**RestTemplate – Access Restful Service**

**Contents**

[**1.** **RestTemplate – GET** 1](#_Toc47387589)

[**2.** **RestTemplate – POST** 7](#_Toc47387590)

[**3.** **RestTemplate – PUT** 10](#_Toc47387591)

[**4.** **RestTemplate – DELETE** 12](#_Toc47387592)

[**5.** **RestTemplateBuilder** 13](#_Toc47387593)

[**6.** **Access Https** 14](#_Toc47387594)

**Clone new Sample API**

<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

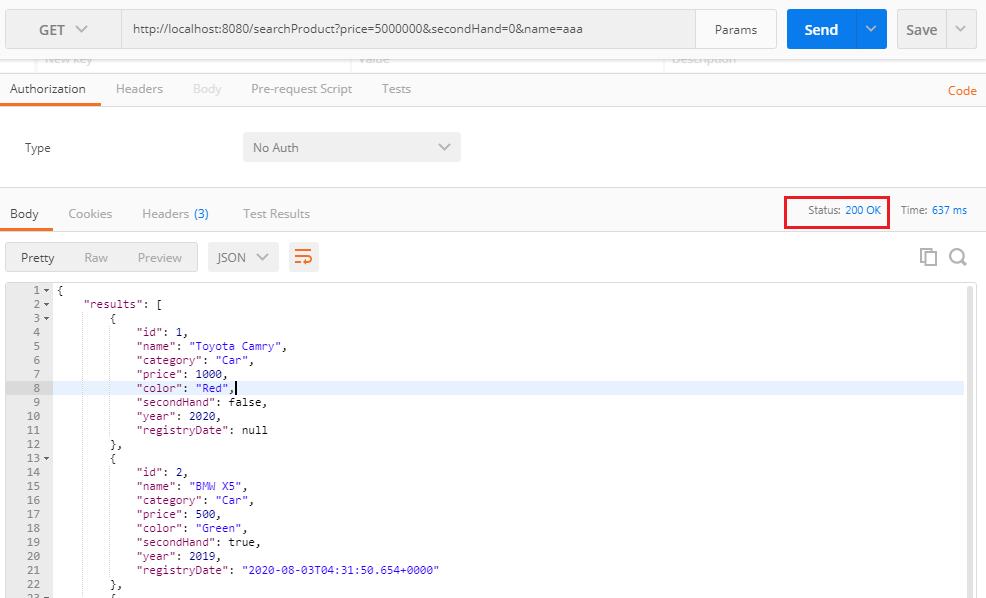
ProductManagementSearchResponse.java

|  |
| --- |
| @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 RestTemplate sẽ throw Exception

Có 2 step xử lý vấn đề này:

* Catch exception do RestClientResponseException throw ra
* Dùng phương pháp exchange()

**Note:** Các Exception mà RestTemplate có thể throw ra:

* HttpClientErrorException -> in case of HTTP status 4xx
* HttpServerErrorException -> in case of HTTP status 5xx
* UnknownHttpStatusCodeException -> in case of an unknown HTTP status
* Cả 3 loại exception trên đều kế thừa RestClientResponseException. Do đó nếu dự án ko yêu cầu phải xử lý cụ thể cho từng loại exception thì chỉ cần cache exception cha: **RestClientException**
  1. ***Get – exchange()***

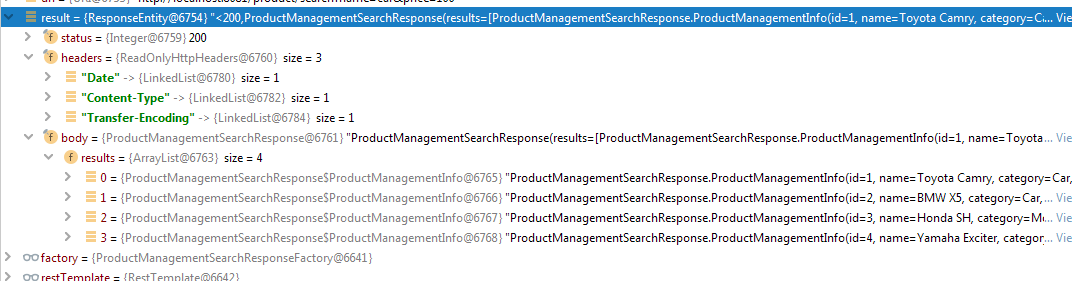
Kết quả RestTemplate trả về chính là ResponseEntity bao gồm:

