Service Based Approach Intro

WS Slides #1

Content

- Web Services
- REST
- Client State

Service Based Background

The Web is a marvelous "application"

- Has been up 24/7 for 30-40 years
- Has been able to expand many magnitudes
- More users, more data, more advanced services … the perfect application?

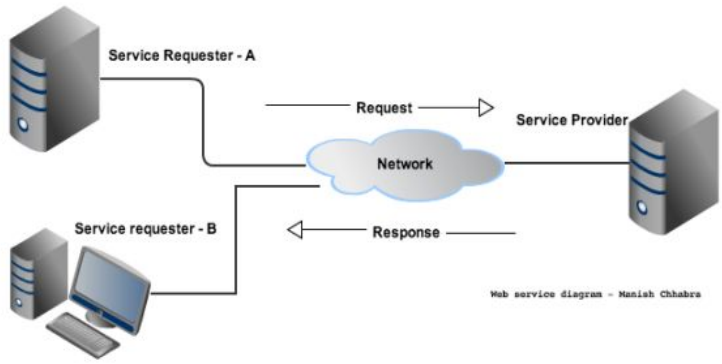
Wouldn't it be good to build our application like that??

- So what are the key principles behind the Web?

In slide

- XML, JSON, data formats
- GET, PUT, POST DELETE, the "HTTP verbs"
- Basic-Auth and OAuth, authorization
- REST, an architectural style

Web Services



A Web service

- Is a software system designed to support interoperable machine-to-machine (m2m) interaction over a network.
- Service has an agreed on/public interface/API
- Other systems interact with the Web service in a manner prescribed by its description

Types of Web Services

- WS-*, A stateless messaging service (Simple Object Access Protocol, SOAP), describing service interfaces in XML (Web Services Description Language, WSDL).
  - Heavyweight.
  - Code generation from WSDL and conversion to objects. WSDL example
  - WS-* is a service oriented approach. The key abstraction is a service (a verb)
- WS-REST, RESTful Web Service, an architectural style ....
  - WS-REST, is not service oriented, it's resource-oriented, the key abstraction is a resource (a noun). Web Service for REST is a bit misleading

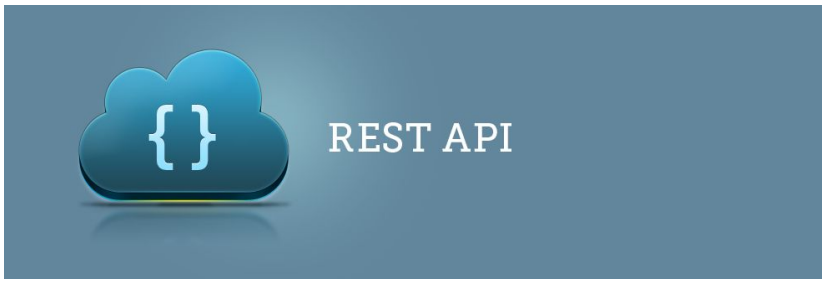
Programmer's view

- The application is composed of loosely coupled, distributed, reusable, platform/language independent services (resources)

Roles

- Consuming a Web Service, i.e a client
- Producing, implement a Web Service

Representational State Transfer (REST)



REST

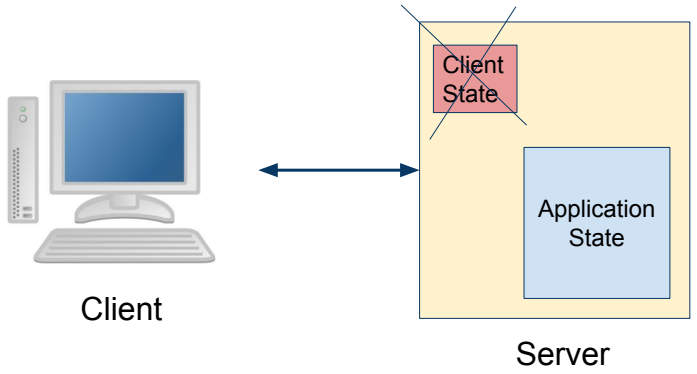
- And from Wikipedia, REST

Key principles that makes the web work and scale

1. Identification of resources (as URIs)
2. Manipulating of resources through representations (in responses we get an representation of the resource, for example as HTML/JSON/XML)
3. Self-descriptive messages (each message contains all the information necessary to complete the task i.e."stateless")
4. Hypermedia as the engine of application state (HATEOAS), the client/server interaction state is in the hypermedia they exchange

// Roy Fielding, author of HTTP specification

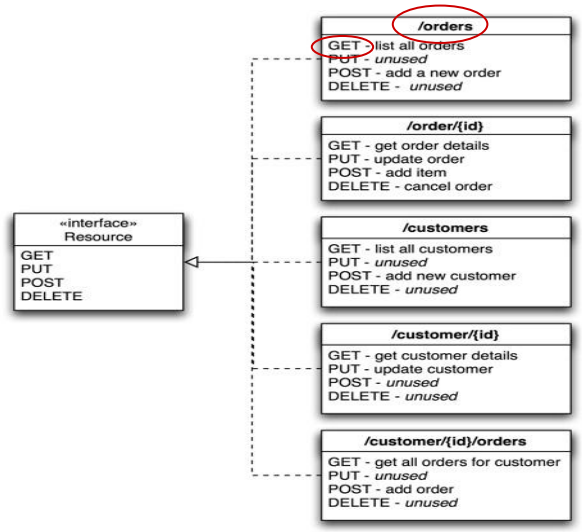
Stateless … What state?



