

Index

■ A

Agile design strategy, 13-14

Ant software, 42

Anypoint system, 28

API

- architecture design, 15

- Client and CORS, 102-106

- comparison, 17-18

- consumers, 25

- description, 14

- design, 26-28

- design strategies, 11-12

- end users, 25

- façade, 23

- Façade Pattern, 97-98

- implementation, 16, 99

- internal subsystems, 98

- layers, 99

- life cycle, 100-101

- management, 100

- methodology, 14

- modeling, 16-17

- monetization, 102

- partial response, 22

- Portfolio, 15

- process, 13

- prototyping, 15

- providers, 25

- publish, 16

- retirement, 101

- solution architecture, 23-24

- version, 21-22

API Engagement platform, 80

API framework

- DAO, 67

- Facade Pattern, 67

- services layer implementation, 66-67

API Platform Architecture

- components, 78

- development, 78-80

- engagement platform, 80

- enterprise, 81-82

- importance, 77

- pre-production, 80

- production, 80

- simulation, 80

- testing, 80

API Portfolio Architecture

- consistency, 63

- customization, 64

- discoverability, 64

- longevity, 64

- requirements, 63

- reuse, 63

API security

- client credentials grant, 115-116

- client registration, 109

- code grant, 110-111

- grant types, 110

- implicit grant, 111-112

- oAuth, 107

- register as a client, 109

- resource owner password credentials

 - grant, 113-114

- roles, 107

- server response, 109

- tokens, 108-109

Application Programming

- Experience (APX), 11

■ B

Back-end systems, 82

BaseURI, 28

Base URL, 19-20

Blueprint, 17–18
Bolt-on strategy, 11

■ **C**

Caching, 5
 conditional cache headers, 118–119
 HTTP, 117
 memory and CPU resources, 116
 server, 117
 time-based cached header, 117–118
 web, 119
Choreography, 66
Client-server interactions, 4, 6
Cloud solutions, 24
Code-on-demand, 6
Command Query Responsibilities
 segmentation (CQRS), 83–85
Community MySQL software, 42
Content negotiation, 9
Cookie parameter, 53
Create, read, update, delete (CRUD), 56
 JAX-RS
 with JSON, 57–61
 with XML, 54–57
Cross-Origin Resource
 Sharing (CORS), 97, 105–106
CRUD architecture, 83
CURL command line tool, 56–57
Curl software, 42

■ **D**

Data Access Object (DAO), 67, 82–83
Data Handler, 84
 CQRS, 83
 DAO, 82–83
 JPA, 85–94
 NoSQL process, 84
 SQL development process, 83
Dependency injection (DI), 67
Development platform, 78–79
 data format transformation, 79
 data integrity and protection, 79
 front-end protocols, 79
 language for designing, 79
 security, 78
 structural transformation, 79
Document Type Definitions (DTDs), 34
Domain analysis, 14

■ **E**

Eclipse-Mars software, 41
E-commerce system, 25
Error code, 21
Error handling, 20–21
eXtensible Markup Language (XML)
 and JSON comparison, 40
 comments, 34–35
 encoding, 35
 importance, 35
 introduction, 33
 JAX-RS, 54–57
 messaging application, 33
 pros and cons, 36
 standalone, 35
 tags, 33
 uses, 35
 version, 35
 vs. HTML, 34

■ **F**

Facade pattern, 23
Facade strategy, 13
Facebook, 22
Form parameter, 53

■ **G**

Governance
 change management, 65
 consistency, 64
 customization, 65
 discoverability, 65
 reuse, 65
Greenfield strategy, 12

■ **H**

Header parameter, 53
HTML, 34
Hypermedia As The Engine Of Application
 State (HATEOAS), 6

■ **I**

Identity and Access Management (IAM), 81
Internet-of-things, 25
Inversion of Control, 67

■ **J, K**

Java API for RESTful Web Services (JAX-RS)

- Cache-Control, 118
- content type, 51
- cookie parameter, 53
- CRUD
 - CURL, 56–57
 - customer resource, 55–56
 - ErrorMessage object, 59
 - Glassfish, 57, 58
 - Java customer object, 54
 - JSON customer object, 57
 - JSON CustomerResource updates, 58, 59
 - JSON pom.xml updates, 57
 - NotFoundException class, 60
 - result in postman, 60, 62
 - XML, 54

- example, 50
- features, 49
- form parameter, 53
- header parameter, 53
- injection, 51–52
- introduction, 49–50
- matrix parameter, 53
- path parameter, 52
- query parameter, 53

Java Persistence API (JPA), 86

- Maven project dependencies, 85
- persistence.xml file, 86
- podcast DAO, 91
- PodcastDAOImpl, 91–95
- podcast domain object, 86–87
- podcast resource, 88–89
- podcast service, 90

JavaScript Object Notation (JSON)

- importance, 38–39
- introduction, 36
- JAX-RS, 57–61
- pros and cons, 39–40
- syntax, 36
 - arrays, 38
 - booleans, 38
 - null, 38
 - numbers, 37
 - objects, 37
 - strings, 37
- uses, 39
- and XML comparison, 40

- JDK 8 software, 41
- Jetty software, 41

■ **L**

- LinkedIn, 22

■ **M**

- Markup, 33
- Matrix parameter, 53
- Maven software, 42
- Mobile app, 24, 63
- MS SQL software, 42
- Multilayer framework
 - data access object, 67
 - Facade pattern, 67
 - services layer implementation, 66
- MySQL, 94–95

■ **N**

- NoSQL, 83–85

■ **O**

- Orchestration (direct calls), 66

■ **P**

- Path parameter, 52
- Plain Object Oriented
 - Java Object (POJO), 72, 86–87
- Podcast domain object, 68–76
- POST, 19
- Postman software, 42

■ **Q**

- Query parameter, 53

■ **R**

- Representational
 - State Transfer (REST)
 - basics, 7
 - fundamentals, 8–9
 - hello exercise
 - instructions, 43–45, 47–48
 - software for installation, 41–42
 - URI processing, 42

■ INDEX

- modifiability, 2
- performance, 2
- portability, 2
- reliability, 2
- scalability, 2
- security, 7
- simplicity of interface, 2
- SOAP *vs.*, 2–4
- structures, 8
- visibility, 2
- web architectural style, 4

RESTful API Modeling

- Language (RAML), 17–18
- Anypoint system, 28
- steps, 28–31

Runtime platform, 79

■ S

Schema model, 16–17

Services layer implementation, 66–67

- podcast domain object, 68–76

Setter injection, 67

Smart TV solutions, 25

SOAP, 1, 3–4

Spring framework, 67, 70–71

SQL development process, 83

Stateless communications, 5–6

Swagger, 17–18

■ T

Tomcat software, 41

Twitter, 22

■ U, V

Uniform resource interface, 5

■ W

Web applications, 24

Web architectural style

- caching, 5
- client-server, 4
- code-on-demand, 5
- HATEOAS, 6
- layered system, 5
- stateless, 5
- uniform resource interface, 5

■ X, Y, Z

XML. *See* eXtensible Markup Language (XML)