

API-Design

Praxishandbuch für Java- und
Webservice-Entwickler

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhaltsübersicht

Teil I	Grundlagen	1
1	Application Programming Interfaces – eine Einführung	3
2	Qualitätsmerkmale	17
3	Allgemeines Vorgehen beim API-Design	33
Teil II	Java-APIs	47
4	Ausprägungen	49
5	Grundlagen für Java-APIs	55
6	Fortgeschrittene Techniken für Java-APIs	103
7	Kompatibilität von Java-APIs	135
Teil III	Remote-APIs	157
8	Grundlagen RESTful HTTP	159
9	Techniken für Web-APIs	177
10	SOAP-Webservices	255
11	Messaging	275

Teil IV	Übergreifende Themen	299
12	Dokumentation	301
13	Caching	325
14	Skalierbarkeit	335
15	Erweiterte Architekturthemen	355
16	API-Management	367
Anhang		379
A	Literaturverzeichnis	381
	Index	389

Inhaltsverzeichnis

Teil I	Grundlagen	1
1	Application Programming Interfaces – eine Einführung	3
1.1	Eine kurze Geschichte der APIs	3
1.2	Web-APIs ab dem Jahr 2000	5
1.3	API-Definition	8
1.4	Vorteile einer API	10
1.5	Nachteile einer API	12
1.6	API als Produkt	12
1.7	Welche Strategien verfolgen Unternehmen mit APIs?	13
1.8	API-first-Ansatz	13
1.9	Zusammenfassung	15
2	Qualitätsmerkmale	17
2.1	Allgemeine Qualitätsmerkmale	17
2.2	Benutzbarkeit	18
2.2.1	Konsistent	18
2.2.2	Intuitiv verständlich	19
2.2.3	Dokumentiert	21
2.2.4	Einprägsam und leicht zu lernen	22
2.2.5	Lesbaren Code fördernd	23
2.2.6	Schwer falsch zu benutzen	24
2.2.7	Minimal	26
2.2.8	Stabil	27
2.2.9	Einfach erweiterbar	28
2.3	Konnaszenz	28
2.4	Zusammenfassung	31

3	Allgemeines Vorgehen beim API-Design	33
3.1	Überblick	33
3.2	Heuristiken und Trade-offs	34
3.3	Anforderungen herausarbeiten	35
3.4	Wenn Use Cases nicht ausreichen	36
3.5	Entwurf mit Szenarien und Codebeispielen	38
3.6	Spezifikation erstellen	40
3.7	Reviews und Feedback	45
3.8	Wiederverwendung	45
3.9	Zusammenfassung	46
Teil II	Java-APIs	47
4	Ausprägungen	49
4.1	Implizite Objekt-API	49
4.2	Utility-Bibliothek	52
4.3	Service	52
4.4	Framework	53
4.5	Eine Frage der Priorität	54
4.6	Zusammenfassung	54
5	Grundlagen für Java-APIs	55
5.1	Auswahl passender Namen	55
5.1.1	Klassennamen	56
5.1.2	Methodennamen	56
5.1.3	Parameternamen	59
5.1.4	Ubiquitäre Sprache	60
5.1.5	Fazit	61
5.2	Effektiver Einsatz von Typen	61
5.2.1	Semantischen Vertrag minimieren	62
5.2.2	Semantische Verletzung der Datenkapselung vermeiden . . .	62
5.2.3	Werden Namen überschätzt?	64
5.2.4	Fazit	66
5.3	Techniken für Objektkollaboration	66
5.3.1	Tell, Don't Ask	66
5.3.2	Command/Query Separation	67
5.3.3	Law of Demeter	70
5.3.4	Platzierung von Methoden	71
5.3.5	Fazit	72

