Spring6 | Spring Boot3有哪些HTTP客户端可以选择

个人博客：[无奈何杨（wnhyang）](http://cxyroad.com/"https://wnhyang.github.io/")

个人语雀：[wnhyang](http://cxyroad.com/"https://www.yuque.com/wnhyang")

共享语雀：[在线知识共享](http://cxyroad.com/"https://www.yuque.com/wnh")

Github：[wnhyang - Overview](http://cxyroad.com/"https://github.com/wnhyang")

参考

[REST Clients :: Spring Framework](http://cxyroad.com/"https://docs.spring.io/spring-framework/reference/integration/rest-clients.html#rest-http-interface")

[【微服务|SpringBoot 3.0】 新特性——内置声明式HTTP客户端_springboot3.0 内置http服务-CSDN博客](http://cxyroad.com/"https://blog.csdn.net/m0_63947499/article/details/131812612")

[七大主流的HttpClient程序比较](http://cxyroad.com/"https://zhuanlan.zhihu.com/p/269208311")

1、Java HttpURLConnection

`JDK`自带的标准`HTTP`客户端`API`，尽管功能相对基础，但因为内置于`JDK`中，无需额外依赖，所以在许多简单场景下仍然被广泛使用。

2、Java11 HttpClient

自`Java 11`起，`JDK`新增了一个更现代化且功能更完善的`HTTP`客户端`API`，旨在替代原有的`HttpURLConnection`，支持`HTTP/2`和异步操作。

3、Apache HttpClient

由`Apache`软件基金会提供的成熟的`HTTP`客户端库，支持同步和异步操作，具备高度可配置性，提供了强大的连接管理和认证支持。

`SpringBoot`下配合`RestTemplate`组件使用示例如下。

```
@Configuration
public class HttpClientConfig {

    @Bean
    public HttpClient httpClient() {
        PoolingHttpClientConnectionManager connectionManager = new PoolingHttpClientConnectionManager();
        // 设置最大连接数
        connectionManager.setMaxTotal(200);
        // 每个路由的基础连接数
        connectionManager.setDefaultMaxPerRoute(20);

        // 其他可能的配置选项...

        CloseableHttpClient httpClient = HttpClients.custom()
                .setConnectionManager(connectionManager)
                // 可以设置重试策略、超时等其他参数
                .build();

        return httpClient;
    }
```

```
    @Bean
    public HttpComponentsClientHttpRequestFactory
clientHttpRequestFactory(HttpClient httpClient) {
        HttpComponentsClientHttpRequestFactory factory = new
HttpComponentsClientHttpRequestFactory();
        factory.setHttpClient(httpClient);
        return factory;
    }

    @Bean
    public RestTemplate
restTemplate(HttpComponentsClientHttpRequestFactory factory) {
        return new RestTemplate(factory);
    }
}
```

4、OkHttp
========

由`Square`公司开发的高性能`HTTP`客户端库，尤其适合移动和服务器端环境，内置连接池和失败重试机制，支持`HTTP/2和SPDY`协议，同时也支持同步和异步`API`。

简单示例如。

```
@Configuration
public class OkHttpConfig {
   @Bean
   public OkHttpClient okHttpClient() {
       return new OkHttpClient.Builder()
               .connectTimeout(10, TimeUnit.SECONDS)
               .readTimeout(10, TimeUnit.SECONDS)
               .build();
   }
}
```

分界
==

上面的都属于底层`Http`类库，基于以上还有很多组件可以使用。既然是组件，就是为了简化调用的，所以使用起来参看`Java Doc`就可以了

5、Retrofit

虽然`Retrofit`最初是为`Android`设计，但同样可以在`Spring Boot`环境中使用。`Retrofit`基于`OkHttp`之上提供了更为简洁优雅的`API`设计，通过注解的方式将接口与`HTTP`请求映射，大大简化了`RESTful API`的调用过程。结合`Gson`或其他`JSON`转换器，它可以轻松地处理`JSON`格式的数据交换。在`Spring Boot`项目中，尽管`Retrofit`主要面向`REST`服务消费，但它也可以与`Spring`框架紧密结合，通过自定义适配器来更好地融入`Spring`生态环境。

