Get Some REST –On Practical RESTful API Design

CodeMash 2016

Priya Rajagopal

Twitter: @rajagp
www.priyaontech.com

Invicara (Director, Mobile Development)
http://invicara.com

dailyKARMA (Principal Tech Consultant)
http://dailykarma.com

Many Popular WebServices…

Many Popular WebServices…

That Strive to be RESTFul

A lot of noise around REST

"Stands for Representational State Transfer "

"RPC mechanism"

"Architecture Specification for consuming web services"

"Light weight alternate to SOAP!"

"HTTP CRUD operations with JSON"

"Standard on how to publish and consume web services"

"Client - server Protocol"

So…what *is* REST?

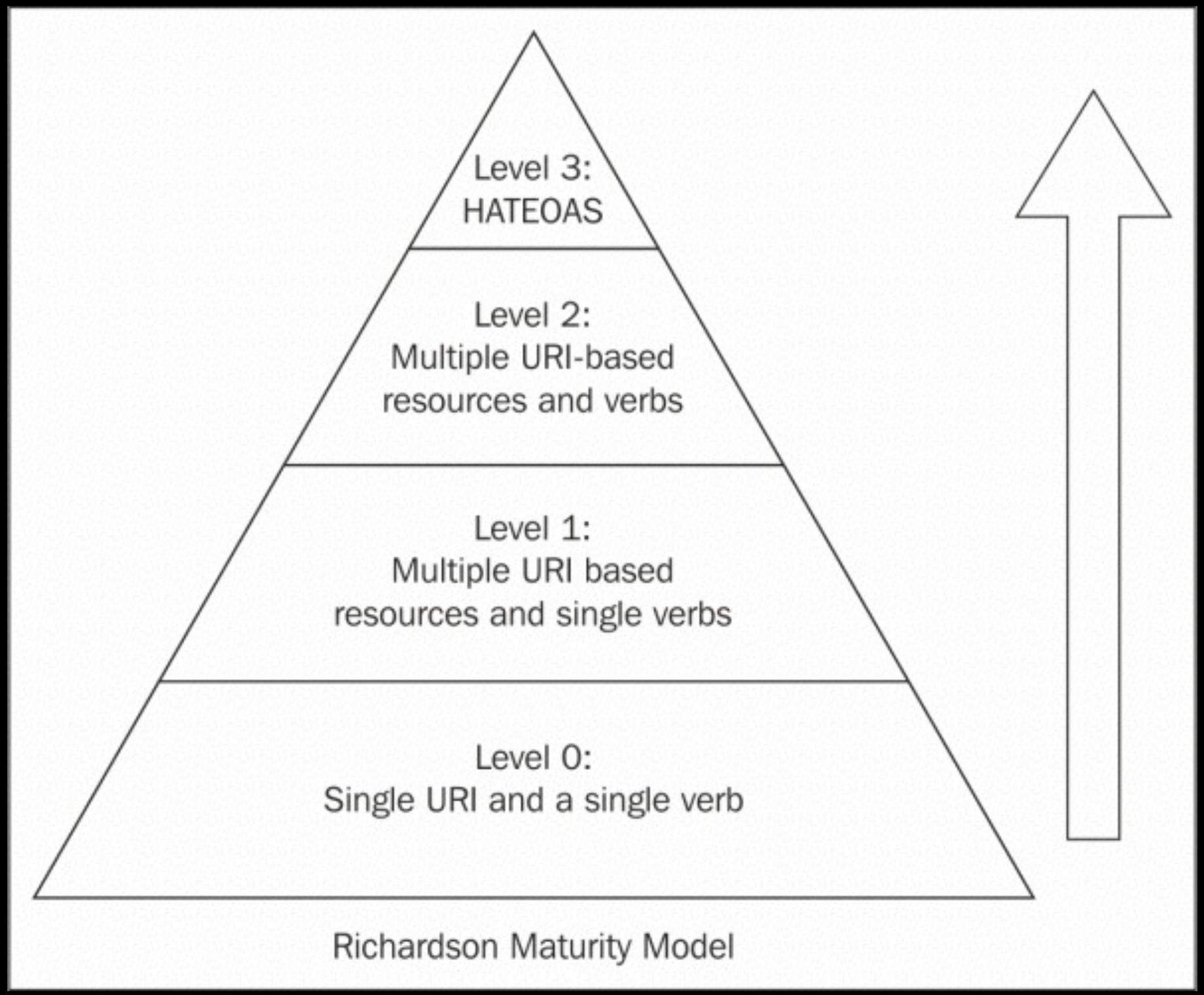
- A set of architectural constraints , guidelines and best practices on how web services can be consumed by a client
  - It is an “architectural style”
  - Roy T. Fielding’s Doctoral Dissertation - Father of REST
  - Not a protocol
  - Not a standard, but standards-based
  - Not tied to specific data transfer protocol (although HTTP used almost universally)
  - Not tied to a specific data representation

Architectural Principles of REST

Academic ….

- Client-Server
- Uniquely Addressable Resources
- Statelessness
- Cacheable
- Layered/ Proxies
- Optional (Code On Demand)

Richardson Maturity Model



RESTifarian (noun): A
***RESTifarian** is a zealous proponent of the REST software architectural style as defined by Roy T. Fielding in Chapter 5 of his PhD. dissertation at UCIrvine**

Key Considerations

- Resources
- Resource URIs
- Data Representation
- Protocol
- Security
- Versioning
- Response Data Control
- Errors
- Caching

Resources

The Core of REST API

Resources

- Objects that clients interact with.
- Limited Actions
- A unique URI
- Associated Data (with a representation)
- Relationships

Actions On Resources

- HTTP works well but Not a REST requirement
- Actions Communicated Over HTTP
  - More than just transport mechanism
- HTTP verbs
  - POST for Creating a resource
  - GET for Fetching resource [representation]
  - PUT for Replacing resource
  - DELETE for Removing a resource

Creating a Resource

(Success Case)

Method	Purpose	Request Body	Response
HTTP POST	Creating Resource, Sub-resource	Details of resource to be created	200 OK w/ Resource created 201 Created w/ location of resource 202 Accepted (async) w/ URI to track

- Not Safe, Not Idempotent
- Can I use POST to update a resource?

Fetching a Resource
(Success Case)

Method	Purpose	Request Body	Response
HTTP GET	Fetch Resource (incl.cache headers)	Optional: filtering, sorting partams, paging	200 OK w/ Requested Resource 204 No Content

- Safe, Idempotent

Replacing a Resource

(Success Case)

Method	Purpose	Request Body	Response
HTTP PUT	Complete Update Resource (incl. cache headers)	Entire Resource incl. fields that do not change	200 OK w/ Updated Resource

- Idempotent
- Can I use PUT to create a resource?

