OBJECT
COMPUTING
HOME TO GRAILS & MICRONAUT

WEBINAR

Micronaut Declarative HTTP Client

Presented By: Puneet Behl

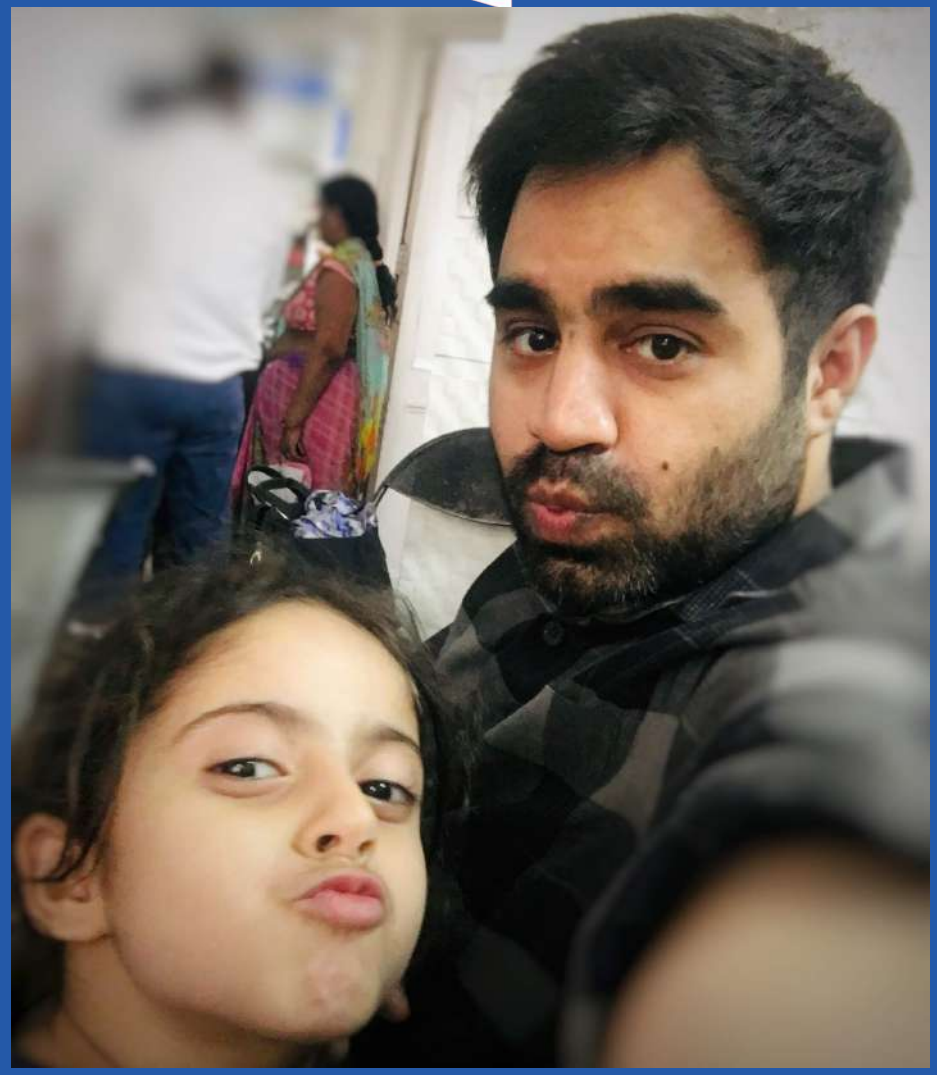
MICRONAUT

About Me

Software Engineer

Member of 2GM Team

@puneetbhl



Motivation

OBJECT COMPUTING

The motivation behind creating a new HTTP client for Micronaut centers around providing a rich set of cloud native features that in total did not exists in any solution at that time.

A non-blocking HTTP client based on Netty that has integration with service-discovery, load-balancing and many more other features is the result.

