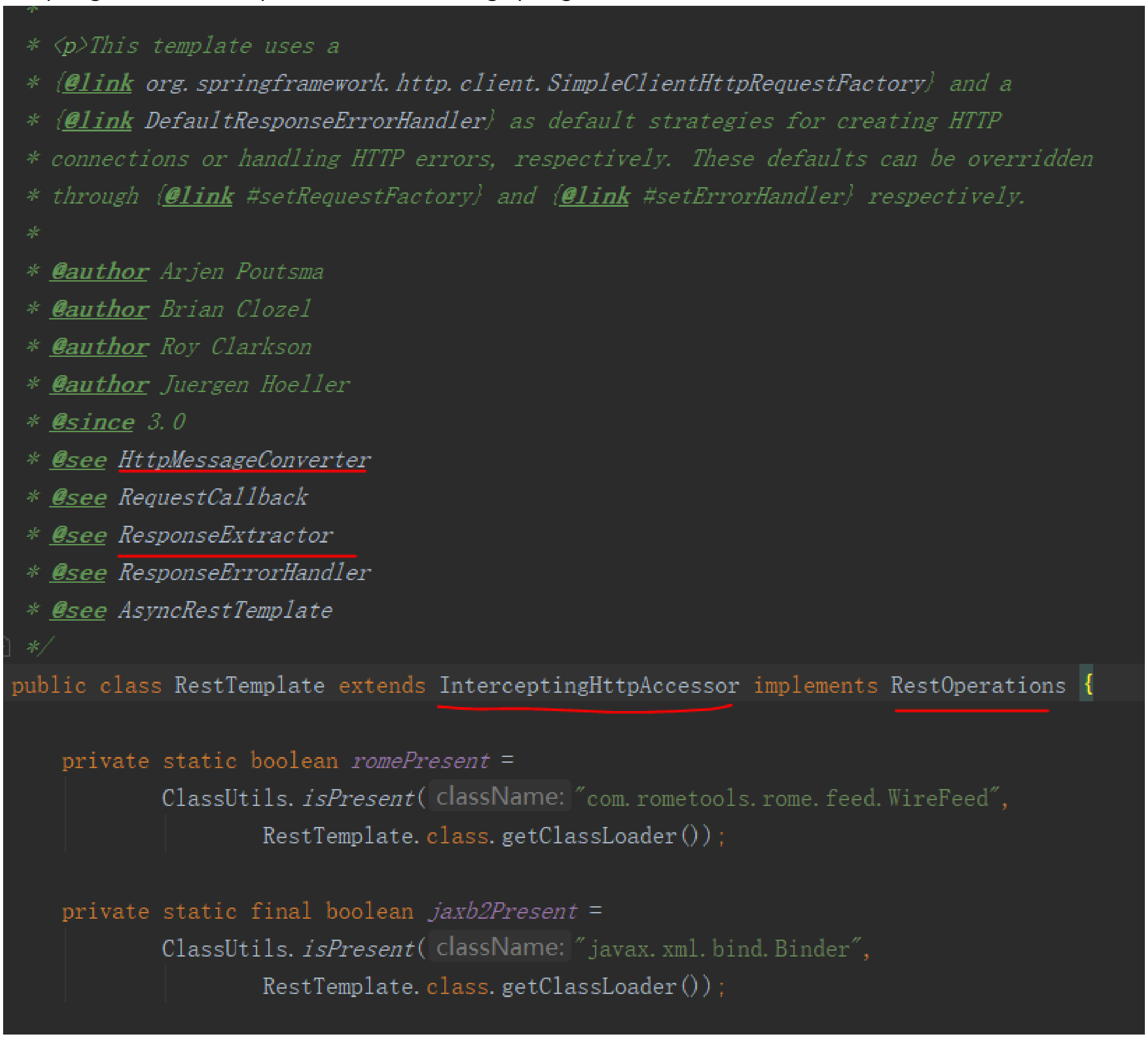
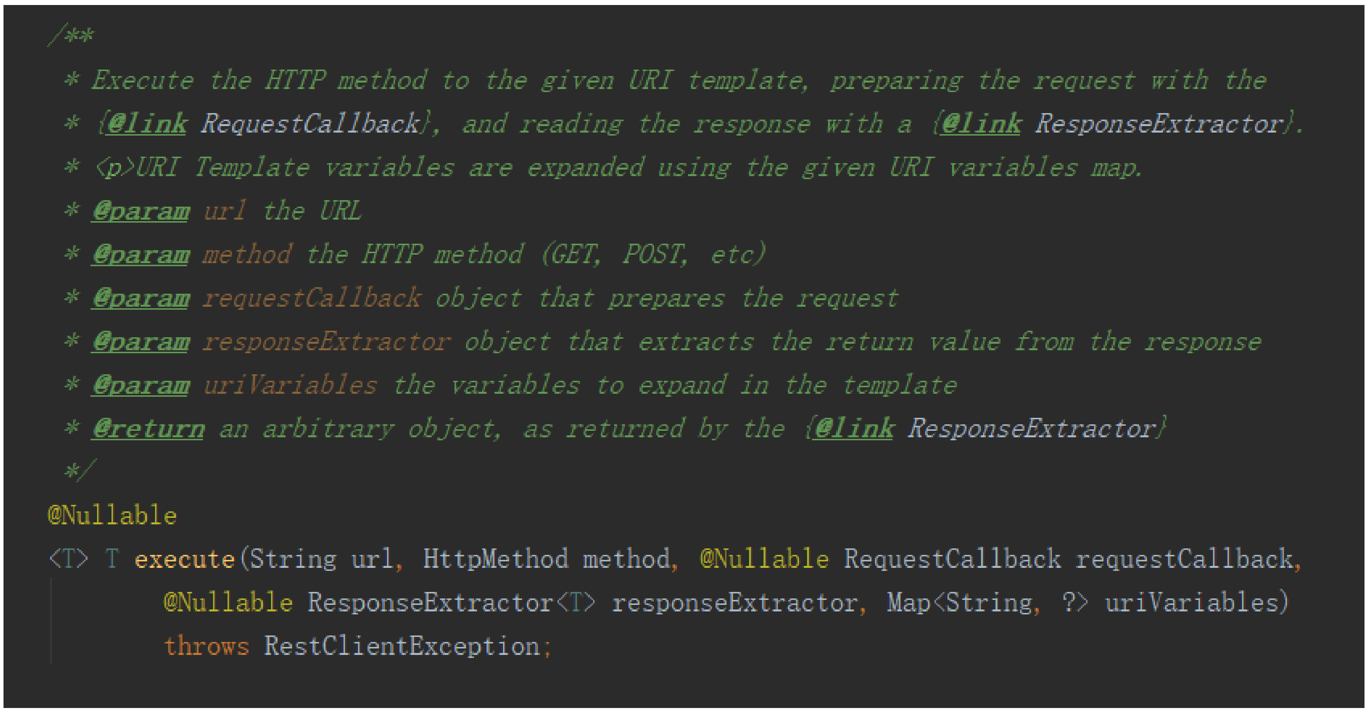
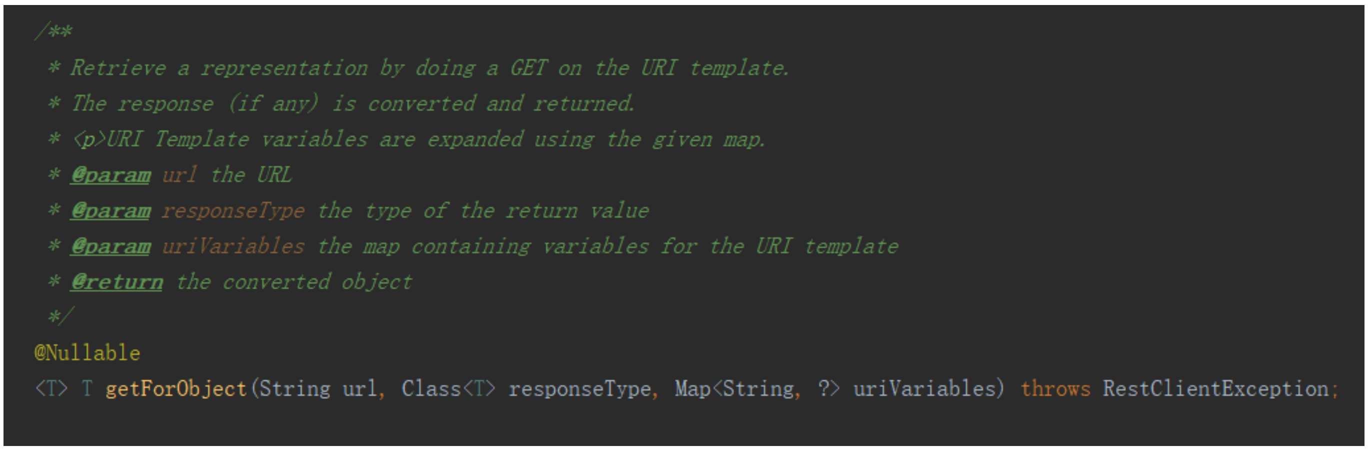
1. 由 spring框架提供的http请求工具 ，属于 org.springframework.web.client 包
2. 

```
 *
 * <p>This template uses a
 * {@link org.springframework.http.client.SimpleClientHttpRequestFactory} and a
 * {@link DefaultResponseErrorHandler} as default strategies for creating HTTP
