REST and the rest of the internet

George Malamidis (george@nutrun.com)
Danilo Sato (danilo@dtsato.com)

Licenced under a Creative Commons Licence

Please refer to the original dissertation about REST published by Roy Fielding entitled:
“Architectural Styles and the Design of Network-based Software Architectures”
http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Licenced under a Creative Commons Licence

Representational State Transfer is the architectural style described in Roy Fielding's "Architectural Styles and the Design of Network-based Software Architectures" dissertation. The design rational as proposed by REST can be viewed as the Web's architecture. With the Rails community having openly adopted a variety of REST's defining elements, we will discuss some of the main concepts behind REST and look at ways of how applying those can lead to improving the way we build and deliver Websites. We will place particular interest in the benefits and trade-offs REST principles introduce to network-based service development. Finally, we will look at how other established Internet standards can be applied to complement REST and potentially counter some of the trade-offs.

Data in REST

Licenced under a Creative Commons Licence

Resources

Licenced under a Creative Commons Licence

Resource is the key data abstraction in REST. Anything that can be named can qualify as a resource. Resources refer to the concept of the data they address.

Resource Identifiers

Licenced under a Creative Commons Licence

Resources are uniquely identified and addressable by URIs/URLs.

Representations

Licenced under a Creative Commons Licence

A Resource can have multiple representations, e.g. HTML, XML, Flash, etc.

Metadata and control

media type
last-modified
if-modified-since
cache-control
...

Licenced under a Creative Commons Licence

Constraints

Licenced under a Creative Commons Licence

REST consists of a set of constraints aimed at standardizing and optimizing component interaction over the Web. Adopting some of REST's constraints can yield numerous benefits, although it is important to understand that most come with a number of associated trade-offs, which implies that they shouldn't be blindly applied to every architecture.

Client-Server

Licenced under a Creative Commons Licence

The Client-Server constraint offers separation of concerns and allows the system's components to evolve independently, e.g. Web-servers / proxies / browsers.

Statelessness

Visibility
Reliability
Scalability

Licenced under a Creative Commons Licence

Visibility: only the content of one request is needed to understand the request
Reliability: easier to recover from failures
Scalability: servers can free resources between requests

Cache

Licenced under a Creative Commons Licence

Caching improves efficiency and scalability by relieving server strain.
“0-th” level caching: Your resources can be cached and you application doesn’t even have to know about it (squid, browser)

Uniform interface

Licenced under a Creative Commons Licence

Simplifies communication. Remote components have a standard way of communicating with each other.
Verbs and Status Codes (HTTP)

Layered System

Licenced under a Creative Commons Licence

Decoupling. Promotes component independence. Scalability (Caches, load balancers, etc).

Code on Demand

Licenced under a Creative Commons Licence

E.g JavaScript, applets. Simplifies clients while making them more flexible and relieves server load.

Hypermedia

Licenced under a Creative Commons Licence

* (non-linear) Graphics, audio, video, text and links as a non-linear medium of information.
* (Little knowledge) Hypermedia is a very powerful concept as it minimizes the amount of knowledge of a service's structure a client needs to have
* (follow and discover) Think of visiting a Website's index page and being able to follow links to navigate to all of the Website's resources
* (same for services) The same concept can be applied to services whose intended consumers are not human.
* (shared understading + microformats) If your consumer is human, he is the state machine navigating your website, if your consumer is another machine, some shared understanding of the semantics of the data is needed

songs

Licenced under a Creative Commons Licence

URI

/songs

Licenced under a Creative Commons Licence

/songs.(html|xml)

Multiple Representations

Licenced under a Creative Commons Licence

GET /songs.xml

Licenced under a Creative Commons Licence

GET /songs.xml

200 OK ← Status Codes

Licenced under a Creative Commons Licence

GET /songs.xml

200 OK

```
<songs>
  <song href="/songs/one" />
  <song href="/songs/sad_but_true" />
</songs>
```

Hypermedia

Licenced under a Creative Commons Licence

Relaxing REST

Licenced under a Creative Commons Licence

You don’t have to support everything just because it’s REST.
If you’re just building a website, why do you need
<input type=”hidden” name=”_method” value=”delete”/>

HTML Forms

GET

Licenced under a Creative Commons Licence

HTML Forms

GET ✓

Licenced under a Creative Commons Licence

HTML Forms

GET ✓

POST

Licenced under a Creative Commons Licence

HTML Forms

GET	✓
POST	✓

Licenced under a Creative Commons Licence

HTML Forms

GET ✓

POST ✓

PUT

Licenced under a Creative Commons Licence

HTML Forms

GET	✓
POST	✓
PUT	POST

Licenced under a Creative Commons Licence

HTML Forms

GET	✓
POST	✓
PUT	POST
DELETE	

Licenced under a Creative Commons Licence

HTML Forms

GET	✓
POST	✓
PUT	POST
DELETE	POST

Licenced under a Creative Commons Licence

405 Fail.

Licenced under a Creative Commons Licence

Facebook only POSTs to your webapp.
Servers get confused to serve static files by POST
405 Method Not Allowed

HTTP

The protocol that powers the Web

Licenced under a Creative Commons Licence

* HTTP is the most prominent implementation of REST
HTTP, the protocol that powers the Web, reflects REST's principles. Everyone who has ever built or even visited a website has been enjoying some of the benefits on offer, long before REST started receiving mainstream praise in development cycles. Following is a non-exhaustive list of reasons why HTTP presents a good candidate for use in developing network-based services.

