VIETNAM NATRIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

Môn: Phát triển Ứng dụng Web Nâng cao

Lớp 20KTPM1

20127237 - NGUYỄN TẤN LỰC
20127507 - BÙI TRẦN HUÂN
20127224 - DƯƠNG ĐẶNG THÀNH LÂM
20127470 - THÂN MINH ĐỨC

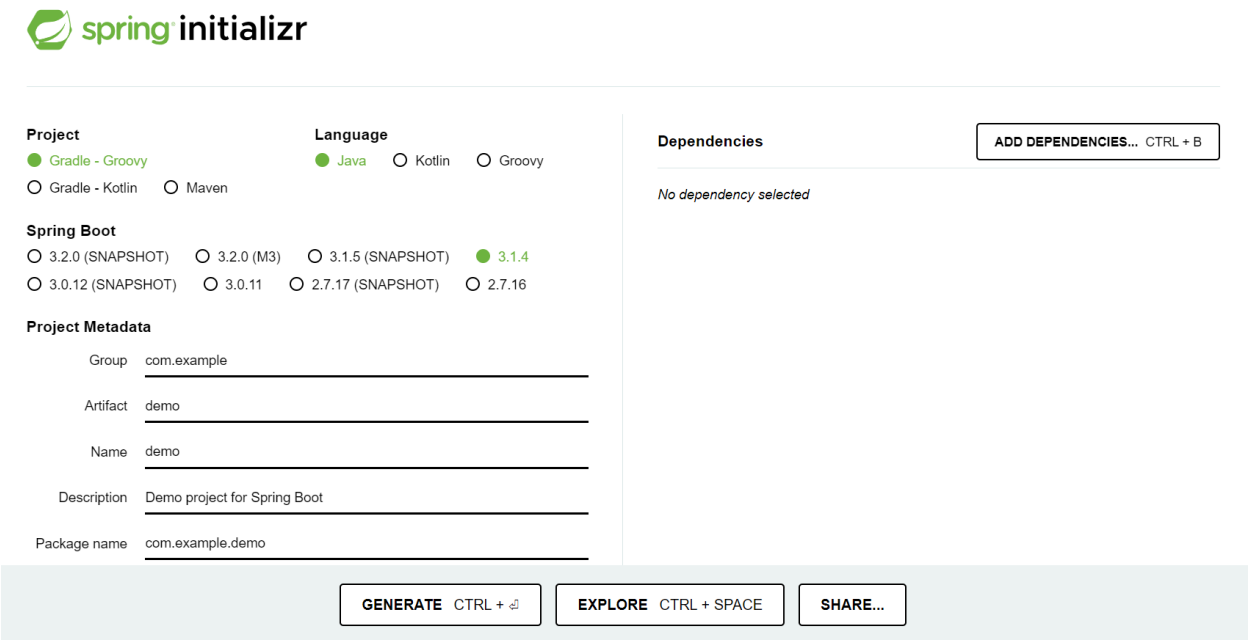
Seminar 01

Giảng viên: Ngô Ngọc Đăng Khoa

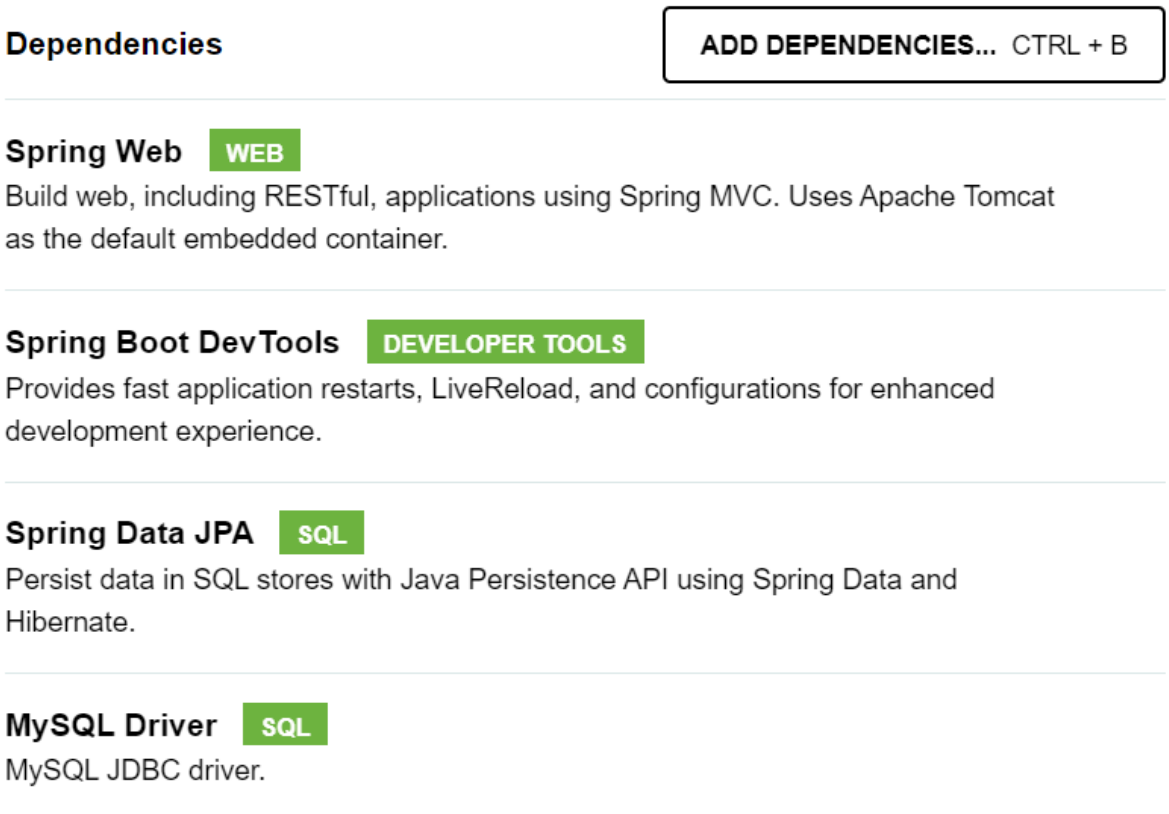
Thành phố Hồ Chí Minh – 2023

1. Khởi tạo dự án:

Truy cập trang https://start.spring.io để generate dự án SpringBoot.



Chọn Hệ thống build tự động Gradle Hoặc Maven
Chọn Version SpringBoot 3.1.4
Cài đặt các thông tin của Project trong mục Project Metadata
Chọn Dependencies:



SpringWeb: Hỗ trợ xây dựng RestFul API

Spring Boot DevTools: Công cụ hỗ trợ Dev
Spring Data JPA: Hỗ trợ giao tiếp với DB
MySQL Driver: Hỗ trợ kết nối với MySQL

Chọn Generate để tải file zip về. Extract ra và mở file đó trong IDE

2. Config Project:

Chọn Version Gradle để build: Ctrl+Alt+S hoặc File->Setting, chọn Build, Excution, Development-> Gradle -> Chọn Gradle JVM
Vào file build.gradle để build các dependencies