 * connections or handling HTTP errors, respectively. These defaults can be overridden
 * through {@link #setRequestFactory} and {@link #setErrorHandler} respectively.
 *
 * @author Arjen Poutsma
 * @author Brian Clozel
 * @author Roy Clarkson
 * @author Juergen Hoeller
 * @since 3.0
 * @see HttpMessageConverter
 * @see RequestCallback
 * @see ResponseExtractor
 * @see ResponseErrorHandler
 * @see AsyncRestTemplate
 */
public class RestTemplate extends InterceptingHttpAccessor implements RestOperations {

    private static boolean romePresent =
            ClassUtils.isPresent( className: "com.rometools.rome.feed.WireFeed",
                    RestTemplate.class.getClassLoader());

    private static final boolean jaxb2Present =
            ClassUtils.isPresent( className: "javax.xml.bind.Binder",
                    RestTemplate.class.getClassLoader());
```

```
org.springframework.web.client.RestOperations  -  定义http请求的核心接口
```

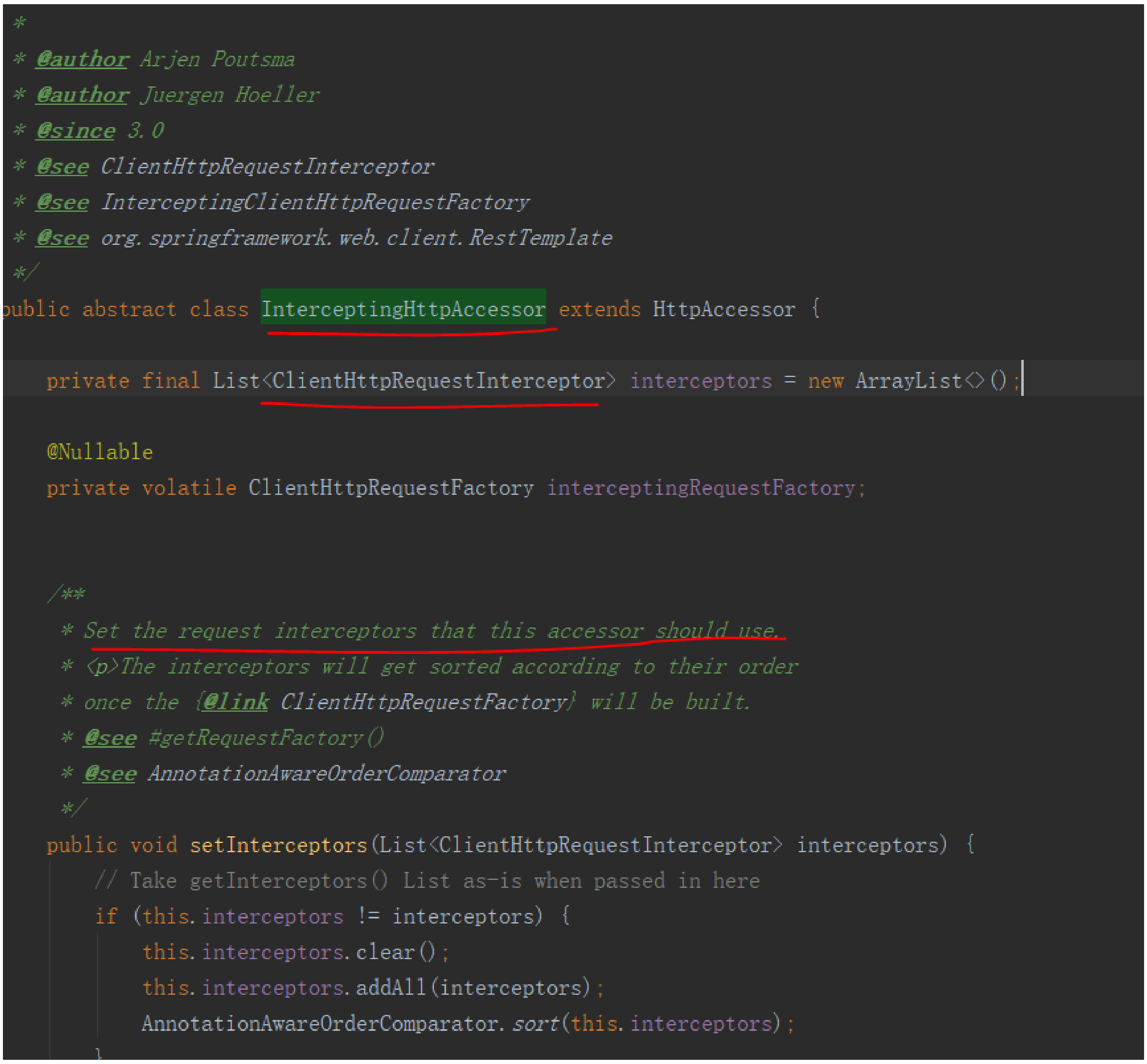


```java
/**
 * Execute the HTTP method to the given URI template, preparing the request with the
 * {@link RequestCallback}, and reading the response with a {@link ResponseExtractor}.
 * <p>URI Template variables are expanded using the given URI variables map.
