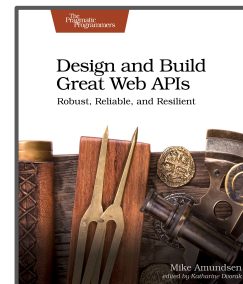


# Learning an API Design Method

Mike Amundsen  
@mamund



# Learning the API Design Method

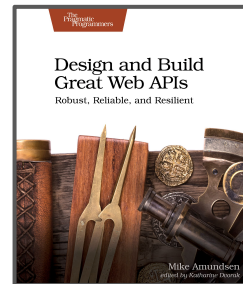
- Design the Interface
- Not the Implementation
- Five Step API Design Method



#mcaTravels

@mamund

#api360



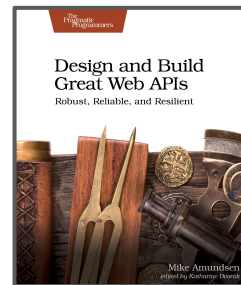
# Design the Interface



#mcaTravels

@mamund

#api360



“ Few people think about it or are aware of it. But there is nothing made by human beings that does not involve a design decision somewhere.”

Bill Moggridge  
Interaction Design Pioneer



# Functionality, Usability, and User Experience: Three Areas of Concern

Niamh McNamara | University College Cork, Ireland | [n.mcnamara@ucc.ie](mailto:n.mcnamara@ucc.ie)

Jurek Kirakowski | University College Cork, Ireland | [jzk@ucc.ie](mailto:jzk@ucc.ie)

design

Functionality

Usability

Experience

A close-up photograph of a stack of four slices of golden-brown toast. The toast is resting on a dark wooden rack with a slatted design. The word "toast" is overlaid in a large, bold, yellow sans-serif font in the center-right of the image. In the background, a white bowl, a white plate, and a silver spoon are visible on a light-colored table. The lighting is warm, highlighting the texture of the toast.

toast



# Functionality



# Usability





# Experience



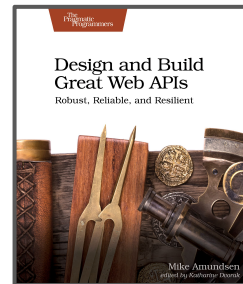
# Not the Implementation



#mcaTravels

@mamund

#api360



# Design Guidelines

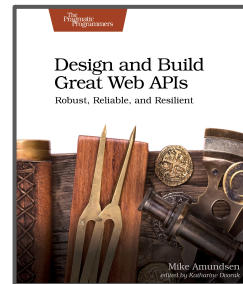
- Craft [good/pretty/usable/stable] URIs



#mcaTravels

@mamund

#api360



Term	Description
Authority	A URI component that identifies the party with jurisdiction over the namespace defined by the remainder of the URI.
Collection	A resource archetype used to model a server-managed <i>directory</i> of resources.
Controller	A resource archetype used to model a procedural concept.
CRUD	An acronym that stands for the four classic storage-oriented functions: create, retrieve, update, and delete.
Developer portal	A Web-based graphical user interface that helps a REST API acquire new clients.
Docroot	A resource that is the hierarchical ancestor of all other resources within a REST API's model. This resource's URI should be the advertised entry point.
Document	A resource archetype used to model a singular concept.
Forward slash separator (/)	Used within the URI path component to separate hierarchically related resources.
Opacity of URIs	An axiom, originally described by Tim Berners-Lee, that governs the visibility of a resource identifier's composition.
Parent resource	The document, collection, or store that governs a given subordinate concept by preceding it within a URI's hierarchical path.
Query	A URI component that comes after the path and before the optional fragment.
Resource archetypes	A set of four intrinsic concepts (document, collection, store, and controller) that may be used to help describe a REST API's model.
Store	A resource archetype used to model a client-managed resource repository.
URI path segment	Part of a resource identifier that represents a single node within a larger, hierarchical resource model.
URI template	A resource identifier syntax that includes variables that must be substituted before resolution.



