HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

Selbstdokumentierende REST-APIs mit OpenAPI

Eine Präsentation über die gleichnamige Studienarbeit im FWP-Fach

“Aktuelle Technologien zur Entwicklung verteilter Java-Anwendungen”

Am 14.06.2019

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Inhalt des Vortrags

1. Einführung
2. Microservices
3. REST
4. Schnittstellendokumentation
5. OpenAPI & Swagger
6. Contract First
    1. Quellcode-Generierung
7. Contract Last
    1. Springfox
    2. Springfox & Maven Model
8. Best Practices

https://www.vollmacht-muster.de/wp-content/uploads/Vollmacht-Inhalt-5-Punkte-e1416143958103.jpg

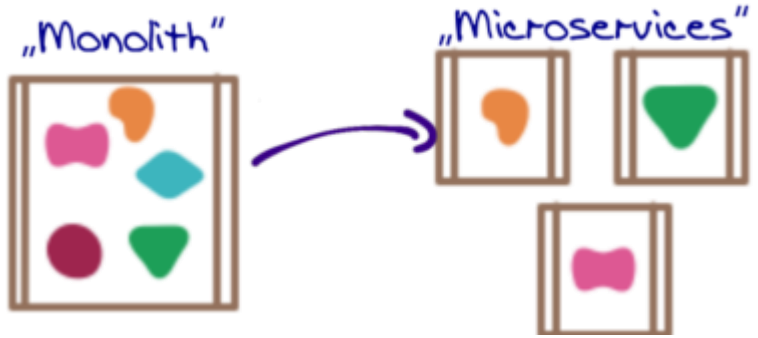
HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Einführung

Generated by https://www.wortwolken.com/

Microservices

- moderne Softwarearchitektur: Monolith → Microservice
  - prinzipientreue Softwareentwicklung (SOLID)
  - Skalierbarkeit
  - Erweiterbarkeit
  - Wartbarkeit
  - Testbarkeit
  - Einsetzbarkeit
- Microservices nutzen Microservices



https://martinfowler.com/articles/microservices.html

Representational State Transfer (REST)

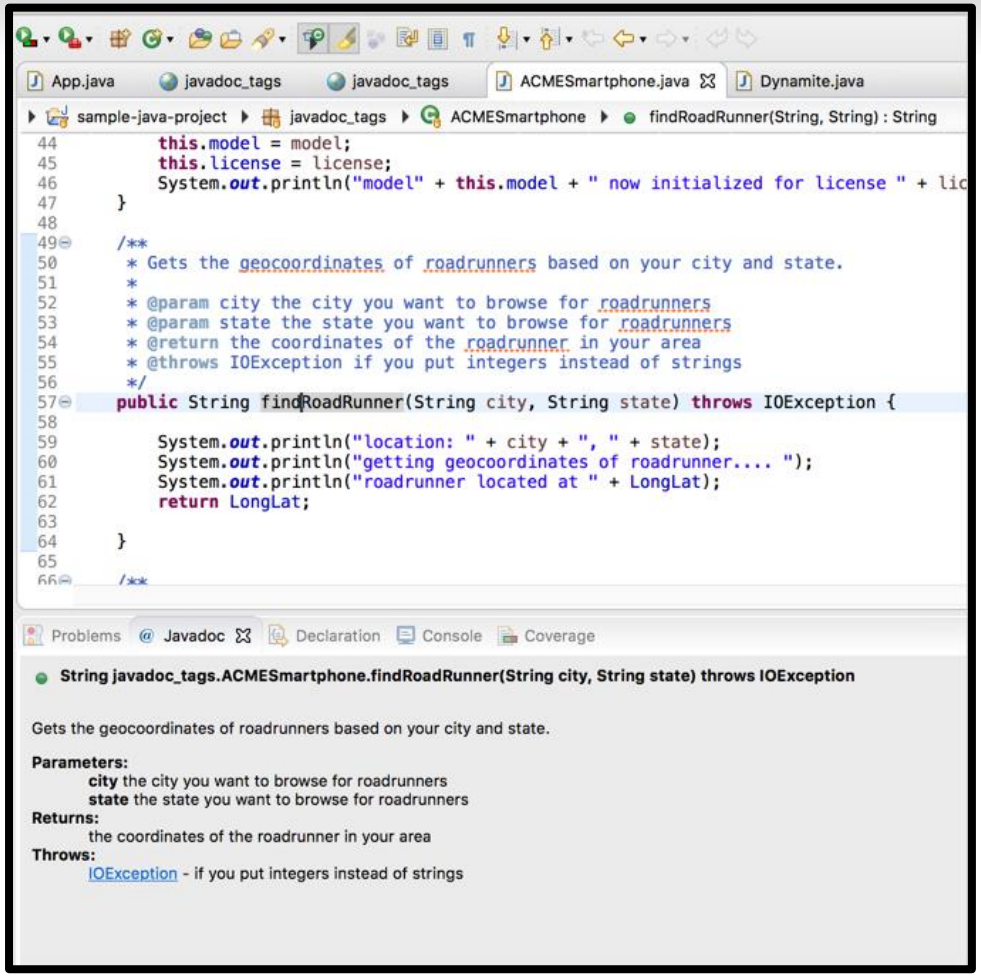
- Entwurfsstil für Dienste
  - jede Ressource erhält eine eindeutige URL
  - URL mit verschiedenen Methoden ansprechbar
    - GET: Ressource anfordern
    - PUT: Ressource erstellen
    - POST: Unterressource hinzufügen
- Nutzen von RESTful APIs in Microservices