 * @param url the URL
 * @param method the HTTP method (GET, POST, etc)
 * @param requestCallback object that prepares the request
 * @param responseExtractor object that extracts the return value from the response
 * @param uriVariables the variables to expand in the template
 * @return an arbitrary object, as returned by the {@link ResponseExtractor}
 */
@Nullable
<T> T execute(String url, HttpMethod method, @Nullable RequestCallback requestCallback,
        @Nullable ResponseExtractor<T> responseExtractor, Map<String, ?> uriVariables)
        throws RestClientException;
```

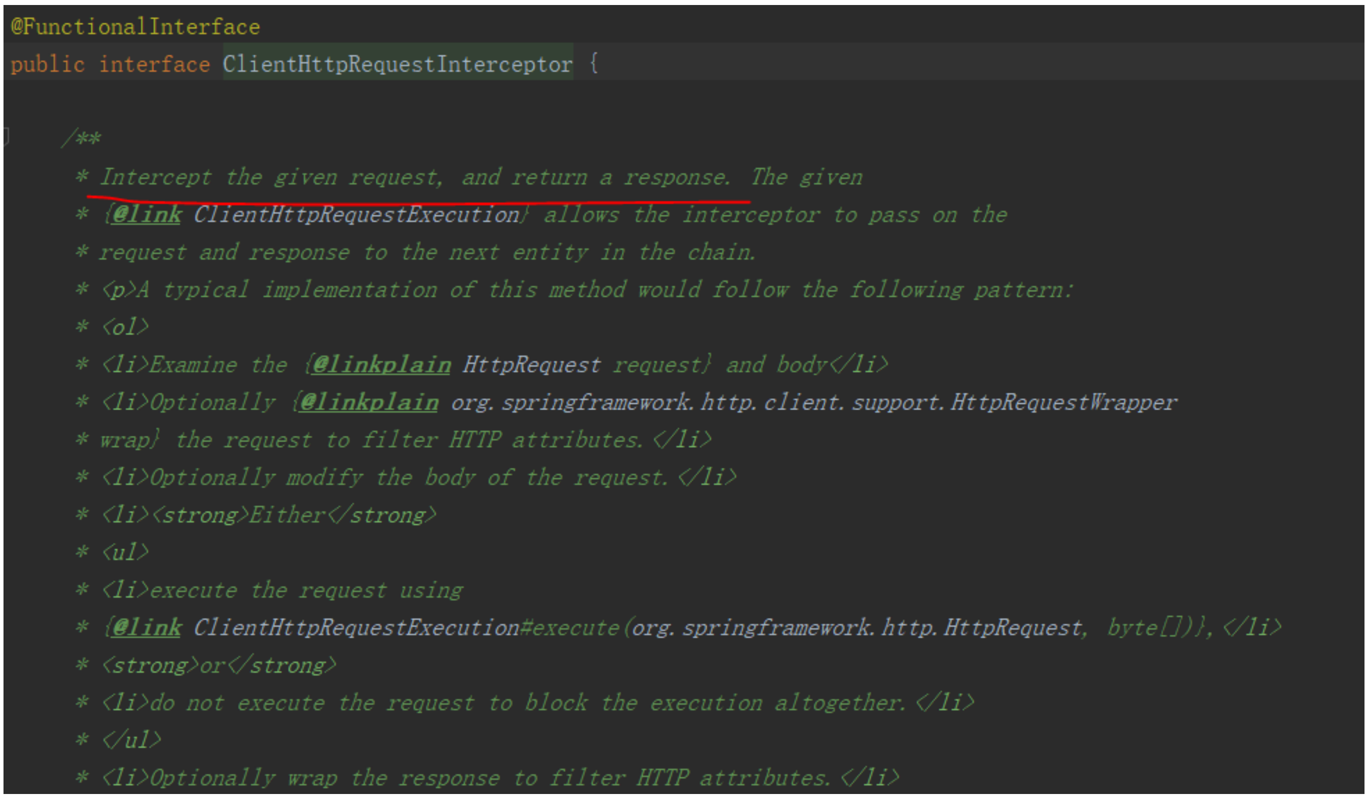


```java
/**
 * Retrieve a representation by doing a GET on the URI template.
 * The response (if any) is converted and returned.
 * <p>URI Template variables are expanded using the given map.
