MicroProfile Rest Client

With MicroProfile Rest Client we can invoke RESTful services over HTTP, and in this tutorial we will create another service, which will call our Book Store application. Open a new terminal window and create a new MicroProfile Maven project using Kodnito MicroProfile Archetype, using the following command:

```
mvn archetype:generate -DarchetypeGroupId=com.kodnito
-DarchetypeArtifactId=kodnito-microprofile-archetype -DarchetypeVersion=1.0.1
-DgroupId=com.kodnito.bookstore.rest -DartifactId=book-store-client -Dversion=1.0
-SNAPSHOT
```

We should have 2 projects now. Now cd into book-store-client application and type mvn clean install to download and install dependencies. Open book-store-client application in your IDE and add the following dependencies to the pom.xml: In the build section add the TomEE Maven runtime:

```
<plugins>
    <plugin>
        <groupId>org.apache.tomee.maven</groupId>
        <artifactId>tomee-maven-plugin</artifactId>
        <version>${tomee.version}</version>
        <configuration>
            <tomeeVersion>${tomee.version}</tomeeVersion>
            <tomeeClassifier>microprofile</tomeeClassifier>
        </configuration>
    </plugin>
</plugins>
```

And in the properties section add the version for the TomEE:

```
<tomee.version>8.0.0-M3</tomee.version>
```

Your pom.xml should look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.kodnito.bookstore.rest</groupId>
    <artifactId>book-store-client</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>
    <dependencies>
        <dependency>
            <groupId>org.eclipse.microprofile</groupId>
            <artifactId>microprofile</artifactId>
            <version>2.0.1</version>
            <type>pom</type>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <finalName>restapi</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.tomee.maven</groupId>
                <artifactId>tomee-maven-plugin</artifactId>
                <version>${tomee.version}</version>
                <configuration>
                    <tomeeVersion>${tomee.version}</tomeeVersion>
                    <tomeeClassifier>microprofile</tomeeClassifier>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <properties>
        <tomee.version>8.0.0-M3</tomee.version>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <failOnMissingWebXml>false</failOnMissingWebXml>
    </properties>
</project>
```

Now in the terminal type `mvn clean install` to download dependencies. It's time to code our book-store-client service which we'll call our book-store. Inside `com.kodnito.bookstore.response` package create a new file called `BookResponse.java` and add the following.

```
package com.kodnito.bookstore.response;

public class BookResponse {
```

```
    private Long id;
    private String title;
    private String description;
    private String isbn;
    private String publisher;
    private String language;
    private String author;
    private float price;
    private int pages;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }

    public String getLanguage() {
```

```
        return language;
    }

    public void setLanguage(String language) {
        this.language = language;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    public int getPages() {
        return pages;
    }

    public void setPages(int pages) {
        this.pages = pages;
    }

}
```

The response from BookStore service will be mapped using this class. We will create two more files, create a new interface called `BookStoreService.java` inside `com.kodnito.bookstore.service` and add the following:

```
package com.kodnito.bookstore.service;

import com.kodnito.bookstore.response.BookResponse;
import java.util.List;
import javax.enterprise.context.Dependent;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;

@Dependent
@RegisterRestClient
@Path("books")
@Produces(MediaType.APPLICATION_JSON)
public interface BookStoreService {

    @GET
    public List<BookResponse> getAll();
}
```

Here we create an interface with method(s) that represent RESTful APIs endpoint, and we can use this interface to invoke, the remote service. Using `@Dependent` and `@RegisterRestClient` on the interface, will make that this interface will be mapped by the CDI. Next thing to do is to create a new resource that will use this interface and invoke our book-store service. Inside `com.kodnito.bookstore.rest` package create `BookStoreEndpoint.java` file and add the following:

```
package com.kodnito.bookstore.rest;

import java.net.MalformedURLException;
import java.net.URL;
import javax.enterprise.context.ApplicationScoped;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import org.eclipse.microprofile.rest.client.RestClientBuilder;
import com.kodnito.bookstore.service.BookStoreService;
import javax.ws.rs.core.Response;

@ApplicationScoped
@Path("/books")
public class BookStoreEndpoint {

    @Inject
    @RestClient
    private BookStoreService bookStoreService;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Response books() throws MalformedURLException {
        return Response.ok(bookStoreService.getAll()).build();
    }
}
```

This is almost identical to the one we have in our book-store application, when we invoke this endpoint on the book-store-client service, it will call the book-store service and retrieve all the books. Before we start the service, we need to add URL to the service we call to the `microprofile-config.properties` file.

```
com.kodnito.bookstore.service.BookStoreService/mp-
rest/url=http://localhost:8080/restapi
```

Now open a new terminal tab and start the `book-store` service first and when the service is up, navigate to the directory where you have the `book-store-client` application and start the application using `mvn clean package tomee:run -Dtomee-plugin.http=8081` and now open your browser and go to http://localhost:8081/restapi/books, and we can see that our services talks to each other.

Summary

In this chapter, we learned how to add MicroProfile Metrics to our application.