# Design Guidelines

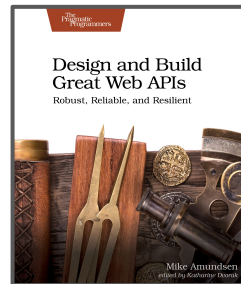
- Craft [good/pretty/usable/stable] URIs
- Map domain actions to HTTP methods (CRUD)



#mcaTravels

@mamund

#api360





Term	Description
DELETE	HTTP request method used to remove its parent.
GET	HTTP request method used to retrieve a representation of a resource's state.
HEAD	HTTP request method used to retrieve the metadata associated with the resource's state.
OPTIONS	HTTP request method used to retrieve metadata that describes a resource's available interactions.
POST	HTTP request method used to create a new resource within a collection or execute a controller.
PUT	HTTP request method used to insert a new resource into a store or update a mutable resource.
Request-Line	RFC 2616 defines its syntax as Method SP Request-URI SP HTTP-Version CRLF
Request method	Indicates the desired action to be performed on the request message's identified resource.
Response status code	A three-digit numeric value that is communicated by a server to indicate the result of a client's request.
Status-Line	RFC 2616 defines its syntax as: HTTP-Version SP Status-Code SP Reason-Phrase CRLF
Tunneling	An abuse of HTTP that masks or misrepresents a message's intent and undermines the protocol's transparency.

# Design Guidelines

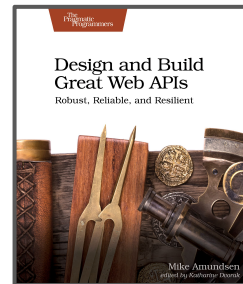
- Craft [good/pretty/usable/stable] URIs
- Map domain actions to HTTP methods (CRUD)
- Use the proper HTTP Status Codes



#mcaTravels

@mamund

#api360



Code	Name	Meaning
400	Bad Request	Indicates a nonspecific client error
401	Unauthorized	Sent when the client either provided invalid credentials or forgot to send them
402	Forbidden	Sent to deny access to a protected resource
404	Not Found	Sent when the client tried to interact with a URI that the REST API could not map to a resource
405	Method Not Allowed	Sent when the client tried to interact using an unsupported HTTP method
406	Not Acceptable	Sent when the client tried to request data in an unsupported media type format
409	Conflict	Indicates that the client attempted to violate resource state
412	Precondition Failed	Tells the client that one of its preconditions was not met
415	Unsupported Media Type	Sent when the client submitted data in an unsupported media type format
500	Internal Server Error	Tells the client that the API is having problems of its own

# Design Guidelines

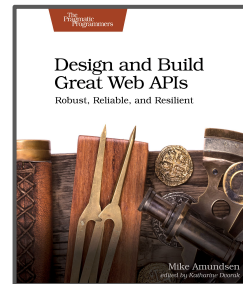
- Craft [good/pretty/usable/stable] URIs
- Map domain actions to HTTP methods (CRUD)
- Use the proper HTTP Status Codes
- Document serialized objects as HTTP bodies



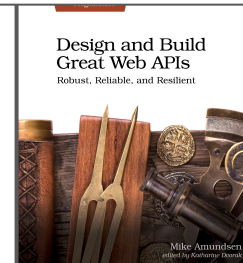
#mcaTravels

@mamund

#api360



Term	Description
Field	A named slot with some associated information that is stored in its value.
Form	A structured representation that consists of the fields and links, which are defined by an associated schema.
Format	Describes a form's presentation apart from its schematic.
Link	An actionable reference to a resource.
Link formula	A boolean expression that may serve as HATEOAS calculator's input in order to determine the availability of state-sensitive hypermedia within a form.
Link relation	Describes a connection between two resources.
Schema	Describes a representational form's structure independent of its format.
State fact	A Boolean variable that communicates a condition that is relevant to some state-sensitive hypermedia.





# Design Guidelines

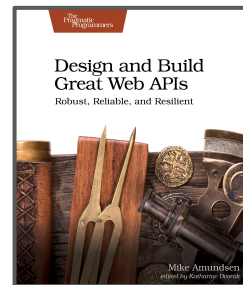
- Craft [good/pretty/usable/stable] URIs
- Map domain actions to HTTP methods (CRUD)
- Use the proper HTTP Status Codes
- Document serialized objects as HTTP bodies
- Use HTTP headers responsibly



#mcaTravels

@mamund

#api360



Code	Purpose
Content-Type	Identifies the entity body's media type
Content-Length	The size (in bytes) of the entity body
Last-Modified	The date-time of last resource representation's change
ETag	Indicates the version of the response message's entity
Cache-Control	A TTL-based caching value (in seconds)
Location	Provides the URI of a resource