Dependency

To use the HTTP client, just add the following dependency:

build.gradle

```
implementation “io.micronaut:micronaut-http-client"
```

pom.xml

```
<dependency>
  <groupId>io.micronaut</groupId>
  <artifactId>micronaut-http-client</artifactId>
  <scope>compile</scope>
</dependency>
```

Usage

To inject the client use @Client with @Inject annotation with a URL of the API

```
@Client("https://newsapi.org") @Inject RxHttpClient rxHttpClient;

rxHttpClient.retrieve(HttpRequest.GET("/v2/top-headlines?country=in")
                .header("X-Api-Key", "API_KEY_VALUE"), Argument.of(Map.class));
```

```
@Singleton
public class NewsService {

    private final HttpClient httpClient;

    public NewsService(ApplicationContext applicationContext) {
        this.httpClient = applicationContext.createBean(HttpClient.class, "/news");
    }
}
```

Create the Client via ApplicationContext

HttpRequest Factory

METHOD	DESCRIPTION	ALLOWS BODY
HttpRequest.GET(java.lang.String)	Constructs an HTTP GET request	FALSE
HttpRequest.OPTIONS(java.lang.String)	Constructs an HTTP OPTIONS request	FALSE
HttpRequest.HEAD(java.lang.String)	Constructs an HTTP HEAD request	FALSE
HttpRequest.POST(java.lang.String,T)	Constructs an HTTP POST request	TRUE
HttpRequest.PUT(java.lang.String,T)	Constructs an HTTP PUT request	TRUE
HttpRequest.PATCH(java.lang.String,T)	Constructs an HTTP PATCH request	TRUE
HttpRequest.DELETE(java.lang.String)	Constructs an HTTP DELETE request	TRUE

Errors


```
@Client("http://localhost:8080") RxHttpClient rxHttpClient;

String uri = UriBuilder.of("/pet/{name}")
        .expand(Collections.singletonMap("name", "dodo"))
        .toString();

Assertions.assertThrows(HttpClientResponseException.class, () -> {
    rxHttpClient.toBlocking().retrieve(uri, Pet.class);
});
```

Debugging

```
<logger name="io.micronaut.http.client" level="TRACE" />
```

```
micronaut:
  http:
    client:
      logger-name: mylogger
    services:
      other-client:
        logger-name: other.client
```

Client Specific Logging Configurations

```
<logger name="mylogger" level="DEBUG"/>
<logger name="other.client" level="TRACE"/>
```

Demo

```
String uri = UriBuilder.of("/v2/top-headlines")
        .queryParam("country", "in")
        .queryParam("apiKey", NEWS_API_KEY)
        .build()
        .toString();

Assertions.assertEquals("/v2/top-headlines?
country=in&apiKey=837fe7ca80a34f48bdbdd2c64c9d91ad", uri);

Map newsApiResult = rxHttpClient.toBlocking().retrieve(uri, Map.class);

Assertions.assertEquals(
        "ok",
        newsApiResult.get("status")
);
```

A Basic HTTP Client Request

Declarative Client

OBJECT COMPUTING

The declarative client can be created by annotating any interface or abstract class with @Client annotation.

```
@Client("https://newsapi.org")
public interface NewsClient {

    @Get("/v2/top-headlines{?country}")
    Map fetchHeadlines(@QueryValue String country);
}
```

@QueryValue is Optional

Client Configurations

1. Proxy
2. Thread Pool
3. Timeouts
4. Follow Redirects
5. Logging
6. SSL
7. Connection Pooling
8. More…

Validation

OBJECT COMPUTING

```
implementation "io.micronaut:micronaut-validation"
```

```
@Post
Single<Pet> save(@NotBlank String name, @Min(1L) int age);
```

JSON Streaming


```
@Client("/news")
public interface NewsSteamingClient {

    @Get(value = "/headlines/{country}", processes = MediaType.APPLICATION_JSON_STREAM)
    Flowable<NewsArticle> headlines(@PathVariable String country);
}
```

Customizing Headers

OBJECT COMPUTING

```
@Client("https://newsapi.org")
@Header(name = "X-Api-Key", value = "${news.api.key}")
public interface NewsClient {

    @Get(value = "/v2/top-headlines", processes = MediaType.APPLICATION_JSON)
    @Override
    Flowable<HttpResponse<NewsArticleResult>> fetchLiveTopAndBreakingNewsHeadlines();

}
```