Removing a Resource

(Success Case)

Method	Purpose	Request Body	Response
HTTP DELETE	Removing Resource	None	204 No Content

- Idempotent

Partial Updates

- Modeling Data To Update As Sub-Resource
  - **PUT /users/:userId/preferences**

```
{
    "max_password_attempts": 3,
    "allow_guest_mode": true
}
```

- POST for partial update (values not provided treated as unchanged)
  - **POST /users/:userId**

```
{
    "max_password_attempts": 3,
    "allow_guest_mode": true
}
```

Partial Updates

- Tunneling PATCH through POST POST /users/:userId

```
X-HTTP-Override-Header: PATCH

{

    "max_password_attempts": 2

}
```

- Directly use PATCH
  - **PATCH /users/:userId**

if-match:”473687dgsd”

```
[

    {“op”:”replace”,”path”: ”max_password_attempts”,”value”: 3},

    {“op”:”replace”,”path”: ”allow_guest_mode”,”value”: true}

]
```

Partial Updates

- Tunneling PATCH through POST POST /users/:userId

```
X-HTTP-Override-Header: PATCH

{

    "max_password_attempts": 2

}
```

- Directly use PATCH
  - **PATCH /users/:userId**

if-match:”473687dgsd”

```
[
    {“op”:”replace”,”path”: ”max_password_attempts”,”value”: 3},
    {“op”:”replace”,”path”: ”allow_guest_mode”,”value”: true}
]
```

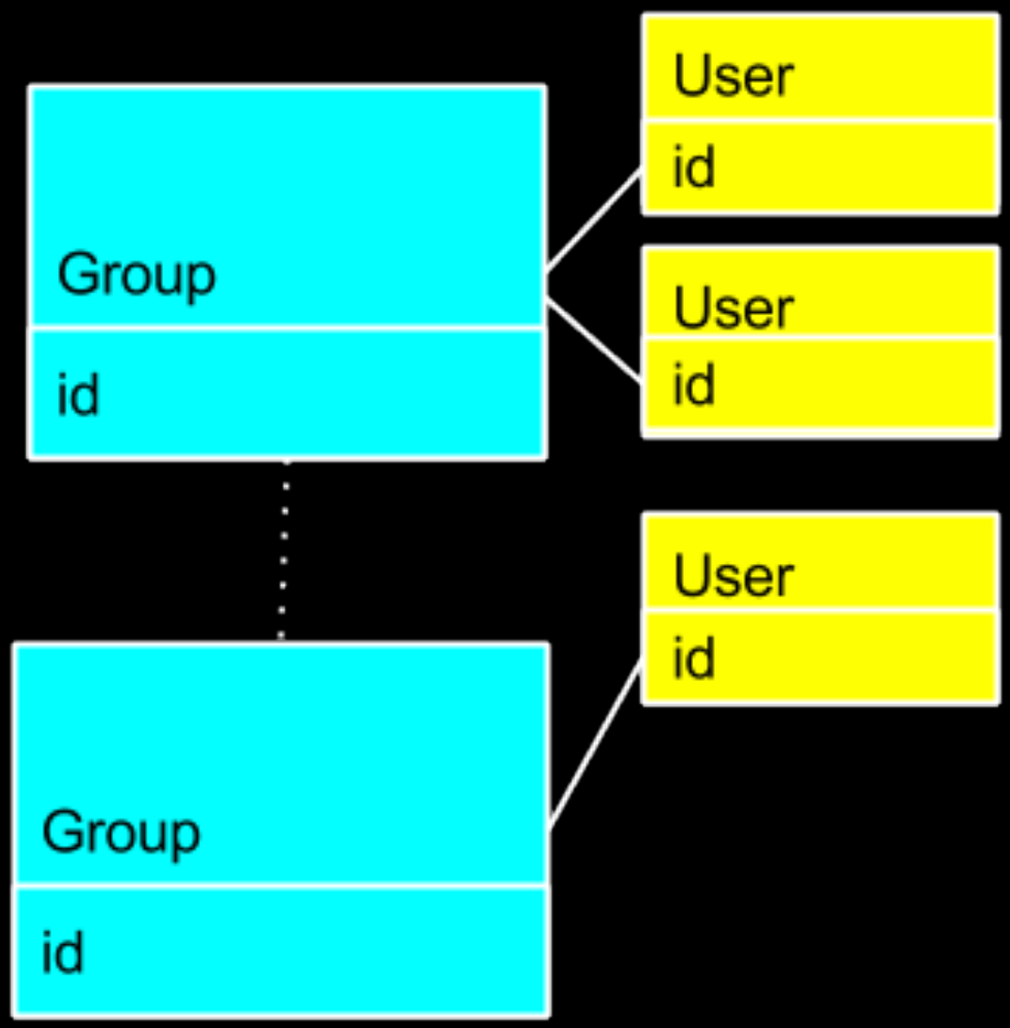
Resource Relationships

- Containment
  - Resources contain sub-resources
- Resource Discoverability
  - Resources hyperlink to related resources
- HATEOAS - Hypertext As The Engine Of Application State

