State-of-the-art RESTful API

GDP Labs

Outline

- REST Definition
- ROA vs SOA
- REST vs SOAP
- REST Users
- REST Properties
- REST Constraints
- Richardson Maturity Model
- How to design State-of-the-art RESTful API

REST Definition

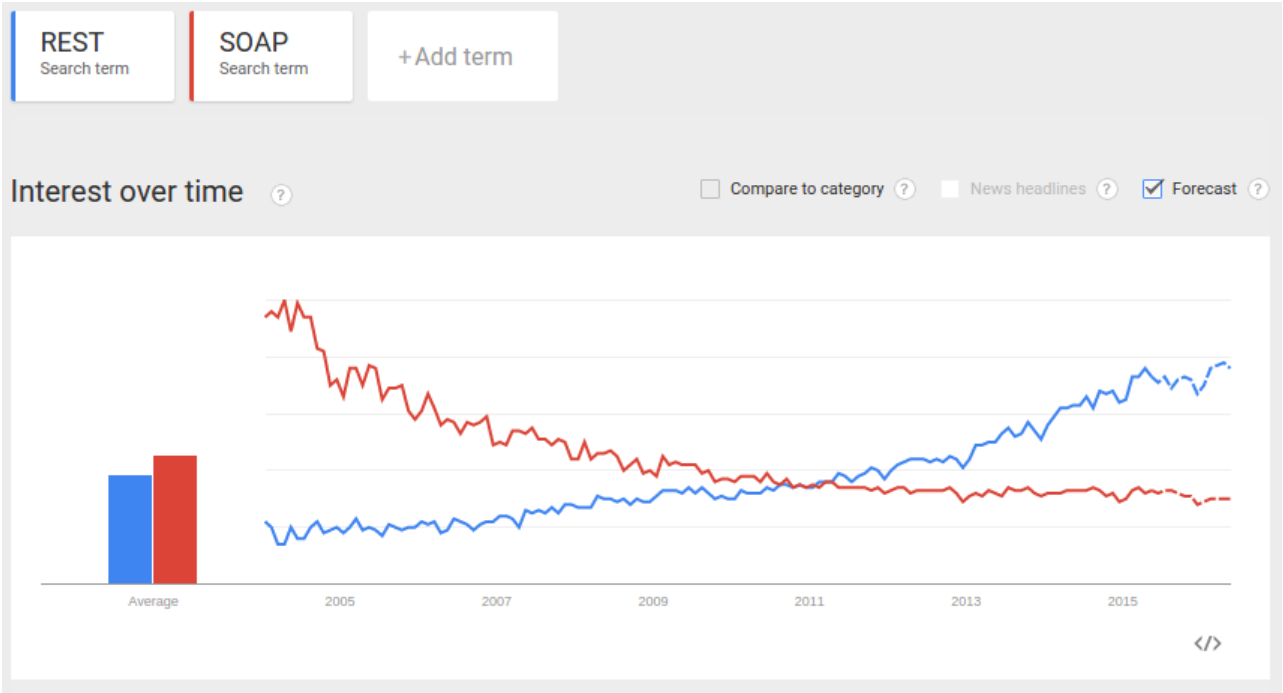
Representational State Transfer (REST) is a software **architecture style** consisting of **guidelines** and **best practices** for creating **scalable web services**

ROA vs SOA

Resource-Oriented Architecture	Service-Oriented Architecture
unique address per resource	one endpoint address per service
cacheable	not cacheable
generic to request mechanism	protocol specific e.g. SOAP
no data format descriptions	data format descriptions are part of the service e.g. WSDL

source: http://research.microsoft.com/pubs/117710/3-arch-styles.pdf

REST vs SOAP



source: [Google Trends - REST vs SOAP]

REST Users

and many more ...