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

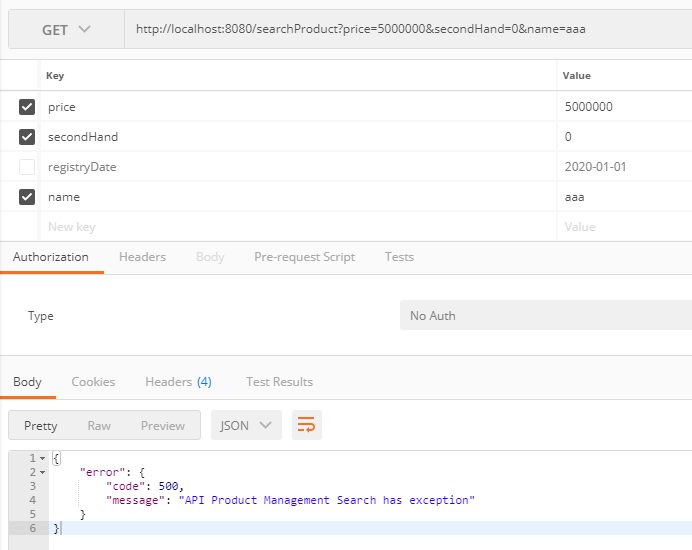
ProductRepositoryImpl.java

|  |
| --- |
| @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 có method POST**

kết quả RestTemplate trả về là 1 Object Response Data, không bao gồm các thông tin chứa trong ResponseEntity như là: ResponseBody, HttpStatus, ...

**ProductRepositoryImpl.java**

|  |
| --- |
| **@Override**  **public void addProduct(Product productRequest) throws ProductAddException {**  **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, null);**  **ProductManagementSuccessResponse response;**  **try {**  **response = restTemplate.postForObject(uri, httpEntity, ProductManagementSuccessResponse.class);**  **} catch (RestClientException ex) {**  **throw new ProductAddException("API Product Management Add has exception");**  **}**  **// do something: Check Response object ProductManagementSuccessResponse ...**  **if(ObjectUtils.isEmpty(response) || response.getCode() != HttpStatus.OK.value()) {**  **throw new ProductAddException("API Product Management Add has exception");**  **}**  **}** |

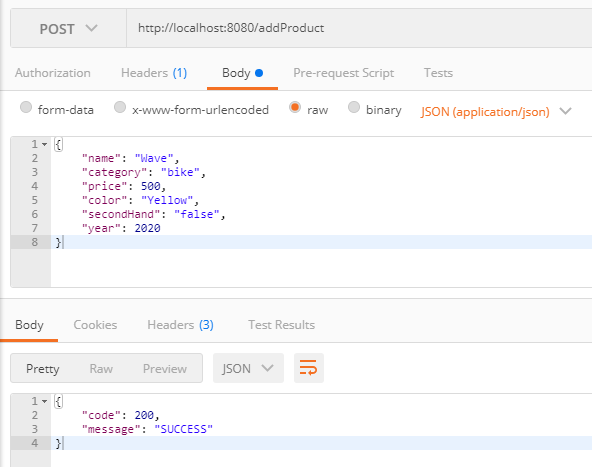
* Vì là method POST nên parameter được gửi vào RequestBody
* Do đó cần phải define HttpEntity để chứa RequestBody
* Ngoài ra HttpEntity còn có thể chứa thông tin Header

