Friday, August 12, 2022



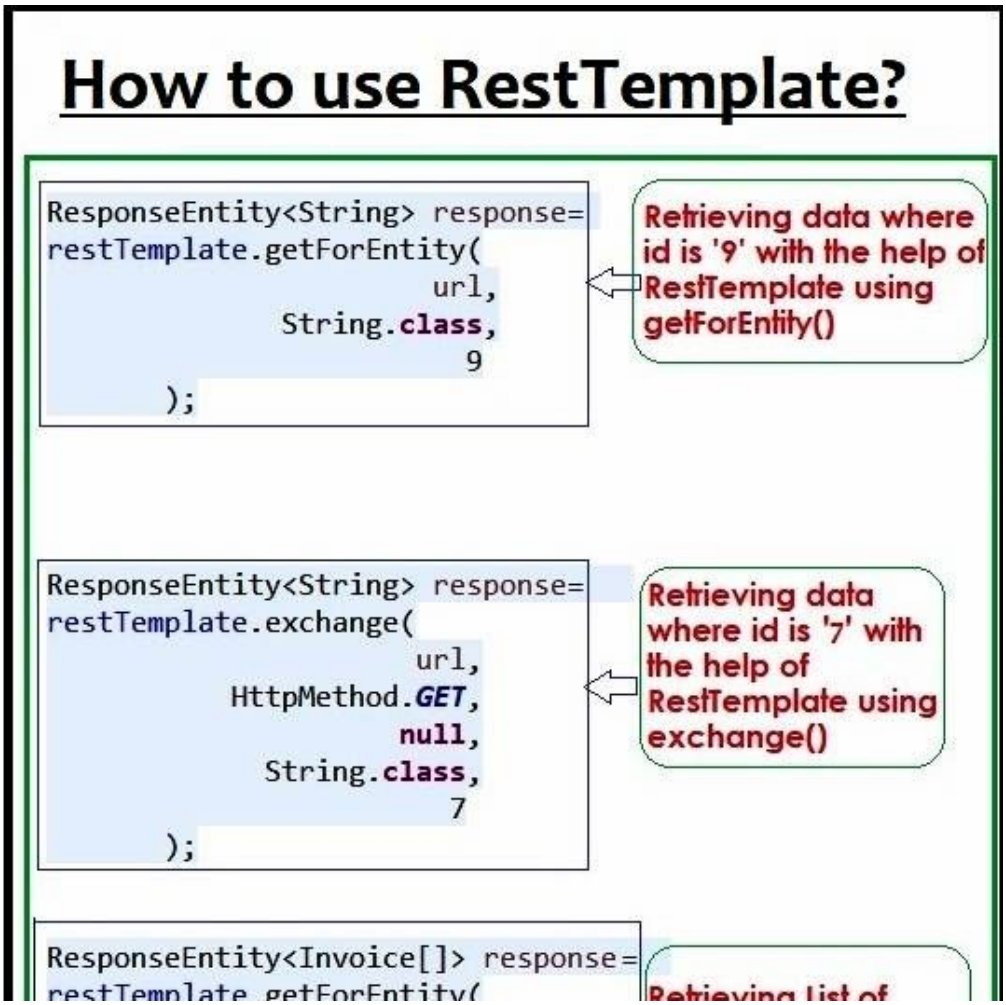
# Making Java easy to learn

Java Technology and Beyond



# How To Write REST Consumer API Using Spring Boot RestTemplate

java RestTemplate Spring Spring Boot *by devs5003 - October 30, 2020* 0




Apart from three well known layers i.e. Presentation Layer, Service Layer and Data Layer, we may also have an Integration Layer. This layer generally works with the web services concept and connects two different applications to exchange data between them. One application refers to producer/provider, whereas other refers


to consumers. However, we have already gone through the REST Producer API in another article where we had developed different operations to produce REST API. Now, we must have a question in mind 'How to write REST Consumer API using Spring Boot RestTemplate'. We had also discussed a bit about RestTemplate( The Spring REST Client).

In fact, RestTemplate provides us Consumer methods to consume services provided by producer applications. We can even have various non-java Consumer applications. However, our RestTemplate is written in Java only. Accordingly, let's start working on 'How to write REST Consumer API using Spring Boot RestTemplate'.

