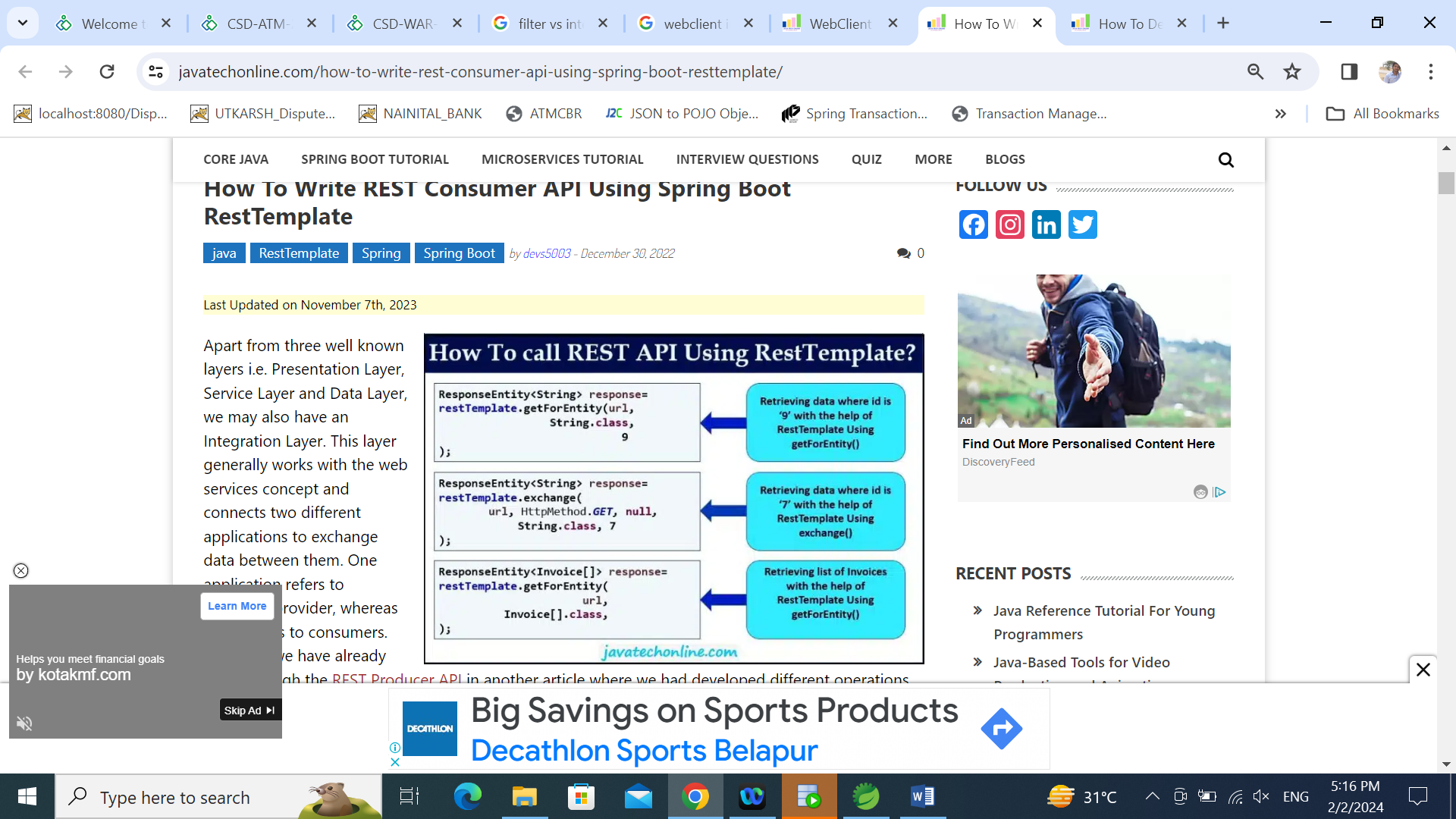
Apart from three well known layers i.e. Presentation Layer, Service Layer and Data Layer, we may also have an Integration Layer. This layer generally works with the web services concept and connects two different applications to exchange data between them. One application refers to producer/provider,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



## What is Rest Template ?

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() ?

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

## 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 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)

Complete Code: How to write REST Consumer API using Spring Boot RestTemplate?

