RestTemplate

dasturlash.uz

RestTemplate

- RestTemplate is a synchronous client to perform HTTP requests.
- It is a higher-order API since it performs HTTP requests by using an HTTP client library like the JDK HttpURLConnection, Apache HttpClient, and others.
- The HTTP client library takes care of all the low-level details of communication over HTTP while the RestTemplate adds the capability of transforming the request and response in JSON or XML to Java objects.
- By default, RestTemplate uses the class java.net.HttpURLConnection as the HTTP client. However, we can switch to another HTTP client library which we will see in a later section

Useful Methods of RestTemplate

- **getForEntity():** executes a GET request and returns an object of ResponseEntity class that contains both the status code and the resource as an object.
- **getForObject()** : similar to getForEntity(), but returns the resource directly.
- **exchange()**: executes a specified HTTP method, such as GET, POST, PUT, etc, and returns a ResponseEntity containing both the HTTP status code and the resource as an object.
- **execute()** : similar to the exchange() method, but takes additional parameters: RequestCallback and ResultSetExtractor.
- **headForHeaders()**: executes a HEAD request and returns all HTTP headers for the specified URL.
- **optionsForAllow()**: executes an OPTIONS request and uses the Allow header to return the HTTP methods that are allowed under the specified URL.

dasturlash.uz

Useful Methods of RestTemplate

- **delete()**: deletes the resources at the given URL using the HTTP DELETE method.
- **put()**: updates a resource for a given URL using the HTTP PUT method.
- **postForObject()** : creates a new resource using HTTP POST method and returns an entity.
- **postForLocation()**: creates a new resource using the HTTP POST method and returns the location of the newly created resource.

RestTemplate - getForObject example

- Simple Get Request

```
RestTemplate restTemplate = new RestTemplate();
ProverbDTO dto = restTemplate.getForObject("some_url", ProverbDTO.class);

RestTemplate restTemplate = new RestTemplate();
List list = restTemplate.getForObject("some_url", List.class);
```

- Get Request with Params

```
RestTemplate restTemplate = new RestTemplate();
int p = 1;
int s = 10;
String response = restTemplate.getForObject("url?page={p}&size={s}",String.class, p, s);
```

RestTemplate - getForEntity example

```
RestTemplate restTemplate = new RestTemplate();

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

System.out.println(response.getStatusCode());
System.out.println(response.getBody());
```

Making an HTTP POST Request

- We are invoking an HTTP POST method on a REST API with the postForObject() method:

```
RestTemplate restTemplate = new RestTemplate();

HttpEntity<FeedBackDTO> request = new HttpEntity<FeedBackDTO>( new FeedBackDTO("dasdasda"));

String response = restTemplate .postForObject(url, request, String.class);
```

dasturlash.uz

HttpEntity<T>

- Represents an HTTP request or response entity, consisting of headers and body.
- Often used in combination with the RestTemplate, like so:

```
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.TEXT_PLAIN);

HttpEntity<String> entity = new HttpEntity<>("Hello World", headers);

URI location = template.postForLocation("https://example.com", entity);
```

```
HttpEntity<String> entity = template.getForEntity("https://example.com", String.class);
String body = entity.getBody();
MediaType contentType = entity.getHeaders().getContentType();
```

ResponseEntity<T>

```
public class ResponseEntity<T> extends HttpEntity<T> {
    private final Object status;
}
```

- ResponseEntity<T> bu HttpEntity<T> dan nasil olgan bo'lib u faqat response da ishlatiladi. ResponceEntity da HttpStatus code ni olish mumkin.

dasturlash.uz

RequestEntity<T>

- RequestEntity<T> extends HttpEntity<T>.
- RequestEntity classi request ning body, header, request method , request URL ni describe qiladigan class hisoblanadi.

```
 public class RequestEntity<T> extends HttpEntity<T> {
    private final HttpMethod method;
    private final URI url;
    private final Type type;
}
```

ParameterizedTypeReference

-
  ```
  ParameterizedTypeReference<List<ProverbDTO>> returnType = new
  ParameterizedTypeReference<List<ProverbDTO>>() {
  };
  ```

dasturlash.uz

.exchange()

- 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.

.exchange() method 1

- **RequestEntity + responseType**

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

dasturlash.uz

.exchange() method 2

- **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)

.exchange() method 3

- **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)

.exchange() Get example 1

```
RestTemplate restTemplate = new RestTemplate();

RequestEntity<String> requestEntity = new RequestEntity<>(HttpMethod.GET, new URI(“url"));
ResponseEntity<List> s = restTemplate.exchange(requestEntity, List.class);

System.out.println(s.getStatusCode());
System.out.println(s.getBody());
```

.exchange() Get example 2

```
 RestTemplate restTemplate = new RestTemplate();

        ParameterizedTypeReference<List<ProverbDTO>> returnType = new
ParameterizedTypeReference<List<ProverbDTO>>() {
        };

RequestEntity<String> requestEntity = new RequestEntity<>(HttpMethod.GET,  new URI("url"));
ResponseEntity<List<ProverbDTO>> s = restTemplate.exchange(requestEntity, returnType);

System.out.println(s.getStatusCode());
System.out.println(s.getBody());
```

.exchange() Get example 3

```
RestTemplate restTemplate = new RestTemplate();

HttpHeaders httpHeaders = new HttpHeaders();
httpHeaders.setContentType(MediaType.APPLICATION_JSON);

HttpEntity<?> requestEntity = new HttpEntity<>(httpHeaders);

ResponseEntity<String> s = restTemplate.exchange("URL", HttpMethod.GET, requestEntity, String.class);
```

.exchange() Get example 4

```
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);
```

.exchange() to Post Data

```
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);
```

.execute() – method.

- **execute()** : similar to the exchange() method, but takes additional parameters: RequestCallback and ResultSetExtractor.

Links

- https://www.concretepage.com/spring-5/spring-resttemplate-exchange
- https://howtodoinjava.com/spring-boot2/resttemplate/spring-restful-client-resttemplate-example/
- https://reflectoring.io/spring-resttemplate/
- https://www.baeldung.com/rest-template