Application.properties:

* Spring.application.name=stock-exchange-service
* spring.security.user.name=admin
* spring.cloud.config.uri=http://localhost:8888/
* spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=create  
spring.datasource.url=jdbc:mysql://localhost:3306/currency\_exchange?serverTimezone=UTC  
spring.datasource.username=root  
spring.datasource.password=adminadmin

manifest.yml

* name: stock-exchange-service
* buildpack: https://github.com/cloudfoundry/java-buildpack
* memory: 1G
* path: build/libs
* services:  
   - myConfigurationServer
* env: spring\_jpa\_hibernate\_ddl-auto: create

build.gradle

* implementation 'io.pivotal.spring.cloud:spring-cloud-services-starter-service-registry'
* compile group: 'io.pivotal.spring.cloud', name: 'spring-cloud-services-starter-config-client', version: '2.1.4.RELEASE'

Few Imp Annotations:

* @SpringBootApplication  
  @EnableFeignClients("com.vivekchutke.stock.exchange.stockexchangeservice")  
  @EnableDiscoveryClient

@EnableHystrix

* @RestController
* @ResponseStatus(HttpStatus.NOT\_FOUND) # On Exception Class
* @Component # use to grab config properties
* @EnableCircuitBreaker # Add this on the controller class at class level
* @Value("${stock-exchange-service.xRapidAPIHost}")
* @RequestMapping(value = "/stock/{stockCode}/book", method = RequestMethod.GET)
* ResponseEntity<OpenHighLowClose> responseEntity = new RestTemplate().exchange("https://host:

/stock/{code}/ohlc", HttpMethod.GET, getHeaderEntity(), OpenHighLowClose.class, uriVariables);

* ResponseEntity<Bean> responseEntity = new RestTemplate().getForEntity("http://host: /service/from/{from}/to/{to}", Bean.class, uriVariables);
* @RibbonClient(name="currency-exchange-service")
* @FeignClient(name="stock-exchange-service", url = "https://investors-exchange-iex-trading.p.rapidapi.com/") ## use on proxy interface
* @PathVariable(name = "symbol") #Define proxy like this in the method definition
* @RequestParam(name = "products") #Defines the Query param
* @XmlRootElement
* @Configuration  
  @EnableWebSecurity # to override the spring security. should extend WebSecurityConfigurerAdapter
* @EnableJpaAuditing
* @CreationTimestamp
* use data.sql or import.sql to pre-insert records
* @Repository # Use it on the Repo interface for Querying Database
* @Query(value = "SELECT \* FROM products where prod\_id in (:productIds)", nativeQuery = true)
* currency-exchange-service.ribbon.listOfServers = http://localhost:8000, http://localhost:8001
* @HystrixCommand(fallbackMethod = "retrieveDefaultExchangeValue")
* <http://localhost:8800/v2/api-docs> && <http://localhost:8800/swagger-ui.html>
* InstanceInfo instanceInfo = eurekaClient.getNextServerFromEureka("currency-service", false);

String baseUrl = instanceInfo.getHomePageUrl().concat("currency-exchange/from/{from}/to/{to}") ;

* eureka.client.service-url.defaultZone=http://localhost:8761/eureka
* currency-exchange-service.ribbon.listOfServers = http://localhost:8000, <http://localhost:8001>
* spring.cloud.config.server.git.uri=https://github.com/vivekchutke/propertiesrepo.git
* <http://localhost:8888/limits-service/qa.properties> or http://localhost:8888/limits-service/qa
* spring.cloud.config.uri=http://localhost:8888/
* spring.cloud.config.discovery.enabled=true # To have the app look for config server in eureka
* java -Dspring.config.additional-location=../config/ -jar build/libs/project-server.jar
* java -Dspring.datasource.url=jdbc:mysql://localhost:3306/products?serverTimezone=UTC -Dserver.port=8550 -jar mySqlTest-0.0.1-SNAPSHOT.jar
* sudo lsof -i :8080 | grep LISTEN # to know the process running
* ps –A # To List processes running on linux
* env:  
   spring\_jpa\_hibernate\_ddl-auto: create  
   spring.security.user.name: ${ vcap.services.my-cfcups-service.credentials.username}  
   spring.security.user.password: ${vcap.services.my-cfcups-service.credentials.password}  
   SPRING\_PROFILES\_ACTIVE: qa
* HttpHeaders createHeaders(String username, String password){
* return new HttpHeaders() {{
* String auth = username + ":" + password;
* byte[] encodedAuth = Base64.encodeBase64(
* auth.getBytes(Charset.forName("US-ASCII")) );
* String authHeader = "Basic " + new String( encodedAuth );
* set( "Authorization", authHeader );
* }};
* }