Table of Contents (Click on links below to navigate) [hide]

# What will you learn from this article ?

1) What is RestTemplate and what is it used for?
2) What is the difference between the various methods of RestTemplate?
3) When to use, which method of RestTemplate?
4) How is the exchange () method used in implementing Rest Consumer Web Services?
5) Also, Which format of data does the producer application provide and which format of data does the consumer application receive?
6) What are all parameters of RestTemplate method to develop the consumer application?
7) How to write REST Consumer API using Spring Boot RestTemplate?
8) Additionally, How to use slf4j Logger effectively while implementing RestTemplate?
9) Finally, How to test the developed application?

# What is Rest Template ?

In a nutshell, RestTemplate is a predefined class in Spring Boot REST project. Moreover  It helps in making HTTP calls to Producer application with all method types eg. GET, POST, PUT, DELETE etc. However Spring Boot framework doesn’t auto configure this class. It also supports JSON/XML to Object and Object to JSON/XML auto-conversion. Moreover, it requires Endpoint details from a producer application like IP, PORT, Paths, Method Type, Input data format and Output data format etc. Additionally, RestTemplate provides the

exchange() method to consume the web services for all HTTP methods. In fact, **RestTemplate** helps in making HTTP Rest Calls.

# What is difference between getForObject() and getForEntity() ?

Having knowledge of getforentity vs getforobject is important for us as we will be using these methods in our implementation.

**getForObject**(url, T.class) : It retrieves an entity using HTTP GET method on the given URL and returns T.  It doesn't return Status, Header params but only Response Body.
**getForEntity**(url, T.class) : It retrieves an entity by using HTTP GET method for the given URL and returns ResponseEntity<T>.

# What is difference between postForObject() and postForEntity() ?

**postForObject**(url, request, T.class) : It saves an entity using HTTP POST method on the given URL and returns T.  It doesn't return Status, Header params but only Response Body.
**postForEntity**(url, request, T.class) : It saves an entity by using HTTP POST method for the given URL and returns ResponseEntity<T>.

# What is exchange() method in RestTemplate used for ?

exchange() method supports making call to any Http method (GET/POST /PUT/DELETE/…..). Generally, it has below syntax.

**exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType, Object… uriVariables)**: ResponseEntity<T>

**url** : Producer application URL (RestController's mrthod path)
**HttpMethod** : is an enum to provide method type.

**HttpEntity** : Request Body + HttpHeaders (it can also be null)
**responseType** : Class type of response
**Object-var/args** : used for sending multiple pathVariables

# What is the format of data Consumer Application receives from Producer Application ?

If any RestController's method returns a non-String type i.e. class or collection type then Data is converted into JSON format and provided to Consumer Application. Consumer application reads that data as a String (JSON is also a String only).

# What all parameters are expected to write consumer methods in RestTemplate ?

In fact consumer method could be predictable on the already existing producer method. The producer method's url, return type, Http method type, path variables etc. will decide the structure of your consumer method. Generally we expect following things to pass as parameters to RestTemplate's methods.

1. URL of Producer webservice
2. Body of the request (in case of POST/PUT…)
3. Media Type like APPLICATION_JSON, APPLICATION_XML, APPLICATION_PDF etc. (in case of POST/PUT…)
4. Http Method type
5. Return type of producer method
6. Path Variables (If any)

# How to implement RestTemplate methods ?

Although we will take reference of REST API producer application that we developed in previous articles and call the already existing methods with the help of RestTemplate. Here, we are considering "Invoice" as our model class. Besides exchange() method, we will take one more alternate method to

develop the template operations. An alternate method is commented in below code. Afterwards, you can un-comment the same as per your requirement to test it accordingly. Let's implement the concept 'How to write REST Consumer API using Spring Boot RestTemplate' step by step.