Resource Relationships

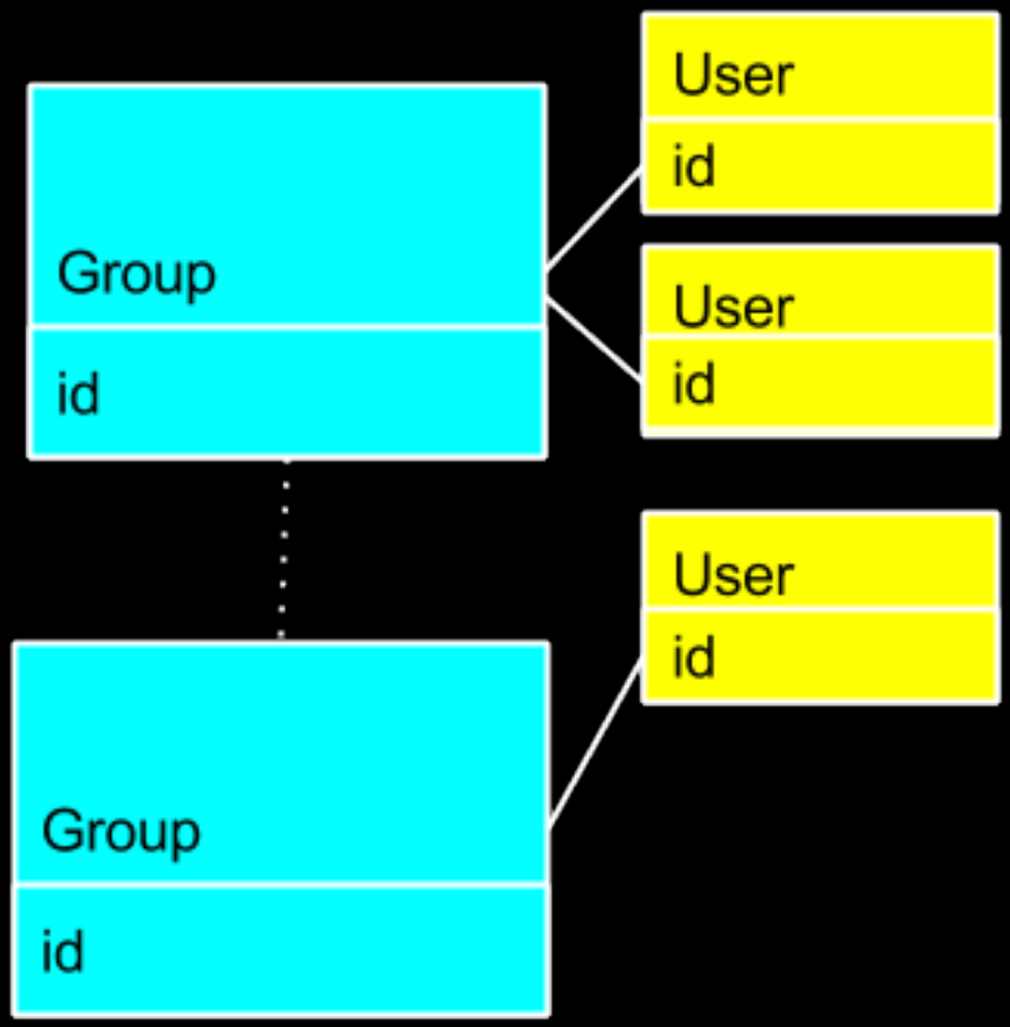
- Containment
  - Resources contain sub-resources
- Resource Discoverability
  - Resources hyperlink to related resources
- HATEOAS - Hypertext As The Engine Of Application State

Self-Describing Relationships



The JSON structure used is not intended to be compliant with any API specification. It is intended to demonstrate the idea of using links to describe the resource hierarchy

Self-Describing Relationships

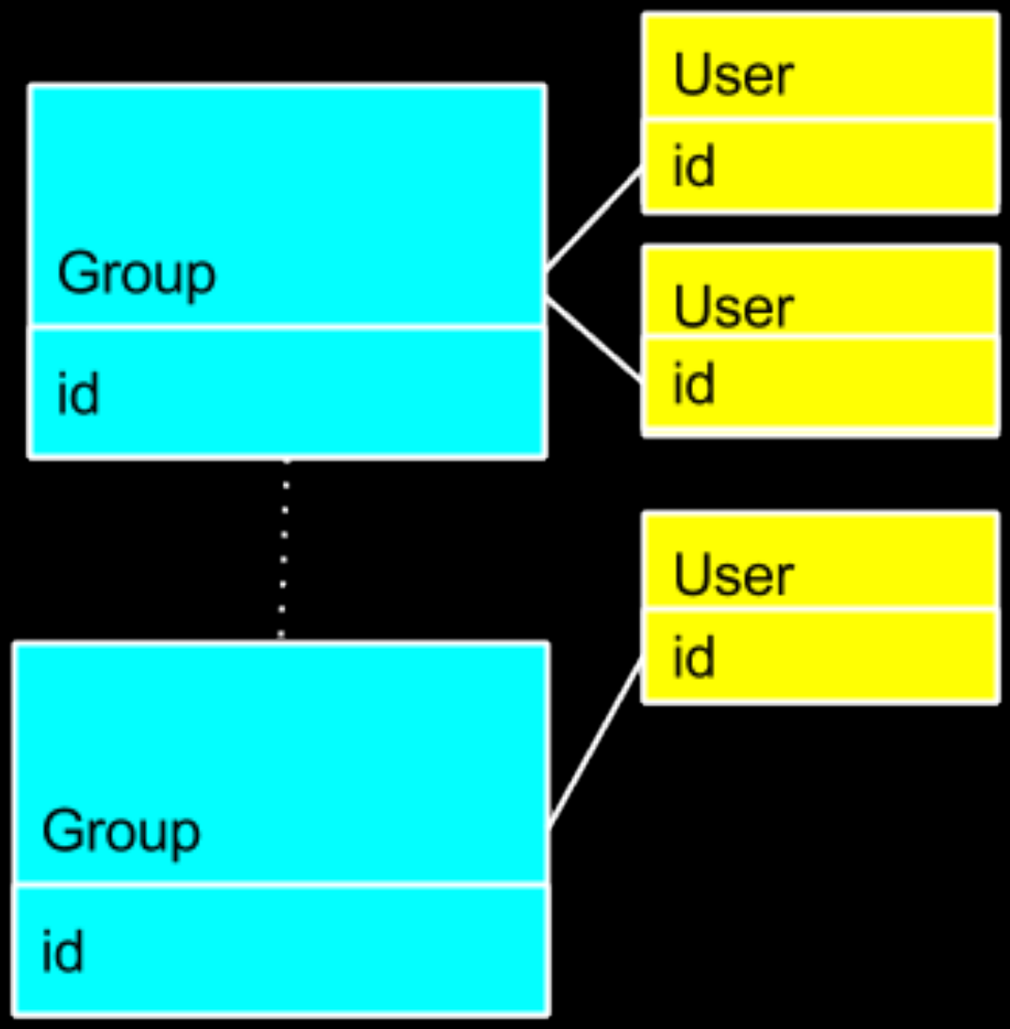



```
POST /groups
{
    "name": “Geeks"
}
```

```
201 Created
{
    "link": {
        "self": "/groups/3232"
    }
}
```

The JSON structure used is not intended to be compliant with any API specification. It is intended to demonstrate the idea of using links to describe the resource hierarchy

