CPAO Sharing session

Rest API
Best practices

CPAO Sharing session - REST API
What is it? Why to use it?

REST (Representational State Transfer) or RESTful

What is it?

- Proposed by Roy Fielding in 2000 for architecturing web services
- It is a Software architecture style and not a protocol or a standard (like SOAP)
- Return the information via HTTP in one of several formats (JSON, XML, HTML, …)
- Stateless client-server communication, meaning no client information is stored between get requests and each request is separated and unconnected

Why to use it?

- Client server communication (frontend to backend, backend to backend, …)
- Standard architecture style in modern web applications
- Easier scalability
- Easy to develop, to test and to use :)

CPAO Sharing session - REST API

Why using best practices ?

- REST stands for representational state transfer. It is a set of constraints that set out how an API should work.
- RESTful refers to an API adhering to those constraints.

A **RESTful** API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to **GET**, **PUT**, **POST** and **DELETE** data types, which refers to the **reading**, **updating**, **creating** and **deleting** of operations concerning resources.

CPAO Sharing session - REST API

Best practices #1

Use HTTP verbs

HTTP Verb	**CRUD action**	**Description**
GET	Read	Retrieve an information
POST	Create	Create an information
PUT	Update	Update an information
DELETE	Delete	Delete an information
PATCH	*Update*	*Update a part of an entity*

CPAO Sharing session - REST API

Best practices #1

Use HTTP verbs in Airbus context

- Theoretically, we should use all the HTTP verbs for the APIs
- In practice it is different because some HTTP verbs are blocked by Airbus

HTTP Verb authorized	CRUD action	Description
GET	Read	Retrieve an information
POST	Create	Create an information
POST	Update	Update an information
DELETE	Delete	Delete an information

PUT and PATCH are not allowed in Airbus context

CPAO Sharing session - REST API

Best practices #2

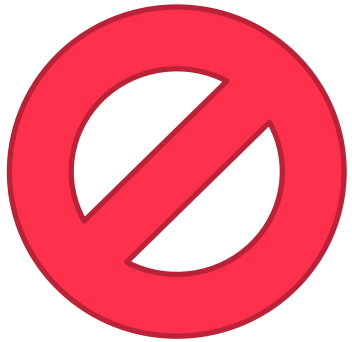
Use nouns instead of verbs

- It is better to use a noun for an API instead of a verb. To explain what to do (read, create,...), we will use the HTTP verb :)
- Use the plural for the nouns

HTTP Verb	URI	Description
GET	/ac-conf/fleets	Retrieve the list of the fleets
POST	/ac-conf/fleets	Create a fleet. The fleet to create will be in the body of the request in JSON format
PUT (POST Airbus)	/ac-conf/fleets/{id}	Update the fleet associated to the id
DELETE	/ac-conf/fleets/{id}	Delete the fleet with id {id}
GET	/ac-conf/fleets/check?thirdId=X&rankId=Y	Check if the newly created fleet does not already exist for the Third and Rank couple
GET	/ac-conf/fleets/series/{acProgram}	Get the fleet series by AC program

CPAO Sharing session - REST API

Best practices #2



Practice to avoid

- Here are some examples of wrong API naming

HTTP Verb	URI	Description
GET	/ac-conf/getAllFleets	Retrieve the list of the fleets
POST	/ac-conf/fleet/create	Create a fleet. The fleet to create will be in the body of the request in JSON format
POST	/ac-conf/fleet/modify	Update the fleet associated

CPAO Sharing session - REST API

Best practices #3

Use sub-resource to identify a unique item or a relation

- Here are some examples of sub-resource usage

HTTP Verb	URI	Description
GET	/ac-conf/fleets/{id}	{id} is a unique identifier to retrieve a fleet (sub-resource of fleets)
GET	/ac-conf/fleets/programs/{program}/roles/{role}	{program} is the identifier of the program and {role } is the user role: retrieve a list of users (relation) according to the role on the program
GET	/ac-conf/fleets/programs/{program}/cms	{program} is the identifier of the program : retrieve a list of CM (relation) according to the program

Think to cut your main controller in sub-controllers

CPAO Sharing session - REST API

Best practices #4

Correct use of HTTP return code

- It is better to use the correct HTTP return code
- It is not mandatory to use all the codes, just the most important like the list below

HTTP Code	Description and usage
200	The server treated the request with success (after a GET, an update, a deletion, …)
201	The resource has been created (with the location attribute set with the URI to retrieve the information)
400	Invalid request (wrong parameter, wrong format of the request, …)
404	Not found resource (when an information does not exist for the requested identifier)
401	The client must be authenticated
403	The client is authenticated but not authorized to access to the resource
500	Error returned by the server and not by the code (when a NPE occurs or an AOOBE)

CPAO Sharing session - REST API

Best practices #4

- With this list of HTTP code, you have all the necessary codes to return for your application.
- GAFAM also uses these practices with sometimes some specificities

CPAO Sharing session - REST API

Best practices #5

Filter by default : usage of path params and query params

- Develop filter capabilities by default with the resources that may return a large number of elements
- Define default limit on each parameter to avoid huge items to return

A good practice is to use offset and limit parameters for filtering

- offset is the starting point of the cursor
- limit is the number of items to return

Example to return the first 50 items

/ac-conf/fleets?offset=0&limit=50

Reminder: **path param** is for functional parameters (exemple: fleet id)

query param is for other optional parameters (exemple: a limit of results, …)

CPAO Sharing session - REST API

Best practices #6

Document your API !

- Don’t forget to document your APIs

Some tools exist to easily document your APIs. Swagger is one of them (maybe the best) and should be use on all the project that exposes an API !

- https://swagger.io/
- https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api

Which is good with Swagger is that it can generate a website to document the API and to test it ☺️

It is also possible to add some « archunit rules » to verify that all your API are annotated with Swagger annotations to be sure that you don’t forget any documentation !

CPAO Sharing session - REST API

Best practices #7

Control API access

- Put in place security mechanisms for your API
- A user should be able to see/modify/delete only public data and its own data. By developing APIs, we offer the ability to the users to modify a request to ask another informations.

Fail fast

- Try to validate the data on API side ASAP (filtering values, sort values, …)
- Format the error to produce a JSON representation of the error

CPAO Sharing session - REST API

Resources

Some resources about Rest APIs

- The famous Apigee guide https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf
- https://swagger.io/
- https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api
- https://pages.apigee.com/rs/351-WXY-166/images/API-Best-Practices-ebook-2016-12.pdf

CPAO Sharing session - REST API

Thank you ☺️

People matter, results count.

About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2016 global revenues of EUR 12.5 billion.

Learn more about us at

www.capgemini.com