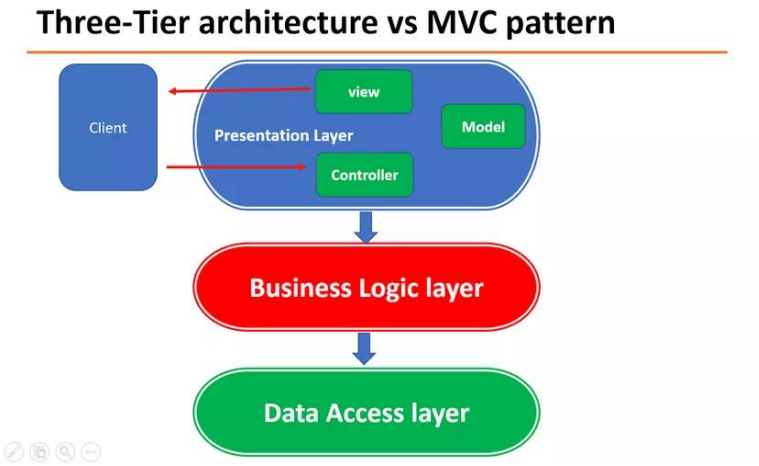
```
group = 'com.example'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'mysql:mysql-connector-java:8.0.27'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.hibernate.validator:hibernate-validator:7.0.1.Final'
}

tasks.named('test') { Task it ->
    useJUnitPlatform()
}
```

SpringBoot Initlizer đã tự động tạo 1 file Application để chạy Project (Có Annotation `@SpringBootApplication`)

Tạo folder Actor để viết các RestAPI liên quan đến Actor, trong folder Actor gồm có các file:

1. Actor: Entity Model đại diện cho bảng Actor trong DB
2. ActorController: Routing các endpoint (Đại diện cho tầng Web Presentation)
3. ActorDTO: 1 Model khác để Controller giao tiếp với Client (Hạn chế giao tiếp với Client thông qua Entity Model - vấn đề bảo mật và hiệu suất)
4. ActorService: Xử lý BusinessLogic (Đại diện cho tầng BusinessLogic)

5. ActorRepository: Giao tiếp trực tiếp với DB (Đại diện cho tầng Data Access)



Config để kết nối với DB tại file src/main/resource/application.properties:

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/
sakila
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql=true
#spring.jpa.show-sql: true
```

trong đó:
spring.jpa.hibernate.ddl-auto: thao tác với DB: update, create, none
spring.datasource.url: đường dẫn tới DB
spring.datasource.username, spring.datasource.password: username và password
spring.datasource.driver-class-name: drive để kết nối DB
spring.jpa.show-sql: true (hiện các query khi giao tiếp với DB), default là false

Cài đặt dự án:

1. Cài đặt Entity Model: Actor (đại diện cho 1 bảng trong Table)

a. Thêm @Entity để SpringBoot Mapping với bảng Actor trong DB, có thể thêm `@Table(name = "actor")` để báo với SPringBoot là đang mapping với Bảng Actor (nếu không ghi chú dòng này, SpringBoot sẽ mapping tự động dựa theo tên Model)

b. Cài đặt các Attributes cho class sao cho khớp với bảng trong DB:

```
public class Actor {
@Id
@GeneratedValue(strategy= GenerationType.IDENTITY)

@Column(name = "actor_id")
private Short id;

@Column(name = "first_name")
private String firstName;

@Column(name = "last_name")
private String lastName;
```

@ID chú thích cho thuộc tính tiếp theo là primary key
@GeneratedValue config cho ID tự động tăng lên khi thêm 1 hàng mới vào bảng

@Column chú thích thuộc tính đó đại diện cho 1 cột trong bảng (có thể chú thích thêm name để xác định trực tiếp cột, nếu không SpringBoot sẽ tự động dựa vào tên thuộc tính)

Cài đặt các getter và setter cho các thuộc tính cần thiết

2. *Cài đặt Repository: ActorRepository*

```
public interface ActorRepository extends CrudRepository<Actor, Short> {
public Optional<Actor> findByFirstNameAndLastName(String firstName, String
lastName);
}
```

CrudRepository<Actor,Short> là 1 class hỗ trợ giao tiếp với DB, cung cấp đầy đủ các hàm cơ bản như findByID, findAll, Delete,...
Actor: chọn Entity Model
Short: kiểu dữ liệu của Primary Key của Entity Model tương ứng

```
public Optional<Actor> findByFirstNameAndLastName(String firstName, String
lastName);
// custom hàm tìm theo firstName và lastName, SpringBoot sẽ tự động cài đặt
và mapping dựa trên tên trong Entity Model đã cài đặt trước
```

3. *Cài đặt Service: ActorService*

