Yuan Cao (yuancao2)

17625 Assignment 1 Reflection

1. What were some of the alternative design options considered? Why did you choose the selected option?

   The task is to design and implement a RESTful API that supports the following capability:

   - Get the current date
   - Get the current day
   - Get the current month
   - Get the current year
   - Add/modify/delete event
   - Get event(s) for a given date

   Overall, it's a clean API with simple functions. I choose Spring Boot with Spring Web plugins. Spring is a framework that's powerful and handy. It's very easy to get start with for a new project like this. Alternatively, we can use Java servlets with Apache TomEE, or other libraries like https://restlet.talend.com/ and https://sparkjava.com/ to implement the HTTP listeners. On the holistic level, I choose Spring over the others because it's easy to implement and extend. It also works well with marshalling and unmarshalling JSON messages.

   Architecturally, I adopted the MVC design pattern that separates an application into three main logical components Model, View, and Controller. Right now, as the front-end is not required, I only implemented the Controller and Model parts that handles the requests and backend logics. The MVC design pattern supports agile and parallel development that programmers can work on different components together with minimum dependencies on each other.

Yuan Cao (yuancao2)

2. What changes did you need to make to your tests (if any) to get them to pass. Why were those changes needed, and do they shed any light on your design?

   I have passed all the tests and the tests have over 70% coverage. The test I've included are integration tests that perform the request and check the results. One improvement on the test can be to add unit test as well. Especially when the functions become complicated, it's better to test different component separately on the minimum granularity. That is to say, to test model and controller without connecting to request before the integration tests. This time I skip this part due to the simplicity of the functions. And integration tests have already overkilled the problem.

3. Pick one design principle discussed in class and describe how your design adheres to this principle.

   The six guiding principles or constraints of the RESTful API are:

   Client-server, stateless, cache, uniform interface, layered system, code-on-demand

   In my implementation, I focused on client-server, stateless, and uniform interface. Clients send request to server via HTTP requests. The requests are under GET/POST/PATCH/DELETE standard interface. Server handles the requests statelessly. It doesn't keep conversational state between requests. All information needed to process the request or response should be included in the messages. My API also supports JSON messages in requests and reply.