Self-Describing Relationships

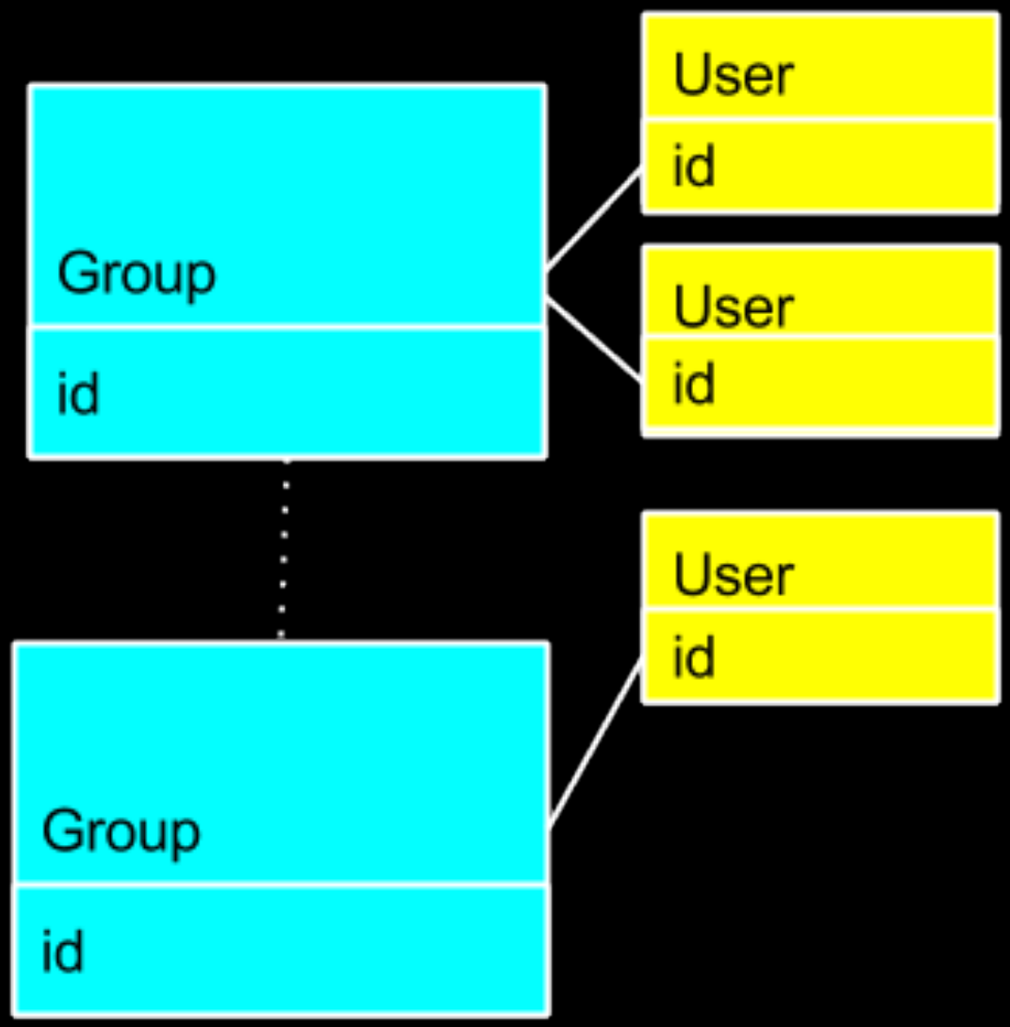


GET /groups/3232

```
200 OK
{
    "id": 3232,
    "name": "Geeks",
    “link”: {
        “self”: “/groups/3232”
    },
    "relationships": {
        "users": {
            "link": {
                "self": “/groups/3232/
users”
            }
        }
```

The JSON structure used is not intended to be compliant with any API specification. It is intended to demonstrate the idea of using links to describe the resource hierarchy

Self-Describing Relationships



POST /groups/3232/users

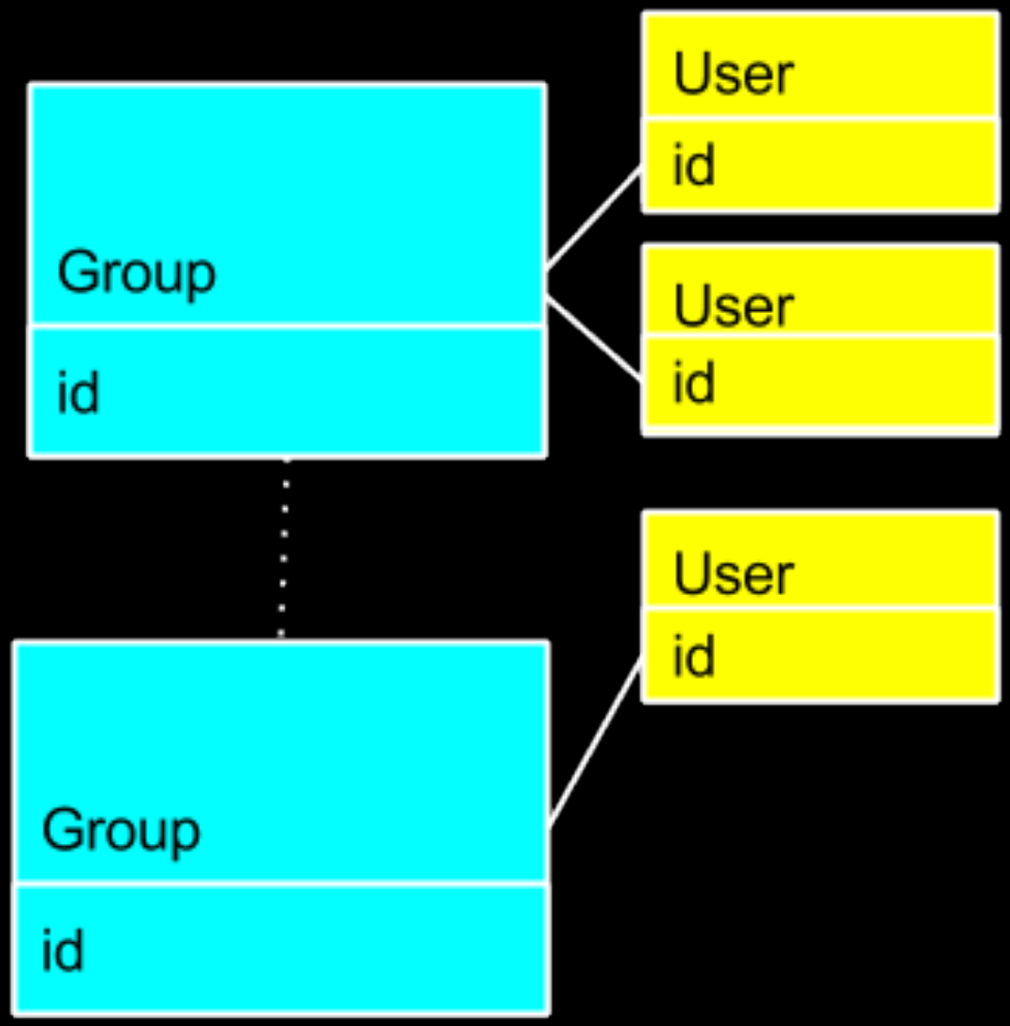
```
{
    "name": “Nicola Tesla"
}
```

201 Created

```
{
    "link": {
        "self": “/groups/3232/users/9890”
    }
}
```

The JSON structure used is not intended to be compliant with any API specification. It is intended to demonstrate the idea of using links to describe the resource hierarchy

