<WA1/>2020

REST API

The glue between clients and servers

Enrico Masala

Fulvio Corno

Luigi De Russis













Roy T. Fielding

Senior Principal Scientist, Adobe Co-founder, Apache HTTP Server Project Director, The Apache Software Foundation Ph.D., Information and Computer Science, UC Irvine

• <u>@fielding</u>; Blog: <u>Untangled</u>

• Email: fielding at (choose **one** of) gbiv.com, adobe.com, apache.org

- Representational State Transfer
- A <u>style of software architecture</u> for distributed systems
- Platform-independent
 - you don't care if the server is Unix, the client is a Mac, or anything else
- Language-independent
 - C# can talk to Java, etc.
- Standards-based
 - runs on top of HTTP
- Can easily be used in the presence of firewalls

What is a Resource?



- A resource can be anything that has identity
 - a document or image
 - a service, e.g., "today's weather in New York"
 - a collection of other resources
 - non-networked objects (e.g., people)
- The resource is the **conceptual mapping** to an entity or set of entities, not necessarily the entity that corresponds to that mapping at any particular point in time!

REST Architecture



Main Principles



- Resource: source of specific information
- Mapping: Resources ⇔ URIs
- Client and server exchange *representations* of the resource
 - the same resource may have different representations
 - e.g., XML, JSON, HTML, RDF, ...

JSON - JavaScript Object Notation < JSON >

- Lightweight Data Interchange Format
 - Subset of JavaScript syntax for object literals
 - Easy for humans to read and write
 - Easy for machines to parse and generate
 - <u>https://www.json.org/</u>
 - ECMA 404 Standard: <u>http://www.ecma-</u> international.org/publications/files/ECMA-ST/ECMA-404.pdf
 - RFC 8259: <u>https://tools.ietf.org/html/rfc8259</u>
- Media type: application/json

/ˈdʒeɪ·sən/

JSON Logical Structure

- Primitive types: string, number, true/false/null
 - Strings MUST use "double" quotes, not 'single'

• Composite type – Array: ordered lists of values

- Composite type Objects: list of key-value pairs
 - Keys are strings (not identifiers)
 - MUST be "quoted"









JSON Example













digit

Using JSON in JavaScript

- JSON.stringify to convert objects into JSON
 - const aString = JSON.stringify(myObj)
 - Works recursively also on nested objects/arrays
 - Excludes function properties (methods) and undefined-valued properties
- JSON.parse to convert JSON back into an object
 - const myObj = JSON.parse(aString)
 - All created objects have the default {} Object prototype
 - Can fix with a *reviver* callback

REST Architecture



Main Types of Resources

Collection resource

- Represents a set (or list) of resources of the same type
- Format: /resource
 - http://api.polito.it/students
 - http://api.polito.it/courses
- Element (Item, Simple) resource
 - Represents a single item, and its properties
 - Has some state and zero or more sub-resources
 - Sub-resources can be simple resources or collection resources
 - Format: /resource/identifier
 - http://api.polito.it/students/s123456
 - http://api.polito.it/courses/01zqp



 $\{REST\}$



Best Practice

- Nouns (not verbs)
- Plural nouns
- Concrete names (not abstract)
 - /courses, not /items



Main Principles



- **Resources** support **Operations** (Actions)
 - Add
 - Delete
 - Update
 - Find
 - Search
 - ...

REST Architecture





Actions use HTTP Methods

{REST}

• GET

- Retrieve the representation of the resource (in the HTTP response body)
- Collection: the list of items
- Element: the properties of the element
- POST
 - Create a new resource (data in the HTTP request body)
 - Use a URI for a Collection
- PUT
 - Update an existing element (data in the HTTP request body)
 - Mainly for elements' properties
- DELETE

REST Architecture



Actions on Resources: Example

Resource	GET	POST	PUT	DELETE
/dogs	List dogs	Create a new dog	Bulk update dogs (<u>avoid</u>)	Delete all dogs (<u>avoid</u>)
/dogs/1234	Show info about the dog with id 1234	ERROR	If exists, update the info about dog #1234	Delete the dog #1234

Relationships



- A given Element may have a (1:1 or 1:N) relationship with other Element(s)
- Represent with: /resource/identifier/resource
- http://api.polito.it/students/s123456/courses (list of courses followed by student s123456)
- http://api.polito.it/courses/01qzp/students (list of students enrolled in course 01qzp)