```
@Service
public class ActorService {
```

gắn Annotation @Service để SpringBoot biết đó là phần Service

```
private final ActorRepository actorRepository;

@Autowired
public ActorService(ActorRepository actorRepository) {
this.actorRepository = actorRepository;
}
```

Dependency Injection với Repository đã cài đặt trước
gắn Annotation @Autowired để áp dụng Dependency Injection
Cài đặt các tác vụ xử lí với từng API

```
public Iterable<Actor> getAll(){
return actorRepository.findAll();
}

public String addNewActor(String firstName, String lastName){
Actor actor=new Actor();
actor.setFirstName(firstName);
actor.setLastName(lastName);
actorRepository.save(actor);
```

```
return "Saved Success";
}

public Optional<Actor> getById(Short id){
return actorRepository.findById(id);
}
public String deleteById(Short id){
actorRepository.deleteById(id);
return "Delete Success";
}

public String updateById(Short id,String firstName, String
lastName){
Optional<Actor> actorOptional=actorRepository.findById(id);

Actor actor=actorOptional.get();
actor.setFirstName(firstName);
actor.setLastName(lastName);

actorRepository.save(actor);
return "Update Success";
}
```

4. Cài đặt Controller: ActorController

```
@RestController
@RequestMapping(path = "/api/v1/actor")
@CrossOrigin(origins = "*")
public class ActorController {
private final ActorService actorService;

public ActorController(ActorService actorService) {
this.actorService = actorService;
}

@GetMapping(path = "/all")
public Iterable<Actor> getAll(){
return actorService.getAll();
}

@PostMapping(path = "/add")
public String addNewActor(@RequestBody ActorDTO form){
String firstName = form.getFirstName();
String lastName = form.getLastName();

return actorService.addNewActor(firstName,lastName);
}

@GetMapping(path = "/{id}")
public Optional<Actor> getById(@PathVariable Short id){
return actorService.getById(id);
}
```

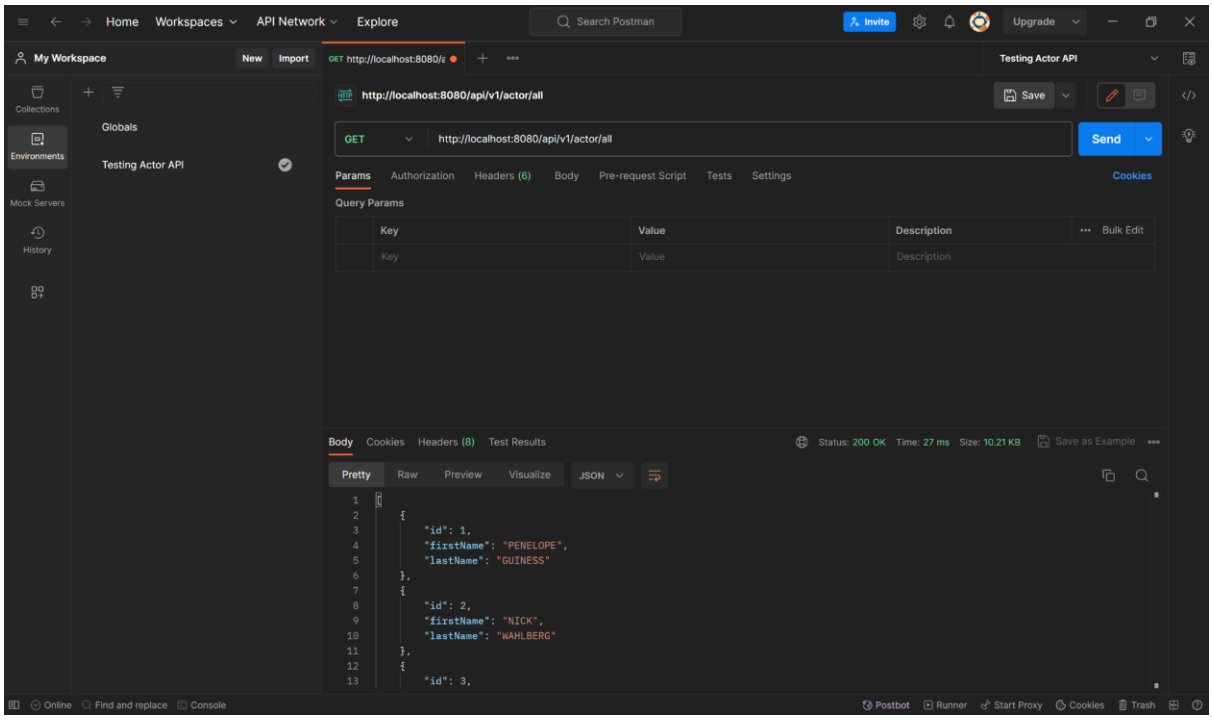
```
@DeleteMapping(path = "/{id}")
public String deleteById(@PathVariable Short id){
return actorService.deleteById(id);
}