Universally understood

Licenced under a Creative Commons Licence

By understanding and employing HTTP we can harness and reuse a plethora of software which understands it (Web servers, proxies, load balancers, frameworks, clients, etc)

Anarchic Scalability

Licenced under a Creative Commons Licence

There is no denying the Web has scaled well. 156 million websites / 1 trillion webpages. It has also scaled in an environment that is difficult to control or predict.

Collaboration
Unpredicted Evolvability

Licenced under a Creative Commons Licence

By offering service endpoints that respect the Web's underlying architecture, our services encourage others to use them in unpredictable, exciting new ways (Mashups, AWS, Delicious, Google).

Trade-offs

Licenced under a Creative Commons Licence

* It is mainstream
* But REST is not a silver bullet
* Applying REST (middleware) requires acknowledging the trade-offs

Stateless

Licenced under a Creative Commons Licence

Can decrease network performance due to repetitive requests. Polling can lead to unnecessary requests.

Cache

Licenced under a Creative Commons Licence

Stale caches are a difficult problem to solve. Also, there are applications that, because of their dynamic nature, don't easily lend themselves to caching. stale-while-revalidate is useful, but is a workaround rather than a complete remedy. There are scenarios where that is not acceptable.

Uniform Interface

Licenced under a Creative Commons Licence

Can degrade efficiency, because is generalized and not optimized to an application's specific needs.

Layered System

Licenced under a Creative Commons Licence

Every additional layer added to a system can incur overhead and latency.

Other established internet protocols

XMPP

BitTorrent

FTP

SMTP

Licenced under a Creative Commons Licence

There are protocols other than HTTP that have enjoyed internet scale success. Based on context, the nature of the problem we are addressing and the environment a system is meant to exist in, these technologies present valuable candidates for efficient network-based systems integration, or for complementing HTTP, while remedying some of its associated trade-offs.

Auction Watch

An imaginary example...

Licenced under a Creative Commons Licence

Auction Watch

- 1 Service
- 3000 Consumers
- Service publishes bid updates / accepts bids
- Consumers subscribe to bid updates / place bids
- Consumers must be authorized to communicate with the service

Licenced under a Creative Commons Licence

In HTTP...

Licenced under a Creative Commons Licence

Service publishes current price feed, consumers subscribe to feed and poll. Polling frequency is once every 10 seconds per consumer (enforced by the service).

In HTTP...

- 6 * 60 * 24 * 3000 = 25,920,000 requests/day
- Authorization = 25,920,000 handshakes/day
- Number of bids on the day = 20,000
- Number of unnecessary requests/handshakes = requests/day - bids/day = 25,900,000

Licenced under a Creative Commons Licence

In HTTP...

- Average bid frequency = 86400/20000 = 4.32 seconds < 10 seconds

Licenced under a Creative Commons Licence

The 20,000 bids/day assumption

The 10 second interval polling frequency is suboptimal when it comes to consumers being able to act on price updates in near real time.

Improvements

- ETag
- Last-Modified
- Conditional GET
- Partial GET

Licenced under a Creative Commons Licence

These reduce some unnecessary network usage, but do not reduce the number of requests, handshakes.
Caching and reverse proxies are also commonly employed for relieving server stress, although, due to the close to real time requirement of this scenario, configuring those effectively can be tricky.

In SMTP...

- Ebay offers email notifications to auction watchers
- The same could be applied to machine consumers and eliminate unnecessary requests

Licenced under a Creative Commons Licence

You don’t have to look very far for a push solution

In XMPP...

- Number of messages = number of bids = 20,000
- Number of handshakes = number of connections = number of consumers = 3,000
- Number of unnecessary requests/ handshakes = 0

Licenced under a Creative Commons Licence

Service publishes updates on XMPP PubSub nodes, consumers subscribe to nodes and receive updates as these happen.
It is important to remember that this would not be appropriate if the number of consumers interacting with the service is outside our control. With each consumer maintaining an open connection, the service never gets the opportunity to release system resources and there is a finite number of persistent connections a physical infrastructure can accommodate.

Summary

Licenced under a Creative Commons Licence

Playing “Back in Black”, AC/DC as suggested by Chad Fowler :)

Summary

Licenced under a Creative Commons Licence

Playing “Back in Black”, AC/DC as suggested by Chad Fowler :)