RESTful API

CYBERSOFT.EDU.VN

Nội dung

- **Spring Sercurity**
  - ✓ Sercurity là gì?
  - ✓ Sercurity Context.
  - ✓ Đối tượng UserDetails.
  - ✓ Lớp UserDetailsService.
  - ✓ Các bước cấu hinhg Security.

API là gì?

- **API** (Application Programming Interface) là một tập các quy tắc, cơ chế để một ứng dụng sẽ tương tác với một ứng dụng khác hay dễ hiểu hơn API là một phần mềm trung gian cho phép 2 ứng dụng có thể giao tiếp với nhau.
- **API** có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.
- **API** có khá nhiều mục đích nhưng mục đích chính của API là cung cấp khả năng truy xuất đến một hàm hay tệp hay dùng.
- ***Ví dụ**: Các lập trình viên của facebook sẽ đưa ra các thư viện có chứa các hàm post, like, share… để cho các lập trình viên khác khai thác, đó chính là các **API.***

RESTful là gì?

- **REST** (Representtational State Transfer) lần đầu tiên được **giới thiệu vào năm 2000** trong luận văn tiến sĩ của Roy Thomas Fielding (đồng sáng lập giao thức HTTP).
- **REST** là một **bộ quy tắc để tạo ra một ứng dụng Web Service**, chỉ cần đảm bảo những điều đó hệ thống của bạn có thể được gọi là **RESTful**.
- **REST** tuân thủ **4 nguyên tắc thiết kế cơ bản** sau:
  - ✓Sử dụng các phương thức HTTP một cách rõ ràng.
  - ✓Phi trạng thái.
  - ✓Hiển thị cấu trúc thư mục như các Urls.
  - ✓Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai.

RESTful

❑ Sử dụng các phương thức HTTP một cách rõ ràng

❖ **REST** đặt ra một **quy tắc đòi hỏi lập trình viên xác định rõ ý định của mình thông qua các phương thức của HTTP**.

- ✓ Để tạo một tài nguyên trên máy chủ, bạn cần sử dụng phương thức POST.
- ✓ Để truy xuất một tài nguyên, sử dụng GET.
- ✓ Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng PUT.
- ✓ Để huỷ bỏ hoặc xoá một tài nguyên, sử dụng DELETE.

❑ Phi trạng thái (Stateless)

- ✓ REST là phi trạng thái (stateless), nó **không lưu giữ thông tin của client, điều đó có nghĩa là REST không quản lý phiên làm việc (Session)**.

RESTful

❑ Hiển thị cấu trúc thư mục như các Urls

✓ Các địa chỉ **REST service** cần phải thật **trực quan đến mức người dùng dễ đoán**.

❖ Một vài **nguyên tắc lưu ý khi cấu trúc địa chỉ của RESTful**:

- ✓ Giấu các đuôi tài liệu mở rộng của bản gốc trong máy chủ (.jsp, .php,...).
- ✓ Để mọi thứ là chữ thường.
- ✓ Thay thế các khoảng trống bằng gạch chân hoặc hoặc gạch nối (một trong hai loại).
- ✓ Thay vì sử dụng mã (404 Not Found) khi yêu cầu địa chỉ cho một phần đường dẫn, luôn luôn cung cấp một trang mặc định hoặc tài nguyên như một phản hồi.

RESTful

❑ **Bốn phương thức chính của HTTP**:

✓ **GET**: Dùng khi lấy dữ liệu – Mã trả về có thể là **200** (OK) + **404** (NOT FOUND) + **400** (BAD REQUEST).

✓ **POST**: Dùng khi tạo mới dữ kiệu – Mã trả về thường là **201** (CREATED).

✓ **PUT**: Dùng khi cập nhật dữ liệu đã tồn tại – Mã code trả về thường là **200** (OK).

✓ **DELETE**: Dùng khi xóa dữ liệu đã tồn tại – Mã code trả về thường là **200** (OK).

Http Status

❑ **Http** định nghĩa một số mã phản hồi cơ bản:

✓ **200** - **OK** : Trả về thành công cho những phương thức GET, PUT hoặc DELETE.

✓ **201** - **Created** : Trả về khi một Resouce vừa được tạo thành công.

✓ **204** - **No Content** : Trả về khi Resource xoá thành công.

✓ **304** - **Not Modified** : Client có thể sử dụng dữ liệu cache.

✓ **400** - **Bad Request** : Request không hợp lệ

✓ **401** - **Unauthorized** : Request cần có auth.

✓ **403** - **Forbidden** : bị từ chối không cho phép.