ProductManagementAddRequest.java

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

ProductManagementSuccessResponse.java

|  |
| --- |
| @AllArgsConstructor  @NoArgsConstructor  @Data  public class ProductManagementSuccessResponse {  private int code;  private String message;  } |



* **Kết luận:** Tuy các thức gửi request khác nhau, nhưng cách nhận response của **postForObject() và getForObject() là giống nhau**
  1. ***Post – postForEntity()***

Kết quả RestTemplate trả về chính là ResponseEntity bao gồm:

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

**ProductRepositoryImpl.java**

|  |
| --- |
| @Override  public void addProduct(Product productRequest) throws ProductAddException {  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, null);  ResponseEntity<ProductManagementSuccessResponse> response;  try {  response = restTemplate. postForEntity (uri, httpEntity, ProductManagementSuccessResponse.class);  } catch (RestClientException ex) {  throw new ProductAddException("API Product Management Add has exception");  }  // Check http status & Object Response  if (ObjectUtils.isEmpty(response)  || response.getStatusCode() != HttpStatus.OK  || ObjectUtils.isEmpty(response.getBody())  || response.getBody().getCode() != HttpStatus.OK.value()) {  throw new ProductAddException("API Product Management Add has exception");  }  } |

**Note**: Có thể dùng exchange() thay cho postForEntity(), 2 method này return kết quả giống nhau

1. **RestTemplate – PUT**

**RestTemplate PUT -> thường được dùng để call api update**

* 1. ***Put default - RestTemplate.put()***

Method put() có kiểu dữ liệu void() -> nên phương thức này chỉ submit data update lên server mà không nhận bất kì thông tin response nào

ProductRepositoryImpl.java

|  |
| --- |
| @Override  public void updateProduct(Product productRequest) throws ProductUpdateException {  UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();  uriBuilder.scheme("http");  uriBuilder.host("localhost");  uriBuilder.port(8081);  uriBuilder.path("product/update");  URI uri = uriBuilder.build().toUri();  ProductManagementUpdateRequest request =  new ProductManagementUpdateRequest(productRequest.getName(), productRequest.getPrice());  HttpEntity<ProductManagementUpdateRequest> httpEntity = new HttpEntity<>(request, null);  try {  restTemplate.put(uri, httpEntity);  } catch (RestClientException ex) {  throw new ProductUpdateException("API Product Management Update has exception");  }  } |

ProductManagementUpdateRequest.java

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

* 1. ***Put – exchange()***

Tương tự Post-exchange()

ProductRepositoryImpl.java

|  |
| --- |
| @Override  public void updateProduct(Product productRequest) throws ProductUpdateException {  UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();  uriBuilder.scheme("http");  uriBuilder.host("localhost");  uriBuilder.port(8081);  uriBuilder.path("product/update");  URI uri = uriBuilder.build().toUri();  ProductManagementUpdateRequest request =  new ProductManagementUpdateRequest(productRequest.getName(), productRequest.getPrice());  HttpEntity<ProductManagementUpdateRequest> httpEntity = new HttpEntity<>(request, null);  ResponseEntity<ProductManagementSuccessResponse> response;  try {  response = restTemplate.exchange(uri, HttpMethod.PUT, httpEntity, ProductManagementSuccessResponse.class);  } catch (RestClientException ex) {  throw new ProductUpdateException("API Product Management Update has exception");  }  // Check http status & Object Response  if (ObjectUtils.isEmpty(response)  || response.getStatusCode() != HttpStatus.OK  || ObjectUtils.isEmpty(response.getBody())  || response.getBody().getCode() != HttpStatus.OK.value()) {  throw new ProductUpdateException("API Product Management Update has exception");  }  } |

1. **RestTemplate – DELETE**

**RestTemplate DELETE -> thường được dùng để call api delete**

* 1. ***Delete default - RestTemplate.delete()***

Method delete() có kiểu dữ liệu void() -> nên phương thức này chỉ submit data delete lên server mà không nhận bất kì thông tin response nào