@SpringBootApplication

public class SpringBootRestTemplateApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootRestTemplateApplication.class, args);

}

@Bean

public RestTemplate getRestTemplate() {

return new RestTemplate();

}

}

RestTemplateRunner.java

@Component

public class RestTemplateRunner implements CommandLineRunner {

private Logger logger = LoggerFactory.getLogger(RestTemplateRunner.class);

@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,\"number\":\"INVOICE11\",\"receivedDate\":\"28-10-2020\",\"type\":\"Normal\",\"vendor\":\"ADHR001\",\"comments\" :\"On Hold\"}";

// Http Header

HttpHeaders headers = new HttpHeaders();

//Set Content Type

headers.setContentType(MediaType.APPLICATION\_JSON);

//requestEntity : Body+Header

HttpEntity<String> request = new HttpEntity<String> (body,headers);

// 2. make HTTP call and store Response (URL,ResponseType)

// ResponseEntity<String> response = restTemplate.postForEntity(url, request, String.class);

ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.POST,request, String.class);

// 3. Print details(body,status..etc)

logger.info("Response Body : {}", response.getBody());

logger.info("Status code value : {}", response.getStatusCodeValue());

logger.info("Status code : {}", response.getStatusCode().name());

}

private void getAllInvoices() {

String url = "http://localhost:8080/api/invoices";

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

// ResponseEntity<Invoice[]> response = restTemplate.exchange(url, HttpMethod.GET, null, Invoice[].class);

Invoice[] invs = response.getBody();

List<Invoice> list = Arrays.asList(invs);

logger.info("Response Body : {}", list);

logger.info("Status code value : {}", response.getStatusCodeValue());

logger.info("Status code : {}", response.getStatusCode().name());

logger.info("Headers {} :", response.getHeaders());

}

private void getOneInvoice() {

String url = "http://localhost:8080/api/invoices/{id}";

// ResponseEntity<String> response= restTemplate.getForEntity(url, String.class, 9);

ResponseEntity<String> response= restTemplate.exchange(url, HttpMethod.GET, null, String.class, 7);

logger.info("Response Body : {}", response.getBody());

logger.info("Status code value : {}", response.getStatusCodeValue());

logger.info("Status code : {}",response.getStatusCode().name());

}

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<String>(body, headers);

// restTemplate.put(url, requestEntity, 7);

ResponseEntity<String> response= restTemplate.exchange(url, HttpMethod.PUT, requestEntity, String.class, 7);

logger.info("Response Body : {}", response.getBody());

logger.info("Status code value : {}", response.getStatusCodeValue());

logger.info("Status code : {}",response.getStatusCode().name());

logger.info("Response Headers : {}", response.getHeaders());

}

private void deleteInvoice() {

String url = "http://localhost:8080/api/invoices/{id}";

// restTemplate.delete(url, 6);

ResponseEntity<String> response= restTemplate.exchange(url, HttpMethod.DELETE, null, String.class,5);

logger.info("Response Body : {}", response.getBody());

logger.info("Status code value : {}", response.getStatusCodeValue());

logger.info("Status code : {}",response.getStatusCode().name());

logger.info("Response Headers : {}", response.getHeaders());

}

}

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, new ParameterizedTypeReference<List<Invoice>>() {});

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, Invoice[].class);

Invoice[] invs = response.getBody();

List<Invoice> list = Arrays.asList(invs);

System.out.println("Response Body : " +list);

## FAQ

**What is the difference between RestTemplate and WebClient in Spring Boot?**

WebClient is a more modern and flexible option to RestTemplate. [WebClient](https://javatechonline.com/webclient-in-spring-boot/) is introduced in Spring WebFlux. It is asynchronous by nature and is a better choice for non-blocking, reactive applications. RestTemplate is a synchronous client and is more appropriate for traditional Spring MVC/REST applications.

**Should I use WebClient in place of RestTemplate in Spring Boot projects?**

RestTemplate was considered as a mature technology. However, WebClient was recommended for new projects and for applications that require non-blocking, reactive behavior. Moreover, people are still using RestTemplate in some legacy projects. Be sure to check the latest Spring Boot documentation and community discussions for any updates on the status of RestTemplate.

**Can I use RestTemplate to perform synchronous and parallel requests to multiple endpoints?**

Yes, RestTemplate offers us to make synchronous requests to multiple endpoints in parallel by leveraging Java’s concurrency mechanisms, such as threads or CompletableFuture. We can create multiple RestTemplate instances or use a single RestTemplate carefully in a multi-threaded environment to accomplish parallelism.

