Great! Since it's working now, here's a **well-structured note** that you can keep for future reference, explaining the **problem, reason, and how you fixed it**:

Problem Faced

- When running the Spring Boot application, the API endpoint `POST /api/cities/insert` returned **404 (Not Found)**.
- Even though the controller and service were implemented correctly, Spring Boot could not detect and create the beans for the controller and service classes.

Root Cause

- Spring Boot automatically scans for components (like `@Controller`, `@Service`, `@Repository`) **only in the package where the main application class is located and its subpackages**.
- In your project:
- Main class package: `com.booking_samp`
- Controller package: `com.booking_samp_controller`
- Service package: `com.booking_samp_service`
- Model package: `com.booking_samp_modal`
- Since these packages are **siblings** (not subpackages of `com.booking_samp`), Spring Boot did **not automatically scan** them.

What Was Happening

- Your controller and service beans were **not being created** by Spring.
- As a result, Spring could not map the API routes defined in your controller !’ leading to 404 errors.

Solution Implemented

Option Chosen : Explicit Component Scanning

- Instead of changing package structure, you **configured Spring Boot to scan all required packages manually** using `@ComponentScan`, `@EnableJpaRepositories`, and `@EntityScan`.

Updated Main Application Class :

```
@SpringBootApplication
@EnableJpaRepositories(basePackages= "com.booking_samp.repository")
@EntityScan(basePackages= "com.booking_samp_modal")
@ComponentScan(basePackages= {
    "com.booking_samp",
    "com.booking_samp_controller",
    "com.booking_samp_service",
    "com.booking_samp_modal"
})
public class MovieTicketSampApplication{
    public static void main(String[] args) {
        SpringApplication.run(MovieTicketSampApplication.class, args);
    }
}
```

- `@ComponentScan`: Ensures Spring scans controller, service, and model packages.
- `@EnableJpaRepositories`: Ensures repository interfaces are detected.
- `@EntityScan`: Ensures JPA entities (models) are detected.

How I Overcame the Problem

1. Identified that 404 was caused by **missing controller beans**.
2. Learned that **Spring scans only the package of the main class and its subpackages** by default.
3. Fixed it by **explicitly adding component scanning configuration**.
4. Verified that after restarting the application, the controller and service beans were registered correctly.
5. Tested API again and confirmed it worked successfully.

Alternative (Best Practice)

- A cleaner way is to **place all packages as subpackages of**

- A cleaner way is to **place all packages as subpackages** of the main package (`com.booking_samp`) like this:

```
com.booking_samp
 %%% controller
 %%% service
 %%% repository
 %%% modal
```

This would remove the need for manual scanning annotations.

Key Takeaways / Lessons Learned

- Spring Boot scans only the package where the main application resides and its subpackages.
- Always keep a **clean package hierarchy** to follow Spring Boot conventions.
- If packages are outside the main package hierarchy, **explicit scanning is required**.
- When facing 404 for valid endpoints, always check if the controller bean is being created by Spring.