MicroProfile Rest Client

This cheat sheet covers the basics of MicroProfile Rest Client specification uses.

CLENT DEFINITION EXAMPLES

```
public interface MyServiceClient {
    @GET
    @Path("/greet")
    Response greet();
}

@Path("/users")
@Produces("application/json")
@Consumes("application/json")
public interface UsersClient {
    @OPTIONS
    Response options();

    @HEAD
    Response head();

    @GET
    List<User> getUsers();

    @GET
    @Path("/{userId}")
    User getUser(@PathParam("userId") String userId);

    @HEAD
    @Path("/{userId}")
    Response headUser(@PathParam("userId") String
    userId);

    @POST
    Response createUser(@HeaderParam("Authorization")
    String authorization, User user);

    @DELETE
    @Path("/{userId}")
    Response deleteUser(@CookieParam("AuthToken")
    String authorization,
                        @PathParam("userId") String
                         userId);
}
```

CLIENT HEADER PARAMS

The `@ClientHeaderParam` annotation can allow users to specify HTTP headers that should be sent without altering the client interface method signature.

override a single header value

```
@POST
@ClientHeaderParam(name="X-Http-Method-Override",
value="PUT")
Response sentPUTviaPOST(MyEntity entity);
```

delagate header computation to a default method

```
@POST
@ClientHeaderParam(name="X-Request-ID",
value="{generateRequestId}")
Response postWithRequestId(MyEntity entity);

default String generateRequestId() {
    return UUID.randomUUID().toString();
}
```

delegate header computation to a public static method

```
@GET
@ClientHeaderParam(name="CustomHeader",
value="{some.pkg.MyHeaderGenerator.generateCustomHe
ader}",required=false)
Response getWithoutCustomHeader();

public class MyHeaderGenerator {
      public static String
      generateCustomHeader(String headerName) {
        return "SomeValue";
      }
}
```

PROGRAMMATIC LOOKUP

Use `RestClientBuilder` class. For instance:

```
RemoteApi remoteApi =
RestClientBuilder.newBuilder()
    .baseUri(apiUri)
    .build(RemoteApi.class);
```

CDI SUPPORT

Rest Client interfaces may be injected as CDI beans. The runtime must create a CDI bean for each interface annotated with `@RegisterRestClient`.

```
@RegisterRestClient(baseUri="http://someHost/
someContextRoot")
public interface MyServiceClient {
    @GET
    @Path("/greet")
    Response greet();
}

@ApplicationScoped
public class MyService {
    @Inject
    @RestClient
    private MyServiceClient client;
}
```

Note: Note the `@RestClient` qualifier which is required when multiple CDI beans of the injected types exist.

INTEGRATION WITH MICROPROFILE CONFIG

supported values

`/com.mycompany.remoteServices.MyServiceClient/mp-rest/url` / base URL

`com.mycompany.remoteServices.MyServiceClient/mp-rest/uri` / base URI

`com.mycompany.remoteServices.MyServiceClient/mp-rest/scope` / fully qualified classname to a CDI scope

`com.mycompany.remoteServices.MyServiceClient/mp-rest/providers` / a comma separated list of fully-qualified provider classnames to include in the client

`com.mycompany.remoteServices.MyServiceClient/mp-rest/connectTimeout` / timeout to connect to the remote endpoint

`com.mycompany.remoteServices.MyServiceClient/mp-rest/readTimeout` / timeout to wait for a response

configuration keys

Config keys are specified in the `@RegisterRestClient` annotation and can be used in place of the fully-qualified classname in MicroProfile Config. For instance:

```
@RegisterRestClient(configKey="myClient")
public interface MyServiceClient {
    @GET
    @Path("/greet")
    Response greet();
}
```

means that you can use config properties like `myClient/mp-rest/url`, `myClient/mp-rest/uri`, `myClient/mp-rest/scope`.

PROVIDER DECLARATIONS

three options

- Through `RestClientBuilder` methods
- Through `@RegisterProvider` annotation
- Through `RestClientBuilderListener` or `RestClientListener` SPI

response exception mappers

Specific provider to the MicroProfile Rest Client. This mapper will take a `Response` object retrieved via an invocation of a client and convert it to a `Throwable`, if applicable:

```
@RegisterRestClient(baseUri = "http://example.com")
@RegisterProvider(MyResponseExceptionMapper.class)
public interface ExampleClient {
    @GET
    @Path("/nonexistent")
    String get();
}

public class MyResponseExceptionMapper implements
ResponseExceptionMapper<SomeException> {
    @Override
    public SomeException toThrowable(Response
response) {
        return new SomeException();
    }
}
```

Author Martin Stefanko