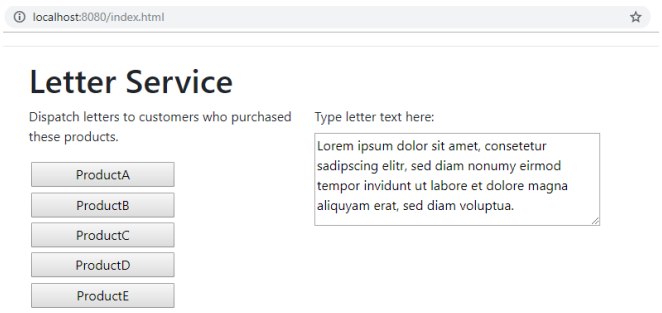
Exercise: Tracing Spring applications with Opentracing and Jaeger

The “Letter service” is a demo application comprised of a set of Java/Spring based demo micro services for demonstrating Opentracing with Jaeger. It simulates dispatching of letters to all customers who purchased a certain product through a web UI. The services are instrumented using the Java Spring Jaeger “`opentracing-spring-web-starter`”. Therefore, basic web requests to the services are instrumented and traced automatically, while some manually added tracing code provides more fine grained traces.

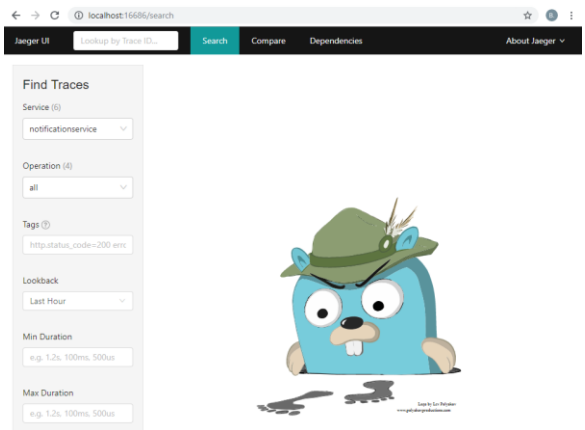
Exercise Setup:

The application is available on GitHub: https://github.com/RETIT/jaeger-spring-demo

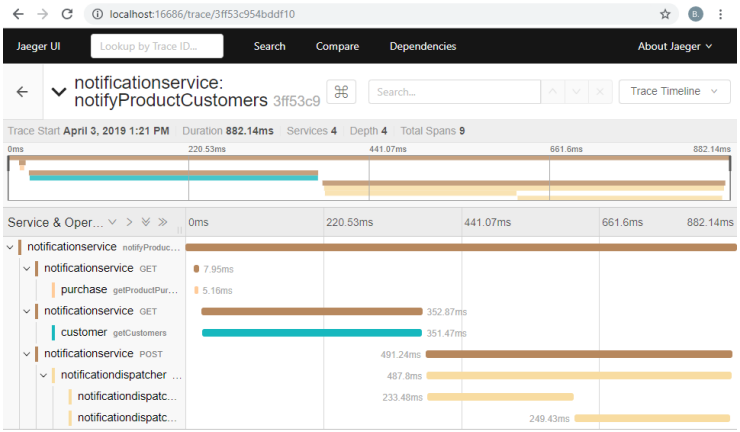
Clone/Download the repository and follow the steps described in the `Readme.me` file to build and run the services. Entry point of the “Letter Service” application is the “notificationservice”, which provides the pictured web UI. After completing the setup, the UI can be accessed under: http://localhost:8080/index.html



The Jaeger UI is available under: http://localhost:16686/search



After having clicked any of the buttons on the “Letter Service” page at least once, Traces for the “notificationservice” are available in the Jaeger UI’s “Search” tab. They may be inspected by clicking on them:



Exercise 1

When clicking different buttons in the demo UI, the response times vary significantly depending on the button. Use the Jaeger UI to investigate the observed pattern and answer the following questions:

1. Why do the response times vary between buttons?
2. Which (design) changes in the applications could theoretically be implemented in order to make the response times more consistent?

Exercise 2

When simulating load by means of multiple clicks in quick succession, the response times increase, even for the buttons that respond more quickly when clicked only once (“ProductD”).

1. Use the Jaeger UI to investigate the observed behavior. Which service is responsible for the deteriorating response times?
2. Implement additional tracing into the responsible “backend” service by means of custom spans in order to make its processing visible as span in the Jaeger UI. The class DemoDispatcherWorker.java may serve as reference for the required code pattern. For stopping, building and restarting all services/containers of the demo application you may use these scripts: `stopContainers.bat (.sh)` and `startAll.bat (.sh)`.
3. Using the additional tracing, observe the service’s behavior and identify the reason for the increasing response times under load.