HTTP, the Web, and REST for API Developers

Mike Amundsen
@mamund


Mike Amundsen
@mamund

Overview

- Clarifying our thinking
- HTTP as Protocol
- WWW as Common Practice
- REST as Style

Clarifying our thinking

Clarifying our thinking

- Naming things
- Concepts vs. instances
- Separation of concerns

Separation of Concerns

"[A] characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation."

-- Edsger W. Dijkstra

https://en.wikipedia.org/wiki/Edsger_W._Dijkstra

HTTP as Protocol

HTTP as Protocol

- HyperText Transfer Protocol first released in 1991
  - First spec was w/ WWW (HTTP/0.9)
  - Current IETF spec is RFC7540 (HTTP/2) from 2015
    https://tools.ietf.org/html/rfc7540
  - HTTP/3 in the works now.
- Three things
  - HTTP Message
  - HTTP Methods
  - Other HTTP Features

HTTP Message

- Message-oriented protocol
- Start Line
- Header collection
- Body (optional)

```
POST /create-user HTTP/2
Host: www.tutorialspoint.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 46

name=mike&email=mike@example.org&sms=123456789
```

HTTP Methods

- Each message has a METHOD
- Methods are NOT function-names
- There are currently close to 40 registered methods
  https://www.iana.org/assignments/http-methods/http-methods.xhtml

- Most used methods (GET, POST, PUT, DELETE)
  - From CRUD coined by James Martin in 1983
  - HEAD , PATCH, OPTIONS often used, too.

HTTP Methods

Method Name	Safe	Idempotent	Reference
ACL	no	yes	[RFC3744, Section 8.1]
BASELINE-CONTROL	no	yes	[RFC3253, Section 12.6]
BIND	no	yes	[RFC5842, Section 4]
CHECKIN	no	yes	[RFC3253, Section 4.4, Section 9.4]
CHECKOUT	no	yes	[RFC3253, Section 4.3, Section 8.8]
CONNECT	no	no	[RFC7231, Section 4.3.6]
COPY	no	yes	[RFC4918, Section 9.8]
DELETE	no	yes	[RFC7231, Section 4.3.5]
GET	yes	yes	[RFC7231, Section 4.3.1]
HEAD	yes	yes	[RFC7231, Section 4.3.2]
LABEL	no	yes	[RFC3253, Section 8.2]
LINK	no	yes	[RFC2068, Section 19.6.1.2]
LOCK	no	no	[RFC4918, Section 9.10]
MERGE	no	yes	[RFC3253, Section 11.2]
MKACTIVITY	no	yes	[RFC3253, Section 13.5]
MKCALENDAR	no	yes	[RFC4791, Section 5.3.1][RFC8144, Section 2.3]
MKCOL	no	yes	[RFC4918, Section 9.3][RFC5689, Section 3][RFC8144, Section 2.3]
MKREDIRECTREF	no	yes	[RFC4437, Section 6]
MKWORKSPACE	no	yes	[RFC3253, Section 6.3]
MOVE	no	yes	[RFC4918, Section 9.9]
OPTIONS	yes	yes	[RFC7231, Section 4.3.7]
ORDERPATCH	no	yes	[RFC3648, Section 7]
PATCH	no	no	[RFC5789, Section 2]
POST	no	no	[RFC7231, Section 4.3.3]
PRI	yes	yes	[RFC7540, Section 3.5]
PROPFIND	yes	yes	[RFC4918, Section 9.1][RFC8144, Section 2.1]
PROPPATCH	no	yes	[RFC4918, Section 9.2][RFC8144, Section 2.2]
PUT	no	yes	[RFC7231, Section 4.3.4]
REBIND	no	yes	[RFC5842, Section 6]
REPORT	yes	yes	[RFC3253, Section 3.6][RFC8144, Section 2.1]
SEARCH	yes	yes	[RFC5323, Section 2]
TRACE	yes	yes	[RFC7231, Section 4.3.8]
UNBIND	no	yes	[RFC5842, Section 5]
UNCHECKOUT	no	yes	[RFC3253, Section 4.5]
UNLINK	no	yes	[RFC2068, Section 19.6.1.3]
UNLOCK	no	yes	[RFC4918, Section 9.11]
UPDATE	no	yes	[RFC3253, Section 7.1]
UPDATEREDIRECTREF	no	yes	[RFC4437, Section 7]
VERSION-CONTROL	no	yes	[RFC3253, Section 3.5]