Applicazioni Web I - Web Applications I - 2019/2020

Representations

- Returned in GET, sent in PUT/POST
- Different formats are possible
- Mainly: XML, JSON
 - But also: SVG, JPEG, TXT, ...
 - In POST: URL-encoding
- Format may be specified in
 - Request headers
 - Accept: application/json
 - URI extension
 - http://api.polito.it/students/s123456.json
 - Request parameter
 - http://api.polito.it/students/s123456?format=json

{REST}

Real World Examples

GitHub Developer	Docs - Blog Forum Versions - 9. Search
REST API v3	Reference Guides Libraries
Overview	* Overview
This describes the resources that make up the official GitHub REST APLV3. If you requests, please contact GitHub Support or GitHub Premium Support. i. Current version	have any problems or Media Types Outh Authorizations API Other Authorization Methods
ii. Schema iii. Autoendication iv. Brannelasia vi. Stock enkoloiti vi. Gandet mono viii. Cliente mono viii. Cliente mono iii. HTTP: vehis iii. HTTP: vehis viii. Brannellia viii. Brannellia viii. Brannellia viii. Brannellia viiii. Schema viiii. Schema viiiii. Schema viiii. Schema viii. Schema	Troubleshooting API Proviews Monitoren
	 Activity Checks
	 ▶ Gists ▶ Git Data
xii. Bate limiting xiii. User agent required xiv. Conditional requests	GitHub Actions GitHub Apps GitHub Apps
xv. Loss organ resource sharing xvi. JSON-P callbacks xvii. Timezones	Interactions Idsues
Current version	▶ Migrations

https://developer.github.com/v3/

🛐 Goog	gle Caler	ndar API			Q. Cerca	Italiano 👻 🦷
Home page	Guide	Riferimento	Esempi	Assistenza		Invia feedbaci
AEST Informace Instructor Standard Ad - Ad - Calcedus - Calcedus - Calcedus - Calcedus - Calcedus - Calcedus - Calcedus - Standard - Standar		Home page API This API represent Resound AcI For AcI	e 3 Produiti 3 G Buile Developer 3 Calendar / Reference is reference is organized by resource type. tations and one or more methods. urce types	M3 > Mitemento 것 것 것 것 것 것 것 Each resource type has one or more data	Contenual Resource types Act Calendaria Calendaria Calendaria Columnals Colu	
Python Ruby [2]			Method	HTTP request	Description	
(100) E			URIs rel	ative to https://www.googleapis.com/calendar/	v3, unless otherwise noted	
			delete	DELETE /calendars/calendarId /acl/ruleId	Deletes an access control rule.	
			get	GET /calendars/ <i>calendarId</i> /acl/ <i>ruleId</i>	Returns an access control rule.	
			insert	POST /calendars/calendarId/acl	Creates an access control rule.	
			list	GET /calendars/ <i>calendarId</i> /acl	Returns the rules in the access control list for the calendar.	
				DATON (aslandans/aslandantd	Underson an account of a distribution of the mothest as an arts	

https://developers.google.com/calendar/v3/reference/

Q. Search all documentation	API reference index	
	AFT reference index	
Basics		
Accounts and users		
	Basics	
Tweets	Busioo	
Direct Messages	Authentication	
Marilia	GET oaut/vauthenticate	
Media	GET oauth/authorize POST oauth/authorize	
Trends	POST cauth/invalidate_token	
Geo	POST cauth/request_token	
deo	POST cauth2roken POST cauth2roken	
Ads		
Metrics	Accounts and users	
Publisher tools & SDKs	Create and manage lists	
	GET Interfact	
Twitter for Websites	GET lists/members	
Labs	GET late/members/see	
	GET lists/ownerships	
Developer utilities	GET lots/show	
API reference index	GET Intr/uctorises GET Intr/uctorises	
	GET lists/subscribers/show	
APT INTERNITOR INDEX	GET Ints/subscriptions	
	POST lists/oreste	
	Polar intervention Polar intervention	
	POIT internet entropy at	
	POST lists/members/destroy	
	 POST lists/members/destroy. all 	
	 POST lists/subscribers/oreste 	
	 POST lists/subscribers/destroy 	

https://developer.twitter.com/en/docs/api-reference-index



https://developers.google.com/youtube/v3/docs

Complex resource search



- Use ?parameter=value for more advanced resource filtering (or search)
 - E.g.,

https://api.twitter.com/1.1/statuses/user_timeline.json?scre
en_name=twitterapi&count=2

Errors



- When errors or exceptions are encountered, use meaningful HTTP Status Codes
 - The Response Body may contain additional information (e.g., informational error messages)

```
{
    "developerMessage" : "Verbose, plain language description of
    the problem for the app developer with hints about how to fix
    it.",
        "userMessage":"Pass this message on to the app user if
    needed.",
        "errorCode" : 12345,
        "more info": "http://dev.teachdogrest.com/errors/12345"
}
```

Authentication

Twitter Streaming API

Authorization: OAuth oauth_consumer_key="xvz1evFS4wEEPTGEFPHBog", ...