**Is RestTemplate suitable for microservices communication within a Spring Boot-based microservices architecture?**

We can use RestTemplate for microservices communication, but we should consider using technologies like Spring Cloud OpenFeign, which provide higher-level abstractions and better integration with service discovery and load balancing features in microservices architectures. These technologies can simplify inter-service communication in a microservices environment.

# **Spring RestTemplate.exchange()**

This page will walk through Spring RestTemplate.exchange() method example. The exchange method executes the request of any HTTP method and returns ResponseEntity instance. The exchange method can be used for HTTP DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT, TRACE methods. Using exchange method we can perform CRUD operation i.e. create, read, update and delete data. The exchange method returns ResponseEntity using which we can get response status, body and headers.  
The exchange method can be used with variety of parameters.  
**RequestEntity + responseType**

ResponseEntity<T> exchange(RequestEntity<?> requestEntity, Class<T> responseType)

ResponseEntity<T> exchange(RequestEntity<?> requestEntity, ParameterizedTypeReference<T> responseType)

**url + HttpMethod + HttpEntity + responseType + uriVariables**

ResponseEntity<T> exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType, Map<String,?> uriVariables)

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

ResponseEntity<T> exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, ParameterizedTypeReference<T> responseType, Map<String,?> uriVariables)

ResponseEntity<T> exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, ParameterizedTypeReference<T> responseType, Object... uriVariables)

**URI + HttpMethod + HttpEntity + responseType**

ResponseEntity<T> exchange(URI url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType)

ResponseEntity<T> exchange(URI url, HttpMethod method, HttpEntity<?> requestEntity, ParameterizedTypeReference<T> responseType)

Here on this page we will discuss using exchange method for CRUD operation in detail.

##### Contents

* [1. Technologies Used](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#Technologies)
* [2. Maven Dependencies](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#Maven)
* [3. exchange() to Post Data](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#Post)
* [4. exchange() to Get Data](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#Get)
* [5. exchange() to Update Data](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#Update)
* [6. exchange() to Delete Data](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#Delete)
* [7. Complete CRUD Example using exchange()](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#CRUD)
* [8. Run Application](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#Run)
* [9. Reference](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#reference)
* [10. Download Source Code](https://www.concretepage.com/spring-5/spring-resttemplate-exchange#download)

### 1. Technologies Used

Find the technologies being used in our example.  
1. Java 11  
2. Spring 5.2.5.RELEASE  
3. Spring Boot 2.2.6.RELEASE  
4. Maven 3.5.2

## 2. Maven Dependencies

Find the Maven dependencies to run the example.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

## 3. exchange() to Post Data

To use exchange to post data, we need to use HTTP method as HttpMethod.POST. For request entity, we can use HttpEntity and RequestEntity. For response type we can pass usual response type or ParameterizedTypeReference.  
Find the **client code** examples to use exchange to post data.  
**URI + HttpMethod + HttpEntity + responseType**

URI uri = new URI("http://localhost:8080/employee");

HttpEntity<Employee> httpEntity = new HttpEntity<Employee>(objEmp, headers);

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Employee> responseEntity = restTemplate.exchange(uri, HttpMethod.POST, httpEntity, Employee.class);

**RequestEntity + ParameterizedTypeReference**

RequestEntity<Employee> requestEntity = new RequestEntity<>(objEmp, headers, HttpMethod.POST, uri);

ParameterizedTypeReference<Employee> typeRef = new ParameterizedTypeReference<Employee>() {};

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Employee> responseEntity = restTemplate.exchange(requestEntity, typeRef);

**Server Code**  
To serve the above client code to post data, our server code will be as follows.

@PostMapping(value = "employee")

public ResponseEntity<Employee> addEmployee(@RequestBody Employee employee, UriComponentsBuilder builder) {

------

}

## 4. exchange() to Get Data

To use exchange to get data, we need to use HTTP method as HttpMethod.GET. To query data for the given properties, we can pass them as URI variables. The exchange method accepts URI variable arguments as Map and Object Varargs. The response type can be usual response type and ParameterizedTypeReference.  
Find the **client code** examples to use exchange to get data.  
**url + HttpMethod + HttpEntity + responseType + Map**

String url = "http://localhost:8080/employee/{profile}/{tech}";

Map<String, String> map = new HashMap<>();

map.put("profile", "Developer");

map.put("tech", "Java");