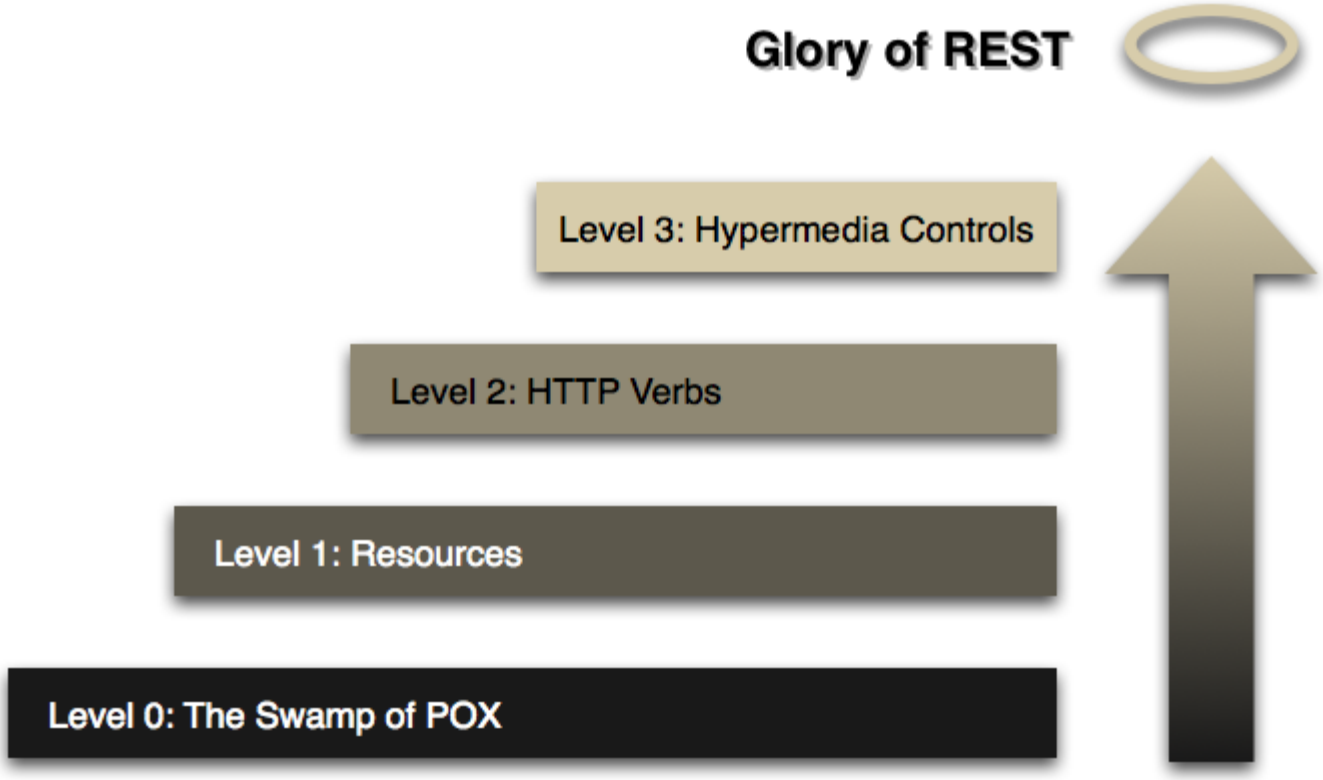
REST Properties

1. Performance
2. Scalability
3. Simplicity
4. Modifiability
5. Visibility
6. Portability
7. Reliability

REST Constraints

1. Client - Server
2. Stateless (w/o cookie & session, but token can be used)
3. Cacheable
4. Layered System
5. Code on Demand (Optional)
6. Uniform Interface
    a. Identification of Resources
    b. Manipulation on Resources through these Representations
    c. Self Descriptive Messages
    d. Hypermedia as the Engine of Application State (HATEOAS)

Richardson Maturity Model



How to Design a RESTful API

Be Consistent

/pied-piper /silicon-valley	/pied-piper /siliconValley

<pre>{
  "created_at": "...",
  "update_at": "..."
}</pre>
	<pre>{
	  "created_at": "...",
	  "updateAt": "..."
	}</pre>

Be Consistent

DOs	DON'Ts
/pied-piper /silicon-valley	/pied-piper /siliconValley

`{`
`  "created_at": "...",`
`  "update_at": "..."`
`}`
	`{`
	`  "created_at": "...",`
	`  "updateAt": "..."`
	`}`

URL Identifies a Resource

- Resource is a noun
- Use plural

Collection of blog posts	/posts
Specific blog post	/posts/12345
Comments of a post	/posts/12345/comments
Specific comment	/posts/12345/comments/5

Utilize HTTP Verbs

Create	Add a new post	`POST`	`/posts`
Read	View list of posts	`GET`	`/posts`
	See one post	`GET`	`/posts/12345`
Update	Edit a post	`PUT`	`/posts/12345`
		`PATCH`	`/posts/12345`
Delete	Remove a post	`DELETE`	`/posts/12345`

What about Non-CRUD Operations?

Resource should be noun, but there is an exception for these

Hide a post	`PUT /posts/12345/hide`
Unhide a post	`DELETE /posts/12345/hide`

Versioning

- Versioning via URL, not headers
  - http://api.site.com/**v1**/posts
- Benefit
  - Simple implementation
  - Ensure browser explorability

Utilize HTTP Status Code

200 - OK	Everything worked as expected.
201 - Created	A new resource was successfully created
400 - Bad Request	Often missing a required parameter.
401 - Unauthorized	No valid API key provided.
402 - Request Failed	Parameters were valid but request failed.
404 - Not Found	The requested item doesn't exist.
500, 502, 503, 504 - Server Errors	Something went wrong on server.

