

Towards a more RESTful world

Anurup Joseph
Elegan Consulting

About Anurup

- ✦ coding professionally since 1994
- ✦ working with Java since 1996
- ✦ different industries/sectors/geographies
- ✦ loves to explore
- ✦ enjoys fostering Agile development and CI/CD
- ✦ volunteer to teach Java to kids

About You

- ✦ CORBA, IIOP, SOAP? REST?
- ✦ XML? JSON?
- ✦ Document and test your web services?

Agenda

- ✦ Interprocess Communication
- ✦ Communication Mechanisms
- ✦ Message formats
- ✦ Documentation & Testing
- ✦ Demo





Interprocess Communication

Interprocess Communication

- ✦ Services consume other services
- ✦ Access via web protocols = Web Services
- ✦ Service chaining



SOAP Part 1

- ✦ originally acronym for Simple Object Access Protocol
- ✦ now, just SOAP
- ✦ developed in 1998 at Microsoft for service communication
- ✦ simpler than an earlier more complex protocol
- ✦ SOAP Faults describe errors
- ✦ tools to generate client code

SOAP Part 2

- ✦ WSDL
 - ✦ acronym for Web Service Definition Language
 - ✦ describes API
- ✦ XSD
 - ✦ acronym for XML Schema Document
 - ✦ defines message formats

XML

- ✦ acronym for eXtensible Markup Language
- ✦ message format that is the basis of XSDs and WSDLs
- ✦ now considered verbose

JSON

- ✦ acronym for JavaScript Object Notation
- ✦ invented by Douglas Crockford, now Senior JavaScript Architect at PayPal
- ✦ derived from JavaScript, but is language-independent message format
- ✦ considered less verbose than XML

XML -> JSON

XML

```
<student>
```

```
  <name>John Doe</name>
```

```
  <age>11</age>
```

```
  <grade>5</grade>
```

```
</student>
```

JSON

```
{
```

```
  "name": "John Doe",
```

```
  "age": 11,
```

```
  "grade": 5
```

```
}
```



REST

Dr. Roy Fielding

- ✦ a principal author of HTTP
- ✦ co-founder of Apache web server (httpd)
- ✦ former Chair of Apache Software Foundation
- ✦ now Senior Principal Scientist at Adobe
- ✦ PhD dissertation: Architectural Styles and the Design of Network-based Software Architectures
- ✦ introduced REST

REST

- ✦ acronym for REpresentational State Transfer
- ✦ stateless web service communication protocol
- ✦ services located by URI's; communication via HTTP operations
- ✦ less verbose, industry standard

CRUD via HTTP

- ✦ acronym for Create, Read, Update, Delete
- ✦ match to HTTP operations
 - ✦ Create = POST
 - ✦ Read = GET
 - ✦ Update = PUT (complete)/PATCH (partial)
 - ✦ Delete = DELETE
- ✦ HTTP status codes used to convey result

Best practices

- ✦ <http://www.restapitutorial.com/lessons/httpmethods.html>
 - ✦ GET operations should be idempotent
 - ✦ any state change on server should be POST, PUT, PATCH, or DELETE
- ✦ <http://www.restapitutorial.com/httpstatuscodes.html>
 - ✦ only return output relevant to client for brevity and security
 - ✦ when possible, use HTTP status codes to convey result



Examples

Create

```
POST /student/12345 HTTP/1.1
Content-Type: application/json
Accept: application/json
{
  "name": "John Doe",
  "age": 11,
  "grade": 5
}
```

```
HTTP/1.1 201 CREATED
Date: Tue, 19 Sep 2017
17:05:35GMT
Content-Type: application/json
{
  "createdOn":
    "2017-09-19T17:05:34.211Z",
  "updatedOn":
    "2017-09-19T17:05:34.211Z",
  "name": "John Doe",
  "age": 11,
  "grade": 5
}
```

Read

```
GET /student/12345 HTTP/1.1  
Accept: application/json
```

```
HTTP/1.1 200 OK  
Date: Tue, 19 Sep 2017  
17:06:35GMT  
Content-Type: application/json  
{  
  "createdOn":  
    "2017-09-19T17:05:34.211Z",  
  "updatedOn":  
    "2017-09-19T17:05:34.211Z",  
  "name": "John Doe",  
  "age": 11,  
  "grade": 5  
}
```

Update (entire)

```
PUT /student/12345 HTTP/1.1
Content-Type: application/json
Accept: application/json
{
  "name": "John Q. Doe",
  "age": 12,
  "grade": 6
}
```

```
HTTP/1.1 200 OK
Date: Tue, 19 Sep 2017
17:07:35GMT
Content-Type: application/json
{
  "createdOn":
  "2017-09-19T17:05:34.211Z",
  "updatedOn":
  "2017-09-19T17:07:34.211Z",
  "name": "John Q. Doe",
  "age": 12,
  "grade": 6
}
```

Update (partial)

```
PATCH /student/12345 HTTP/1.1
Content-Type: application/json
Accept: application/json
{
  "nickName": "Johnny"
}
```

```
HTTP/1.1 200 OK
Date: Tue, 19 Sep 2017
17:08:35GMT
Content-Type: application/json
{
  "createdOn":
  "2017-09-19T17:05:34.211Z",
  "updatedOn":
  "2017-09-19T17:08:34.211Z",
  "name": "John R. Doe",
  "age": 12,
  "grade": 6,
  "nickName": "Johnny"
}
```

Delete

DELETE /student/12345 HTTP/1.1
Accept: application/json

HTTP/1.1 204 NO CONTENT
Date: Tue, 19 Sep 2017
17:09:35GMT

Errors: Create

```
POST /student/12345 HTTP/1.1  
Content-Type: application/json  
Accept: application/json
```

```
{  
  "name": "John Doe",  
  "age": 11,  
  "grade": 5  
}
```

```
HTTP/1.1 409 CONFLICT  
Date: Tue, 19 Sep 2017  
17:10:35GMT
```

Errors: Read

GET /student/no_such HTTP/1.1
Content-Type: application/json
Accept: application/json

HTTP/1.1 404 NOT FOUND
Date: Tue, 19 Sep 2017
17:11:35GMT

Errors: Delete

DELETE /student/12345 HTTP/1.1
Accept: application/json

HTTP/1.1 404 NOT FOUND
Date: Tue, 19 Sep 2017
17:12:35GMT

RESTful + JSON

- ✦ web services consume HTTP operations
- ✦ web services produce HTTP status codes whenever possible
- ✦ simple message format
- ✦ communicate concisely
- ✦ more efficient communication; enhances scalability



Documentation
& Testing



Swagger/OpenAPI

- enables documenting & testing REST APIs
- started as side project at Wordnik in 2010
- bought by SmartBear in 2015, renamed as OpenAPI
- <https://swagger.io/>
- language and platform-agnostic
- code-generation possible



SpringFox

- ✦ automatic API documentation for Java services built with Spring
- ✦ based on Swagger/OpenAPI
- ✦ annotation-based configuration
- ✦ demo