*https://miro.medium.com/max/790/1*uHzooF1EtgcKn9_XiSST4w.png*

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Schnittstellendokumentation

- Dokumentation?
  - Lesbarkeit & Verständlichkeit erhöhen
  - komplizierte Zusammenhänge und Implementierungsentscheidungen beschreiben
  - Zusammenarbeit & Einarbeitung erleichtern
  - Generierung von Dokumenten
- Schnittstellendokumentation?
  - oft abteilungsübergreifende Zusammenarbeit
  - klare API-Spezifikation notwendig



https://idratherbewritingmedia.com/images/api/javadocpane.png

OpenAPI & Swagger

https://www.openapis.org/wp-content/uploads/sites/3/2018/02/OpenAPI_Logo_Pantone-1.png

- OpenAPI
  - quelloffene Initiative um REST-APIs zu standardisieren
    - OpenAPI 2.0 weit verbreitet und unterstützt
    - neuste Version: OpenAPI 3.0.2
  - Mitglieder: Google, Paypal…

https://cdn-images-1.medium.com/fit/t/1600/480/1*aKVg84SP5oPV9fwOnbl6yQ.png

- Swagger
  - bietet verschiedene Werkzeugen für Schnittstellenentwicklung
    - Swagger-Editor: OpenAPI Spezifikationen schreiben & betrachten
    - Swagger-Codegen: Quellcodegenerierung anhand OpenAPI Spezifikationen
    - Swagger Inspector: Test-Framework
  - Swagger-Spezifikation äquivalent zu OpenAPI 2.0 Spezifikation

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Contract First

- Fokus liegt auf Planung & Integration der Applikationen
  - Entwerfen von Projekt-Architektur
  - Definition der Zuständigkeitsbereiche von Microservices
  - Konkrete Definition aller Schnittstellen

  → *Ergebnis: Ausgehandelter Vertrag mit allen Beteiligten*
- *Danach: Zügiger Einstieg in Implementierungsphase*
  - *Aufgaben, Abhängigkeiten, Schnittstellen klar definiert*
  - *Codegenerierung anhand der Spezifikation*

```yaml
swagger: "2.0"
info:
  description: "This is a sample server Petstore server.  Y
  version: "1.0.0"
  title: "Swagger Petstore"
  termsOfService: "http://swagger.io/terms/"
  contact:
    email: "apiteam@swagger.io"
  license:
    name: "Apache 2.0"
    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
host: "petstore.swagger.io"
basePath: "/v2"
tags:
- name: "pet"
  description: "Everything about your Pets"
  externalDocs:
    description: "Find out more"
    url: "http://swagger.io"
- name: "store"
  description: "Access to Petstore orders"
- name: "user"
  description: "Operations about user"
  externalDocs:
    description: "Find out more about our store"
    url: "http://swagger.io"
schemes:
- "https"
- "http"
paths:
  /pet:
    post:
      tags:
      - "pet"
      summary: "Add a new pet to the store"
      description: ""
      operationId: "addPet"
      consumes:
      - "application/json"
      - "application/xml"
      produces:
      - "application/xml"
      - "application/json"
      parameters:
      - in: "body"
        name: "body"
```

https://editor.swagger.io/

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Quellcode-Generierung

- quelloffener „Swagger-Editor“
  - im Editor OpenAPI-Spezifikation erstellen
  - daraus Projekt generieren
  - einfach, intuitiv, über UI erstellbar
  - aber: Nicht automatisierbar, wenig Anpassungsmöglichkeit
- *quelloffener „Swagger-Codegen“*
  - *Command-Line-Interface (CLI)*
  - *vorhandene OpenAPI-Spezifikation einlesen und Projekt generieren*
  - *sehr gut automatisierbar, viele Einstellungen möglich*
  - *aber: unhandlich, komplex, nicht intuitiv*