Amazon Web Services API

Authorization: AWS AKIAIOSFODNN7EXAMPLE:frJIUNo//yllqDzg=

Google API



Authorization: Bearer 1/fFBGRNJru1FQd44AzqT3Zg

API Design

- How to design a set of APIs for your application?
- Practical guidelines, with applied standard practices
- Suggestion: Google API Design Guide
 - <u>https://cloud.google.com/apis/design/</u>



http://apistylebook.com/design/guidelines/

Atlansian REST API Design Guidelines version	Atlassian REST API Policy	API Design Guide
And a second sec		na (Na (
		No analysis of the second sec
		in addas in addas in a main in a main in addas in a main in addas in

API Design Flow

- 1. Determine what types of **resources** an API provides.
- 2. Determine the **relationships** between resources.
- 3. Decide the resource **name schemes** based on types and relationships.
- 4. Decide the **resource schemas**.
- 5. Attach minimum set of **methods** to resources.

API Service Name	Collection ID	Resource ID	Resource ID	Resource ID
//mail.googleapis.com	/users	/name@example.com	/settings	/customFrom

Example (Gmail API)

- API service: gmail.googleapis.com
- A collection of users: users/*. Each user has the following resources.
 - A collection of messages: users/*/messages/*.
 - A collection of threads: users/*/threads/*.
 - A collection of labels: users/*/labels/*.
 - A collection of change history: users/*/history/*.
 - A resource representing the user profile: users/*/profile.
 - A resource representing user settings: users/*/settings.

Standard Methods

Standard Method	HTTP Mapping	HTTP Request Body	HTTP Response Body
List	GET <collection url=""></collection>	N/A	Resource* list
Get	GET <resource url=""></resource>	N/A	Resource*
Create	POST <collection url=""></collection>	Resource	Resource*
Update	PUT or PATCH <resource url=""></resource>	Resource	Resource*
Delete	DELETE <resource url=""></resource>	N/A	<pre>google.protobuf.Empty**</pre>

Let's read: <u>https://cloud.google.com/apis/design/standard_methods</u>

Guidelines (1/2)

URL Design	
Plural nouns for collections	/dogs
ID for entity	/dogs/1234
Associations	/owners/5678/dogs
HTTP Methods	POST GET PUT DELETE
Bias toward concrete names	/dogs (not animals)
Multiple formats in URL	/dogs.json /dogs.xml
Paginate with limit and offset	?limit=10&offset=0
Query params	?color=red&state=running
Partial selection	?fields=name,state
Use medial capitalization	"createdAt": 1320296464 myObject.createdAt;
Use verbs for non-resource requests	/convert?from=EUR&to=CNY&amount=100
Search	/search?q=happy%2Blabrador
DNS	api.foo.com developers.foo.com

Guidelines (2/2)

Versioning	
Include version in URL	/v1/dogs
Keep one previous version long enough for developers to migrate	/v1/dogs /v2/dogs

Errors	
Status Codes	200 201 304 400 401 403 404 500
Verbose messages	{"msg": "verbose, plain language hints"}

Client Considerations	
Client does not support HTTP status codes	?suppress_response_codes=true
Client does not support HTTP methods	GET /dogs?method=post GET /dogs GET /dogs?method=put GET /dogs?method=delete
Complement API with SDK and code libraries	1. JavaScript 2 3

License

- These slides are distributed under a Creative Commons license "Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)"
- You are free to:
 - Share copy and redistribute the material in any medium or format
 - Adapt remix, transform, and build upon the material
 - The licensor cannot revoke these freedoms as long as you follow the license terms.
- Under the following terms:
 - Attribution You must give <u>appropriate credit</u>, provide a link to the license, and <u>indicate if changes were</u> <u>made</u>. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - NonCommercial You may not use the material for <u>commercial purposes</u>.
 - ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the <u>same license</u> as the original.
 - No additional restrictions You may not apply legal terms or <u>technological measures</u> that legally restrict others from doing anything the license permits.
- <u>https://creativecommons.org/licenses/by-nc-sa/4.0/</u>