5.4	Jigsaw-Module	72
5.5	Minimale Sichtbarkeit	75
5.5.1	Jigsaw-Module	75
5.5.2	Packages	76
5.5.3	Klassen	76
5.5.4	Methoden	76
5.5.5	Felder	76
5.5.6	Fazit	76
5.6	Optionale Hilfsmethoden	77
5.6.1	Komfort	77
5.6.2	Utility-Klassen	77
5.6.3	Fazit	78
5.7	Optionale Rückgabewerte	78
5.7.1	Ad-hoc-Fehlerbehandlung	79
5.7.2	Null-Objekte	80
5.7.3	Verwendung der Klasse <code>java.util.Optional</code>	81
5.7.4	Fazit	82
5.8	Exceptions	82
5.8.1	Ausnahmesituationen	82
5.8.2	Checked Exception versus Unchecked Exception	83
5.8.3	Passende Abstraktionen	84
5.8.4	Dokumentation von Exceptions	85
5.8.5	Vermeidung von Exceptions	86
5.8.6	Fazit	87
5.9	Objekterzeugung	87
5.9.1	Erzeugungsmuster der GoF	88
5.9.2	Statische Factory-Methode	88
5.9.3	Builder mit Fluent Interface	90
5.9.4	Praktische Anwendung der Erzeugungsmuster	91
5.9.5	Fazit	93
5.10	Vererbung	93
5.10.1	Ansätze zum Einsatz von Vererbung	94
5.10.2	Stolperfallen bei Vererbung	95
5.10.3	Bedeutung für API-Design	97
5.10.4	Fazit	98
5.11	Interfaces	98
5.11.1	Typen nachrüsten	99
5.11.2	Unterstützung für nicht triviale Interfaces	99
5.11.3	Markierungsschnittstellen	100
5.11.4	Funktionale Interfaces	100
5.11.5	Fazit	101
5.12	Zusammenfassung	101

6	Fortgeschrittene Techniken für Java-APIs	103
6.1	Fluent Interface	103
6.1.1	DSL-Grammatik	104
6.1.2	Schachteln versus Verketteten	107
6.1.3	Fluent Interface von jOOQ	107
6.1.4	Ist der Aufwand gerechtfertigt?	108
6.1.5	Fazit	108
6.2	Template-Methoden	108
6.2.1	API versus SPI	109
6.2.2	Erweiterbare Parameter	111
6.2.3	Fazit	111
6.3	Callbacks	111
6.3.1	Synchrone Callbacks	112
6.3.2	Asynchrone Callbacks	113
6.3.3	Fazit	114
6.4	Annotationen	115
6.4.1	Auswertung zum Kompilierzeitpunkt	115
6.4.2	Auswertung zur Laufzeit	117
6.4.3	Fazit	118
6.5	Wrapper-Interfaces	119
6.5.1	Proxy	119
6.5.2	Adapter	121
6.5.3	Fassade	121
6.5.4	Fazit	124
6.6	Immutability	124
6.6.1	Wiederverwendung	125
6.6.2	Threadsicherheit	126
6.6.3	Einfachheit	126
6.6.4	Umsetzung	127
6.6.5	Automatische Überprüfung mit dem Mutability Detector	128
6.6.6	Codegenerierung mit Immutables	129
6.6.7	Fazit	129
6.7	Threadssichere APIs	130
6.7.1	Vorteile	130
6.7.2	Nachteile	130
6.7.3	Was bedeutet Threadsicherheit?	131
6.7.4	Fazit	134
6.8	Zusammenfassung	134

7	Kompatibilität von Java-APIs	135
7.1	Kompatibilitätsstufen	135
7.1.1	Codekompatibilität	135
7.1.2	Binäre Kompatibilität	136
7.1.3	Funktionale Kompatibilität	137
7.2	Verwandtschaftsbeziehungen	139
7.3	Design by Contract	140
7.4	Codeänderungen	142
7.4.1	Package-Änderungen	143
7.4.2	Interface-Änderungen	144
7.4.3	Klassenänderungen	145
7.4.4	Spezialisierung von Rückgabetypen	146
7.4.5	Generalisierung von Parametertypen	147
7.4.6	Generics	147
7.4.7	Ausnahmen	148
7.4.8	Statische Methoden und Konstanten	148
7.5	Praktische Techniken für API-Änderungen	149
7.6	Test Compatibility Kit	153
7.7	Zusammenfassung	155