# Design Guidelines

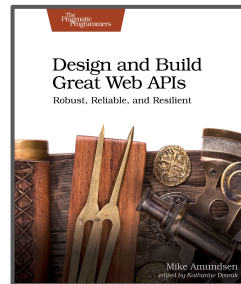
- Craft [good/pretty/usable/stable] URIs
- Map domain actions to HTTP methods (CRUD)
- Use the proper HTTP Status Codes
- Document serialized objects as HTTP bodies
- Use HTTP headers responsibly
- Describe edge cases (async, errors, authN/Z)



#mcaTravels

@mamund

#api360



## # Response

HTTP/1.1 202 Accepted ❶

Content-Type: application/xml; charset=UTF-8

Content-Location: <http://www.example.org/images/task/1>

Date: Sun, 13 Sep 2009 01:49:27 GMT

```
<status xmlns:atom="http://www.w3.org/2005/Atom">
```

```
  <state>pending</state>
```

```
  <atom:link href="http://www.example.org/images/task/1" rel="self"/>
```

```
  <message>Your request has been accepted for processing.</message>
```

```
  <ping-after>2009-09-13T01:59:27Z</ping-after> ❷
```

```
</status>
```



## # Response

HTTP/1.1 409 Conflict

Content-Type: application/xml; charset=UTF-8

Content-Language: en

Date: Wed, 14 Oct 2009 10:16:54 GMT

Link: <http://www.example.org/errors/limits.html>; rel="help"

```
<error xmlns:atom="http://www.w3.org/2005/Atom">
```

```
  <message>Account limit exceeded. We cannot complete the transfer due to  
  insufficient funds in your accounts</message>
```

```
  <error-id>321-553-495</error-id>
```

```
  <account-from>urn:example:account:1234</account-from>
```

```
  <account-to>urn:example:account:5678</account-to>
```

```
  <atom:link href="http://example.org/account/1234"  
             rel="http://example.org/rels/transfer/from"/>
```

```
  <atom:link href="http://example.org/account/5678"  
             rel="http://example.org/rels/transfer/to"/>
```

```
</error>
```



# Request to obtain a request token

POST /request\_token HTTP/1.1 ❶

Host: www.example.org

Authorization: OAuth realm="http://www.example.com/photos",  
                  oauth\_consumer\_key=a1191fd420e0164c2f9aeac32ed35d23,  
                  oauth\_nonce=109843dea839120a,  
                  oauth\_signature=d8e19bb988110380a72f6ca33b2ba5903272fe1,  
                  oauth\_signature\_method=HMAC-SHA1,  
                  oauth\_timestamp=1258308730,  
                  oauth\_version=1.0 ❷

Content-Length: 0

# Response containing a request token and a secret

HTTP/1.1 200 OK

Content-Type: application/x-www-form-urlencoded

oauth\_token=0e713d524f290676de8aff4073b1bb52e37f065c

&oauth\_token\_secret=394bc633d4c93f79aa0539fd554937760f05987c ❸

---

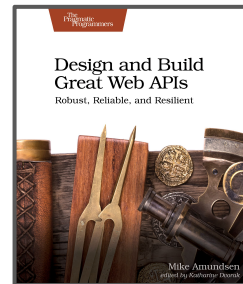
*But that's implementation detail!*



#mcaTravels

@mamund

#api360



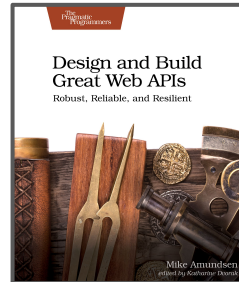
# API Design Method



#mcaTravels

@mamund

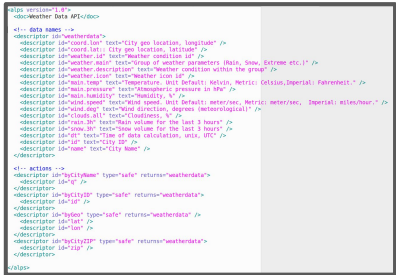
#api360



## *A repeatable process to govern the creation of interfaces*

- ```
Get Weather Data
=====

Home -> ByCityName(name)      -> WeatherData
Home -> ByCityID(id)          -> WeatherData
Home -> ByLatLong(lat,lon)    -> WeatherData
Home -> ByZipCode(zip)        -> WeatherData
```

