Atbash MicroProfile rest client

Rudy De Busscher

Version 0.5.1, ??/??/2018

Table of Contents

Release notes

0.5.1

1. Included RestClientBuilder from MP rest Client 1.1 (to be able to define providers globally)

0.5

1. First release, compatible with Rest Client 1.0
2. Implementation using Java 7, compatible with Java SE and Java EE.

Usage

Java EE environment

Atbash MP Rest client can be used in any Java EE 7 or Java EE 8 certified application server.

Project setup

Add the following artefacts to the maven project file to have the Atbash Rest Client functionality

```
<dependency>
    <groupId>be.atbash.mp.rest-client</groupId>
    <artifactId>atbash-rest-client-impl</artifactId>
    <version>0.5</version>
</dependency>
```

```
<dependency>
    <groupId>be.atbash.config</groupId>
    <artifactId>geronimo-config</artifactId>
    <version>0.9.1</version>
</dependency>
```

```
<dependency>
    <groupId>org.apache.deltaspike.modules</groupId>
    <artifactId>deltaspike-partial-bean-module-impl</artifactId>
    <version>${deltaspike.version}</version>
</dependency>
```

These dependencies are required because

1. geronimo-config, MP Config is used to read the endpoint URL (see further on) and you can choose which MP config compliant implementation you like.

2. deltaspike is used for the proxy functionality. Dependency is set to provided so that developer can specify the version they like

Warning

Deltaspike 1.8.1 is required as a minimum due to the usage of for example DeltaSpikeProxyInvocationHandler.

Calling endpoint

Define the interface which describes the remote endpoint

```
@Path("/other")
@RegisterRestClient
@ApplicationScoped
public interface OtherService {
```

```
@Path("json/{parameter}")
@GET
Data sayHello(@PathParam("parameter") String parameter);
```

```
}
```

By specifying the *@RegisterRestClient* annotation, a CDI bean is created that can be injected and used to call the remote endpoint.

```
@ApplicationScoped
public class SomeService {
```

```
@Inject
@RestClient
private OtherService otherService;
```

```
    public void doSomething() {
       Data data = otherService.sayHello("Rudy");
    }
}
```

Rest client configuration

The most important part which is missing in the above construct, is the definition of the URL where the endpoint can be found.

By using atbash config, the property file (even YAML is supported) can be defined by the developer.

```
public class ClientBaseName implements BaseConfigurationName {
    @Override
    public String getBase() {
        return "client";
    }
}
```

And define this class for using the ServiceLoader mechanism (contents of file *src/main/resources/META-INF/services/be.atbash.config.spi.BaseConfigurationName*

```
be.atbash.mp.rest_client.demo.client.config.ClientBaseName
```

The URL value is then retrieved from the file *client.properties* (define within maven *src/main/resources* folder.

```
be.atbash.mp.rest_client.demo.client.OtherService/mp-
rest/url=http://localhost:8080/server/data
```

Java SE environment

Atbash MP Rest client can also be used in a Java SE 7 (or 8) environment with a JAX-RS client available.

Project setup

Add the following artefacts to the maven project file to have the Atbash Rest Client functionality

```
<dependency>
    <groupId>be.atbash.mp.rest-client</groupId>
    <artifactId>atbash-rest-client-impl</artifactId>
    <version>0.5</version>
</dependency>
```

```
<!-- need a JAX-RS client -->
<dependency>
    <groupId>org.glassfish.jersey.core</groupId>
    <artifactId>jersey-client</artifactId>
    <!-- Latest JAX-RS 2.0 version -->
    <version>2.25.1</version>
</dependency>
```

```
<dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-json-jackson</artifactId>
    <version>2.22</version>
</dependency>
```

```
<dependency>
    <groupId>be.atbash.config</groupId>
    <artifactId>geronimo-config</artifactId>
    <version>0.9.1</version>
</dependency>
```

```
<dependency>
    <groupId>org.apache.deltaspike.modules</groupId>
    <artifactId>deltaspike-partial-bean-module-impl</artifactId>
    <version>${deltaspike.version}</version>
</dependency>
```

These dependencies are required because

1. jersey-client, because we need a JAX-RS client since the MicroProfile Rest Client is build on top of that.
2. geronimo-config, because the Atbash implementation uses config for CDI implementation (TODO this requirement needs to be removed in next version in Java SE usage !!)
3. deltaspike is used for the proxy functionality. Dependency is set to provided so that developer can specify the version they like

Warning

Deltaspike 1.8.1 is required as a minimum due to the usage of for example DeltaSpikeProxyInvocationHandler.

Calling endpoint

Define the interface which describes the remote endpoint

```
@Path("/other")
public interface OtherService {
```

```
@Path("json/{parameter}")
@GET
Data sayHello(@PathParam("parameter") String parameter);
```

```
}
```

Just define the interface and the methods you like. No special annotations required. Rest client capable of calling remote endpoint can be retrieved programmatically.

```
public void doSomething() {
   OtherService otherService = AbstractRestClientBuilder.newBuilder()
        .baseUrl(new URL("http://localhost:8080/server/data"))
        .build(OtherService.class);
   Data data = otherService.sayHello("Rudy")
}
```

Rest client configuration

See Java EE use case for defining the file with the configuration values. But it can be empty as it is not used within Java SE.

In a future version, this requirement will be removed.

Global provider

With the help of the interface **org.eclipse.microprofile.rest.client.spi.RestClientBuilderListener**, we can specify the providers which are applied to all generated rest clients.

Define the implementing class through ServiceLoader configuration, and by implementing the method *onNewBuilder* have have the change of adding providers to the Rest client in a global way.

You no longer need to add the provider to each individual interface with the use of the @RegisterProvider annotation.