1. What is Restful API

* Based on REST architectural style
* Uses Http Request to access and use data
* Uses HTTP protocol
* Has Metadata such as status code and content-type
* Used when Requirement is use CURD operations

1. What is a Non-Restful API?

Which does not follow REST architecture

1. Characteristics of a RESTful API?

The RESTful API must follow REST principles. These are

* Stateless,
* Cacheable,
* Client-server,
* uniform Identifier,
* layered system,
* Code on demand

1. What is the use of designing an API?

Mulesoft follows design first approach in which APIs are designed first and will be tested and published to Exchange for review. With this approach the developers will get quick feedback from the client and they can identify bugs also at the beginning stage itself

1. Is it mandatory to design an API? Explain.

Yes, with designing API, developers will get feedback from their client and also can detect bugs at the early stage

1. What ways can we design a RESTful API?

RESTful API can be created by using

RAML

OAS

1. What is RAML?

* RAML (RESTful API Modelling Language)
* Provides all information necessary to describe an API
* Based on Yaml
* Supports entire API life cycle
* Lacks strong documentation and tutorials outside of specification

1. What is OAS?

* OAS stands for Open API Specification.
* It was named as Swagger.
* It was designed to solve the need for keeping the API design and documentation in sync.
* It is a type of framework that was designed to describe, produce, visualize, and consume RESTful web services.
* Can be written in JSON or YAML.
* The recommend language is YAML, because it is easier to read and write.

1. What is APIARY?

Apiary allows us to first create the design of an API and then implement it. Apiary helps in creating a mock API framework and also generate its documentation fairly quickly.

1. What are the differences between RAML vs OAS vs APIARY?

|  |  |  |  |
| --- | --- | --- | --- |
|  | **RAML** | **OAS** | **APIARY** |
| **Goals** | Provides all the necessary information to describe RESTful API | Keep documentation, client libraries and source code in sync | It has an online interface and integrates with API Blueprint, a high-level API description language that sits atop Markdown |
| **Current version** | RAML Specification 1.0 | Open API Specification 3.0.1 | Apiary Interactive Documentation v4 |
| **Supporters** | MuleSoft, Agular, PayPal, Cisco,VMware | Google IBM, Microsoft, Atlassian | Oracle, github |
| **Code Language** | Node.js, .NET, Python, Mule, IOT | Node.js, .NET, Python, Mule, IOT, PHP, Ruby, Swift, Java, JavaScript |  |
| **Document Format** | YAML | JSON, YAML | API blueprint , JSON, YAML |
| **Editors** | API WorkBench | Swagger Tools | API Editor |

1. When should you use each of these languages?

* If the requirement is to use CURD operation, REST Api is the recommended option
* If the requirement is to keep API design and document in synch, the recommended option is to be OAS

1. Example of RAML and OAS?

Raml Example



OAS

openapi: 3.0.0

info:

version: 1.0.0

title: Simple API

description: A simple API to illustrate OpenAPI concepts

servers:

- url: https://example.io/v1

components:

securitySchemes:

BasicAuth:

type: http

scheme: basic

security:

- BasicAuth: []

paths:

/artists:

get:

description: Returns a list of artists

# ----- Added lines ----------------------------------------

parameters:

- name: limit

in: query

description: Limits the number of items on a page

schema:

type: integer

- name: offset

in: query

description: Specifies the page number of the artists to be displayed

schema:

type: integer

# ---- /Added lines ----------------------------------------

responses:

'200':

description: Successfully returned a list of artists

content:

application/json:

schema:

type: array

items:

type: object

required:

- username

properties:

artist\_name:

type: string

artist\_genre:

type: string

albums\_recorded:

type: integer

username:

type: string

'400':

description: Invalid request

content:

application/json:

schema:

type: object

properties:

message:

type: string

1. What ways can you design a Non-RESTFul API?

* RPC
* SOAP
* GraphQL
* Async API

1. Explain GraphQL API with example?

* Developed by Facebook
* Uses Schema Definition language
* exposes a single endpoint and respond with precise data
* Can fetch user specific data with single API call
* Mainly used for high performance
* GraphQL is a query language for APIs.
* GraphQL enables declarative data fetching where a client can specify exactly what data it needs from an API.
* Instead of multiple endpoints that return fixed data structures, a GraphQL server only exposes a single endpoint and responds with precisely the data a client asked for