ProductRepositoryImpl.java

|  |
| --- |
| @Override  public void deleteProduct(int id) throws ProductDeleteException {  // URI parameters (path variable)  Map<String, Integer> urlParams = new HashMap<>();  urlParams.put("id", id);  UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();  uriBuilder.scheme("http");  uriBuilder.host("localhost");  uriBuilder.port(8081);  uriBuilder.path("product/delete/{id}");  URI uri = uriBuilder.buildAndExpand(urlParams).toUri();  try {  restTemplate.delete(uri);  } catch (RestClientException ex) {  throw new ProductDeleteException("API Product Management Delete has exception");  }  } |

* 1. ***Delete – exchange()***

Tương tự Post-exchange()

ProductRepositoryImpl.java

|  |
| --- |
| @Override  public void deleteProduct(int id) throws ProductDeleteException {  // URI parameters (path variable)  Map<String, Integer> urlParams = new HashMap<>();  urlParams.put("id", id);  UriComponentsBuilder uriBuilder = UriComponentsBuilder.newInstance();  uriBuilder.scheme("http");  uriBuilder.host("localhost");  uriBuilder.port(8081);  uriBuilder.path("product/delete/{id}");  URI uri = uriBuilder.buildAndExpand(urlParams).toUri();  ResponseEntity<ProductManagementSuccessResponse> response;  try {  response = restTemplate.exchange(uri, HttpMethod.DELETE, null, ProductManagementSuccessResponse.class);  } catch (RestClientException ex) {  throw new ProductDeleteException("API Product Management Delete has exception");  }  // Check http status & Object Response  if (ObjectUtils.isEmpty(response)  || response.getStatusCode() != HttpStatus.OK  || ObjectUtils.isEmpty(response.getBody())  || response.getBody().getCode() != HttpStatus.OK.value()) {  throw new ProductDeleteException("API Product Management Delete has exception");  }  } |

1. **RestTemplateBuilder**

**Thay vì sử dụng instance mặc định của RestTemplate, mình có thể tự config và tạo 1 RestTemplate thông qua RestTemplateBuilder**

RestTemplateConfig.java

|  |
| --- |
| @Configuration  public class RestTemplateConfig {  @Value("${api.timeout}")  private int timeout;  @Bean  public RestTemplate productRestTemplate(RestTemplateBuilder builder) {  return builder  .setConnectTimeout(Duration.ofMillis(timeout))  .setReadTimeout(Duration.ofMillis(timeout))  .build();  }  } |

Đoạn code trên tao 1 Bean với name: productRestTemplate

ProductRepositoryImpl.java

|  |
| --- |
| @Autowired  public ProductRepositoryImpl(@Qualifier("productRestTemplate") final RestTemplate restTemplate,  final ProductManagementSearchResponseFactory factory) {  this.restTemplate = restTemplate;  this.factory = factory;  } |

Để tìm hiểu thêm các phương thức của RestTemplateBuilder: <https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/web/client/RestTemplateBuilder.html>

1. **Access Https**

**RestTemplate có thể access đến Https, cách xử lý hoàn toàn giống access vào Http**

**VD: access vào page:** <https://vnexpress.net/>

|  |
| --- |
| **@Override**  **public List<Product> findAll(final Product productRequest) throws ProductSearchException {**  **String url = "https://vnexpress.net/";**  **ResponseEntity<String> response;**  **try {**  **response = restTemplate.exchange(url, HttpMethod.GET, null, String.class);**  **} catch (RestClientException ex) {**  **throw new ProductSearchException("API Product Management Search has exception");**  **}**  **// Check http status**  **if (ObjectUtils.isEmpty(response)**  **|| response.getStatusCode() != HttpStatus.OK**  **|| ObjectUtils.isEmpty(response.getBody())) {**  **throw new ProductSearchException("API Product Management Search has exception");**  **}**  **return null;**  **}** |