Other HTTP Features

- Safety (READ/WRITE)
  - GET, HEAD are *safe*
  - PUT, POST, DELETE are *unsafe*
- Idempotence (REPEATABLE)
  - GET, PUT, DELETE are idempotent
  - POST is *non-idempotent*
- The Missing Response Problem

HTTP as Protocol

- HyperText Transfer Protocol first released in 1991
- Message-oriented
- Methods for intent
- Safety and idempotence

WWW as Common Practice

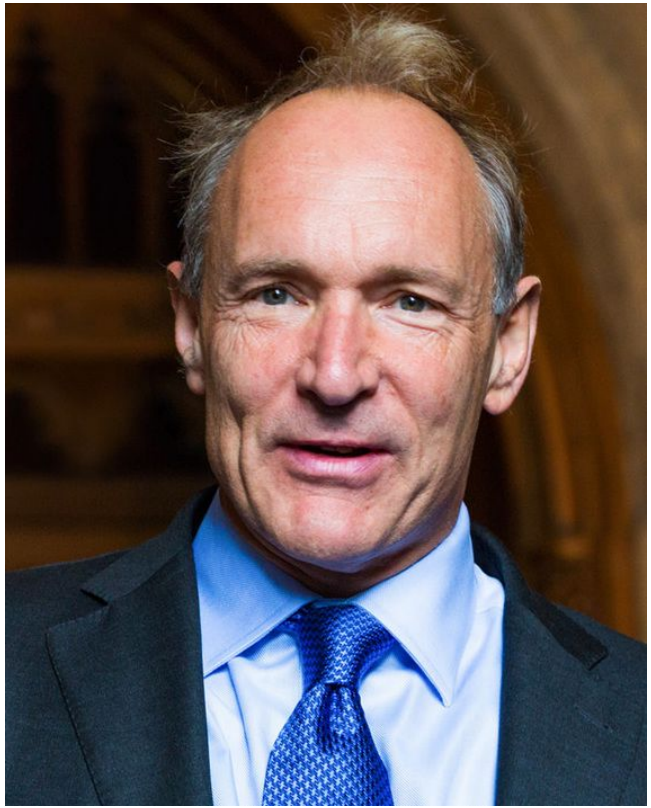
WWW as Common Practice

- CERN proposal in 1989 by Tim Berners-Lee
  https://www.w3.org/History/1989/proposal.html
- Used to be called the "World Wide Web"
  - Now just "the web" (we conquered "the world" part)
- Web is not a standard, it's a practice
- It's a decoupled collection, not a monolith
  - HTTP, HTML, CSS, JS, ...
  - PNG, GIF, JPEG, MP4, ...

The Web as Common Practice

"[A] non-linear text system known as hypertext."

Tim Berners-Lee



Practices of the Web

- It should be easy to publish and link to resources
- It should be decentralized (no single control/coordination)
- It should be easily extended (without breakage)

The power of the Web comes from clear standards everyone can agree upon, and an easy on-ramp for those just getting started.

REST as Style

- Roy Fielding PhD dissertation from 2000
  - "Architectural Styles and the Design of Network-based Software Architectures." https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- Chapter 5 of the dissertation is titled "Representational State Transfer (REST)"
- A style is a principled way to do something.

REST as Style

"[T]he architectural design of network-based application software through principled use of architectural constraints, thereby obtaining the functional, performance, and social properties desired of an architecture."