1. Explain RPC with example?

* RPC – Remote Procedure Call enables “Call a function on another server.”
* Executing a block of code on another server
* Instead of resources in RAML, RPC url will have function name
* Payloads will be of light weight
* Mostly uses Http methods GET and POST
* Mainly used when requirement is action based or command based
* An API is built by defining public methods; then, the methods are called with arguments.
* Example to “send a message” the syntax is as below:



1. Explain SOAP with an example?

* SOAP is the Simple Object Access Protocol. It is an open-standard, XML-based messaging protocol for accessing web services over HTTP.
* SOAP uses an XML data format to declare its request and response messages, relying on XML Schema and other technologies to enforce the structure of its payloads.
* It uses Web Service Description Language (WSDL) wot write API.
* This API description language defines the endpoints and describes all processes that can be performed.
* It defines the operations available and what input/output fields to expect

1. Explain Async API with example?

Async API is based on an event driven architecture. It uses a method by which an API can be represented using a commonly defined and understood language. It support YAML and JSON language to write API



1. What are the constraints of REST?

Stateless,

Cacheable,

Client-server,

uniform Identifier,

layered system,

Code on demand

1. What is connectedness, stateless and unique resources in REST means?

* **stateless**

Implementing stateless requests means the communication between consumer and service is initiated by the request, and the request contains all the information necessary for the server to respond.

Session state is kept entirely on the client.

Statelessness makes a service more reliable and easier to scale.

Statelessness promotes session independence.

Because of session independence, any server can process the client request. This makes load balancing possible.

The trade-off of statelessness is that it increases network bandwidth since every information needs to be passed in every request.

* + **Connectedness**

The client and server know about each other only during the session. So client need to send all the required information so that server can response.

Since client and server are loosely couples they are independent and can be updated independent of one another

* + **unique resources**

By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behaviour of components. REST is defined by four interface constraints:

1. identification of resources;
2. manipulation of resources through representations;
3. self-descriptive messages; and,
4. hypermedia as the engine of application state (HATEOAS).
5. What is hypermedia?

Hypermedia is an extension to what is known as hypertext, or the ability to open new Web pages by clicking text links on a Web browser. Hypermedia extends upon this by allowing the user to click images, movies, graphics and other media apart from text to create a nonlinear network of information

1. Why do you want to use hypermedia?

Hypermedia extends upon this by allowing the user to click images, movies, graphics and other media apart from text to create a nonlinear network of information

1. What is Richardson Maturity Model? Explain?

Richardson Maturity Model is a way to grade your API according to the constraints of REST

The better your API adheres to these constraints, the higher its score is

The Richardson Maturity Model knows 4 levels (0-3), where level 3 designates a truly RESTful API

1. Explain how to use a visual editor in the design centre.

Visual editor is an option available in Design Centre to create API specification

With visual editor non-technical person can also design API

1. What are API Fragments?

An API fragment is a portion of an API specification, which is why understanding it starts at the API specification level. Instead of starting every project from scratch, you can reuse fragments and APIs to accelerate project delivery

1. Why do you need Fragments?

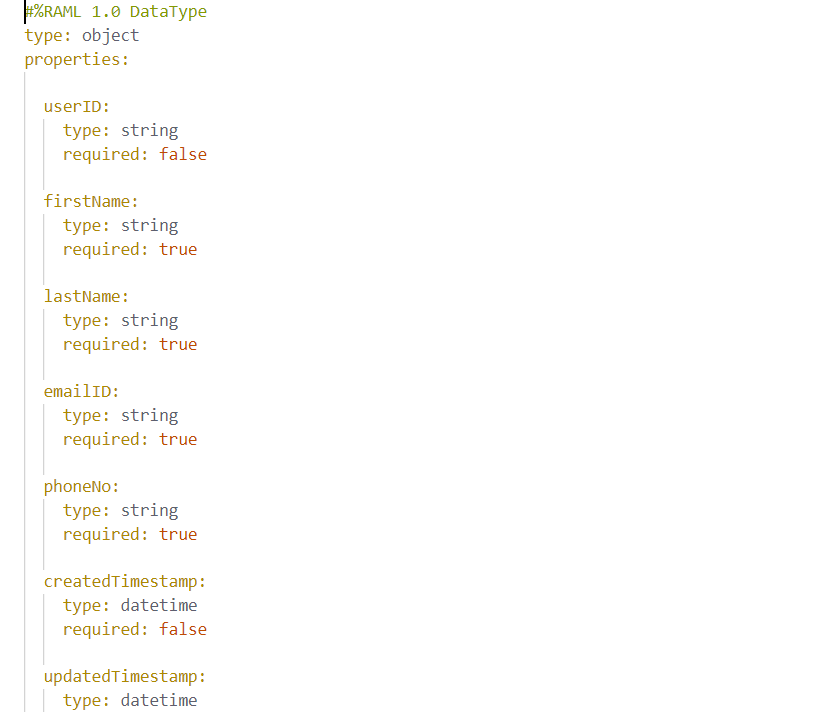
An API fragment is a portion of an API specification, which is why understanding it starts at the API specification level. Instead of starting every project from scratch, you can reuse fragments and APIs to accelerate project delivery