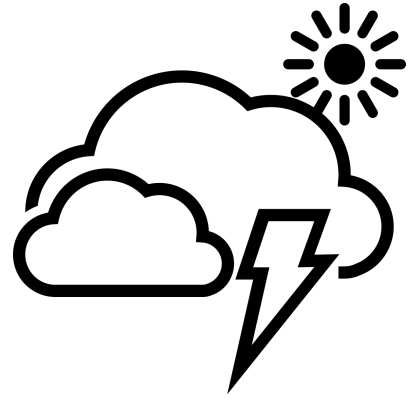


## Produce a Workflow Sketch

[illegible]



# Workflow Sketch: Weather API



Get Weather Data

=====

|      |    |                    |    |             |
|------|----|--------------------|----|-------------|
| Home | -> | ByCityName(name)   | -> | WeatherData |
| Home | -> | ByCityID(id)       | -> | WeatherData |
| Home | -> | ByLatLong(lat,lon) | -> | WeatherData |
| Home | -> | ByZipCode(zip)     | -> | WeatherData |

# Draw a Diagram

```

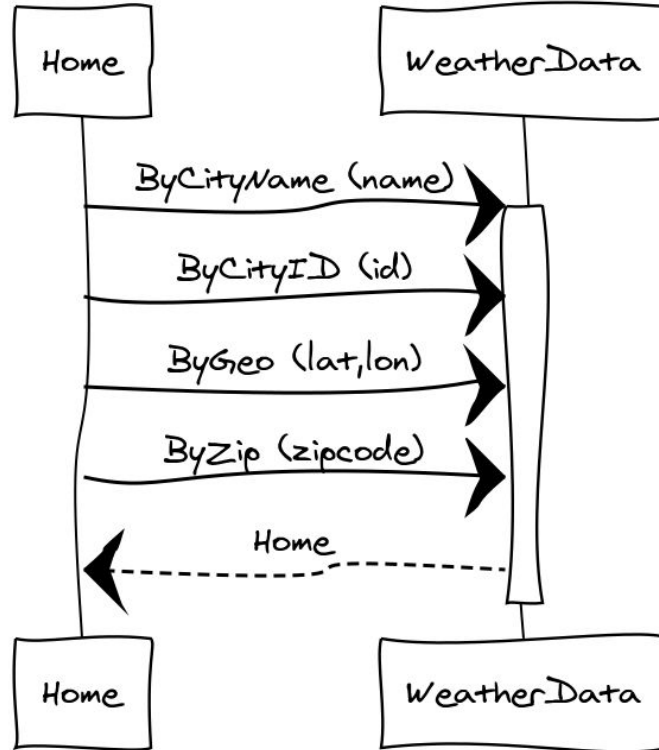
@data version 1.0.0
@uri weather Data API: /v1/

<!-- Data Names -->
<descriptor id="weatherData">
  <descriptor id="coord lat" text="City geo location, latitude" />
  <descriptor id="coord lon" text="City geo location, longitude" />
  <descriptor id="coord alt" text="City geo location, altitude" />
  <descriptor id="weather id" text="Weather condition id" />
  <descriptor id="weather desc" text="Description of weather conditions (Rain, Snow, Extreme etc.)" />
  <descriptor id="weather description" text="Weather condition within the group" />
  <descriptor id="temp" text="Temperature (Unit Default: Celsius, Metric: Celsius, Imperial: Fahrenheit)" />
  <descriptor id="temp max" text="Maximum Temperature Pressure in MPa" />
  <descriptor id="temp min" text="Minimum Temperature" />
  <descriptor id="wind speed" text="Wind speed (Unit Default: meters/sec, Metric: meters/sec, Imperial: miles/hour)" />
  <descriptor id="wind dir" text="Wind direction, degrees (meteorological)" />
  <descriptor id="cloud all" text="Cloudiness, %" />
  <descriptor id="cloud 3h" text="Cloudiness, % for the last 3 hours" />
  <descriptor id="precip" text="Precipitation volume for the last 3 hours" />
  <descriptor id="sun" text="Time of data calculation, units, UTC" />
  <descriptor id="city" text="City Name" />
</descriptor>

<!-- Actions -->
<descriptor id="getWeather" type="safe" returns="weatherData">
  <descriptor id="q" />
</descriptor>
<descriptor id="getWeather" type="safe" returns="weatherData">
  <descriptor id="lat" />
</descriptor>
<descriptor id="getWeather" type="safe" returns="weatherData">
  <descriptor id="lon" />
</descriptor>
<descriptor id="getWeather" type="safe" returns="weatherData">
  <descriptor id="alt" />
</descriptor>
</data>

