Hysterisch gewachsen - Wege zu einer stabilen API

The Author

Marco Schulz ElmarDott

studied at HS Merseburg, Germany, computer science and holds an engineers degree in software engineering. The main topics in his field of work are Software Architectures, automatism of the software development process and Software Configuration Management. Since more than 15 years he work in different large Web Application Projects. Currently he work as independent Consultant, Trainer and publish plently articles in several computer magazines. Mail: marco.schulz@outlook.com

+ Project Manager + Consultant + Writer + Speaker + Trainer +

Agenda

- Beispiele zu einer API
- Design Pattern
- Änderungen und Wartbarkeit
- Dokumentation mit API Guardian
- RESTful API & Swagger
- API Checkliste

Bekannte APIs

XML

Extensible Markup Language

// mehrere APIs

SAX – Simple API for XML
DOM – Document Object Model
Stax – Streaming API for XML

JSON

JavaScript Object Notation

// rudimentäre APIs

<!-- simple -->
flexjson

<!-- popular -->
jackson

PDF

Portable Document Format

// Herstellerformat

iText vs. OpenPDF

Pattern: Adaptor / Fassade / Proxy

Adaptor: Auch bekannt als Wrapper, koppelt eine Schnittstelle zu einer anderen, die nicht kompatibel sind.

Facade: bündelt mehrere Schnittstellen zu einer vereinfachten Schnittstelle.

Proxy: ist eine Verallgemeinerung einer komplexen Schnittstelle. Es kann als Komplement zur Fassade verstanden werden, die mehrere Schnittstellen zu einer Einzigen zusammenfasst.

Änderungen und Wartbarkeit

```
// Java Standard API
List<String> collection = new ArrayList<>();

// Userdefined Project based API
IList<String> collection = new ListImpl<>();
```

Architektur & Strucktur

- my.pkg.business: Interfaces
- my.pkg.application: Implementierungen
- my.pkg.application.hepler: Hilfsklassen

Dokumentation mit API Guardian

```
<dependency>
  <groupId>org.apiguardian</groupId>
  <artifactId>apiguardian-api</artifactId>
  <version>1.1.0</version>
</dependency>
```

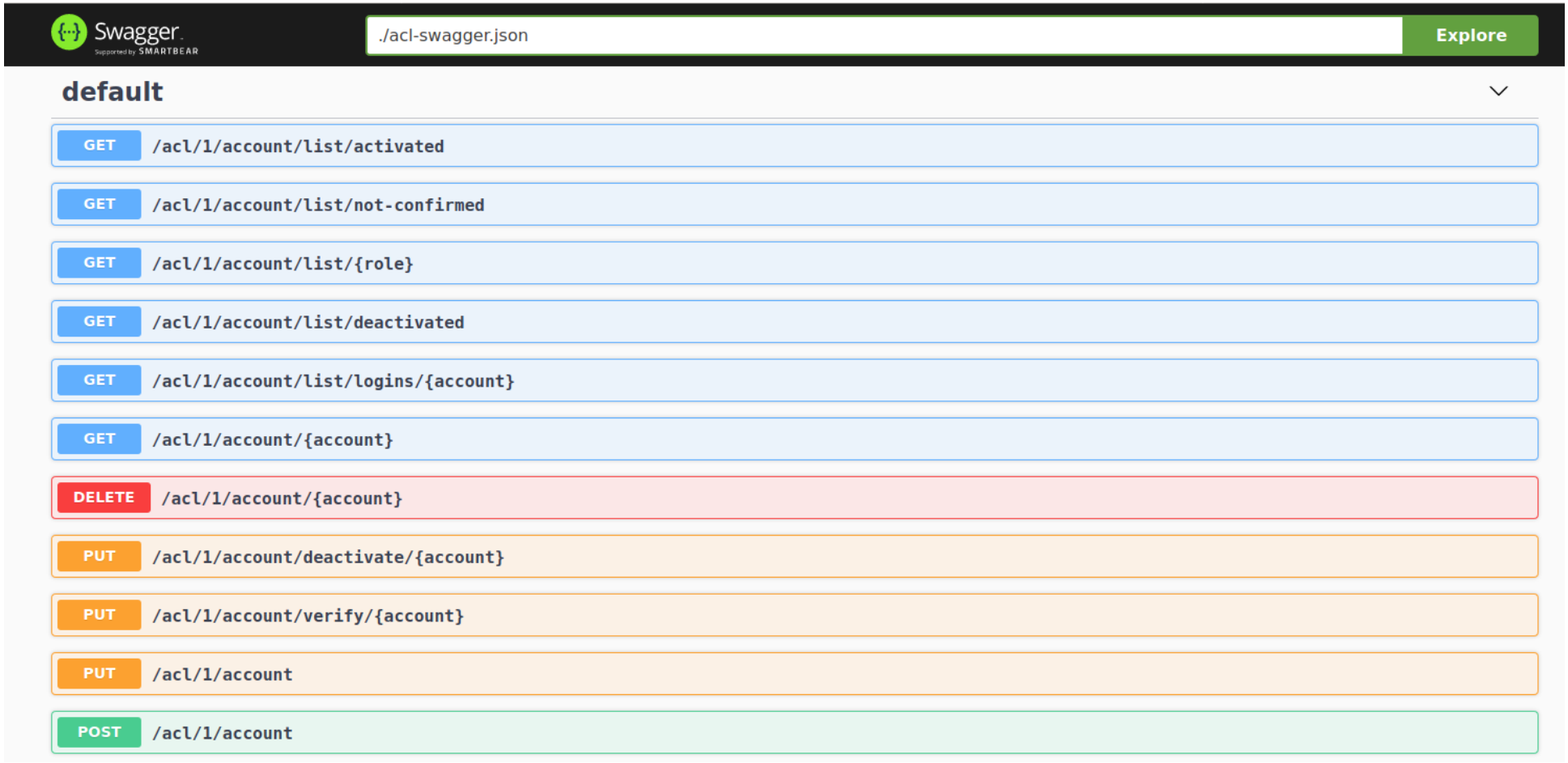
- DEPRECATED: Veraltet, sollte nicht weiterverwendet werden.
- EXPERIMENTAL: Kennzeichnet neue Funktionen, auf die der Hersteller gerne Feedback erhalten würde. Mit Vorsicht verwenden, da hier stets Änderungen erfolgen können.
- INTERNAL: Nur zur internen Verwendung, kann ohne Vorwarnung entfallen.
- STABLE: Rückwärts kompatibles Feature, das für die bestehende Major-Version unverändert bleibt.
- MAINTAINED: Sichert die Rückwärtsstabilität auch für das künftige Major-Release zu.

RESTful

```
01: RolesDO role = rolesDAO.find(roleName);
02: String json = rolesDAO.serializeAsJson(role);

03: if (role != null) {
04:     response = Response.status(Response.Status.OK)
05:             .type(MediaType.APPLICATION_JSON)
06:             .entity(json)
07:             .encoding("UTF-8")
08:             .build();
09: } else {
10:    response = Response
11:            .status(Response.Status.NOT_FOUND)
12:            .build();
13: }
```

Swagger



API Checkliste

- Möglichst gegen Interfaces programmieren
- KISS: Keep it simple, stupid
- Wiederkehrende Strukturen verwenden
- Interfaces möglichst generell benennen
- Implementierungen nach deren Spezialisierung benennen
- Ausführliche Dokumentation der Interfaces
- API Änderungen in einem Release bündeln
- Auf Kompatibilität achten

Thank you / Danke / Gracias