Dưới đây là một ví dụ hoàn chỉnh về cách kết hợp Java Spring Boot với GraphDB để thực hiện truy vấn SPARQL. Bạn cần thay đổi một số phần như URL của GraphDB endpoint và xử lý kết quả truy vấn SPARQL theo nhu cầu của ứng dụng cụ thể của bạn.

Bước 1: Thêm Dependency vào pom.xml

xml
Copy code
```
<dependencies>
    <!-- ... các dependency khác ... -->
    <dependency>
        <groupId>com.ontotext.graphdb</groupId>
        <artifactId>graphdb-free-runtime</artifactId>
        <version>INSERT_VERSION_HERE</version>
    </dependency>
</dependencies>
```
Bước 2: Cấu Hình Kết Nối đến GraphDB trong application.properties hoặc application.yml

properties
Copy code
```
# application.properties

graphdb.url=http://localhost:7200/repositories/your-repository-name
```
Thay thế your-repository-name bằng tên repository bạn đang sử dụng.

Bước 3: Tạo Configuration Class Cho GraphDB

java
Copy code
```
import com.ontotext.graphdb.repository.GraphDBRepository;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class GraphDBConfig {

    @Value("${graphdb.url}")
    private String graphDBUrl;

    @Bean
    public GraphDBRepository graphDBRepository() {
        return new GraphDBRepository(graphDBUrl);
    }
}
```
Trong đoạn mã trên:

@Configuration đánh dấu lớp là một lớp cấu hình.

@Value("${graphdb.url}") sử dụng để chú thích giá trị của thuộc tính từ file cấu hình (application.properties).
@Bean được sử dụng để tạo một bean của GraphDBRepository để quản lý kết nối và tương tác với GraphDB.
Bước 4: Tạo Service Class Cho Xử Lý Dữ Liệu RDF và Truy Vấn SPARQL

java
Copy code
```
import com.ontotext.graphdb.repository.GraphDBRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class RDFService {

    private final GraphDBRepository graphDBRepository;

    @Autowired
    public RDFService(GraphDBRepository graphDBRepository) {
        this.graphDBRepository = graphDBRepository;
    }

    public String executeSparqlQuery(String query) {
        // Sử dụng GraphDBRepository để thực hiện truy vấn SPARQL
        return graphDBRepository.executeSparqlQuery(query);
    }
}
```
Trong đoạn mã trên:

@Service đánh dấu lớp là một service Spring.
@Autowired được sử dụng để inject GraphDBRepository vào RDFService.
executeSparqlQuery là một phương thức để thực hiện truy vấn SPARQL sử dụng GraphDBRepository.
Bước 5: Tạo Controller Class để Xử Lý Yêu Cầu HTTP

java
Copy code
```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class RDFController {

    private final RDFService rdfService;

    @Autowired
    public RDFController(RDFService rdfService) {
        this.rdfService = rdfService;
```

```
    }

    @GetMapping("/sparql")
    public String executeSparqlQuery(@RequestParam String query) {
        // Gọi service để thực hiện truy vấn SPARQL
        return rdfService.executeSparqlQuery(query);
    }
}
```

Trong đoạn mã trên:

@RestController đánh dấu lớp là một controller Spring.
@Autowired được sử dụng để inject RDFService vào RDFController.
@GetMapping("/sparql") định nghĩa một API endpoint cho việc thực hiện truy vấn SPARQL.

==========

Dưới đây là phiên bản build.gradle tương đương với đoạn mã pom.xml và cấu hình trong ví dụ trước đó:

groovy
Copy code

```
plugins {
    id 'org.springframework.boot' version '2.5.4'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'com.ontotext.graphdb:graphdb-free-runtime:INSERT_VERSION_HERE'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

springBoot {
    buildInfo()
}

// Thay thế INSERT_VERSION_HERE bằng phiên bản GraphDB bạn đang sử dụng
```

Lưu ý rằng bạn cần thay thế INSERT_VERSION_HERE bằng phiên bản cụ thể của GraphDB mà bạn đang sử dụng.

Bạn cũng có thể sử dụng plugin org.springframework.boot để tạo ứng dụng có thể chạy độc lập. Nếu bạn muốn tạo file thực thi JAR chạy được bằng java -jar, thêm cấu hình như sau:

groovy
Copy code
```
springBoot {
    buildInfo()
    executable = true
}
```
Sau khi tạo file build.gradle mới, bạn có thể sử dụng lệnh sau để tải các dependencies và xây dựng dự án:

bash
Copy code
```
./gradlew build
```
Lệnh trên sẽ tạo ra file build/libs/your-application-name.jar. Bạn có thể chạy ứng dụng bằng lệnh:

bash
Copy code
```
java -jar build/libs/your-application-name.jar
```
Lưu ý rằng bạn cần phải cài đặt Gradle trên máy tính của mình để sử dụng các lệnh trên.

========== Python

Nếu bạn muốn sử dụng FastAPI trong Python để tạo một ứng dụng tương tác với GraphDB, bạn cũng cần sử dụng một thư viện RDF Python để thực hiện truy vấn SPARQL và tương tác với dữ liệu RDF. Một trong những thư viện phổ biến là rdflib.

Dưới đây là một ví dụ về cách có thể thiết lập một ứng dụng FastAPI đơn giản để thực hiện truy vấn SPARQL và tương tác với GraphDB. Trước hết, bạn cần cài đặt các thư viện cần thiết:

bash
Copy code
```
pip install fastapi uvicorn rdflib
```
Sau đó, tạo một file main.py với nội dung sau:

python
Copy code
```
from fastapi import FastAPI, HTTPException
from rdflib import Graph, URIRef

app = FastAPI()
```

```
# Đường dẫn đến GraphDB endpoint
graphdb_endpoint = "http://localhost:7200/repositories/your-repository-name"

# Tạo đối tượng Graph từ đường dẫn
graph = Graph()
graph.load(graphdb_endpoint, format="turtle")


@app.get("/sparql")
async def execute_sparql_query(query: str):
    try:
        # Thực hiện truy vấn SPARQL
        results = graph.query(query)

        # Chuyển kết quả thành danh sách Python để trả về
        result_list = [dict(zip(results.vars, binding)) for binding in
results.bindings]

        return {"results": result_list}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Trong đoạn mã trên:

FastAPI được sử dụng để tạo ứng dụng web API.
rdflib được sử dụng để tương tác với dữ liệu RDF và thực hiện truy vấn SPARQL.
Endpoint SPARQL /sparql nhận một truy vấn SPARQL thông qua query parameter và trả về kết quả dưới dạng JSON.
Chạy ứng dụng FastAPI bằng lệnh sau:

bash
Copy code
uvicorn main:app --reload
Sau đó, bạn có thể truy cập http://127.0.0.1:8000/sparql?query=YOUR_SPARQL_QUERY trong trình duyệt hoặc sử dụng các công cụ khác như curl hoặc Postman để gửi truy vấn SPARQL và nhận kết quả.

Lưu ý: Bạn cần thay thế your-repository-name và YOUR_SPARQL_QUERY bằng tên repository và truy vấn SPARQL của bạn.