Self-Describing Relationships



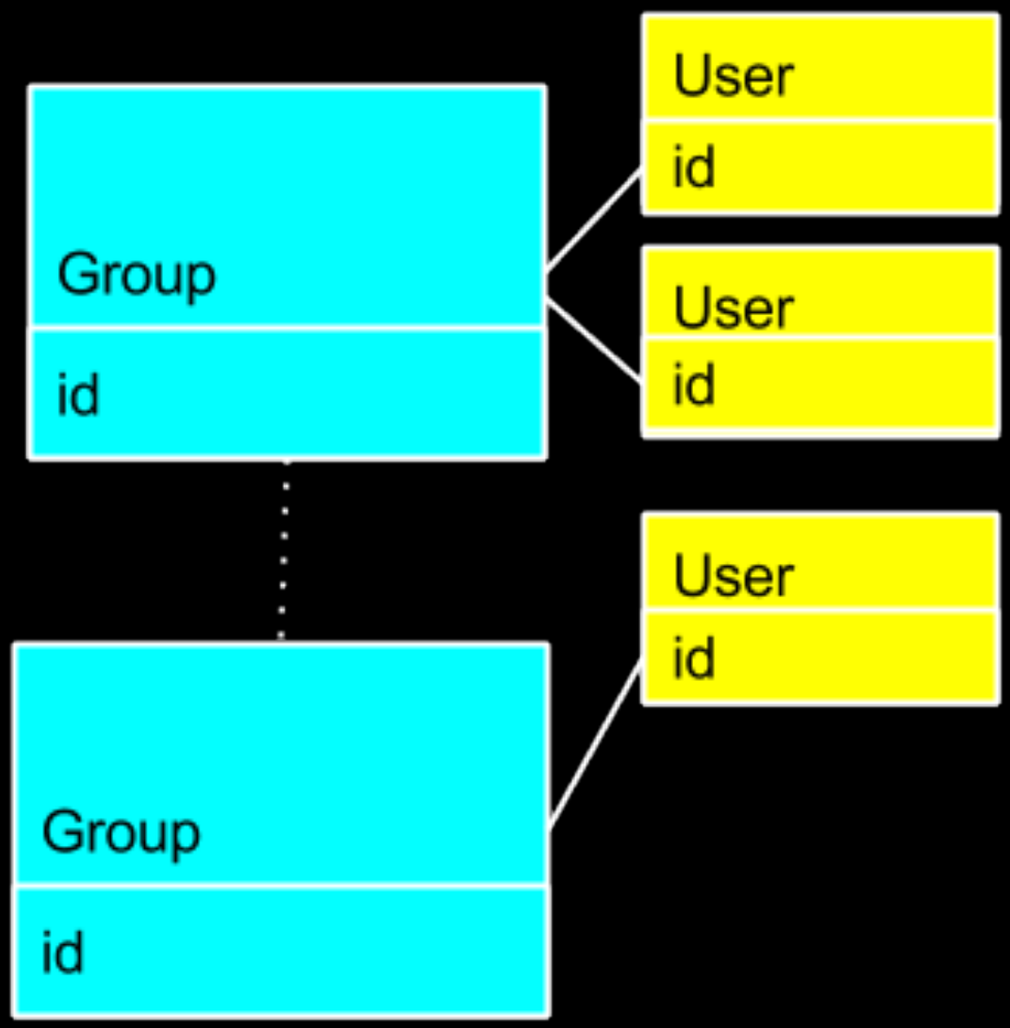
GET /groups/3232/

200 OK

```
{
    "id": 3232,
    "name": “Geeks",
    “link”: {
        “self”: “/groups/3232”
    },
    "relationships": {
        "users": {
            "name": "Nicola Tesla",
            "link": {
                "self": “/groups/3232/users/9890“
            }
        }
```

The JSON structure used is not intended to be compliant with any API specification. It is intended to demonstrate the idea of using links to describe the resource hierarchy

Self-Describing Relationships



The JSON structure used is not intended to be compliant with any API specification. It is intended to demonstrate the idea of using links to describe the resource hierarchy

Resource URIs

“What’s your name & where do you live?”

Naming Conventions

- lowercase or snake_case or camelCase or hyphen-value or PascalCase ?
  - Typically based on platform development languages
- All plurals or all Singular or mixed?
- Pick a convention and Be Consistent

Think Nouns…

Think Nouns… 

Think Nouns…

/add_new_employee

/update_employee_details?id=123

/get_all_employees

/get_all_employees?dept=ABC

/remove_employee?id=123

Think Nouns...

/add_new_employee	POST /employees
/update_employee_details?id=123	PUT /employees/123
/get_all_employees	GET /employees
/get_all_employees?dept=ABC	GET /depts/ABC/employees
/remove_employee?id=123	DELETE /employees/123

Not Always That Obvious…

- All Actions Cannot Be Mapped to CRUD
- Reset a Password
  - POST /users/resetPassword

```
{
    “emailId”: “bob@example.com”
}
```

