

# Designing RESTful APIs

**<https://github.com/anandology/restful-apis>**

# Outline

- Introduction to HTTP
- Representational State Transfer (REST)
- Examples of RESTful APIs
- Designing an API
- Authentication and Security
- Exercises

# Introduction to HTTP

# Internet vs. World Wide Web

What is the difference between Internet and World Wide Web?

Internet is the network of computers.

World Wide Web is an application on top of internet.  
(Like many others including email, ftp, telnet, ssh etc.)

World Wide Web is the killer app of the internet.  
It revolutioned the internet.

# World Wide Web - Key Concepts

- Uniform Resource Locator (URL)
- Hyper Text Markup Language (HTML)
- Hyper Text Transfer Protocol (HTTP)



# Uniform Resource Location

Locate any resource with a single string.

Examples:

`https://rootconf.in/2017/building-restful-apis`

`https://www.cleartrip.com/account/trips/17041292873`

Revolutionary idea!

# Hyper Text

Document with references to other documents, which can be accessed immediately.

The term hypertext is coined by  
`<a href="https://en.wikipedia.org/wiki/Ted_Nelson">Ted Nelson</a>`  
in 1963.

Very simple idea. Nothing comes closer even after half a century.

Think: how do you manage related word documents?

# Hyper Text Transfer Protocol (HTTP)

HTTP is the protocol to transfer hypertext.

Simple text-based protocol.

# HTTP - Sample Request

```
GET /hello?name=web HTTP/1.1  
Host: example.com  
User-Agent: Mozilla/5.0  
Accept: */*
```

# HTTP - Sample Response

HTTP/1.1 200 OK

Server: gunicorn/19.7.1

Date: Thu, 11 May 2017 10:46:00 GMT

Content-Type: text/html; charset=utf-8

Content-Length: 11

Hello, web!

# HTTP - Important Parts

- HTTP Methods
  - GET, POST, PUT, DELETE
- Headers
  - Content-Type, Content-length, ...
- Status Codes
  - 200 OK, 404 Not Found

# Demo

Demo using curl and netcat.

# Send Email using netcat

```
$ nc -c mail.mailinator.com 25
S: 220 mail.mailinator.com ESMTP Postfix "Seqrite Terminator"
C: HELO mail.mailinator.com**
S: 250 mail.mailinator.com
C: MAIL FROM:<netcat@anandology.com>
S: 250 Ok
C: RCPT TO:<restful-apis@mailinator.com>
S: 250 Ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: "Netcat" <netcat@anandology.com>
  To: RESTful APIs Workshop <restful-apis@mailinator.com>
  Subject: HORN - 200 OK - PLEASE

  Hello Everyone,
  Enjoy your workshop!
  With Love,
  Netcat
  .
S: 250 Ok
C: QUIT
S: 221 Bye
```



# Safe and Idempotent Methods

- Safe - no side effects
  - GET and HEAD
- idempotence - the side-effects of more than one identical requests is the same as for a single request.
  - GET, HEAD, PUT and DELETE

# **Representational State Transfer (REST)**

# What is REST?

Nobody Understands REST or HTTP!

# What is REST?

Architectural principles and constraints for building network-based application software.

Defined by *Roy Fielding* in his PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures"

# Practical REST

- Thinking in Resources
  - model your application around resources/topics (nouns) instead of actions (verbs)
- Use HTTP methods and headers for metadata and control data

# Practical REST - Resources

BAD

`/show-page?id=5`

`/add-comment.php?post_id=5`

GOOD

`/pages/5`

`/pages/5/comments`

# Practical REST - HTTP Methods

Use HTTP methods for verbs. Common CRUD operations can be mapped to standard HTTP methods.

GET - read

POST - create

PUT - create or update

DELETE - delete

# Practical REST - HTTP Status Codes

Use HTTP Status codes to indicate success and error cases.



# SUCCESS

200 OK - Success

201 Created - New resource is created successfully.

# CLIENT ERRORS

400 Bad Request - malformed syntax

401 Unauthorized - authorization required

403 Forbidden - the current user doesn't have permission to access this resource

404 Not Found - requested resource is not found

# SERVER ERRORS

500 Internal Error - Oops! something went wrong

501 Not Implemented - Not yet implemented!

# Practical REST - HTTP Headers

## Sample Request Headers

Accept: application/json

Accept-Language: te, en;q=0.9, kn;q=0.5

Authorization: Basic dGVzdDp0ZXN0

## Sample Response Headers

Content-Type: application/json

Content-Language: en

# Alternatives to REST

- SOAP
- XML-RPC
- HTTP-RPC (even with JSON)

# SOAP - URL

Single URL for all API calls.

`https://api.flickr.com/services/soap/`

# SOAP - Sample Request

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
  <s:Body>
    <x:FlickrRequest xmlns:x="urn:flickr">
      <method>flickr.test.echo</method>
      <name>value</name>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

# SOAP - Sample Response - SUCCESS

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Body>
    <FlickrResponse xmlns="/ns/api#">
      [xml-payload]
    </FlickrResponse>
  </s:Body>
</s:Envelope>
```



# SOAP - Sample Response - ERROR

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Body>
    <s:Fault>
      <faultcode>flickr.error.[error-code]</faultcode>
      <faultstring>[error-message]</faultstring>
      ...
    </s:Fault>
  </s:Body>
</s:Envelope>
```

# HTTP RPC

```
$ curl -i https://slack.com/api/api.test
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
...

{"ok":true}
```

# Good Examples of RESTful APIs

Github

<https://developers.github.com/>

Digital Ocean

<https://developers.digitalocean.com/documentation/v2>

# Bad Examples of RESTful APIs

Flickr

<https://www.flickr.com/services/api/>

Bitly

<http://dev.bitly.com/links.html>

# Blog API

version 0 - Naive CRUD API for blog posts.

version 1 - blog api made RESTful

version 2 - add support for tags

version 3 - add support for comments

version 4 - add support for authors

version 5 - authentication

# Exercise - 1

Design a RESTful API for for bitly.

Current API:

<http://dev.bitly.com/links.html>

# Exercise - 2

Look at Twitter REST API and see how can it be made better.

<https://dev.twitter.com/rest/reference>

# Authentication Patterns

- Basic Auth - simple
- Digest access authentication - I don't understand
- API Keys - autogenerated pair of access key and secret key
- OAuth - third-party authentication



# Advanced Topics

- What is the right identifier?
- Versioning APIs
- Pagination

# References

- Cool URIs don't change
- Best Practices for Designing a Pragmatic RESTful API - Vinay Sahni
- Zalando RESTful API and Event Scheme Guidelines

# Thanks

Anand Chitipothu

@anandology

anandology@gmail.com