**RestTemplate – Access Restful Service**

Clone new api sample

<https://github.com/java-spring-training/product-management-api/tree/product-management-api>

Trong Micro Service, 2 service tương tác với nhau (Web app -> API hoặc API -> API ) thông qua HTTP protocol. Java cung cấp các phương pháp phổ biến hay được sử dụng là: HttpClient, HttpUrlConnection.

Spring Framework cung cấp 1 đối tượng để xử lý vấn đề này đơn giản hơn, đó là RestTemplate.

RestTemplate quản lý các HTTP Connection, và nó đóng vai trò như một phương tiện trung gian kết nối giữa các RESTful Web Service, giúp các service có thể giao tiếp với nhau dễ dàng

Một RESTful API sẽ được access thông qua một trong các Http method sau: GET, POST, PUT, DELETE, PATCH

1. **RestTemplate – GET**

**RestTemplate hỗ trợ 3 method để xử lý Http Method GET : getForObject(), exchange(), Basic Authentication**

* 1. ***Get - getForObject()***

kết quả RestTemplate trả về chỉ là Object Response Data, không bao gồm các thông tin chứa trong ResponseEntity như là: ResponseBody, HttpStatus, ...

**getForObject() không thể nhận kết quả response là List data -> do đó chỉ có thể wrap tất cả thông tin response vào Object**

ProductRepositoryImpl.java

|  |
| --- |
| @Repository  public class ProductRepositoryImpl implements ProductRepository {  private RestTemplate restTemplate;  private ProductManagementSearchResponseFactory factory;  @Autowired  public ProductRepositoryImpl(final RestTemplate restTemplate,  final ProductManagementSearchResponseFactory factory) {  this.restTemplate = restTemplate;  this.factory = factory;  }  @Override  public List<Product> findAll(final Product productRequest) {  UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();  uriBuilder.scheme("http");  uriBuilder.host("localhost");  uriBuilder.port(8081);  uriBuilder.path("product/search");  uriBuilder.queryParam("name", "car");  uriBuilder.queryParam("price", 100);  URI uri = uriBuilder.build().toUri();  ProductManagementSearchResponse result = restTemplate.getForObject(uri, ProductManagementSearchResponse.class);  // Factory List<ProductManagementInfo> -> List<Product>  return factory.createProducts(result);  }  } |

* Step 1: Khởi tạo default instant của RestTemplate
* Step 2: chuẩn bị Url cần access đến
  + Ở đây RestTemplate có thể nhận URI, hoặc Chuỗi Url
  + Có thể truyền trực tiếp full url string vào: **http://localhost:8081/product/search?name=car&price=100**
* Step 3: tạo Object để nhận kết quả response
  + Có thể nhận bằng 1 Object
  + Có thể nhận bằng String: thay thế ProductManagementSearchResponse -> String

