

Json Rest Client

Motivation

Moderne Server Applikationen bieten Json Rest Endpoints für die Interaktion mit Clients an. Json Rest Endpoints basieren in der Regel auf dem Http Protokoll. Java bietet viele verschiedene Varianten für die Programmierung von Http Clients an, die einfachste Variante bietet die Klasse `java.net.URL`. Den Einsatz von `java.net.URL` findet man unter dem Blog Mockito Unit Test mit Eclipse Maven und der Klasse `UrlFakeRandomImpl`. In diesem Blog verwenden wir den Java Standard `java.net.http.HttpClient`, welcher seit Java 11 vorhanden ist.

Json Random Rest Service

Für diesen Blog verwenden wir default den Json Random Rest Service Endpoint. Die Beschreibung hierzu findet man unter dem Link [Json Random Service](#).

`java.net.http.HttpClient`

```
Das folgende Listing zeigt die Klasse ch.std.blog.rest.client.JsonRestClient:
package ch.std.blog.rest.client;
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.util.HashMap;
import java.util.Map;
ch.std.blog.rest.json.JsonBodyHandler;
public class JsonRestClient {
    static void main(String[] args) throws IOException, InterruptedException {
        String url = "https://www.simtech-ag.ch/std-ajax/randomservice?min=0&max=1000";
        if (args.length > 1) {
            url = args[0];
        }
        // create a client
        HttpClient client = HttpClient.newHttpClient();
        // create a request
        HttpRequest request = HttpRequest.newBuilder(URI.create(url))
            .header("accept", "application/json")
            .build();
        // use the client to send the request
        var response = client.send(request, new JsonBodyHandler<Map<String, String>>());
        Map responseMap = response.body().get();
        System.out.println(responseMap);
    }
}
```

Json Body Handler

Der `HttpClient` übernimmt nur die Kommunikation via Http Protokoll. Die Auswertung der Json Response übernimmt die Klasse `ch.std.blog.rest.json.JsonBodyHandler`:

```
package ch.std.blog.rest.json;
import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.net.http.HttpResponse;
import java.util.function.Supplier;
import com.fasterxml.jackson.databind.ObjectMapper;
public class JsonBodyHandler<T> implements
    HttpResponse.BodyHandler<Supplier<T>>> {
    private static final ObjectMapper om = new ObjectMapper();
    private final Class<T> targetClass;
    public JsonBodyHandler(Class<T> targetClass) {
        this.targetClass = targetClass;
    }
    @Override
    public HttpResponse.BodySubscriber<Supplier<T>> apply(
        HttpResponse.ResponseInfo responseInfo) {
        return asJSON(this.targetClass);
    }
    public static <W> HttpResponse.BodySubscriber<Supplier<W>> asJSON(
        Class<W> targetType) {
        return new HttpResponse.BodySubscriber<InputStream>() {
            upstream =
                HttpResponse.BodySubscribers.ofInputStream();
            return
                HttpResponse.BodySubscribers.mapping(
                    upstream,
                    inputStream ->
                        toSupplierOfType(inputStream, targetType));
            public static
                <W> Supplier<W> toSupplierOfType(
                    InputStream inputStream,
                    Class<W> targetType) {
                    return () -> {
                        try {
                            InputStream stream = inputStream;
                            ObjectMapper objectMapper = new ObjectMapper();
                            return objectMapper.readValue(stream, targetType);
                        } catch (IOException e) {
                            throw new UncheckedIOException(e);
                        }
                    };
                }
        };
    }
}
```

Jackson ObjectMapper

Für die Konversion der Json Textdaten in die `java.util.Map` verwenden wir den Open Source Jackson ObjectMapper (`com.fasterxml.jackson.databind.ObjectMapper`). Hierzu haben wir im Eclipse Projekt die folgenden Jar Dateien referenziert: `jackson-annotations-2.12.3.jar`, `jackson-core-2.12.3.jar`, `jackson-databind-2.12.3.jar`. Die Dateien findet man im Eclipse Projekt oder über die Jar Download Site <https://jar-download.com>.

Komplettes Eclipse Projekt

Das komplette Eclipse Projekt findet man unter dem Link `jsonrestclient.zip`.

Feedback

War dieser Blog für Sie wertvoll. Wir danken für jede Anregung und Feedback

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz. Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsausschluss. Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by `stack.ch` written in Golang by Daniel Schmutz

<https://www.simtech-ag.ch/blog/java/jsonrestclient>