ResponseEntity<Employee[]> responseEntity =

restTemplate.exchange(url, HttpMethod.GET, httpEntity, Employee[].class, map);

**url + HttpMethod + HttpEntity + ParameterizedTypeReference + Object Varargs**

String url = "http://localhost:8080/employee/{profile}/{tech}";

ParameterizedTypeReference<List<Employee>> typeRef = new ParameterizedTypeReference<List<Employee>>() {};

ResponseEntity<List<Employee>> responseEntity =

restTemplate.exchange(url, HttpMethod.GET, httpEntity, typeRef, profile, tech);

**Server Code**  
To serve the above client code to get data, our server code will be as follows.

@GetMapping(value = "employee/{profile}/{tech}")

public ResponseEntity<List<Employee>> getEmployeesByProfileNTech(@PathVariable("profile") String profile,

@PathVariable("tech") String technology) {

------

}

## 5. exchange() to Update Data

To use exchange to update data, we need to use HTTP method as HttpMethod.PUT. For request entity, we can use HttpEntity and RequestEntity. For response type we can pass usual response type or ParameterizedTypeReference.  
Find the **client code** examples to use exchange to update data.  
**URI + HttpMethod + HttpEntity + responseType**

URI uri = new URI("http://localhost:8080/employee");

HttpEntity<Employee> httpEntity = new HttpEntity<Employee>(objEmp, headers);

ResponseEntity<Employee> responseEntity = restTemplate.exchange(uri, HttpMethod.PUT, httpEntity, Employee.class);

**RequestEntity + ParameterizedTypeReference**

RequestEntity<Employee> requestEntity = new RequestEntity<>(objEmp, headers, HttpMethod.PUT, uri);

ParameterizedTypeReference<Employee> typeRef = new ParameterizedTypeReference<Employee>() {};

ResponseEntity<Employee> responseEntity = restTemplate.exchange(requestEntity, typeRef);

**Server Code**  
To serve the above client code to update data, our server code will be as follows.

@PutMapping(value = "employee")

public ResponseEntity<Employee> updateEmployee(@RequestBody Employee employee) {

------

}

## 6. exchange() to Delete Data

To use exchange to delete data, we need to use HTTP method as HttpMethod.DELETE. To pass properties to delete data, we can use URI variables as Map and Object Varargs.  
Find the client code using URI variable as Object Varargs.

String url = "http://localhost:8080/employee/{id}";

ResponseEntity<Void> responseEntity =

restTemplate.exchange(url, HttpMethod.DELETE, httpEntity, Void.class, empId);

Find the client code using URI variable as Map.

String url = "http://localhost:8080/employee/{id}";

Map<String, Integer> map = new HashMap<>();

map.put("id", 200);

ResponseEntity<Void> responseEntity =

restTemplate.exchange(url, HttpMethod.DELETE, httpEntity, Void.class, map);

**Server Code**  
To serve the above client code to delete data, our server code will be as follows.

@DeleteMapping(value = "employee/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable("id") Integer empId) {

------

}

## 7. Complete CRUD Example using exchange()

Find client code for POST.  
**RestClientUtilPost.java**

package com.concretepage.client;

import java.net.URI;

import java.net.URISyntaxException;

import org.springframework.core.ParameterizedTypeReference;

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpMethod;

import org.springframework.http.MediaType;

import org.springframework.http.RequestEntity;

import org.springframework.http.ResponseEntity;

import org.springframework.web.client.RestTemplate;

import com.concretepage.domain.Employee;

public class RestClientUtilPost {

// exchange(URI url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType)

public void addEmployeeDemo() throws URISyntaxException {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

URI uri = new URI("http://localhost:8080/employee");

Employee objEmp = new Employee();

objEmp.setName("Krishna");

objEmp.setCity("Noida");

HttpEntity<Employee> httpEntity = new HttpEntity<Employee>(objEmp, headers);

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Employee> responseEntity = restTemplate.exchange(uri, HttpMethod.POST, httpEntity,

Employee.class);

System.out.println("Status Code: " + responseEntity.getStatusCode());

System.out.println("Id: " + responseEntity.getBody().getEmpId());

System.out.println("Location: " + responseEntity.getHeaders().getLocation());

}

// exchange(RequestEntity<?> requestEntity, ParameterizedTypeReference<T> responseType)

public void addEmployeeDemoPTRef() throws URISyntaxException {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

URI uri = new URI("http://localhost:8080/employee");

Employee objEmp = new Employee();

objEmp.setName("Krishna");

objEmp.setCity("Noida");

