VI Cloud Bus

projekt startowy : CLOUD_BUS_START

rozwiązanie : CLOUD_BUS_END

walkthrough

- zainstaluj i uruchom RabbitMQ
    - https://www.rabbitmq.com/
- limit-service
    - booststrap.properties dodaj
        - management.endpoints.web.exposure.include=*
- dodaj zależność spring-cloud-starter-bus.ampq
    - spring-cloud-config-server
    - limit-service
- restart (najpierw spring-cloud-config-server, potem multirun, kilka instancj limit-service)
- przetestuj w przeglądarce działanie obu instancji
- zmień w git plik configuracyjny dla profilu dev
- odśwież wszystkie instancje jednym komunikatem [POST] (wybierz dowolny port)
    - localhost:PORT/actuator/bus-refresh
    - sprawdź czy zaszła zmiana na obu instancjach

VII Zuul

projekt startowy : API_GATEWAY_START

rozwiązanie : API_GATEWAY_END

walkthrough I

- utwórz komponent rozszerzający ZuulFilter
  - główna
    - @EnableZuulProxy
  - maven
    - spring-cloud-starter-netflix-zuul
- application.properties:
  - spring.application.name=gateway-server
  - server.port=8765
  - eureka.client.service.url.default-zone=http://localhost:8761

walkthrough II

- w metodzie zaimplementuj logowanie URI
  - utwórz @Component LoggingFilter extends ZuulFilter
  - metoda run
    - użyj w RequestContext.getCurrentContext().getRequest()
    - zwróci null
  - filterType
    - “pre”
- request przez gateway ‘maska’
    - http://localhost:**8765**/<app-name>/<url>
- **przykład**
  - http://localhost:8000/currency-exchange/from/EUR/to/PLN
  - http://localhost:8765/exchange-service/currency-exchange/from/EUR/to/PLN

walkthrough II

- refactor conversion-service: przekierowanie przez Zuul
  - test
- na jaki URL wyślemy request do cs, żeby przeszedł przez API Gateway?
  - wyślij i podejrzyj logi w Zuul