for this application:

https://documenter.getpostman.com/view/20246616/UVyrUbtK

Self-Evaluation

The solution presented with this report is, capable of performing almost all use cases it was intended to do. it conforms to HATEOAS application architecture constraints and is a modular application capable of extension. This is by making use of its handler functions and the extensive libraries provided to this project by the Spring Boot framework.

Maven, used for handling package dependencies, has provided many different open-source libraries that allow this application to perform its functionality.

- **JPA**, which is used to persist data between Java objects and a relational database. These are presented as pages, which are collections of objects that allow for limiting or sorting of the persistent object collection.
- The **H2 relational database** runtime library is used in tandem with JPA; it presents an intermediate representation between a database and java class objects. Together these allow for dependency injection, where the Spring framework "injects" objects into other objects or "dependencies".
- **ZXing**, a barcode scanning library that is used to deliver brewery information to its end user. This is done by making use of the vCard file format standard. This allows for the conversion of a string to an ImageMatrix, that is converted to a bytecode array. This is then delivered to the user through Springboot's Request Mapping annotation.
- The Google Map API is extended to present the static map of a brewery's location. An authentication key was required to send a request to this API, and it has been included in the GeoMapHandler utility.
- **Lowagie**, a package library allows for the manipulation of strings to pdfs. Instead of saving these pdfs they are returned through the HttpServletResponse which like ZXing is interpreted by the Request Mapping.
- The **java.utils.zip** package library allows for the compression of files to a zipped array by constructing a bytestream containing the content of that file. RequestMapping annotation again allows for the end-user to receive the file, as it is returned to the end user as a file made of that byte stream representation.

Making use of these packages, and the extensive libraries presented by the spring framework have allowed for rapid development throughout this project. These packages are used by separated handler functions called by the controller. These utilities act as a way to present a view, or in this possible case, a file to the end user. This allows for the enforcing of the MVC design pattern as the data transfer objects are handled by JPA presenting a form of model. Constructing the application in this way allows for extension, where these utilities could be called by other controllers as the project grows in scope.

The faults of this application are:

- A lack of bean validation, making the creation of new beer objects quite difficult. As it does not present any particular error to the user.
- The static map presented provides an approximate location of the brewery but is not capable of pinpointing its exact location.
    - This could be solved by using the brewery geocode object instead of the information stored in the brewery class.
- The API key used to call google maps API is exposed. This information shouldn't be shared over a repository but for the purposes of this assignment is stored in the project to make testing easy. This information should be stored in a separate config file.

Benchmark & Evaluation

If there was more time to improve the application, other than resolving the presented faults, it would be to enforce the use of authentication keys when calling the API. This would be obtained by making a call to a login controller that would validate the user information.

The application could also be improved by implementing versioning. As the scope of this application may grow in size it would be in the best interest of preserving its current functionality, by separating the API into calls likes v1/beer or v2/beer ect.

APIs provide end-users with a means of interacting with a system, using APIs to call or gather information about a system would have many benefits when developing in an enterprise environment. APIs allow for integration into other applications by providing an entry point for data collection. This could be used by a developer to communicate information between two systems or collect and manipulate this data in a myriad of different ways. Such as utilizing the information gathered by an API to create visualizations, filtering through the provided data for information that a developer of a product may require.

Providing a data access layer to an application is the greatest strength of using APIs. Should a developer work in an environment where information wasn't exposed through an API it would require the creation of an independent data layer that manually gathers information from a data stream. It also allows for the controlled exposure of information to users of the API, allowing a developer to select the information returned to the end user. This can be seen in this application through the use of getter methods in class objects.