Consuming REST Services

RestClient

Using the RestClient

- RestClient was introduced in Spring 6.1
- RestClient is a new synchronous HTTP client
- Offers a fluent API, allows method chaining
- Can be created like this:

```
RestClient restClient = RestClient.builder()
        .baseUrl( baseUrl: "http://localhost:8080")
        .build();
```

RestClient example code

- Getting an object of type Customer.class by using the get method of the restClient to get a customer with a specific id from a REST Service with the URI /customer (compared to the RestTemplate below):

```
Customer customer = restClient.get()
        .uri( uri: "/customer/{id}",  ...uriVariables: id)
        .retrieve()
        .body( bodyType: Customer.class);
```

Method

URI

Type of response

```
Customer customer = restTemplate.getForObject( url: "http://localhost:8080/customer/" + id,  responseType: Customer.class);
```

Method

URI

Type of response

RestClient example code

- Posting an object of type Customer.class by using the post method of the restClient to post a customer object to a REST Service with the URI / customer (compared to the RestTemplate below):

```
restClient.post()
        .uri( uri: "/customer")
        .body( body: customer)
        .retrieve();
```

Method

URI

Object

```
restTemplate.postForObject( url: "http://localhost:8080/customer", request: customer, responseType: Customer.class);
```

Method URI Object Type of response

HTTP methods

- RestClient methods for sending different HTTP methods:
- POST - to Create - restClient.post()
- GET - to Read - restClient.get()
- PUT - to Update - restClient.put()
- DELETE - to Delete - restClient.delete()

Demo/Exercise - RestClient

- Download and open the ConsumingRESTRestClientStarter project
- It’s a working application. Try it out and look at the code and comments
- All calls to the DogRepository should be replaced by calls made by a RestClient to the REST Service created in the exercise for Creating REST Services
- A RestClient is already declared in the Controller class. Run the REST Service in another IntelliJ window on port 8080, this application will run on port 8081. Try it out when you have switched from repository to REST with the RestClient!