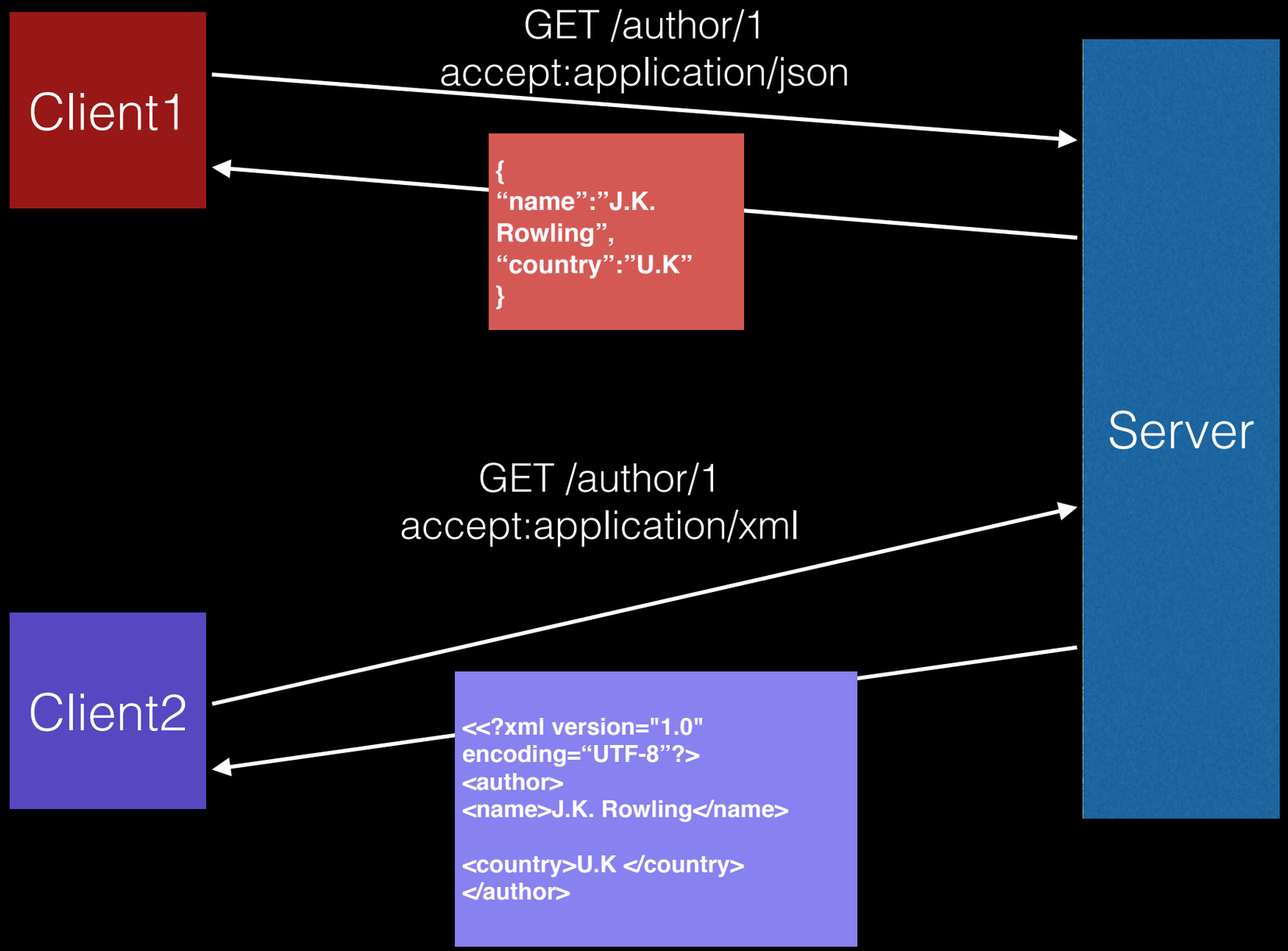
- POST /users/passwordReset

```
{
    “emailId”: “bob@example.com”
}
```

Resource Representation

Content Negotiation

Single Resource, Many Representations…



What Should be the Representation?

- JSON or XML or HTML or XHTML or custom types….
  - Hypermedia support to be truly RESTful
- JSON Popular
  - Built-In Data Types Quite Sufficient
    - Some missing types - Date, URI…
  - Simple
  - Ubiquitous Parsers
- Many JSON Media Type Extensions (JSON-LD, JSON-HAL etc)

What Should be the Representation?

- JSON or XML or HTML or XHTML or custom types….
  - Hypermedia support to be truly RESTful
- JSON Popular
  - Built-In Data Types Quite Sufficient
    - Some missing types - Date, URI…
  - Simple
  - Ubiquitous Parsers
- Many JSON Media Type Extensions (JSON-LD, JSON-HAL etc)

Content Negotiation : Client Requests

- HTTP accept header
  - Accept: application/json
  - Accept: application/xml
  - Accept: application/vnd.api+json
  - Accept:application/hal+json
  - Accept:application/vnd.mycompany.api+json
- Request URI could include extension
  - GET /employees.json
  - GET /employees.xml

Content Negotiation : Client Requests

- HTTP accept header
  - Accept: application/json
  - Accept: application/xml
  - Accept: application/vnd.api+json
  - Accept:application/hal+json
  - Accept:application/vnd.mycompany.api+json
- Request URI could include extension
  - GET /employees.json
  - GET /employees.xml

Content Negotiation: Server Response

- HTTP content-type Header (General , Vendor Specific)
  - Content-Type: application/json
  - Content-Type: application/xml
  - Content-Type: application/vnd.mycompany.myapp.resource+json
  - Content-Type: application/vnd.mycompany.myapp.resource+xml

