🗑️ officially deprecating RestTemplate

Use RestClient! 💯

RESTTEMPLATE DEPRECATION & THE RISE OF RESTCLIENT

- RestTemplate was the classic synchronous HTTP client in Spring since 3.0.
  - Spring Framework 6.1 introduced RestClient as a modern, fluent, synchronous HTTP client.
  - In “The state of HTTP clients in Spring” (Sept 30, 2025), **Spring announced the official deprecation of RestTemplate**.
  - Docs: As of 6.1, **RestClient offers a more modern API** for synchronous HTTP access.
  - Rule of thumb: synchronous → RestClient, reactive/streaming → WebClient, legacy → RestTemplate (start migrating).

GET REQUESTS – BEFORE & AFTER

- Fetch a Customer by id at https://api.example.com/customers/{id}.
  - Same use case, less overloads, more fluent & readable.

BEFORE – RestTemplate

```
RestTemplate restTemplate = new
RestTemplate();

String url =
"https://api.example.com/customers/{i
d}";

Customer customer =
restTemplatea.getForObject(
    url,
    Customer.class,
    id);
```

NOW – RestClient

```
RestClient restClient =
RestClient.create();

String url =
"https://api.example.com/customers/{i
d}";

Customer customer = restClient.get()
    .uri(url, id)
    .retrieve()
    .body(Customer.class);
```

POST REQUESTS – CREATE A RESOURCE

- Create a new Customer at https://api.example.com/customers.
  - Fluent builder: post() → uri() → body() → retrieve().

BEFORE – RestTemplate

```
RestTemplate restTemplate = new
RestTemplate();

String url =
"https://api.example.com/customers";

CustomerRequest request = new
CustomerRequest("Alice",
"alice@example.com");

// Directly get created entity:
Customer created =
restTemplate.postForObject(
   url,   request,   Customer.class);
```

NOW – RestClient

```
RestClient restClient = RestClient.create();

String url =
"https://api.example.com/customers";

CustomerRequest request = new
CustomerRequest("Alice",
"alice@example.com");

// Get created entity:
Customer created = restClient.post()
   .uri(url)
   .body(request)
   .retrieve()
   .body(Customer.class);
```

PUT REQUESTS – UPDATE A RESOURCE

- Update an existing Customer at https://api.example.com/customers/{id}.
  - With RestClient, even void-like operations can expose status & headers.

BEFORE – RestTemplate

```
RestTemplate restTemplate = new
RestTemplate();

String url =
"https://api.example.com/customers/{id}";

CustomerUpdate update = new
CustomerUpdate("Alice V.",
"alice.v@example.com");

restTemplate.put(url, update, id);
// If something goes wrong →
RestClientException or
HttpStatusCodeException
```

NOW – RestClient

```
RestClient restClient = RestClient.create();

String url =
"https://api.example.com/customers/{id}";

CustomerUpdate update = new
CustomerUpdate("Alice V.", "alice.v@example.com");

ResponseEntity<Void> response = restClient.put()
    .uri(url, id) .body(update)
    .retrieve() .toBodilessEntity();

HttpStatusCode status = response.getStatusCode();
HttpHeaders headers = response.getHeaders();
```

DELETE REQUESTS – REMOVE A RESOURCE

- Delete a Customer at https://api.example.com/customers/{id}.
  - All HTTP verbs now share the same fluent pattern.

BEFORE – RestTemplate

```
RestTemplate restTemplate = new
RestTemplate();

String url =
"https://api.example.com/customers/{i
d}";

restTemplate.delete(url, id);
// Check for RestClientException /
HttpStatusCodeException if needed
```

NOW – RestClient

```
RestClient restClient = RestClient.create();

String url =
"https://api.example.com/customers/{id}";

// Basic fire-and-forget:
restClient.delete()
    .uri(url, id)
    .retrieve()
    .toBodilessEntity();
```

https://bit.ly/2v7222

https://spring-book.mystrikingly.com