**Step 1 : Install the client**

For the feign cline to work we need Okhttp and a HTTP client. Add the below script to the build.gradle

compile group: 'org.springframework.cloud', name: 'spring-cloud-starter-openfeign', version: '2.2.3.RELEASE'

compile 'org.apache.httpcomponents:httpclient:4.5.12'

implementation 'com.squareup.okhttp3:okhttp:4.7.2'

**Step 2 : Include Feign client on the project**

On our main Application add *@EnableFeignClients* Along with the spring annotations

@SpringBootApplication

@EnableFeignClients

public class OpenFeignApplication {

public static void main(String[] args) {

SpringApplication.run(OpenFeignApplication.class, args);

}

}

**Step 3 : Creating the Configuration**

We can configure the client either via properties file configuration or by the Java configuration. Create a configuration file and add the below contents

@Configuration

public class ClientConfiguration {

//Documentation https://cloud.spring.io/spring-cloud-netflix/multi/multi\_spring-cloud-feign.html

@Bean

public Logger.Level feignLoggerLevel() {

return Logger.Level.FULL;

}

@Bean

public ErrorDecoder errorDecoder() {

//Can use our own if required -> new CustomErrorDecoder()

return new CustomErrorDecoder();

//return new ErrorDecoder.Default();

}

@Bean

public OkHttpClient client() {

return new OkHttpClient();

}

@Bean

public RequestInterceptor requestInterceptor() {

return requestTemplate -> {

requestTemplate.header("Accept", ContentType.APPLICATION\_JSON.getMimeType());

};

}

}

**feignLoggerLevel** : To configure the log. We have 4 options (None,Basic, Headers,Full)

**errorDecoder** : To customise handling the error callbacks. If you want to use default use ErrorDecoder.Default()

**Client** : The HTTP Client

**requestInterceptor** : Customise the request before sending. We can use this to add our own logic, custom headers

**Step 4 : Creating the Client**

Now we have to create a client. See a demo client below

@FeignClient(value = "demo-client",

url = "${feign.client.url}",

configuration = ClientConfiguration.class,

fallback = JSONPlaceHolderFallback.class)

public interface DemoClient {

@GetMapping("/posts")

List < Post > getPosts(@RequestHeader(value = "Authorization", required = true) String authorizationHeader);

@GetMapping("/test")

ResponseEntity < Void > getTest();

}

The client should have the FeignClient annotation. Each feign client is part of an ensemble of components that work together to contact a remote server on demand, and the ensemble has a name that you give it as an application developer using the *@FeignClient* annotation

Spring Cloud creates a new ensemble as an ApplicationContext on demand for each named client using *@FeignClientsConfiguration*. This contains (amongst other things) an feign.Decoder, a feign.Encoder, and a feign.Contract.

On the annotation we have option to configure the configuration we created on the step 3

**Url** : it is the base url for the feign client. Here the url is configurable via application.properties

**Step 5 : Integration on Controller.**

This is straightforward. Add the client to the controller with the spring Autowired annotation.

On the controller function can use the demo client instance to call the required method

@Autowired

private DemoClient demoClient;

ResponseEntity<Void> result = demoClient.getTest();

**Step 6 : Error Handling.**

You can create the custom error handler by implementing the ErrorDecoder.

public class CustomErrorDecoder implements ErrorDecoder {

@Override

public Exception decode(String methodKey, Response response) {

if (response.status() >= 400 && response.status() <= 499) {

return new ClientException(

response.status(),

response.reason()

);

}

if (response.status() >= 500 && response.status() <= 599) {

return new ServerException(

response.status(),

response.reason()

);

}

return errorStatus(methodKey, response);

}

}

**Step 4 : Handling Exception in Controller.**

Use try catch to handle in the controller. Code below

try {

ResponseEntity<Void> result = demoClient.getTest();

System.*out*.println(result.getStatusCode());

}catch (Exception exception){

if(exception.getCause() instanceof ClientException){

System.*out*.println("Error Code : =" + ((ClientException) exception.getCause()).getStatus());

}

}