6、SpringBoot RestTemplate

`Spring Boot` 自带的 `RestTemplate` 是一个轻量级且全面的`HTTP`客户端，内置在`Spring`框架中。它简化了与`RESTful`服务之间的交互，提供了丰富的模板方法来执行`GET`、`POST`等各种`HTTP`操作，并能够自动转换`HTTP`响应为`Java`对象。然而，`RestTemplate` 在并发场景下性能并不突出，且不具备连接池和流式`API`支持，对于复杂的`HTTP`调用可能需要更多手动配置。

7、Spring5 WebClient

自`Spring 5`引入`WebFlux`之后，`WebClient` 成为了非阻塞式的`HTTP`客户端首选。其基于`Reactor`项目，完全支持异步和响应式编程模型，允许构建高效、可扩展的服务。`WebClient` 提供了强大的`API`来处理各种`HTTP`请求和响应，包括流式处理、多路复用以及灵活的背压控制，尤其适合高并发和低延迟的应用场景。

8、Spring Cloud Feign

`Spring Cloud Netflix Feign`是一个声明式的伪客户端，它使编写`Web`服务客户端变得更简单。

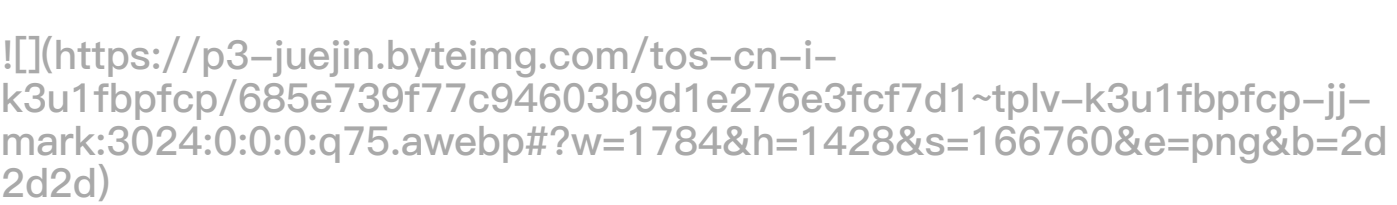
它允许你定义一个接口并使用注解来指定`HTTP`方法、`URL`模板以及参数绑定，然后通过动态代理生成实现类来进行远程调用。

适合微服务架构下服务间通信，具有良好的可读性和一致性。

9、Spring6.1 RestClient
==========

官方描述：`RestClient`是一个同步`HTTP`客户端，它提供了一个现代、流畅的`API`。它提供了对`HTTP`库的抽象，允许从`Java`对象到`HTTP`请求的方便转换，以及从`HTTP`响应创建对象。

10、Spring6 HTTP interface
==========



官方描述：`Spring Framework`允许您使用`@HttpExchange`方法将`HTTP`服务定义为`Java`接口。您可以将这样的接口传递给`HttpServiceProxyFactory`，以创建一个代理，该代理通过HTTP客户端（如`RestClient`或`WebClient`）执行请求。您还可以从`@Controller`实现用于服务器请求处理的接口。

简单的来讲，可以类比为`OpenFeign`，使用方法是几乎一样的。

小结
==

综上，关于底层的`Http`类库，比较推荐`Apache HttpClient`和`OkHttp`。关于上层组件，如果追求现代非阻塞`IO`及响应式编程，`WebClient`是首选；如果偏好简单的同步操作和`Spring`框架的无缝集成，`RestTemplate`是基础选项，`Spring6`可以使用`RestClient`；而对于更高级别的`API`调用抽象和微服务架构，`Feign`或`Retrofit`则是值得考虑的候选者，当然未来`HTTP interface`有可能更加通用。

写在最后

拙作艰辛，字句心血，望诸君垂青，多予支持，不胜感激。

个人博客：[无奈何杨（wnhyang）](http://cxyroad.com/”https://wnhyang.github.io/”)

个人语雀：[wnhyang](http://cxyroad.com/”https://www.yuque.com/wnhyang”)

共享语雀：[在线知识共享](http://cxyroad.com/”https://www.yuque.com/wnh”)

Github：[wnhyang - Overview](http://cxyroad.com/”https://github.com/wnhyang”)

原文链接: https://juejin.cn/post/7350199693753581619