@PutMapping(path = "/{id}")
public String updateById(@PathVariable Short id,@RequestBody ActorDTO form){
String firstName = form.getFirstName();
String lastName = form.getLastName();
return actorService.updateById(id,firstName,lastName);
}
}
```

@RestController khai báo Đây là Controller của Restful API khi đó dữ liệu sẽ nằm trong Response Body
@RequestMapping Config endpoint cho Controller
@CrossOrigin cài đặt CORS
@PostMapping,@GetMapping, @DeleteMapping, @PutMapping: cài đặt endpoint và method
@RequestBody, @PathVariable: PathVariable dùng để xác định biến trong path URL và chọn RequestBody

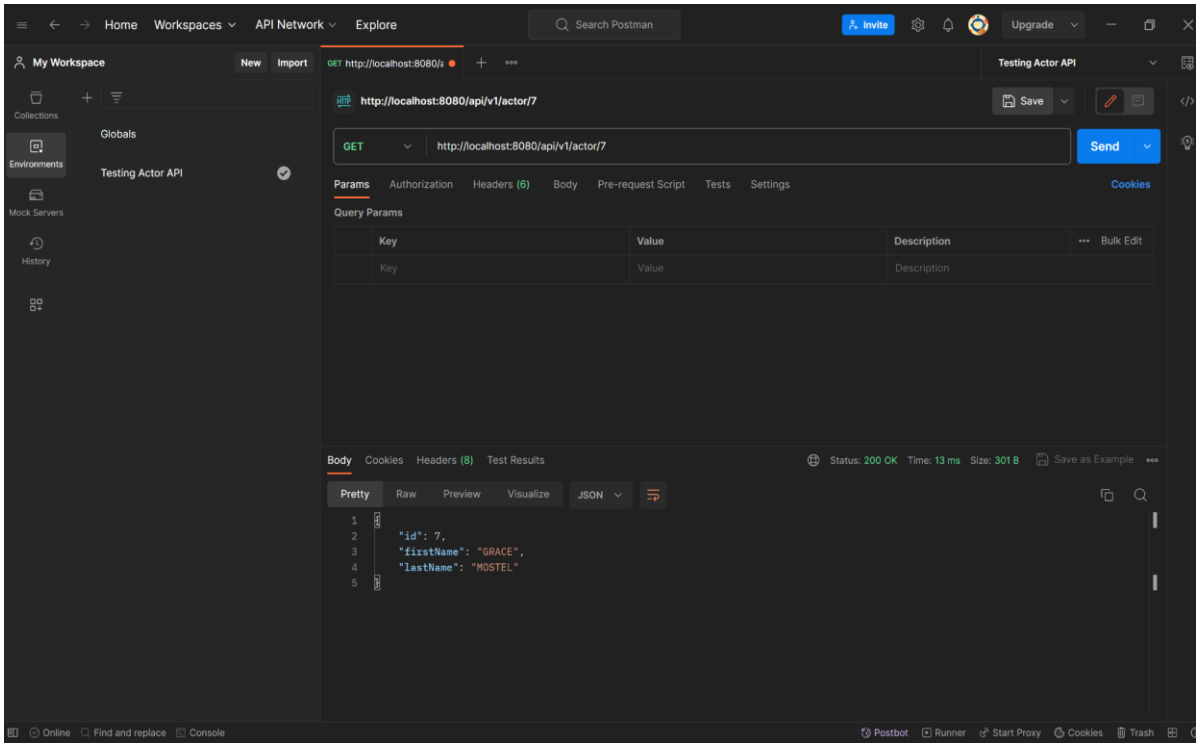
Kiểm tra bộ API đã cài đặt bằng Postman:

- Tải Postman, tạo Enviromenst mới, đổi tên thành Testing Actor API.
- Chạy Spring Boot App trên port đã cấu hình (8080)
- Sử dụng Postman để kiểm ra các API đã cài đặt.

- Xem danh sách tât cả các Actor: Chọn GET method + http://localhost:8080/api/v1/actor/all

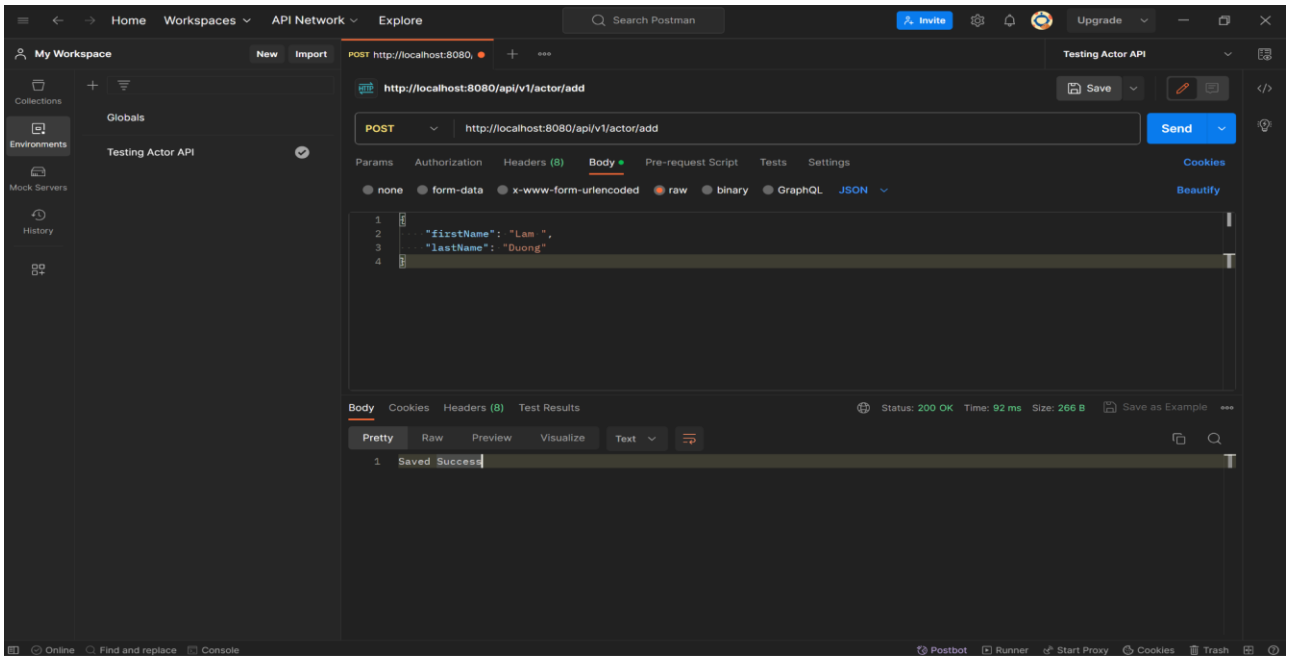