## Save Invoice

saveInv()

```
private void saveInv() {
        // 1. Producer application URL
        String url = "http://localhost:8080/api/invoices";
        // Send JSON data as Body
        String body = "{\"name\":\"INV11\", \"amount\":234.11,
        // Http Header
        HttpHeaders headers = new HttpHeaders();
        //Set Content Type
        headers.setContentType(MediaType.APPLICATION_JSON);
        //requestEntity : Body+Header
        HttpEntity<String> request = new HttpEntity<String> (b
        // 2. make HTTP call and store Response (URL,ResponseT
    //  ResponseEntity<String> response =  restTemplate.postFo
        ResponseEntity<String> response =  restTemplate.exchan
        // 3. Print details(body,status..etc)
        logger.info("Response Body : {}", response.getBody());
        logger.info("Status code value : {}", response.getStat
        logger.info("Status code : {}", response.getStatusCode

    }
```

### Output:

Output

```
Response Body : Invoice '9' created
```

```
Status code value : 201
Status code : OK
```

# Get All Invoices

getAllInvoices()

```
private void getAllInvoices() {
        String url = "http://localhost:8080/api/invoices";
        ResponseEntity<Invoice[]> response = restTemplate.getF
    //  ResponseEntity<Invoice[]> response = restTemplate.exch
        Invoice[] invs = response.getBody();
        List<Invoice> list = Arrays.asList(invs);

        logger.info("Response Body : {}", list);
        logger.info("Status code value : {}", response.getStat
        logger.info("Status code : {}", response.getStatusCode
        logger.info("Headers {} :", response.getHeaders());
    }
```

## Output:

output

```
Response Body : [Invoice(id=1, name=Inv1, amount=135.0, finalA
Status code value : 200
Status code : CREATED
Headers [Content-Type:"application/json", Transfer-Encoding:"c
```

# Get One Invoice

getOneInvoice()

```
private void getOneInvoice() {
        String url = "http://localhost:8080/api/invoices/{id}"
    //  ResponseEntity<String> response= restTemplate.getForEn
        ResponseEntity<String> response= restTemplate.exchange
        logger.info("Response Body : {}", response.getBody());
        logger.info("Status code value : {}", response.getStat
        logger.info("Status code : {}",response.getStatusCode(
    }
```

### Output:

output

```
Response Body : {"id":9,"name":"INV11","amount":234.11,"finalA
Status code value : 200
Status code : OK
```

## Update Invoice

updateInvoice()

```
private void updateInvoice() {
        String url = "http://localhost:8080/api/invoices/{id}"
        String body = "{\"name\":\"INV13\",\"amount\":888}";
        // Request Header
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        // requestEntity = Body + header
        HttpEntity<String> requestEntity = new HttpEntity<Stri
    //  restTemplate.put(url, requestEntity, 7);
        ResponseEntity<String> response= restTemplate.exchange
        logger.info("Response Body : {}", response.getBody());
        logger.info("Status code value : {}", response.getStat
        logger.info("Status code : {}",response.getStatusCode(
```

```
            logger.info("Response Headers : {}", response.getHeade
        }
```

### Output:

output

```
Response Body : null
Status code value : 205
Status code : RESET_CONTENT
Response Headers : [Content-Length:"0", Date:"Thu, 29 Oct 2020
```

## Delete Invoice

deleteInvoice()

```
private void deleteInvoice() {
        String url = "http://localhost:8080/api/invoices/{id}"
    //  restTemplate.delete(url, 6);
        ResponseEntity<String> response= restTemplate.exchange
        logger.info("Response Body : {}", response.getBody());
        logger.info("Status code value : {}", response.getStat
        logger.info("Status code : {}",response.getStatusCode(
        logger.info("Response Headers : {}", response.getHeade
    }
```

### Output:

output

```
Response Body : Invoice '5' deleted
Status code value : 200
Status code : OK
```

```
Response Headers : [Content-Type:"text/plain;charset=UTF-8", C
```

# Complete Code

Let’s write the complete code for our topic ‘How to write REST Consumer API using Spring Boot RestTemplate’. We created one Spring Boot Starter Project named “SpringBootRestTemplate” with dependencies ‘Lombok’ and ‘Spring Web’. Still, if you are new to SpringBoot & Lombok, visit internal links as SpringBoot & Lombok accordingly. Furthermore, in the main class(SpringBootRestTemplateApplication.java) we wrote code of RestTemplate object creation and utilized it via auto-wiring. Besides this modification, we created two new classes Invoice.java(Model class) and a Runner class RestTemplateRunner.java(class having all methods). From the run() method of RestTemplateRunner.java uncomment the respective method required to test accordingly. In the end, Project structure will look like below screenshot.