Our application will have state, so how is REST stateless?

- Answer: No client state!
  - I.e no session on server
- Everything needed must be in request (also authorization, more later …)

Implementing REST



Example of practical interpretation of RESTful CRUD interface (create, read, update, delete,... the basic operations on data)

1. All resources accessible with URLs (http://www.server.com/application/orders)
2. We will get object representations as JSON (or other)
3. HTTP is stateless and self descriptive (simple unified interface: GET, POST, PUT, DELETE, … the verbs used for operating on the resources)
4. Embed links in response (upcoming)

Implementing HATEOAS

```
GET /account/12345

HTTP/1.1 HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
    <link rel="withdraw" href="/account/12345/withdraw" />
    <link rel="transfer" href="/account/12345/transfer" />
    <link rel="close" href="/account/12345/close" />
</account>
```

HATEOAS means that hypertext should be used to find your way through the API.

- The GET response contains requested data and …
- … links to associated data.
- API is self exposing! The hypertext is actually telling us what is allowed and what not
- Slide example is XML (we mostly use JSON, more to come …)

Testing some RESTful Services

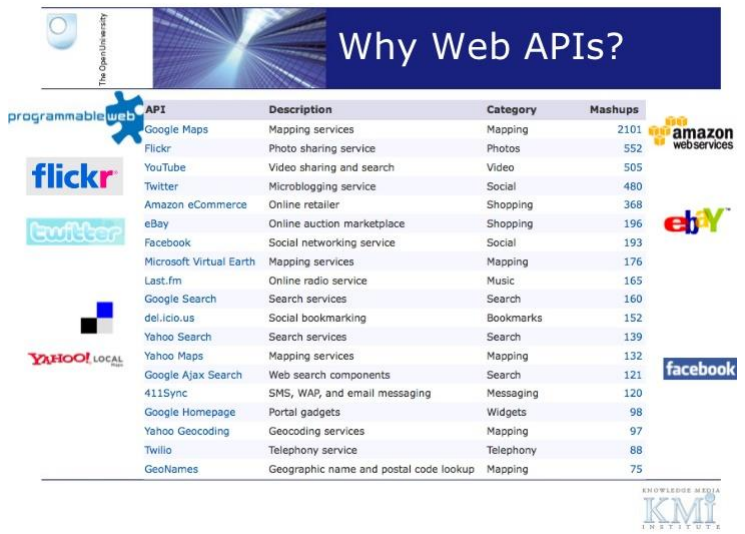
Flickr (photo service, no API key)
http://api.flickr.com/services/feeds/photos_public.gne?tags=flower&lang=en-us&format=atom (try change format)

Google Maps (no API key)
https://www.google.se/maps/@59.7009921,11.8938365,10z?hl=sv (try change coordinates)

Many public RESTful services available.

- WebAPIs
- NOTE: Some APIs are
  - … client side, i.e. possible to call with client side JavaScript, some are …
  - … server side, must call from server side code (often involving OAuth … a pain)
  - … and some are both.
- Possibly need an API-key to send with requests (must get one from the producer)
  - Example: https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyCQKzCM8wvY…
- Possibly must register application
  - If so need an account (Google, FaceBook, Twitter, Amazon, …) …
  - Search for Developer pages

Web APIs

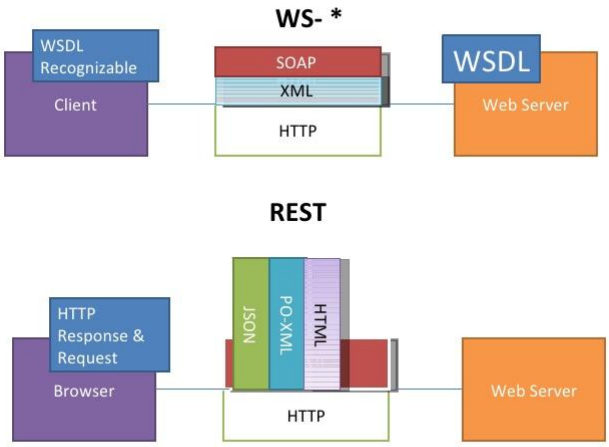


API	Description	Category	Mashups
Google Maps	Mapping services	Mapping	2101
Flickr	Photo sharing service	Photos	552
YouTube	Video sharing and search	Video	505
Twitter	Microblogging service	Social	480
Amazon eCommerce	Online retailer	Shopping	368
eBay	Online auction marketplace	Shopping	196
Facebook	Social networking service	Social	193
Microsoft Virtual Earth	Mapping services	Mapping	176
Last.fm	Online radio service	Music	165
Google Search	Search services	Search	160
del.icio.us	Social bookmarking	Bookmarks	152
Yahoo Search	Search services	Search	139
Yahoo Maps	Mapping services	Mapping	132
Google Ajax Search	Web search components	Search	121
411Sync	SMS, WAP, and email messaging	Messaging	120
Google Homepage	Portal gadgets	Widgets	98
Yahoo Geocoding	Geocoding services	Mapping	97
Twilio	Telephony service	Telephony	88
GeoNames	Geographic name and postal code lookup	Mapping	75

Many many APIs exposed as RESTful services

- Find your API

The RESTful Hype



REST in rather new, REST is very hyped right now … (seems to work for many cases, but …)

- … watch this … and this
- Still, .. we only use WS-REST