RequestEntity<Employee> requestEntity = new RequestEntity<>(objEmp, headers, HttpMethod.POST, uri);

ParameterizedTypeReference<Employee> typeRef = new ParameterizedTypeReference<Employee>() {

};

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Employee> responseEntity = restTemplate.exchange(requestEntity, typeRef);

System.out.println("Status Code: " + responseEntity.getStatusCode());

System.out.println("Id: " + responseEntity.getBody().getEmpId());

System.out.println("Location: " + responseEntity.getHeaders().getLocation());

}

public static void main(String args[]) throws URISyntaxException {

RestClientUtilPost util = new RestClientUtilPost();

util.addEmployeeDemo();

util.addEmployeeDemoPTRef();

}

}

Output for addEmployeeDemo() method.

Status Code: 201 CREATED

Id: 201

Location: http://localhost:8080/employee/201

Find client code for GET.  
**RestClientUtilGet.java**

package com.concretepage.client;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.springframework.core.ParameterizedTypeReference;

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.web.client.RestTemplate;

import com.concretepage.domain.Employee;

public class RestClientUtilGet {

// exchange(String url, HttpMethod method, HttpEntity<?> requestEntity,

// Class<T> responseType, Map<String,?> uriVariables)

public void getEmployeeDemo() {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

String url = "http://localhost:8080/employee/{profile}/{tech}";

HttpEntity<?> httpEntity = new HttpEntity<>(headers);

Map<String, String> map = new HashMap<>();

map.put("profile", "Developer");

map.put("tech", "Java");

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Employee[]> responseEntity = restTemplate.exchange(url, HttpMethod.GET, httpEntity,

Employee[].class, map);

System.out.println("Status Code: " + responseEntity.getStatusCode());

for (Employee e : responseEntity.getBody()) {

System.out.println(e);

}

}

// exchange(String url, HttpMethod method, HttpEntity<?> requestEntity,

// ParameterizedTypeReference<T> responseType, Object... uriVariables)

public void getEmployeeDemoPTRef() {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

String url = "http://localhost:8080/employee/{profile}/{tech}";

HttpEntity<?> httpEntity = new HttpEntity<>(headers);

String profile = "Developer";

String tech = "Java";

ParameterizedTypeReference<List<Employee>> typeRef = new ParameterizedTypeReference<List<Employee>>() {

};

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<List<Employee>> responseEntity = restTemplate.exchange(url, HttpMethod.GET, httpEntity, typeRef,

profile, tech);

System.out.println("Status Code: " + responseEntity.getStatusCode());

responseEntity.getBody().forEach(e -> System.out.println(e));

}

public static void main(String args[]) {

RestClientUtilGet util = new RestClientUtilGet();

util.getEmployeeDemo();

util.getEmployeeDemoPTRef();

}

}

Output for getEmployeeDemo() method.

Status Code: 200 OK

101, Lakshman, Agra

102, Bharat, Mathura

103, Shatrughan, Noida

Find client code for PUT.  
**RestClientUtilPut.java**

package com.concretepage.client;

import java.net.URI;

import java.net.URISyntaxException;

import org.springframework.core.ParameterizedTypeReference;

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpMethod;

import org.springframework.http.MediaType;

import org.springframework.http.RequestEntity;

import org.springframework.http.ResponseEntity;

import org.springframework.web.client.RestTemplate;

import com.concretepage.domain.Employee;

public class RestClientUtilPut {

// exchange(URI url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType)

public void updateEmployeeDemo() throws URISyntaxException {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

URI uri = new URI("http://localhost:8080/employee");

Employee objEmp = new Employee(100, "Krishna", "Noida");

HttpEntity<Employee> httpEntity = new HttpEntity<Employee>(objEmp, headers);

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Employee> responseEntity = restTemplate.exchange(uri, HttpMethod.PUT, httpEntity,

Employee.class);

System.out.println("Status Code: " + responseEntity.getStatusCode());

System.out.println(responseEntity.getBody());

}

// exchange(RequestEntity<?> requestEntity, ParameterizedTypeReference<T> responseType)

public void updateEmployeeDemoPTRef() throws URISyntaxException {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

URI uri = new URI("http://localhost:8080/employee");

Employee objEmp = new Employee(100, "Krishna", "Noida");

RequestEntity<Employee> requestEntity = new RequestEntity<>(objEmp, headers, HttpMethod.PUT, uri);

ParameterizedTypeReference<Employee> typeRef = new ParameterizedTypeReference<Employee>() {

};

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Employee> responseEntity = restTemplate.exchange(requestEntity, typeRef);

System.out.println("Status Code: " + responseEntity.getStatusCode());

System.out.println(responseEntity.getBody());

}

public static void main(String args[]) throws URISyntaxException {

RestClientUtilPut util = new RestClientUtilPut();

util.updateEmployeeDemo();

util.updateEmployeeDemoPTRef();

}

}

Output for updateEmployeeDemo() method.