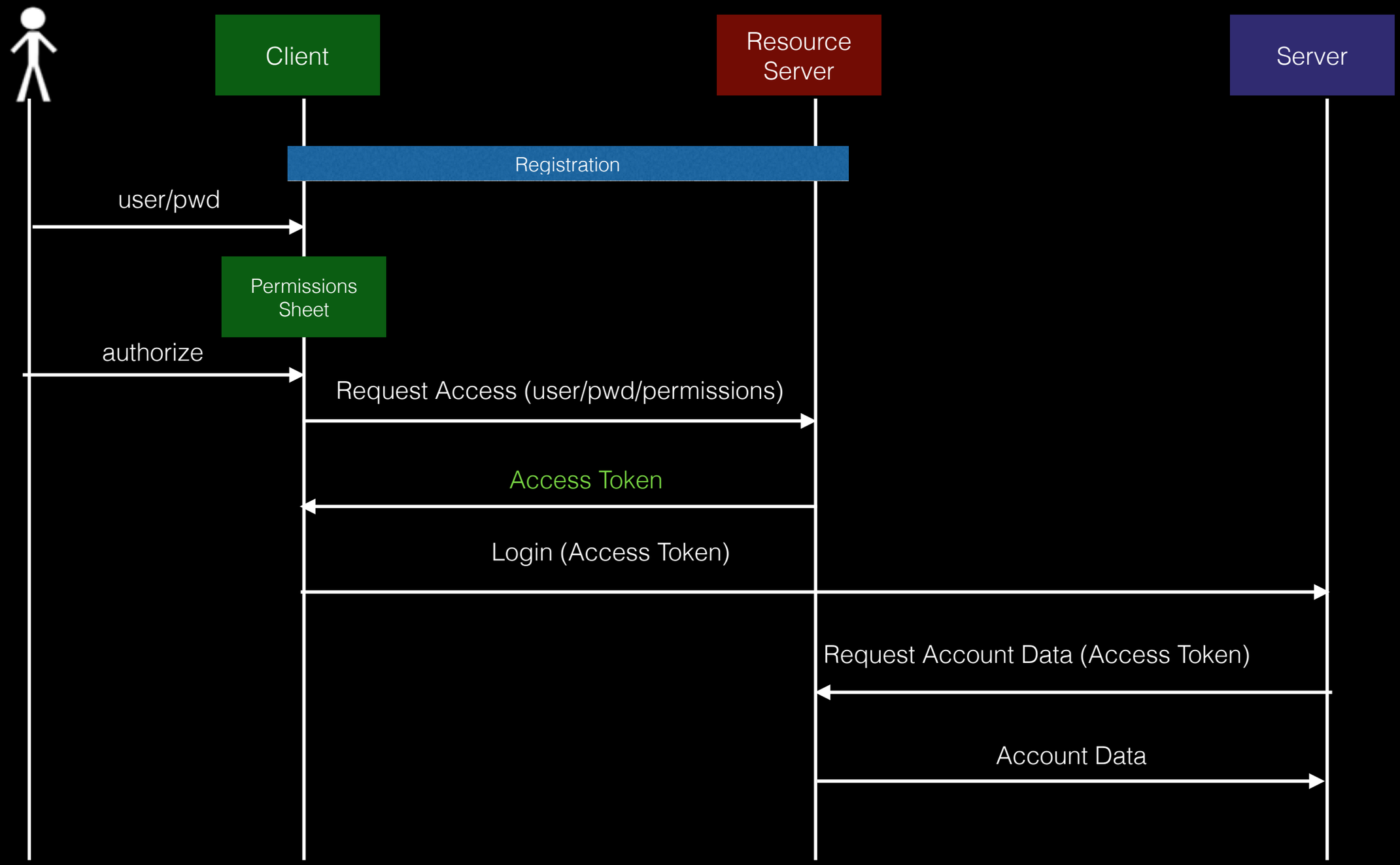
Security

Securing Resource Access

- SSL . (HTTPS). No Excuses.
- Token Based Authentication
  - Exchange username/password for a token
  - Time bound, Scoped
  - Stateless paradigm
  - HTTP Authorization Header

OAuth 2

Temporary Access to subset of resources



JSON Web Token (RFC 7519)

(jwt.io)

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3
MiOiJodHRwczovL2RhaWx5a2FybWEuY29tIiwic3ViI
joibWFpbHRvOnByaXlhLnJhamFnb3BhbEBkYWlseWth
cm1hLmNvbSIsIm5iZiI6MTQ1MTMzNjg4MSwiZXhwIjo
xNDUxMzQwNDgxLCJpYXQiOjE0NTEzMzY4ODEsImp0aS
I6ImlkMTIzNDU2In0.l2-KdHGZOh6PjCVG-
KoE65NU3t9NcxJWwjTJERe0nLM
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

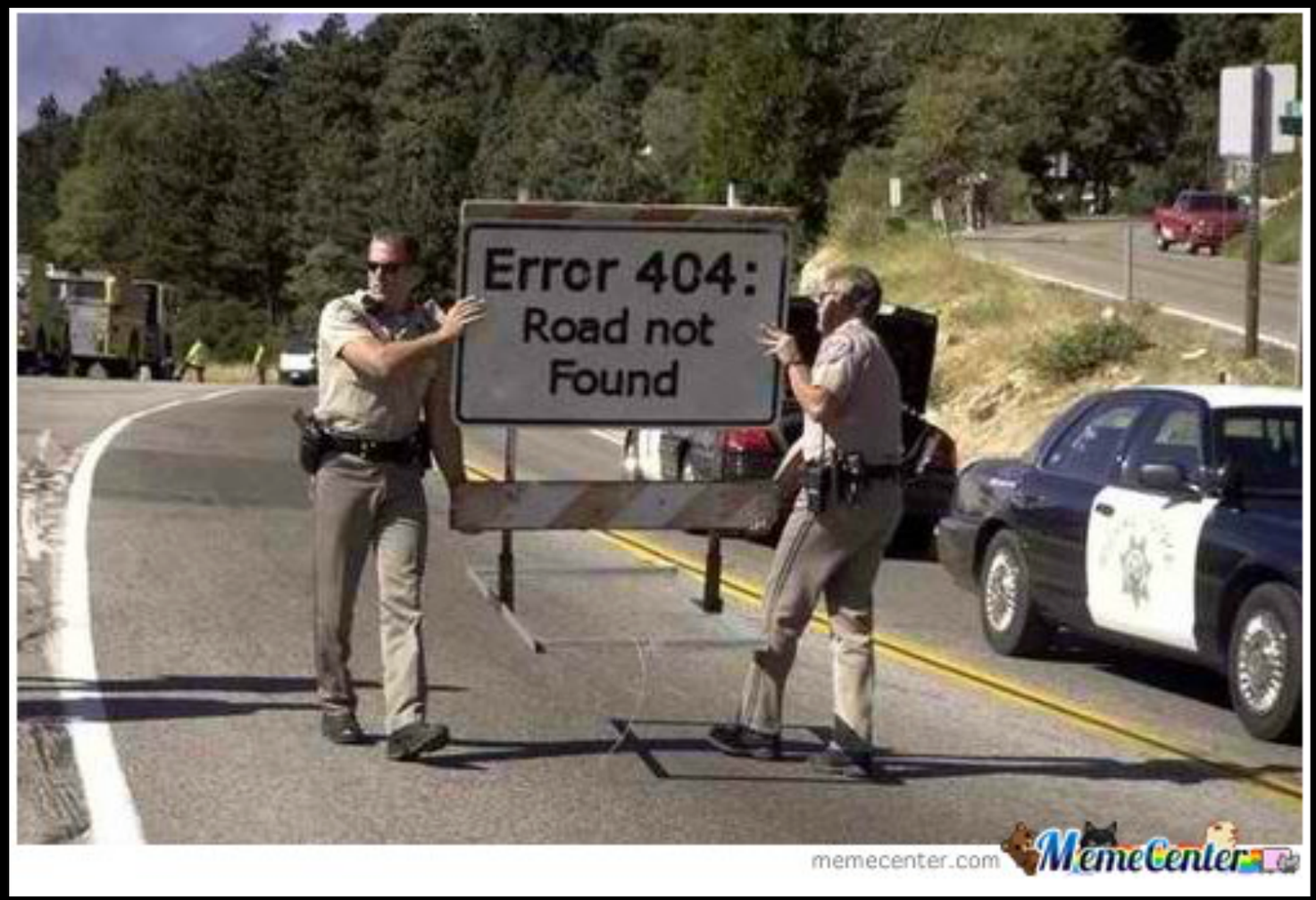
PAYLOAD: DATA

```
{
  "iss": "https://dailykarma.com",
  "sub": "mailto:priya.rajagopal@dailykarma.com",
  "nbf": 1451336881,
  "exp": 1451340481,
  "iat": 1451336881,
  "jti": "id123456"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) secret base64 encoded
```