Use JSON

- For both request & response body
- Why?
  - Simplicity
  - Readability
  - Flexibility
  - Support data types (i.e string, int, float)

Descriptive & Consumable Errors

```
{
  "code" : 1234,
  "message" : "Something bad happened :(",
  "description" : "More details about the error"
}
```

Provide Partial Response

```
GET /posts/12345?fields=title,author

{
  "title": "RESTful API",
  "author": "GDP Labs"
}
```

Provide Pagination

```
GET /posts?page=1&count=10
```

```
[
  {
    "title": "RESTful API",
    "author": "GDP Labs",
    "content": "....",
  },
  {
    "title": "How to design good RESTful API",
    "author": "GDP Labs",
    "content": "....",
  }
]
```

Provide Sorting & Filtering

`GET /posts?sort=created_at`
Sort by `created_at` ascending

`GET /posts?sort=-created_at,author`
Sort by `created_at` descending & `author` ascending

`GET /posts?author=Bob&sort=-created_at`
Show posts from `Bob`
Sort by `created_at` descending

Provide Hypermedia Control

```
GET /posts/12345
```

```
{
  "title": "RESTful API",
  "author": "GDP Labs",
  "content": "....",
  "_links": {
    "self": {
      "href": "https://localhost/posts/12345
    },
    "comments" {
      "href": "https://localhost/posts/12345/comments"
    }
  }
}
```

Security

- Use OAuth2
  - It become the industry standard for RESTful API authentication and authorization
  - Used by big tech companies (i.e Google, Microsoft, Facebook, Paypal, Stripe etc)
  - Will explain further in the next slide
- Use HTTPS
  - If you implement OAuth2, HTTPS is a must
  - Encouraged to use HTTPS even though you don’t deploy OAuth2

OAuth2 - Client Credentials

1. Request Access Token

```
curl -X POST https://api.site.
com/oauth2/token \
-H "Content-Type: application/json" \
-u "APP_CLIENT_ID:APP_CLIENT_SECRET" \
-d "grant_type=client_credentials"

// HTTP OK 200
{
    "access_token": "A0151oM-YvNU",
    "token_type": "Bearer",
    "scope": "read write",
    "expires_in": 28800
}
```

2. Request resources

```
curl -X GET https://api.site.
com/v1/posts/12345 \
-H "Content-Type: application/json" \
-H "Authorization: Bearer A0151oM-YvNU"

// HTTP OK 200
{
  "title": "RESTful API",
  "author": "GDP Labs",
  "content": "...."
}
```

OAuth2 - Password

1. Request Access Token

```
curl -X POST https://api.site.
com/oauth2/token \
-H "Content-Type: application/json" \
-u "APP_CLIENT_ID:APP_CLIENT_SECRET" \
-d "grant_type=password"
-d "username=george"
-d "password=p@$$w0rd"

// HTTP OK 200
{
    "access_token": "RtY019-UkfKvN",
    "token_type": "Bearer",
    "scope": "read write",
    "expires_in": 28800
}
```

2. Request resources

```
curl -X GET https://api.site.
com/v1/posts/12345 \
-H "Content-Type: application/json" \
-H "Authorization: RtY019-UkfKvN"

// HTTP OK 200
{
  "title": "RESTful API",
  "author": "GDP Labs",
  "content": "...."
}
```


ANY
QUESTIONS
?

References

- Architectural Styles and the Design of Network-based Software Architectures
- Best Practices for Designing a Pragmatic RESTful API
- Best Practices for Architecting a Pragmatic Web API
- RESTful API Design
- Designing REST + JSON APIs
- HTTP API Design Guide


THANK YOU
GRACIAS
BOLZÏN
MERCI
ARIGATO
SHUKURIA
GOZAIMASHITA
EFCHARISTO
JUSPAXAR
DANKSCHEEN
BÏYAN
SHUKRIA
TASHAKKUR
ATU
SUKSAMA
EKHMET
YAQHANYELAY
TINGKI
GRAZIE
MEHRBANI
MAAKE
KOMAPSUMNIDA
PALDIES
SPASSIBO
SNACHALHUYA
NUHUN
CHALTU
WABEEJA
MAITEKA
YUSPAGABÁTAM
HUI
GUI
HATUR
EKOJU
SIKOMO
MAKETAI
MINMONCHAR
UNALCHÉESH
SPASIBO
DENKAUJA
NENACHALHYA
MERSI
DHANYABAAD
AHHA
ATTO
SANCO
MERASTAWHY
GAEJTHO
LAH
AGUYJE
FAKAAUE
TAVTAPUCH
MEDAWAGSE
BAIKA