SpringBootRestTemplateApplication.java

```
package com.dev.springboot.rest.template;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplic
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class SpringBootRestTemplateApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootRestTemplateApplicatio
    }

    @Bean
    public RestTemplate getRestTemplate() {
        return new RestTemplate();
    }
}
```

Invoice.java

```
package com.dev.springboot.rest.template.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Invoice {
```

```
    private Long id;
    private String name;
    private Double amount;
    private Double finalAmount;
    private String number;
    private String receivedDate;
    private String type;
    private String vendor;
    private String comments;

}
```

RestTemplateRunner.java

```
package com.dev.springboot.rest.template.runner;

import java.util.Arrays;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

import com.dev.springboot.rest.template.entity.Invoice;

@Component
```

```
public class RestTemplateRunner implements CommandLineRunner {

    private Logger logger = LoggerFactory.getLogger(RestTempla

    @Autowired
    RestTemplate restTemplate;

    @Override
    public void run(String... args) throws Exception {
        saveInv();
//      getAllInvoices();
//      getOneInvoice();
//      updateInvoice();
//      deleteInvoice();
    }


    private void saveInv() {
        // 1. Producer application URL
        String url = "http://localhost:8080/api/invoices";
        // Send JSON data as Body
        String body = "{\"name\":\"INV11\", \"amount\":234.11,
        // Http Header
        HttpHeaders headers = new HttpHeaders();
        //Set Content Type
        headers.setContentType(MediaType.APPLICATION_JSON);
        //requestEntity : Body+Header
        HttpEntity<String> request = new HttpEntity<String> (b
        // 2. make HTTP call and store Response (URL,ResponseT
    //  ResponseEntity<String> response =  restTemplate.postFo
        ResponseEntity<String> response =  restTemplate.exchan
        // 3. Print details(body,status..etc)
        logger.info("Response Body : {}", response.getBody());
        logger.info("Status code value : {}", response.getStat
        logger.info("Status code : {}", response.getStatusCode

    }
```

```
private void getAllInvoices() {
    String url = "http://localhost:8080/api/invoices";
    ResponseEntity<Invoice[]> response = restTemplate.getF
//  ResponseEntity<Invoice[]> response = restTemplate.exch
    Invoice[] invs = response.getBody();
    List<Invoice> list = Arrays.asList(invs);

    logger.info("Response Body : {}", list);
    logger.info("Status code value : {}", response.getStat
    logger.info("Status code : {}", response.getStatusCode
    logger.info("Headers {} :", response.getHeaders());
}

private void getOneInvoice() {
    String url = "http://localhost:8080/api/invoices/{id}"
//  ResponseEntity<String> response= restTemplate.getForEn
    ResponseEntity<String> response= restTemplate.exchange
    logger.info("Response Body : {}", response.getBody());
    logger.info("Status code value : {}", response.getStat
    logger.info("Status code : {}",response.getStatusCode(
}

private void updateInvoice() {
    String url = "http://localhost:8080/api/invoices/{id}"
    String body = "{\"name\":\"INV13\",\"amount\":888}";
    // Request Header
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    // requestEntity = Body + header
    HttpEntity<String> requestEntity = new HttpEntity<Stri
//  restTemplate.put(url, requestEntity, 7);
    ResponseEntity<String> response= restTemplate.exchange
    logger.info("Response Body : {}", response.getBody());
    logger.info("Status code value : {}", response.getStat
    logger.info("Status code : {}",response.getStatusCode(
    logger.info("Response Headers : {}", response.getHeade
```

```
        }

        private void deleteInvoice() {
            String url = "http://localhost:8080/api/invoices/{id}"
        //  restTemplate.delete(url, 6);
            ResponseEntity<String> response= restTemplate.exchange
            logger.info("Response Body : {}", response.getBody());
            logger.info("Status code value : {}", response.getStat
            logger.info("Status code : {}",response.getStatusCode(
            logger.info("Response Headers : {}", response.getHeade
        }

    }
```

# How to test the application ?

To run the application for testing , right click on Project then select Run As >> Spring Boot App. Additionally, please uncomment the respective method called in run() of RestTemplateRunner.java to test the methods one by one. Further, you can verify your output by comparing the output given in the previous section.

# How to convert ResponseEntity to Java object ?

Sometimes many developers face issue in converting ResponseEntity to Java Objects. Although this part is somewhat tricky. Further, in order to clarify this problem, here we will discuss three solutions to get it done.