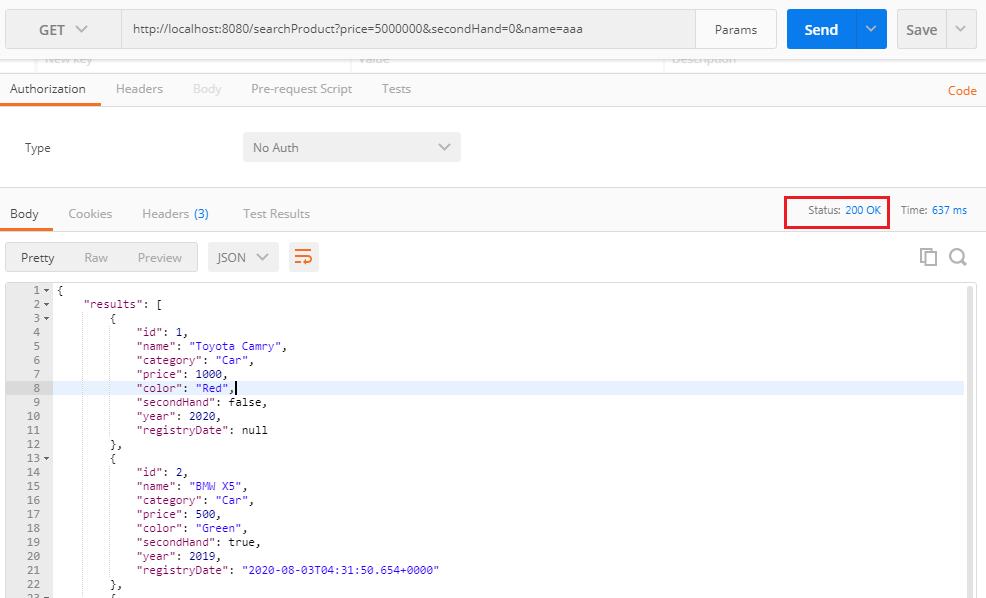
ProductManagementSearchResponse.java

|  |
| --- |
| @AllArgsConstructor  @NoArgsConstructor  @Data  public class ProductManagementSearchResponse {  private List<ProductManagementInfo> results;  @AllArgsConstructor  @NoArgsConstructor  @Data  public static class ProductManagementInfo {  private int id;  private String name;  private String category;  private int price;  private String color;  private boolean secondHand;  private int year;  private Date registryDate;  }  } |

ProductSearchResponseFactory.java

|  |
| --- |
| @Component  public class ProductSearchResponseFactory {  public ProductSearchResponse createProductSearchResponse(List<Product> products) {  List<ProductInfo> productInfos = products.stream().map(this::createProductInfo).collect(Collectors.toList());  return new ProductSearchResponse(productInfos);  }  private ProductInfo createProductInfo(Product product) {  return new ProductSearchResponse.ProductInfo(  product.getId(),  product.getName(),  product.getCategory(),  product.getPrice(),  product.getColor(),  product.isSecondHand(),  product.getYear(),  product.getRegistryDate()  );  }  } |

Đến đây có thể run để xem kết quả response trong trường hợp Normal



Nhưng trường hợp khi ProductManagement api trả về Error response (VD: Http status 400) thì Spring sẽ không Parse ErrorRessponse vào ProductSearchResponse -> Sẽ xảy ra Exception

1 cách để xử lý vấn đề này là check Http Status của RestTemplate. Nhưng vì là getForObject() nên kết quả nhận được chỉ là Object Response mà thôi. Khi đó Spring sẽ throw HttpClientErrorException

Có 2 cách xử lý vấn đề này:

* Catch HttpClientErrorException
* Dùng phương pháp exchange()
  1. ***Get – exchange()***

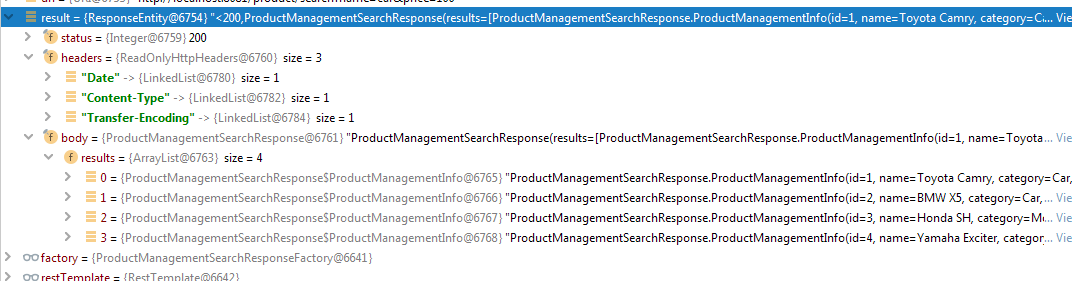
Kết quả RestTemplate trả về chính là ResponseEntity bao gồm:

