**RESTful Web Services và Tích Hợp Cơ Sở Dữ Liệu**

1. **Xây dựng RESTful API với Spring Boot**
2. **RESTful API là gì?**

- **RESTful API:** tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các tài nguyên.

- **API** (**A**pplication **P**rogramming **I**nterface): tập các quy tắc và cơ chế mà một ứng dụng hay một thành phần tương tác với một ứng dụng hay thành phần khác. - **REST** (**RE**presentational **S**tate **T**ransfer): một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API.

- **REST** quy định cách sử dụng các giao thức HTTP và cách định dạng URL cho ứng dụng để quản lý tài nguyên.

1. **Giao thức HTTP reqquest**

- REST hoạt động chủ yếu dựa trên giao thức HTTP.

- Giao thức HTTP reqquest: tập các phương thức được định nghĩa để thực hiện một số hành động dựa trên các tài nguyên mà client gửi cho web server.

- Cấu trúc HTTP request:

* Request line: gồm method (GET, POST,...), path (URL), và HTTP version.
* Headers: các thông tin bổ sung như cookie, thông tin xác thực,...
* Message body: máy chủ dùng message body để cung cấp những thông tin cần thiết cho máy khách. Gồm các dòng yêu cầu, thông tin, tiêu đề, dòng trống, nội dung.

1. **GET**

- Cho phép client gửi yêu cầu thông qua URL lên server.

- Không có request body.

- Một số đặc điểm:

* Giới hạn độ dài của các giá trị là 255 kí tự.
* Chỉ hỗ trợ các dữ liệu kiểu String.
* Có thể lưu vào bộ nhớ cache.
* Các tham số truyền vào được lưu trữ trong lịch sử trình duyệt.
* Có thể được bookmark do được lưu trong lịch sử trình duyệt.

1. **POST**

- Là phương thức gửi dữ liệu đến server để có thể thêm mới hoặc cập nhật dữ liệu.

- Thông tin cần thêm hoặc cập nhật nằm trong phần body request.

- Một số đặc điểm:

* Dữ liệu cần thêm hoặc cập nhật không được hiển thị trong URL.
* Dữ liệu không được lưu trong lịch sử trình duyệt và không hạn chế về độ dài.
* Hỗ trợ nhiều kiểu dữ liệu.

1. **PUT**

- Hoạt động tương tự phương thức Post nhưng chỉ được sử dụng để cập nhật dữ liệu.

- Cần phải sửa toàn bộ dữ liệu của một đối tượng khi sử dụng.

- Dữ liệu sẽ được thêm mới nếu đối tượng được yêu cầu sửa chưa tồn tại.

1. **PATCH**

- Tượng tự như Post và Put, nhưng Patch được sử dụng khi phải cập nhật một phần dữ liệu của đối tượng.

- Không thể hoạt động nếu đối tượng được yêu cầu chưa tồn tại.

1. **DELETE**

- Xóa các dữ liệu của server về tài nguyên thông qua URI.

- Không có body request.

1. **HEAD**

- Giống với GET, tuy nhiên không có response body.

- Phù hợp khi sử dụng để kiểm tra tình trạng API.

1. **Lớp generic: ResponseEntity và Optional**
2. **ResponseEntity**

- Một class trong Spring Framework, đại diện cho toàn bộ phản hồi HTTP, gồm:

* **HTTP Status Code**: Mã trạng thái HTTP.
* **HTTP Headers**: Các thông tin header.
* **Body**: Nội dung phản hồi có thể là bất kỳ đối tượng nào.

- Được dùng nhằm kiểm soát chi tiết phản hồi HTTP, tăng tính linh hoạt và khả năng phản hồi.

- Một số phương thức:

* **ok(T body)**: Trả về phản hồi với mã trạng thái **200 OK**
* **status(HttpStatus status)**: Chỉ định mã trạng thái tùy chỉnh cho phản hồi.
* **noContent()**: Phản hồi với mã trạng thái **204 No Content** (xử lý yêu cầu thành công và không có nội dung trả về).
* **notFound()**: Phản hồi với mã trạng thái **404 Not Found** (không tìm thấy tài nguyên được yêu cầu).
* **badRequest()**: Phản hồi với mã trạng thái **400 Bad Request** (yêu cầu từ client không hợp lệ).
* **created(URI location)**: Phản hồi với mã trạng thái **201 Created** (tạo tài nguyên thành công) và cung cấp địa chỉ tài nguyên mới được tạo.
* **headers(HttpHeaders headers)**: Thêm các header tùy chỉnh vào phản hồi.
* **body(T body)**: Thiết lập body cho phản hồi.
* **build()**: Tạo ResponseEntity mà không có body.

- Ví dụ:

1. **Optional**

- Dùng để kiểm tra một biến có tồn tại giá trị không.

- Giúp hạn chế kiểm tra null và tránh lỗi **NullPointerException** lúc runtime.

- Một số phương thức:

* Tạo Optional: of(), ofNullable(), empty()
* Truy xuất giá trị: get(), orElse(), orElseGet(), orElseThrow()
* Xử lý luồng: stream()

- Có thể kết hợp với lớp **ResponseEntity** để xử lý HTTP request và HTTP response.