Status Code: 200 OK

100, Shri Krishna, Mathura

Find client code for DELETE.  
**RestClientUtilDelete.java**

package com.concretepage.client;

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.web.client.RestTemplate;

public class RestClientUtilDelete {

// exchange(String url, HttpMethod method, HttpEntity<?> requestEntity,

// ParameterizedTypeReference<T> responseType, Object... uriVariables)

public void deleteEmployeeDemo() {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

String url = "http://localhost:8080/employee/{id}";

HttpEntity<?> httpEntity = new HttpEntity<>(headers);

Integer empId = 200;

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<Void> responseEntity = restTemplate.exchange(url, HttpMethod.DELETE, httpEntity, Void.class,

empId);

System.out.println("Status Code: " + responseEntity.getStatusCode());

}

public static void main(String args[]) {

RestClientUtilDelete util = new RestClientUtilDelete();

util.deleteEmployeeDemo();

}

}

Output

Status Code: 204 NO\_CONTENT

Find the server code.  
**EmployeeController.java**

package com.concretepage.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.util.UriComponentsBuilder;

import com.concretepage.domain.Employee;

import com.concretepage.service.EmployeeService;

@RestController

public class EmployeeController {

@Autowired

private EmployeeService empService;

@GetMapping(value = "employee/{profile}/{tech}")

public ResponseEntity<List<Employee>> getEmployeesByProfileNTech(@PathVariable("profile") String profile,

@PathVariable("tech") String technology) {

List<Employee> list = empService.getEmployees(profile, technology);

return new ResponseEntity<List<Employee>>(list, HttpStatus.OK);

}

@PostMapping(value = "employee")

public ResponseEntity<Employee> addEmployee(@RequestBody Employee employee, UriComponentsBuilder builder) {

empService.addEmployee(employee);

HttpHeaders headers = new HttpHeaders();

headers.setLocation(builder.path("/employee/{id}").buildAndExpand(employee.getEmpId()).toUri());

return new ResponseEntity<Employee>(employee, headers, HttpStatus.CREATED);

}

@PutMapping(value = "employee")

public ResponseEntity<Employee> updateEmployee(@RequestBody Employee employee) {

empService.updateEmployee(employee);

return new ResponseEntity<Employee>(employee, HttpStatus.OK);

}

@DeleteMapping(value = "employee/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable("id") Integer empId) {

empService.deleteEmployee(empId);

return new ResponseEntity<Void>(HttpStatus.NO\_CONTENT);

}

}

**Employee.java**

package com.concretepage.domain;

public class Employee {

private int empId;

private String name;

private String city;

public Employee() {}

public Employee(int empId, String name, String city) {

this.empId = empId;

this.name = name;

this.city = city;

}

//Setters and Getters

}

**EmployeeService.java**

package com.concretepage.service;

import java.util.Arrays;

import java.util.List;

import org.springframework.stereotype.Service;

import com.concretepage.domain.Employee;

@Service

public class EmployeeService {

public void addEmployee(Employee emp) {

// Perform database operation

emp.setEmpId(201);

System.out.println("Name: " + emp.getName());

System.out.println("City: " + emp.getCity());

}

public List<Employee> getEmployees(String profile, String technology) {

// Perform database operation

System.out.println(profile);

System.out.println(technology);

Employee e1 = new Employee(101, "Lakshman", "Agra");

Employee e2 = new Employee(102, "Bharat", "Mathura");

Employee e3 = new Employee(103, "Shatrughan", "Noida");

return Arrays.asList(e1, e2, e3);

}

public void updateEmployee(Employee emp) {

// Perform database operation

emp.setName("Shri Krishna");

emp.setCity("Mathura");

}

public void deleteEmployee(int empId) {

// Perform database operation

System.out.println("Employee deleted with id " + empId);

}

}

https://javatechonline.com/how-to-develop-rest-crud-api-using-spring-boot/