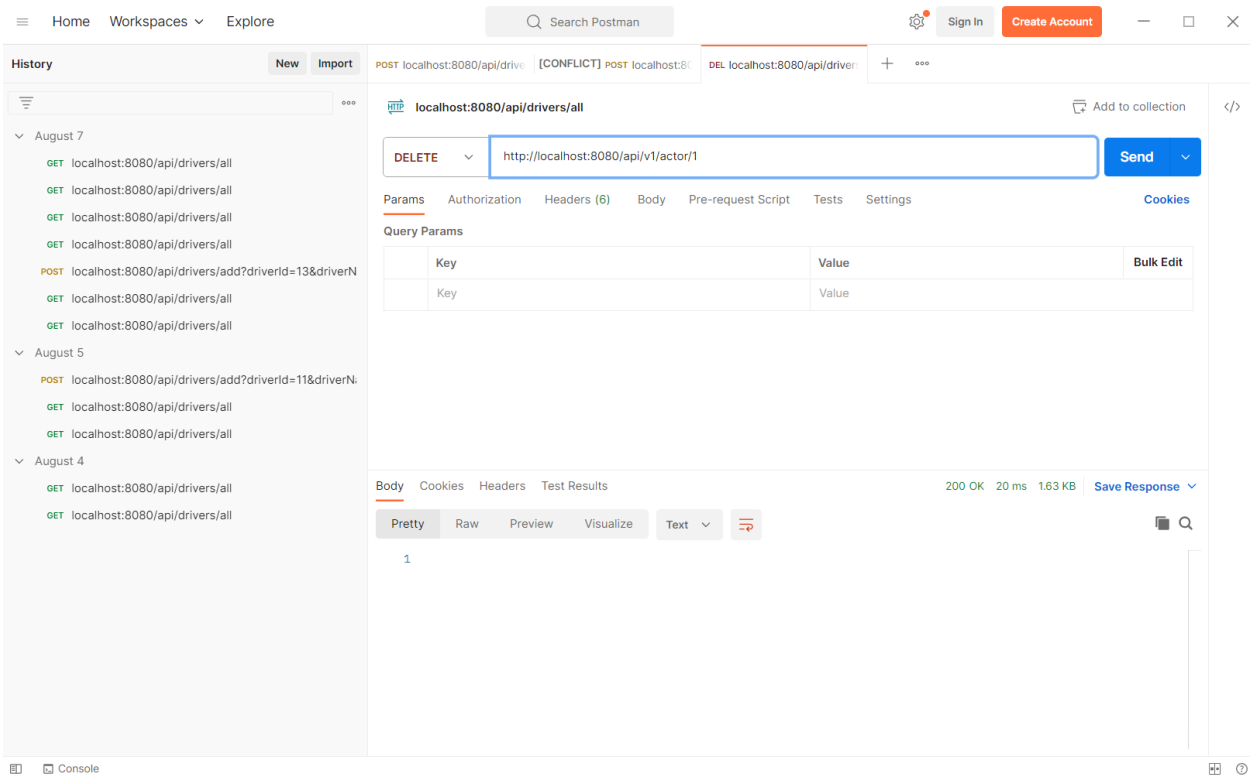
- Xem chi tiết thông tin một Actor theo id: Chọn GET method + http://localhost:8080/api/v1/actor



- Thêm một 'Actor':

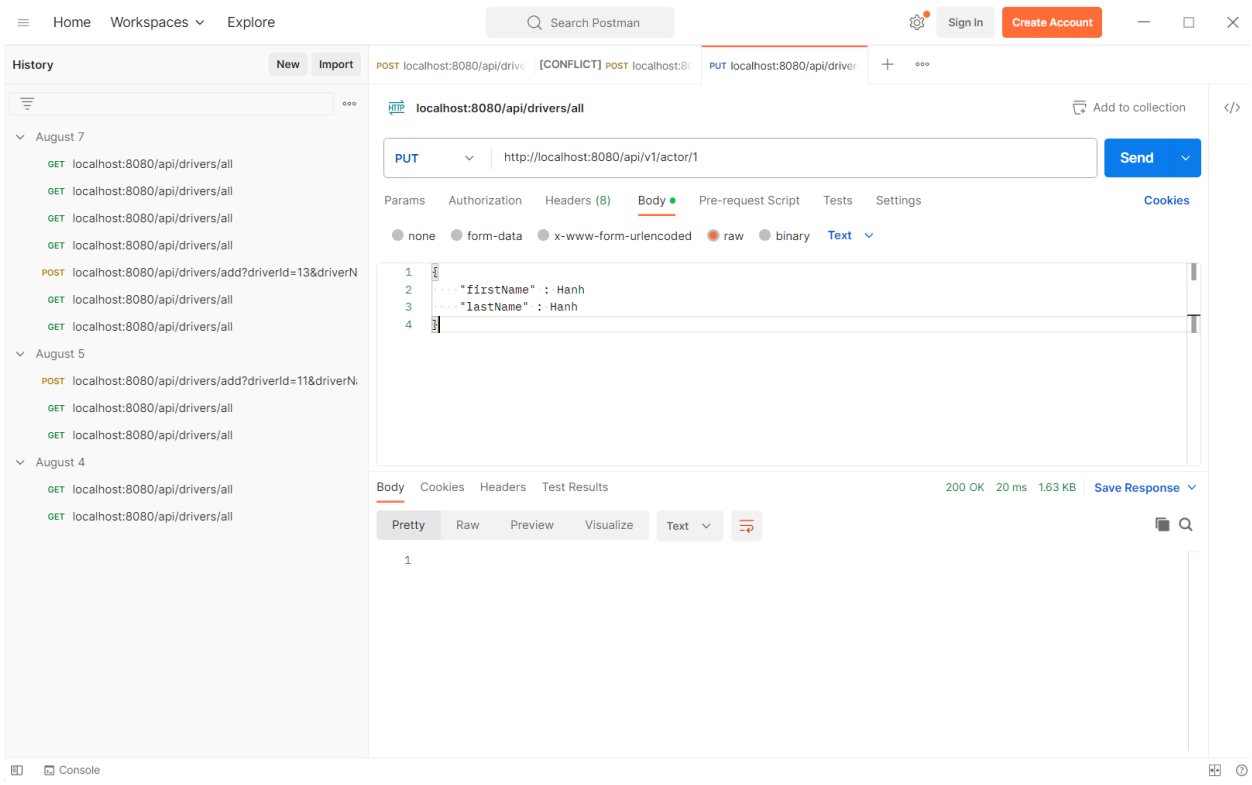
Chọn POST method + http://localhost:8080/api/v1/actor/add
Ở phần Body, chọn raw + JSON, thêm thông tin của Actor
Nếu thêm thành công trả về "Saved Success"
Nếu thông tin Actor đã tồn tại trả về "Actor already exists"
Nếu thông tin không thoả điều kiện (null, empty) trả về "Invalid Input"





- Xóa 1 Actor theo id :

- Chọn DELETE Method + http://localhost:8080/api/v1/actor/{id}
- Nếu id không hợp lệ trả về “Actor does not exists”
- Nếu xóa thành công trả về “Delete success”



- Update 1 Actor theo id:

- Chọn PUT method + http://localhost:8080/api/v1/actor/{id}
- Ở phần Body, chọn raw + JSON, thêm thông tin của Actor
- Nếu id không hợp lệ trả về “Actor does not exists”
- Nếu firstName hoặc lastName là chuỗi rỗng hoặc bằng null trả về “Invalid input”
- Nếu update thành công trả về “Update Success”