*https://cdn-images-1.medium.com/fit/t/1600/480/1*aKVg84SP5oPV9fwOnbl6yQ.png*

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Contract Last

- Generierung der OpenAPI-Spezifikation durch Frameworks auf Basis des Codes
- Dokumentation fest an den Quellcode gebunden (ähnlich Javadoc)
  - „nachdokumentieren“ wird seltener vergessen
  - aktualisiert sich zu großen Teilen selbst
- Bereitstellung der aktuellen Spezifikation zur Laufzeit

Springfox

```java
@GetMapping
@ApiOperation(value = "Returns a pet of a certain colour")
@ApiResponses({@ApiResponse(code = 200, message = "OK"), @ApiResponse(code = 400, message = "error")})
public String getPet(@ApiParam(required = true) @RequestParam int color) {
    // mechanics
    return "";
}
```

Dokumentieren mit Springfox - Eigener Screenshot

- Einfache Einrichtung:
  - Abhängigkeit hinzufügen
  - In das Springprojekt importieren *@Import*
  - Konfigurationsklasse erstellen
- Erkennt automatisch Spring-Annotations
  - z.B. *@RestController, @PostMapping, @RequestParam*
  - Auch möglich: Constraints, z.B. *@Size, @NotNull*

```java
@EnableSwagger2
@Configuration
public class SpringfoxConfig {

    @Bean
    public Docket apiConfig() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.any())
                .paths(PathSelectors.any())
                .build();
    }

}
```

Springfox Konfiguration – Eigener Screenshot

- Bietet zur Laufzeit die aktuelle Spezifikation und eine UI zur besseren Ansicht
  - http://<host>:<port>/v2/api-docs
  - http://<host>:<port>/swagger-ui.html

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Springfox & Maven Model

- Unschön: Redundante Informationen
  - pom.xml: Version, Projektname, Lizenz, Projektbeschreibung
  - Springfox Configuration: Version, Projektname, Lizenz, Projektbeschreibung
- Lösung: Auslesen der pom.xml mit Maven Model
- Maven Model bietet pom.xml Inhalte als Java-Objekte
- Vorgehen:
  - Speicherort der pom.xml zur Laufzeit des Programms:
    META-INF/maven/<groupId>/<artifactId>/pom.xml
  - Pfad zur pom.xml bestimmen
  - Parsen der pom.xml mit Maven Modules
  - Konfigurieren der ApiInfo durch Auslesen der Maven-Daten

https://maven.apache.org/images/maven-logo-black-on-white.png

Springfox & Maven Model

```java
    private ApiInfo apiInfo() throws Exception {

        InputStream projectProperties = this.getClass().getResourceAsStream("/project.properties");
        Properties properties = new Properties();
        properties.load(projectProperties);

        String artifactId = properties.getProperty("project-artifactId");
        String groupId = properties.getProperty("project-groupId");

        String pathToPom = String.format("META-INF/maven/%s/%s/pom.xml", groupId, artifactId);

         MavenXpp3Reader reader = new MavenXpp3Reader();
        Model model =  reader.read(this.getClass().getClassLoader().getResourceAsStream(pathToPom));

        Developer firstDeveloper = model.getDevelopers().get(0);
        Contact contact = new Contact(firstDeveloper.getName(), firstDeveloper.getOrganizationUrl(), firstDeveloper.getEmail());

        return new ApiInfoBuilder()
                .title(model.getName())
                .description(model.getDescription())
                .version(model.getVersion())
                .contact(contact)
                .build();
    }
```

Springfox ApiInfo Konfiguration mit Maven Mode – eigener Screenshot

```
project-groupId=@project.groupId@
project-artifactId=@project.artifactId@
```

Inhalt project.properties – eigener Screenshot

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

Best Practices?

- Kombination von beiden Ansätzen:
  - Zu Beginn des Projektes: Contract first
    - konzeptionieren, planen
    - Architektur entwerfen
    - Schnittstellen definieren
  - Anschließend für jedes Projekt automatisiert Quellcode generieren und in CI und CD Pipelines dafür erschaffen
  - Weiterentwicklung: Contract last
    - Schnittstellen werden selbstständig nachdokumentiert
    - API ist immer abrufbar

Fragen

- Haben Sie Fragen?

https://neuformation.de/wp-content/uploads/2016/03/Frage.jpg