14. What are the possible ways to call remote API using REST? Have you ever used third party clients for it ?

Here are the **four primary ways** to call remote REST APIs in a Spring Boot application, along with when to choose each.

1. `RestClient` (The Modern Standard)

Introduced in Spring Boot 3.2 (Java 17+), this is the modern, synchronous replacement for `RestTemplate`. It offers a fluent, readable API similar to WebClient but operates on the standard Servlet stack.[1] [2]

- **Best for:** Standard, synchronous microservice calls in modern Spring Boot 3 apps.
- **Example:**

```
RestClient restClient = RestClient.create();
String result = restClient.get()
  .uri("https://api.example.com/users/1")
  .retrieve()
  .body(String.class);
```

2. `WebClient` (The Reactive Powerhouse)

Part of Spring WebFlux, this is a non-blocking, asynchronous client. It is essential for high-concurrency applications where you need to make parallel calls without blocking threads.[3] [4]

- **Best for:** High-throughput systems, reactive microservices, or when you need to call 3 APIs in parallel and aggregate results.
- **Example:**

```
webClient.get().uri("/users/1")
  .retrieve()
  .bodyToMono(User.class)
  .subscribe(user -> System.out.println(user.getName()));
```

3. `Spring Cloud OpenFeign` (The Declarative "Magic")

Feign allows you to define an interface, annotate it with `@GetMapping`, and Spring generates the implementation at runtime. It drastically reduces boilerplate code.[5] [3]

- **Best for:** Rapid development in microservice architectures where clean code is prioritized over raw performance tuning.
- **Example:**

```
@FeignClient(name = "user-service")
public interface UserClient {
    @GetMapping("/users/{id}")
    User getUser(@PathVariable("id") Long id);
}
```

4. `RestTemplate` (The Legacy)

The old standard. It is currently in "maintenance mode" (not deprecated, but no new features). It is widely used in legacy Spring applications.[6] [4]

- **Best for:** Maintaining older Spring Boot 2.x applications. Avoid for new projects.

Have I used third-party clients?

"Yes, apart from the Spring ecosystem, I have used:"

1. **Retrofit (from Square):**
   - "I used Retrofit in a project where we needed strict type safety and cleaner API interfaces, similar to Feign but standalone. It uses OkHttp under the hood and is significantly faster than Feign for high-load scenarios."[7]
2. **OkHttp:**
   - "I used OkHttp directly when I needed low-level control, like fine-tuning connection pooling, managing specific timeout behaviors, or handling complex interceptors that were hard to configure in RestTemplate."[8]

Summary Table for Interview

Client	Type	Use Case
RestClient	Synchronous	**Default choice** for Spring Boot 3.x+ apps.
WebClient	Asynchronous	**High concurrency** or reactive apps.
Feign	Declarative	**Cleanest code**, fastest dev speed.
Retrofit	Declarative	**Performance + Clean code** outside Spring context.

⁂

1. https://dev.to/devcorner/simplified-rest-api-calls-in-spring-boot-3-with-restclient-full-crud-example-1i59
2. https://www.geeksforgeeks.org/advance-java/a-guide-to-restclient-in-spring-boot/
3. https://www.baeldung.com/spring-boot-feignclient-vs-webclient
4. https://www.baeldung.com/spring-webclient-resttemplate
5. https://github.com/akash-coded/spring-framework/discussions/174
6. https://www.linkedin.com/pulse/webclient-vs-feign-client-resttemplate-which-one-should-talha-adeel-mbldf
7. https://www.javacodemonk.com/retrofit-vs-feign-for-server-side-d7f199c4
8. https://www.linkedin.com/pulse/comparing-feign-webclient-other-http-clients-kotlin-spring-eslami-7fbsf
9. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/24f18c0a-b56e-4862-a619-e2959036a5c2/image.jpg
10. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/97e9ed86-ed11-4035-a18b-1eb05c1bab4e/image.jpg
11. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/87f62423-96f2-4071-9802-8f6699e0ecd8/image.jpg
12. https://digma.ai/restclient-vs-webclient-vs-resttemplate/
13. https://www.wiremock.io/post/java-http-client-comparison
14. https://www.reddit.com/r/SpringBoot/comments/191l7ya/a_comparison_between_restclient_webclient_and/
15. https://www.javaguides.net/2024/08/resttemplate-vs-webclient-vs-feign-client.html
16. https://stackoverflow.com/questions/31483874/resttemplate-vs-apache-http-client-for-production-code-in-spring-project
17. https://dev.to/nullvoidkage/webclient-vs-resttemplate-vs-feignclient-a-comparative-guide-4028
18. https://github.com/nlinhvu/rest-client-demo-2024
19. https://www.freecodecamp.org/news/best-choice-openfeign-or-webclient/
20. https://developers.ascendcorp.com/spring-restclient-vs-resttemplate-a742aa499e6f
21. https://stackoverflow.com/questions/67191617/springboot-feignclient-vs-webclient
22. https://www.reddit.com/r/java/comments/sax4vb/i_created_a_library_like_retrofit_and_openfeign/
23. https://alankar-srivastava.hashnode.dev/springs-http-trio-resttemplate-vs-webclient-vs-feignclient-whos-the-real-mvp
24. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/429036f4-3bc3-4a7a-8c6f-90246e73b9b9/image.jpg