* Response.body: chứa Object response
* HttpStatus
* Header

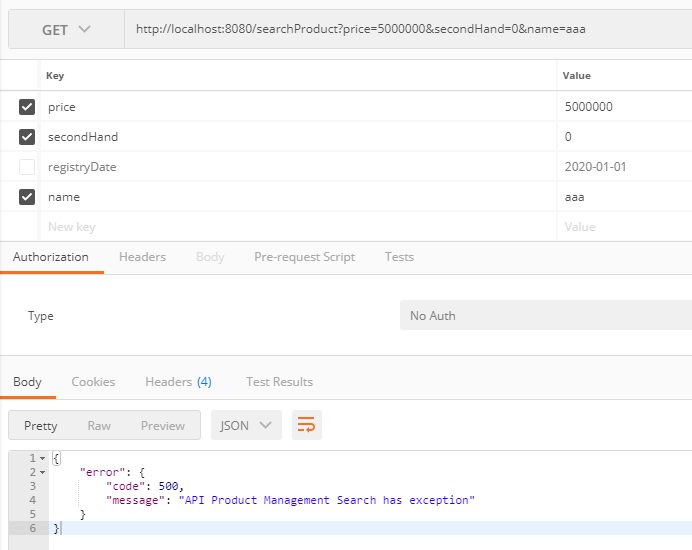
ProductRepositoryImpl.java

|  |
| --- |
| @Override  public List<Product> findAll(final Product productRequest) throws ProductSearchException {  UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();  uriBuilder.scheme("http");  uriBuilder.host("localhost");  uriBuilder.port(8081);  uriBuilder.path("product/search");  uriBuilder.queryParam("name", "car");  uriBuilder.queryParam("price", 10);  URI uri = uriBuilder.build().toUri();  ResponseEntity<ProductManagementSearchResponse> result;  try {  result = restTemplate.exchange(uri, HttpMethod.GET, null, ProductManagementSearchResponse.class);  } catch (HttpClientErrorException ex) {  throw new ProductSearchException("API Product Management Search has exception");  }  // Check http status  if (ObjectUtils.isEmpty(result)  || result.getStatusCode() != HttpStatus.OK  || ObjectUtils.isEmpty(result.getBody())) {  throw new ProductSearchException("API Product Management Search has exception");  }    // Factory List<ProductManagementInfo> -> List<Product>  return factory.createProducts(result.getBody());  } |

Kết quả debug ResponseEntity



Trường hợp RestTemplate nhận được ErrorResponse



1. **RestTemplate – POST**
   1. ***Post – postForObject()***

**Tương tự như getForObject(),postForObject() thực hiện gọi 1 api co method POST**

kết quả RestTemplate trả về là 1 Object vừa được submit lên server, không bao gồm các thông tin chứa trong ResponseEntity như là: ResponseBody, HttpStatus, ...

**ProductRepositoryImpl.java**

|  |
| --- |
| **@Override**  **public void addProduct(Product productRequest) throws ProductSearchException {**  **UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();**  **uriBuilder.scheme("http");**  **uriBuilder.host("localhost");**  **uriBuilder.port(8081);**  **uriBuilder.path("product/add");**  **URI uri = uriBuilder.build().toUri();**  **ProductManagementAddRequest request = new ProductManagementAddRequest("Car", 100);**  **HttpEntity<ProductManagementAddRequest> httpEntity = new HttpEntity<>(request, new HttpHeaders());**  **ProductManagementAddRequest productManagementResponse =**  **restTemplate.postForObject(uri, httpEntity, ProductManagementAddRequest.class);**  **// do something like: Check object productManagementAddRequest ...**  **if(ObjectUtils.isEmpty(productManagementResponse) || StringUtils.isEmpty(productManagementResponse.getName())) {**  **throw new ProductSearchException("API Product Management Add has exception");**  **}**  **}** |

ProductManagementAddRequest.java

|  |
| --- |
| @NoArgsConstructor  @AllArgsConstructor  @Data  public class ProductManagementAddRequest {  private String name;  private int price;  } |

* Vì là method POST nên parameter được gửi vào RequestBody
* Do đó cần phải define HttpEntity để chứa RequestBody
* Ngoài ra HttpEntity còn có thể chứa thong tin Header
  1. ***Post – postForEntity()***

Kết quả RestTemplate trả về chính là ResponseEntity bao gồm:

* Response.body: chứa Object vừa được submit lên server
* HttpStatus
* Header

**ProductRepositoryImpl.java**

|  |
| --- |
| @Override  public void addProduct(Product productRequest) throws ProductSearchException {  UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();  uriBuilder.scheme("http");  uriBuilder.host("localhost");  uriBuilder.port(8081);  uriBuilder.path("product/add");  URI uri = uriBuilder.build().toUri();  ProductManagementAddRequest request = new ProductManagementAddRequest("Car", 100);  HttpEntity<ProductManagementAddRequest> httpEntity = new HttpEntity<>(request, new HttpHeaders());  ResponseEntity<ProductManagementAddRequest> response;  try {  response = restTemplate.postForEntity(uri, httpEntity, ProductManagementAddRequest.class);  } catch (HttpClientErrorException ex) {  throw new ProductSearchException("API Product Management Add has exception");  }    // Check http status & Object Response  if (ObjectUtils.isEmpty(response)  || response.getStatusCode() != HttpStatus.OK  || ObjectUtils.isEmpty(response.getBody())  || ObjectUtils.isEmpty(response.getBody().getName())) {  throw new ProductSearchException("API Product Management Add has exception");  }  } |