Alternatively you can supply a NEWS_API_KEY environment variable and the value will be populated

```
news:
  api:
    key: 837fe7ca80a34f48bdbdd2c64c9d91ad
```

Client Filter

OBJECT COMPUTING

```
@Filter("/v2/top-headlines/**")
public class NewsApiFilter implements HttpClientFilter {

    private String apiKey;

    public NewsApiFilter(@Value("${news.api.key}") String apiKey) {
        this.apiKey = apiKey;
    }

    @Override
    public Publisher<? extends HttpResponse<?>> doFilter(MutableHttpRequest<?>
request, ClientFilterChain chain) {
        return chain.proceed(request.header("X-Api-Key", apiKey));
    }

}
```

Multipart Upload

OBJECT COMPUTING

```
MultipartBody multipartBody = MultipartBody.builder()
    .addPart("file", "dodo.jpg", new File("dodo.jpg"))
    .build()


@Client("http://localhost:8080/pet")
public interface PetClient {


    @Post(uri = "/image", produces = MediaType.MULTIPART_FORM_DATA)
    Single<HttpResponse<String>> uploadImage(@Body MultipartBody body);


}
```

Demo

Retry

OBJECT COMPUTING

Enable retries with a simple annotation. If all attempts fail the original exception is thrown.

```
@Retryable(delay = "2s", attempts = "3")
@Client("http://localhost:8080/pet")
public interface PetClient extends PetApi {

    @Override
    @Get("{name}")
    Single<HttpResponse<Pet>> get(@PathVariable String name);

}
```

Circuit Breaker

OBJECT COMPUTING

Same thing as @Retry but with reset option

```
@CircuitBreaker(delay = "2s", attempts = "3")
@Client("http://localhost:8080/pet")
public interface PetClient extends PetApi {

    @Override
    @Get("{name}")
    Single<HttpResponse<Pet>> get(@PathVariable String name);
```

Fallback

OBJECT COMPUTING

The @Client annotation is annotated with @Recoverable

When an exception is thrown, a fallback is searched for and executed

```
@Fallback
public class PetFallback implements PetApi {

    private static final Logger LOG = LoggerFactory.getLogger(PetFallback.class);
    public static final int FALLBACK_PET_AGE = 10;

    @Override
    public Single<HttpResponse<Pet>> get(String name) {
        if (LOG.isDebugEnabled()) {
            LOG.debug("Fallback called for PetApi get method call");
        }
        return Single.just(HttpResponse.ok(new Pet(name, FALLBACK_PET_AGE)));
    }
}
```

Demo

Service Discovery

```
implementation ‘io.micronaut:micronaut-discovery-client'
```

```
@Client("pets-server")
public interface PetClient {

  ...

}
```

Load Balancing

Round robin by default

Static list or service discovery

Provide your own implementation through the LoadBalancer interface

Tracing

All tracing integrations automatically support the HTTP client; declarative and imperative styles.

Zipkin, Jaeger

https://guides.micronaut.io/tags/distributed-tracing.html

Demo

Using Micronaut HTTP Client with Spring


```
@Client("http://localhost:8080")
public interface PetClient {

    @GetMapping(“/pet/{name}")
    public Pet get(@PathVariable String name)

}
```

Using Micronaut HTTP Client with Grails

Micronaut is the parent application context of Grails 4 so you can take advantage of many Micronaut features including HTTP Client.

```
@Client("https://start.grails.org")
interface GrailsAppForgeClient {

    @Get("/{version}/profiles")
    List<Map> profiles(String version)
}
```

OBJECT
COMPUTING

Questions?

Micronaut Resources

- gitter.im/micronautfw
- docs.micronaut.io
- guides.micronaut.io
- micronaut.io/faq.html
- github.com/micronaut-projects/micronaut-core
- github.com/micronaut-projects/micronaut-examples
- objectcomputing.com/products/micronaut

LEARN MORE ABOUT OCI EVENTS AND TRAINING

OBJECT COMPUTING
HOME TO GRAILS & MICRONAUT

Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a private tech talk or custom workshop for your team.
We can deliver these events on site, online, or in our state-of-the-art, Midwest training lab.

OBJECT COMPUTING

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com