```

# Diagram: Weather API



## Apply Vocabularies

[illegible]

# Weather Data

- Gather all the data points
- Gather all the action names
- Normalize vocabularies
  - Open
  - Industry
  - Internal

[illegible]

# Before Applying Vocabularies

### === Weather Data

```

longitude:: City geo location, longitude
latitude:: City geo location, latitude
id:: Weather condition id
forecast:: Group of weather parameters (Rain, Snow, Extreme etc.)
description:: Weather condition within the group
icon:: Weather icon id
tempDegrees:: Temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit
pressureHPA:: Atmospheric pressure in hPa
humidityPCT:: Humidity, %
wind:: Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: mile/hour
direction:: Wind direction, degrees (meteorological)
cloudStatus:: Cloudiness, %
recentRain:: Rain volume for the last 3 hours
recentSnow:: Snow volume for the last 3 hours
dateTime:: Time of data calculation, unix, UTC
cityId:: City ID
cityName:: City name

```

[illegible]

# Sources for Vocabularies

- IANA Link Relation Values
- schema.org, Activity Streams
- microformats, Dublin Core
- Industry Vocabularies (Semantic Sensor Web)
- Internal Vocabularies

[illegible]



# After Applying Vocabularies

## === Weather Data

coord.lon:: City geo location, longitude  
coord.lat:: City geo location, latitude  
weather.id:: Weather condition id  
weather.main:: Group of weather parameters (Rain, Snow, Extreme etc.)  
weather.description:: Weather condition within the group  
weather.icon:: Weather icon id  
main.temp:: Temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: F  
main.pressure:: Atmospheric pressure in hPa  
main.humidity:: Humidity, %  
wind.speed:: Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: mph  
wind.deg:: Wind direction, degrees (meteorological)  
clouds.all:: Cloudiness, %  
rain.3h:: Rain volume for the last 3 hours  
snow.3h:: Snow volume for the last 3 hours  
dt:: Time of data calculation, unix, UTC  
id:: City ID  
name:: City name

**\*byCityName\***

**\*byCityID\***

**\*byGeo\***

**\*byZIP\***

```
JSON version 1.0
{
  "weather": {
    "data": {
      "city": {
        "id": "1",
        "name": "New York",
        "coord": {
          "lon": -74.006,
          "lat": 40.7128
        },
        "weather": {
          "id": "1",
          "main": {
            "temp": 293.15,
            "pressure": 1013,
            "humidity": 75,
            "wind": {
              "speed": 1.6,
              "deg": 90
            },
            "clouds": {
              "all": 80
            },
            "rain": {
              "3h": 0
            },
            "snow": {
              "3h": 0
            },
            "dt": 1415160000
          },
          "description": "Partly cloudy",
          "icon": "02d"
        }
      }
    }
  }
}
```

# After Applying Vocabularies

## === Weather Data

coord.lon:: City geo location, longitude  
coord.lat:: City geo location, latitude  
weather.id:: Weather condition id  
weather.main:: Group of weather parameters (Rain, Snow, Extreme etc.)

## Normalize Data and Specify Actions

main.humidity:: Humidity, %  
wind.speed:: Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: miles/hour  
wind.deg:: Wind direction, degrees (meteorological)  
clouds.all:: Cloudiness, %  
rain.3h:: Rain volume for the last 3 hours  
snow.3h:: Snow volume for the last 3 hours  
dt:: Time of data calculation, unix, UTC  
id:: City ID  
name:: City name