Versioning



API Versioning

- Include version in the URI

  https://mywebservice/api/V1/resource1

- Specify in Media Type

  Accept: application/
  vnd.company.mywebservice.resource1+json;vers
  ion=2.0

- HATEOAS Could get you away from Versioning

Handling Errors

“Failure is not an option. It comes bundled with the software”

Error Responses

- Use HTTP Status Codes (1:1 Mapping)
  - HTTP/1.1 200 OK
  - HTTP/1.1 401 Unauthorized
  - HTTP/1.1 500 Internal Server Error and so on…
- Application Level Errors
  - Always return HTTP 200 OK
  - Include error object

```
{
  "code": 401,
  "message": "The username\/password was incorrect"
}
```

Error Responses

- Use HTTP Status Codes (1:1 Mapping)
  - HTTP/1.1 200 OK
  - HTTP/1.1 401 Unauthorized
  - HTTP/1.1 500 Internal Server Error and so on…
- Application Level Errors
  - Always return HTTP 200 OK
  - Include error object

```
{
  "code": 401,
  "message": "The username\/password was incorrect"
}
```

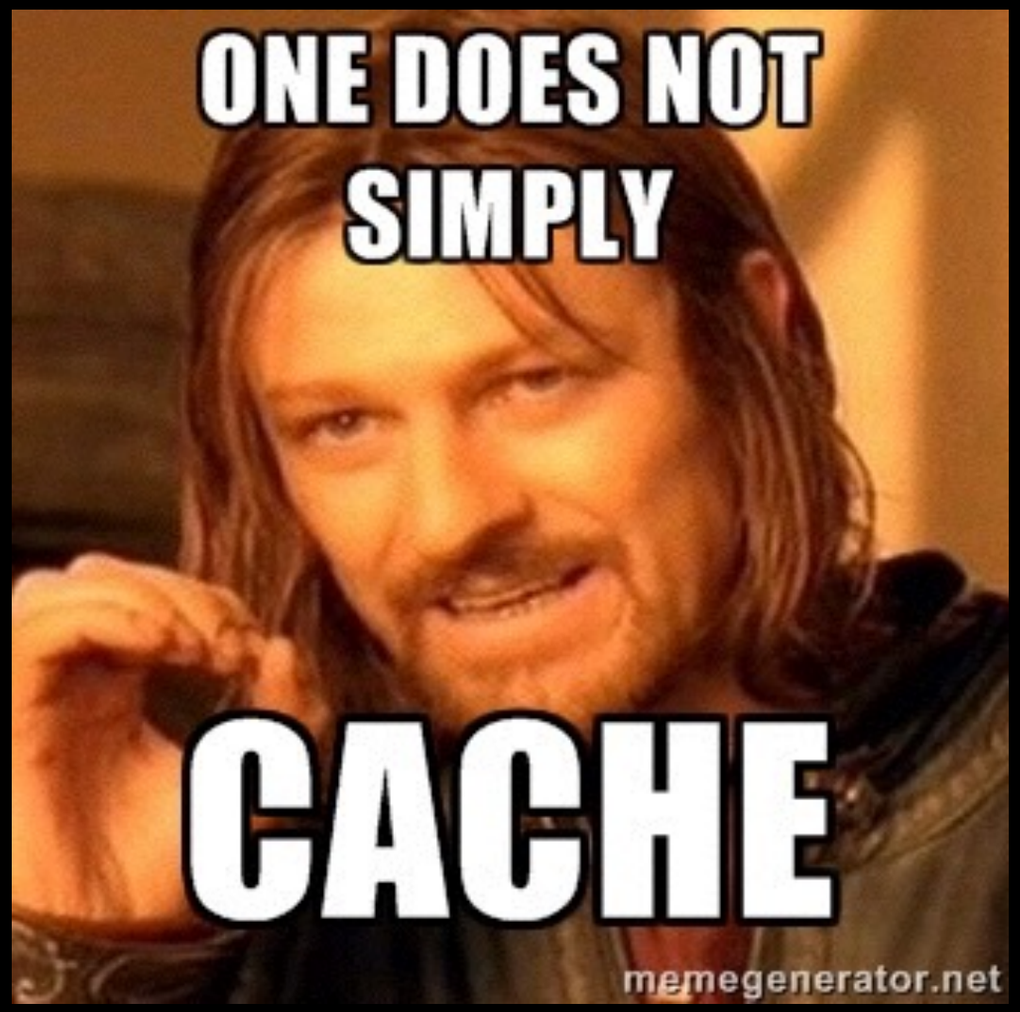
Error Responses

- Hybrid (Piggyback on Subset of HTTP Status Codes)
  - HTTP 2XX - Success
  - HTTP 404 - Not Found
  - HTTP 403 - Unauthorized
  - HTTP 301 - Redirect
  - HTTP 303
  - HTTP 5XX - Server Error
    - Include Application Specific Error Object

{“code”:34422,”message”:”Invalid parameter type”}

Caching

Performance & Scalability



Caching

- Content cached locally or intermediaries
- Scalability / Performance
- Safe Methods
- “Caching is a pain”

Caching

- Content cached locally or intermediaries
- Scalability / Performance
- Safe Methods
- “Caching is a pain”

Caching

- Don’t reinvent the wheel
  - HTTP Cache-Control Headers
- Static Resources
  - Expires/max-age
- Dynamic Resources
  - If-None-Match / eTag
  - If-Modified-Since / Last-Modified

In Summary

- REST is an architectural style
- Resource URIs & Limited Actions on Resources
- Resource Relationships w/ HATEOAS
- Think Nouns
- Many Resource Representations - Use Content Negotiation
- Token Based Authentication
- Cache Resources
- Unambiguous Error Responses

Describing your APIs

- API Blueprint -
  - https://apiblueprint.org (www.apiary.io)
- Swagger
  - http://swagger.io
- RAML - RESTful API Modeling Language.
  - http://raml.org
- Many more ….

No Rest From REST…

- Testing RESTful Web Services, Mark Winteringham,(Indigo Bay,January 6, 2016 1:00 PM)
- Get Some REST- On Practical RESTful API Design, Priya Rajagopal (Orange,January 7, 2016 9:15 AM)
- Consuming REST APIs, for all interpretations of REST, Darrel Miller, (Indigo Bay, January 7, 2016 10:30 AM)
- Making life a bit easier for mobile app developers through better REST API Design, Priya Rajagopal (Mangrove, January 7, 2016 11:45 AM)
- Hypermedia APIs: The rest of REST, Chris Marinos (Salon A, January 7, 2016 1:00 PM)
- Ember Data. The key to good relationships is communication (to your REST server), Brian Gantzler (Portia, Wisteria, January 7, 2016 3:30 PM)

Thank You !

Priya Rajagopal

Twitter: @rajagp