### 走一遍

systemctl stop firewalld.service #停止firewall

systemctl disable firewalld.service #禁止firewall开机启动

firewall-cmd --state #查看默认防火墙状态（关闭后显示notrunning，开启后显示running）

pom.xml

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.graphql</groupId>  <artifactId>graphqlDemo</artifactId>  <version>0.0.1-SNAPSHOT</version>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.1.RELEASE</version>  </parent>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>com.graphql-java</groupId>  <artifactId>graphql-spring-boot-starter</artifactId>  <version>4.0.0</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

application

|  |
| --- |
| **package** com.graphql;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  **public** **class** Application {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(Application.**class**, args);  }  } |

controller

|  |
| --- |
| **package** com.graphql.controller;  **import** **static** graphql.schema.idl.RuntimeWiring.*newRuntimeWiring*;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** graphql.ExecutionResult;  **import** graphql.GraphQL;  **import** graphql.schema.GraphQLSchema;  **import** graphql.schema.StaticDataFetcher;  **import** graphql.schema.idl.RuntimeWiring;  **import** graphql.schema.idl.SchemaGenerator;  **import** graphql.schema.idl.SchemaParser;  **import** graphql.schema.idl.TypeDefinitionRegistry;  @RestController  **public** **class** GraphQLController {  // public static void main(String[] args) {  @GetMapping("/get")  **public** String get() {  String schema = "type Query{hello: String, Jim: String}";  SchemaParser schemaParser = **new** SchemaParser();  TypeDefinitionRegistry typeDefinitionRegistry = schemaParser.parse(schema);  RuntimeWiring runtimeWiring = *newRuntimeWiring*()  .type("Query", builder -> builder  .dataFetcher("hello", **new** StaticDataFetcher("world"))  .dataFetcher("Jim", **new** StaticDataFetcher("welcome")))  .build();  SchemaGenerator schemaGenerator = **new** SchemaGenerator();  GraphQLSchema graphQLSchema = schemaGenerator.makeExecutableSchema(typeDefinitionRegistry, runtimeWiring);  GraphQL build = GraphQL.*newGraphQL*(graphQLSchema).build();  ExecutionResult executionResult = build.execute("{Jim}");  System.***out***.println(executionResult.getData().toString());  **return** executionResult.getData().toString();  }  } |