Http Status

- ✓ **404** - **Not Found** : Không tìm thấy resource từ URI
- ✓ **405** - **Method Not Allowed** : Phương thức không cho phép với user hiện tại.
- ✓ **410** - Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.
- ✓ **415** - **Unsupported Media Type** : Không hỗ trợ kiểu Resource này.
- ✓ **422** - **Unprocessable Entity** : Dữ liệu không được xác thực
- ✓ **429** - **Too Many Requests** : Request bị từ chối do bị giới hạn

JSON

- **JSON** (JavaScript Object Notation) là một **định dạng để lưu trữ và vận chuyển dữ liệu**.
- Định dạng JSON có nguồn gốc từ cú pháp đối tượng của Javascript vì vậy nó thừa kế sự **đơn giản và hoàn toàn dựa trên văn bản**.
- **Ưu điểm của JSON**:
  - ✓JSON là một định dạng để trao đổi dữ liệu gọn nhẹ (Lightweight).
  - ✓Dữ liệu JSON tự mô tả chính nó, vì vậy nó dễ hiểu cho tất cả mọi người.
  - ✓JSON là một ngôn ngữ độc lập, và là một văn bản. Bạn có thể sử dụng một ngôn ngữ bất kỳ để đọc hoặc tạo ra dữ liệu JSON.
  - ✓Hầu hết các ngôn ngữ lập trình đều có thư viện đọc và ghi dữ liệu JSON.

Cú pháp JSON

- Cú pháp của JSON rất đơn giản là mỗi thông tin dữ liệu sẽ có 2 phần đó là key và value.
- ✓ Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}.
- ✓ Các key, value của JSON bắt buộc phải đặt trong dấu nháy kép "".
- ✓ Nếu có nhiều dữ liệu (nhiều cặp key, value) thì ta dùng dấu phẩy (,) để ngăn cách.
- ✓ Các key của JSON bạn nên đặt chữ cái không dấu hoặc số, dấu _ và không có khoảng trắng., ký tự đầu tiên không nên đặt là số.

```
{
  "name" : "Amazon",
  "ceo" : "Jeff Bezos",
  "employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
  ]
}
```

Thư viện Jackson

- **Spring** cung cấp thư viện **JACKSON** hỗ trợ chuyển đổi các kiểu dữ liệu khác thành kiểu dữ liệu JSON.

```
<!-- jackson-core -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.9.8</version>
</dependency>
<!-- jackson-databind -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.8</version>
</dependency>
```

Thư viện Jackson

```java
Student student = new Student();
student.setName("Cybersoft");
student.setAge(30);
```

Chuyển Object thành JSON

```java
ObjectMapper om = new ObjectMapper();
String json = om.writeValueAsString(student);
System.out.println(json);
```

Chuyển JSON thành Object

```java
ObjectMapper om = new ObjectMapper();
Student obj = om.readValue(json, Student.class);
System.out.println(json);
```

ResponseEntity

- **ResponseEntity** là đối tượng được Spring cung cấp để đóng gói toàn bộ phản hồi HTTP: mã trạng thái, tiêu đề và nội dung. Do đó, chúng ta có thể sử dụng nó để định cấu hình đầy đủ phản hồi HTTP.

```
@RequestMapping(value = "/products", method = RequestMethod.GET)
public ResponseEntity<List<Product>> findAllProduct() {
    List<Product> products = productService.findAllProduct();
    if (products.isEmpty()) {
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }
    return new ResponseEntity<>(products, HttpStatus.OK);
}
```

Phương thức GetAll

```java
@GetMapping(value = "/products")
public ResponseEntity<List<Product>> get() {
    List<Product> products = productService.findAll();
    if (products.isEmpty()) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
    return new ResponseEntity<>(products, HttpStatus.OK);
}
```

Phương thức GetById

```java
@GetMapping(value = "/products/{id}")
public ResponseEntity<Product> get(@PathVariable("id") Integer id) {

    Product product = productService.findById(id);
    if (product == null) {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
    return new ResponseEntity<>(product, HttpStatus.OK);
}
```

Phương thức Insert

```
@PostMapping(value = "/products/post")
@RequestBody
public ResponseEntity<Product> post(
    @Valid @ModelAttribute("product") Product product,
    BindingResult errors) {
    if(errors.hasError()){
        return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
    }
    productService.save(product);
    return new ResponseEntity<>(product, HttpStatus.CREATED);
}
```

Phương thức Update

```
@PutMapping(value = "/products/put")
@RequestBody
public ResponseEntity<Product> put(
        @Valid @ModelAttribute("product") Product product,
        BindingResult errors) {
    if(errors.hasError()){
        return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
    }
    productService.update(product);
    return new ResponseEntity<>(HttpStatus.OK);
}
```

Phương thức Delete

```java
@DeleteMapping(value = "/products/{id}")
public ResponseEntity<Product> delete(@PathVariable("id") Integer id) {
    productService.removeById(id);
    return new ResponseEntity<>(HttpStatus.OK);
}
```