## Solution#1 : By using getForEntity() method and returning Array Of Objects

```
ResponseEntity<Invoice[]> response = restTemplate.getForEntity(url, Invoice[].class);

Invoice[] invs = response.getBody();
```

```
List<Invoice> list = Arrays.asList(invs);
System.out.println("Response Body : " +list);
```

## Solution#2 : By using exchange() method and returning List Of Objects

```
ResponseEntity<List<Invoice>> response = restTemplate.exchange(url, HttpMethod.GET, null, ne

List<Invoice> list = response.getBody();
System.out.println("Response Body : " +list);
```

## Solution#3 : By using exchange() method and returning Array Of Objects

```
ResponseEntity<Invoice[]> response = restTemplate.exchange(url, HttpMethod.GET, null, Invoic

Invoice[] invs = response.getBody();
List<Invoice> list = Arrays.asList(invs);
System.out.println("Response Body : " +list);
```

Moreover, we can use any solution to get it done based on our requirement and scenario. Furthermore, in order to get complete code implementation, visit getAllInvoices() method of this article.

# Summary

Almost every REST application will have these operations, we learnt in this article. In other words, no REST API can be developed without these operations. So, you have learned the mandatory concepts of ‘How to write REST Consumer API using Spring Boot RestTemplate’. Now, you should be able to work on ‘How to write REST Consumer API using Spring Boot RestTemplate’. In addition, for further learning on RestTemplate, kindly visit official site. Also,

in upcoming articles we will be learning other types of REST Clients.

How to implement Fault Tolerance in Microservices using Resilience4j?
July 23, 2021
In "Resilience4j"

How to develop a Reactive CRUD REST API with Spring WebFlux?
February 3, 2021
In "Spring Boot"

Spring Boot Features
February 1, 2022
In "java"


Tagged A Guide to the RestTemplate call api using resttemplate consume rest api in java consume rest api in spring boot consume rest service java consumer and producer in rest api consumer api Consuming a RESTful Web Service -Spring consuming rest api in java Convert responseentity string to pojo difference between getForEntity() and getForObject() difference between getForObject() and getForEntity() exchange() getforentity vs getforobject getforobject getforobject vs getforentity how to call api in java how to call rest api in java how to consume rest api in java how to consume restful webservice in java spring boot How to convert Java object to response entity? How to convert responseentity string to Java object? How to convert responseentity to Java object? How To Use an API with Spring RestTemplate How to write REST Consumer API using Spring Boot : RestTemplate How to write RestTemplate methods ? Spring Rest Client java call rest api get example java call rest api post example java resttemplate java spring rest client produces and consumes in rest api rest api java rest consumer spring boot rest template rest template example rest template spring boot restclient spring boot resttemplate resttemplate configuration in spring boot resttemplate example resttemplate exchange resttemplate exchange example resttemplate exchange post example resttemplate get example resttemplate getforentity vs getforobject resttemplate getforobject resttemplate in spring boot resttemplate java resttemplate methods resttemplate post RestTemplate POST example resttemplate postforentity resttemplate postforobject resttemplate postforobject example resttemplate put example resttemplate spring resttemplate spring boot resttemplate spring boot example Spring Boot - Rest Template spring boot responseentity spring boot rest client spring boot rest consumer api spring boot resttemplate spring boot resttemplate exchange post example spring boot resttemplate post example spring rest client spring rest template Spring Restful client spring resttemplate Spring RestTemplate Example what is resttemplate in spring boot

‹ **Previous article**
How to reduce boilerplate code in REST API Spring Data REST

**Next article** ›
How to Schedule a Task/Job in Java? : Spring Boot Scheduler

## Leave a Reply

Name *

Email Address *

Website

Comment *

☐ Yes, add me to your mailing list

Post Comment

## FOLLOW US

## RECENTLY PUBLISHED POSTS

- **Spring Transaction Annotations With Examples**
- **Spring Scheduling Annotations With Examples**
- **Spring Security Annotations With Examples**
- **Spring Boot Interview Questions & Answers**
- **Java Interface**
- **Spring Security Without WebSecurityConfigurerAdapter**
- **Spring Boot Batch Example CSV to MySQL Using JPA**
- **Spring Batch Tutorial**
- **Java 14 Features**
- **Java Features After Java 8**

## DO YOU HAVE A QUERY ?

**Submit your Query**

Email *

**Subscribe to receive Updates !**

© 2022

Powered by javatechonline.com | https://javatechonline.com