Roy T. Fielding

REST in one slide

Properties

- Performance
- Scalability
- Simplicity
- Modifiability
- Visibility
- Portability
- Reliability

REST in one slide

Properties

- Performance
- Scalability
- Simplicity
- Modifiability
- Visibility
- Portability
- Reliability

+ Requirements

- Low-Entry Barrier
- Extensibility
- Distributed Hypermedia
- Internet Scale

REST in one slide

Properties	+	Requirements	=	Constraints
● Performance		● Low-Entry Barrier		● Client-Server
● Scalability		● Extensibility		● Stateless
● Simplicity		● Distributed Hypermedia		● Cache
● Modifiability		● Internet Scale		● Uniform Interface
● Visibility				● Layered System
● Portability				● Code on Demand
● Reliability				

REST Style

*When combined, the **properties** and **constraints** provide a clear set of goals and principles developers can reference when creating APIs for the Web.*

Putting it all together...

Clarifying our thinking

- Naming things
- Concepts vs. instances
- Separation of concerns

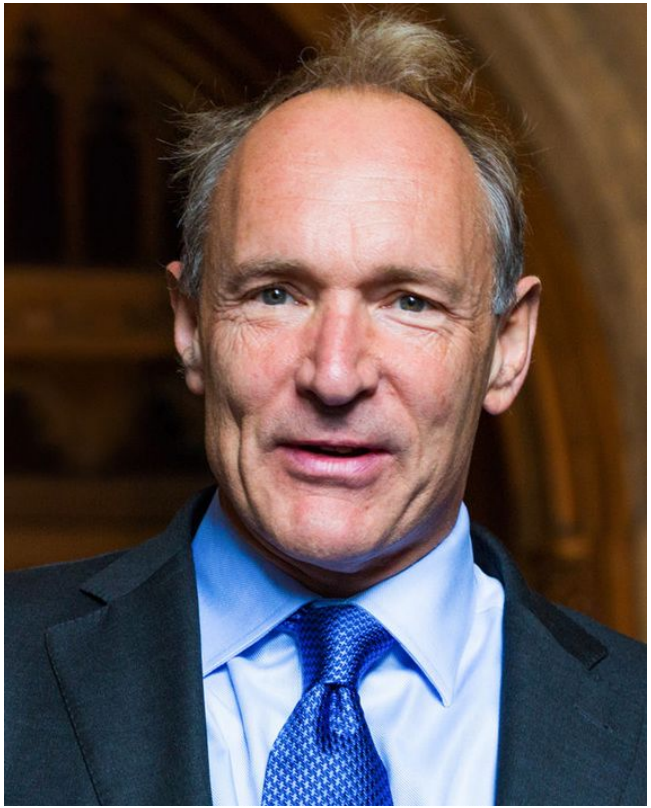
HTTP as Protocol

- HyperText Transfer Protocol first released in 1991
- Message-oriented
- Methods for intent
- Safety and idempotence

Practices of the Web

- It should be easy to publish and link to resources
- It should be decentralized (no single control/coordination)
- It should be easily extended (without breakage)

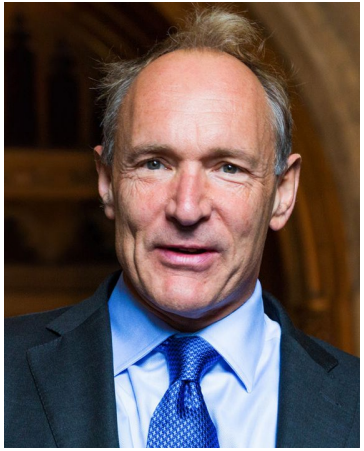
The power of the Web comes from clear standards everyone can agree upon, and an easy on-ramp for those just getting started.



REST Style

*When combined, the **properties** and **constraints** provide a clear set of goals and principles developers can reference when creating APIs for the Web.*

Using separation of concerns (SoC),
apply
protocols, practices, & style
to create
successful decentralized systems.



HTTP, the Web, and REST for API Developers

Mike Amundsen
@mamund