**\*byCityName\***  
**\*byCityID\***  
**\*byGeo\***  
**\*byZIP\***

```
<?xml version="1.0"?>
<!-- Weather Data API Schema -->
<!-- Data Names -->
<!-- Data Name: weatherData -->
<!-- Data Name: coord -->
<!-- Data Name: weather -->
<!-- Data Name: main -->
<!-- Data Name: wind -->
<!-- Data Name: clouds -->
<!-- Data Name: rain -->
<!-- Data Name: snow -->
<!-- Data Name: dt -->
<!-- Data Name: id -->
<!-- Data Name: name -->
<!-- Actions -->
<!-- Action: byCityName -->
<!-- Action: byCityID -->
<!-- Action: byGeo -->
<!-- Action: byZIP -->
</xml>
```

# Create a Description Document

[illegible]

# Description vs. Definitions

- Describing the interface doesn't define it.
- Description languages
  - ALPS
  - DCAP
  - JSON Home

[illegible]

# Description vs. Definitions

- Describing the interface doesn't define it.
- Description languages
  - ALPS
  - DCAP
  - JSON Home
- Definition languages
  - WSDL
  - Swagger
  - RAML
  - Blueprint

[illegible]

# Description vs. Definitions

- Describing the interface doesn't define it.
- Description languages
  - **ALPS**
  - DCAP
  - JSON Home
- Definition languages
  - WSDL
  - Swagger
  - RAML
  - Blueprint

[illegible]

```
<alps version="1.0">
  <doc>Weather Data API</doc>

  <!-- data names -->
  <descriptor id="weatherdata">
    <descriptor id="coord.lon" text="City geo location, longitude" />
    <descriptor id="coord.lat:: City geo location, latitude" />
    <descriptor id="weather.id" text="Weather condition id" />
    <descriptor id="weather.main" text="Group of weather parameters (Rain, Snow, Extreme etc.)" />
    <descriptor id="weather.description" text="Weather condition within the group" />
    <descriptor id="weather.icon" text="Weather icon id" />
    <descriptor id="main.temp" text="Temperature. Unit Default: Kelvin, Metric: Celsius,Imperial: Fahrenheit." />
    <descriptor id="main.pressure" text="Atmospheric pressure in hPa" />
    <descriptor id="main.humidity" text="Humidity, %" />
    <descriptor id="wind.speed" text="Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: miles/hour." />
    <descriptor id="wind.deg" text="Wind direction, degrees (meteorological)" />
    <descriptor id="clouds.all" text="Cloudiness, %" />
    <descriptor id="rain.3h" text="Rain volume for the last 3 hours" />
    <descriptor id="snow.3h" text="Snow volume for the last 3 hours" />
    <descriptor id="dt" text="Time of data calculation, unix, UTC" />
    <descriptor id="id" text="City ID" />
    <descriptor id="name" text="City Name" />
  </descriptor>

  <!-- actions -->
  <descriptor id="byCityName" type="safe" returns="weatherdata">
    <descriptor id="q" />
  </descriptor>
  <descriptor id="byCityID" type="safe" returns="weatherdata">
    <descriptor id="id" />
  </descriptor>
  <descriptor id="byGeo" type="safe" returns="weatherdata" />
    <descriptor id="lat" />
    <descriptor id="lon" />
  </descriptor>
  <descriptor id="byCityZIP" type="safe" returns="weatherdata">
    <descriptor id="zip" />
  </descriptor>

</alps>
```



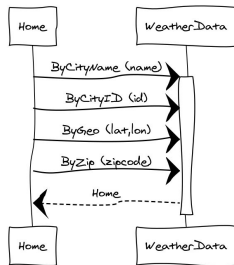
# Design Artifacts

- Workflow Sketch
- Diagram
- Description Document (ALPS)

## Check these into source control

## Get Weather Data

```
Home -> ByCityName(name)      -> WeatherData
Home -> ByCityID(id)          -> WeatherData
Home -> ByLatLong(lat,lon)    -> WeatherData
Home -> ByZipCode(zip)        -> WeatherData
```

[illegible]

# Learning the API Design Method

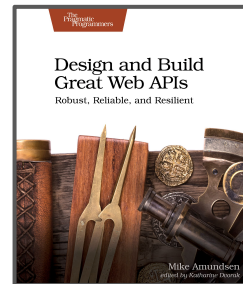
- Design the Interface
- Not the Implementation
- Four Step API Design Method



#mcaTravels

@mamund

#api360



# Learning an API Design Method

Mike Amundsen  
@mamund

