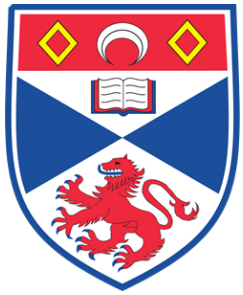
University of St Andrews

CS4103 Coursework 1

Middleware

Author:
150008022

February 19, 2019



Goal

The goal of this practical was to implement a distributed application using communication middleware to collect data on user responses to the n-person prisoner's dilemma.

Part I
Communication Set-Up

The application was set up using Spring Initializr [1], which provided a Spring Boot application template, with the web and test starter dependencies. A controller class to provide REST request mappings was implemented, and the test connection method was created in the ProsecutorService class. Unit tests were written for both, and Postman was used to test the request when the application server was running, and to fulfill the criteria for part one.

To provide a user-friendly client, create-react-app [2] was used to boostrap a react.js application. By using a proxy setting, CORS issues were avoided while developing locally. A simple test button was supplied that would perform an asynchronous call to the API test endpoint, and then print the response to the screen (Figures 1 and 2). When building distributed systems, it is advised to always design for failure, and so a clear "Failed to Connect" message is shown to the user if the request fails for whatever reason.

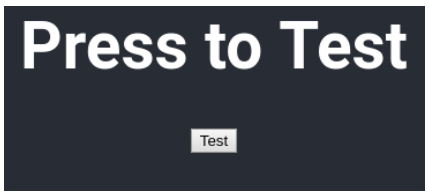


Figure 1: Before Press

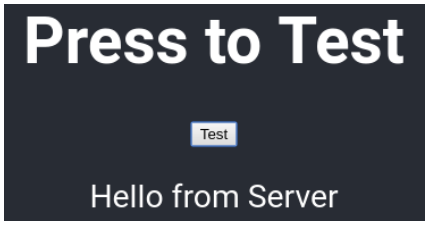


Figure 2: After Press

Part II
Single Client Game

First, the Persecutor service in the Spring Boot application was extended to include the *chooseOption* method, which would take the choice of prisoner one and return the number of years reduction based on a random choice from prisoner two. To reliably test all outcomes produced the desired result when using a random choice generator, the random choice was provided through a separate service which could then be mocked in tests.

Since only one game would exist at this stage, the endpoint */prosecutor/game* was chosen to be where the prisoner would send their choice in a JSON format. By PUTing their choice to this endpoint, the user could expect a return value of the number of years reduction. PUT was chosen as it is defined by the HTTP standard to be idemptotent, and so it shouldn't matter how many times a user submitted the same choice as it would have the same result (though the current random nature of prisoner 2 would not provide this behaviour). Spring boot provided the (de)serialization between JSON and the Java object representation through the Jackson library.

Conclusion

[1] Pivotal Web Services. Spring initializr.

[2] Facebook. create-react-app.