Teil III Remote-APIs **157**

8	Grundlagen RESTful HTTP	159
8.1	REST versus HTTP	159
8.2	REST-Grundprinzipien	160
8.3	Ressourcen – die zentralen Bausteine	165
8.4	HTTP-Methoden	167
8.5	HATEOAS	172
8.6	Zusammenfassung	175
9	Techniken für Web-APIs	177
9.1	Anwendungsbeispiel: Onlineshop	177
9.2	URI-Design	187
9.3	Medientypen	191
9.4	Fehlerbehandlung	205

9.5	Versionierung	210
9.5.1	Daten- und Sprachversionierung	210
9.5.2	Kompatibilität und Perspektive	211
9.5.3	Versionsidentifikation	213
9.6	API-Sicherheit	217
9.6.1	API-Sicherheitsmechanismen	217
9.6.2	Authentifizierung	218
9.6.3	API-Keys	222
9.6.4	Distributed Denial of Service (DDoS)	223
9.6.5	Injection-Angriff	229
9.7	Partielle Rückgaben	231
9.8	GraphQL	236
9.9	OData	244
9.10	gRPC	247
9.11	Zusammenfassung	252
10	SOAP-Webservices	255
10.1	SOAP-Grundlagen	255
10.2	WSDL-Grundlagen	258
10.3	Entwurfsansätze und -muster	261
10.4	Versionierung	268
10.5	SOAP versus REST	271
10.6	Zusammenfassung	273
11	Messaging	275
11.1	Routenplanung für Lkw-Transporte (Teil 1)	276
11.2	Message Broker	277
11.3	Produkte	281
11.4	Standards und Protokolle	284
11.5	Routenplanung für Lkw-Transporte (Teil 2)	288
11.6	Transaktionen und garantierte Nachrichtenzustellung	290
11.7	Asynchrone Verarbeitung und REST	292
11.8	Push Notifications	294
11.9	Zusammenfassung	298

Teil IV Übergreifende Themen **299**

12	Dokumentation	301
12.1	Motivation	301
12.2	Zielgruppen unterscheiden	302
12.3	Allgemeiner Aufbau	302
12.4	Beispiele	305
12.5	Dokumentation von Java-APIs	307
12.6	Dokumentation von Web-APIs	314
12.7	Zusammenfassung	324
13	Caching	325
13.1	Anwendungsfälle	325
13.2	Performancevorteil	326
13.3	Verdrängungsstrategien	326
13.4	Cache-Strategien für Schreibzugriffe	327
13.5	Cache-Topologien für Webanwendungen	328
13.6	HTTP-Caching	329
13.7	Zusammenfassung	334
14	Skalierbarkeit	335
14.1	Anwendungsfall	335
14.2	Grundlagen	336
14.3	Load Balancing	339
14.4	Statuslose Kommunikation	343
14.5	Skalierung von Datenbanken	345
14.6	Skalierung von Messaging-Systemen	349
14.7	Architekturvarianten	352
14.8	Zusammenfassung	353
15	Erweiterte Architekturthemen	355
15.1	Consumer-Driven Contracts	355
15.2	Backends for Frontends	358
15.3	Vernachlässigte Frontend-Architektur	361
15.4	Netflix-APIs	362
15.5	Zusammenfassung	365

16	API-Management	367
16.1	Überblick	367
16.2	Funktionen einer API-Management-Plattform	368
16.3	API-Management-Architektur	370
16.4	Open-Source-Gateways	375
16.5	Zusammenfassung	378

Anhang	379
---------------	------------

A	Literaturverzeichnis	381
	Index	389