 * @param url the URL
 * @param responseType the type of the return value
 * @param uriVariables the map containing variables for the URI template
 * @return the converted object
 */
@Nullable
<T> T getForObject(String url, Class<T> responseType, Map<String, ?> uriVariables) throws RestClientException;
```

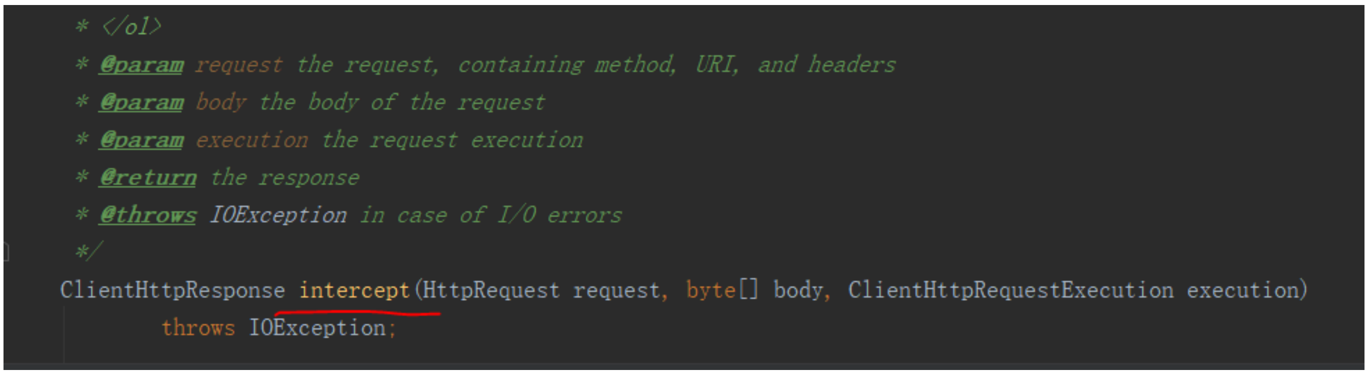
```
org.springframework.http.client.support.InterceptingHttpAccessor
ClientHttpRequestInterceptor - 微服务的核心接口，加载拦截器，对服务地址进行预处理
```



```java
 *
 * @author Arjen Poutsma
 * @author Juergen Hoeller
 * @since 3.0
 * @see ClientHttpRequestInterceptor
 * @see InterceptingClientHttpRequestFactory
 * @see org.springframework.web.client.RestTemplate
 */
public abstract class InterceptingHttpAccessor extends HttpAccessor {

    private final List<ClientHttpRequestInterceptor> interceptors = new ArrayList<>();

    @Nullable
    private volatile ClientHttpRequestFactory interceptingRequestFactory;


    /**
     * Set the request interceptors that this accessor should use.
     * <p>The interceptors will get sorted according to their order
     * once the {@link ClientHttpRequestFactory} will be built.
     * @see #getRequestFactory()
     * @see AnnotationAwareOrderComparator
     */
    public void setInterceptors(List<ClientHttpRequestInterceptor> interceptors) {
        // Take getInterceptors() List as-is when passed in here
        if (this.interceptors != interceptors) {
            this.interceptors.clear();
            this.interceptors.addAll(interceptors);
            AnnotationAwareOrderComparator.sort(this.interceptors);
        }
```



```java
@FunctionalInterface
public interface ClientHttpRequestInterceptor {

    /**
     * Intercept the given request, and return a response. The given
     * {@link ClientHttpRequestExecution} allows the interceptor to pass on the
     * request and response to the next entity in the chain.
     * <p>A typical implementation of this method would follow the following pattern:
     * <ol>
     * <li>Examine the {@linkplain HttpRequest request} and body</li>
     * <li>Optionally {@linkplain org.springframework.http.client.support.HttpRequestWrapper
     * wrap} the request to filter HTTP attributes.</li>
     * <li>Optionally modify the body of the request.</li>
     * <li><strong>Either</strong>
     * <ul>
     * <li>execute the request using
     * {@link ClientHttpRequestExecution#execute(org.springframework.http.HttpRequest, byte[])},</li>
     * <strong>or</strong>
     * <li>do not execute the request to block the execution altogether.</li>
     * </ul>
     * <li>Optionally wrap the response to filter HTTP attributes.</li>
```



```
	 * </ol>
	 * @param request the request, containing method, URI, and headers
	 * @param body the body of the request
	 * @param execution the request execution
	 * @return the response
	 * @throws IOException in case of I/O errors
	 */
	ClientHttpResponse intercept(HttpRequest request, byte[] body, ClientHttpRequestExecution execution)
			throws IOException;
```

## spring cloud 与 restTemplate

```
spring cloud 通过对 restTemplate中的InterceptingHttpAccessor 进行处理
```