Used for Reusability

1. Explain data type with example?

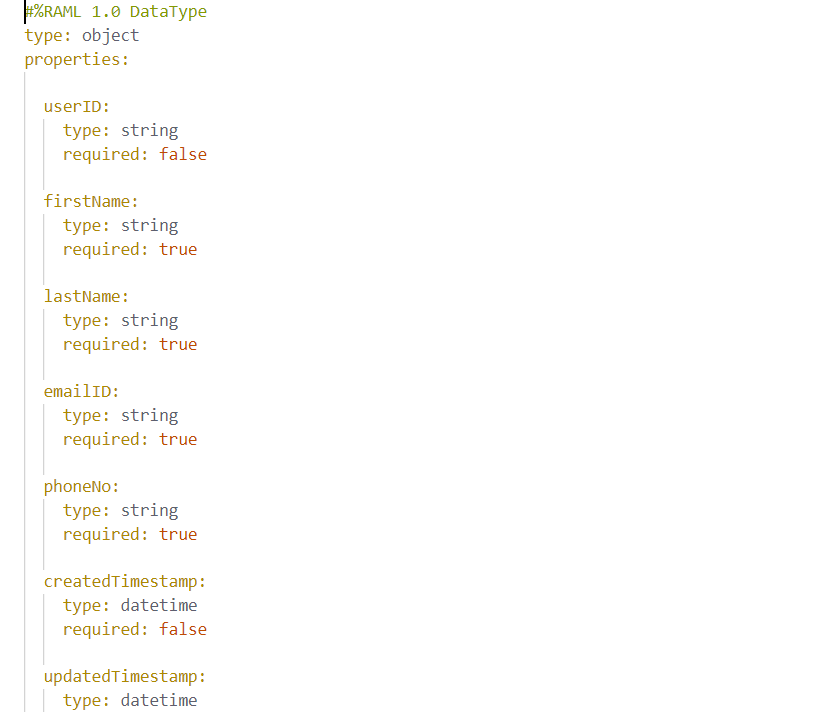
Data Types, a concise and versatile way to describe and validate data inside your API

definition.



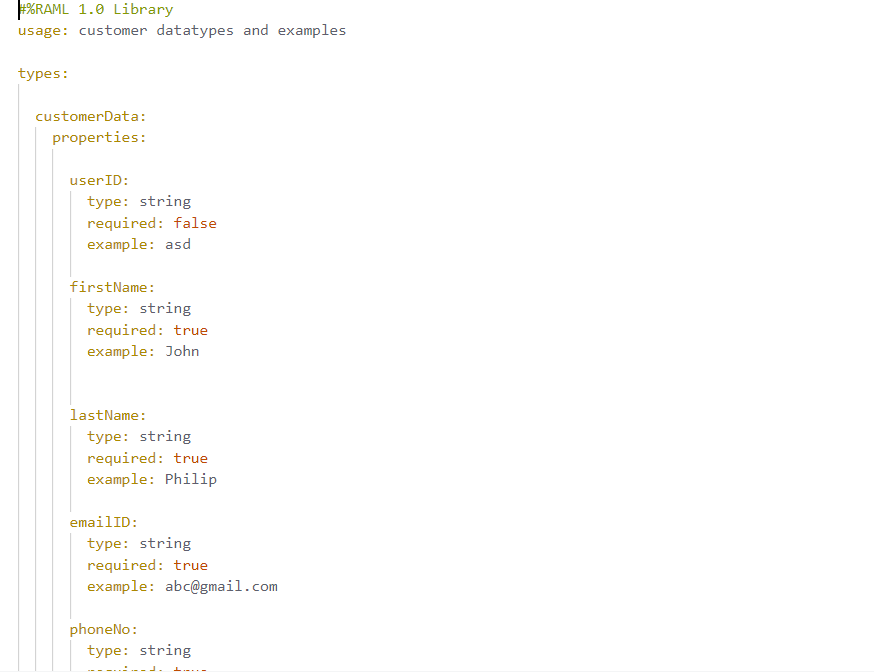
1. Explain Traits with examples?

* Trait is a type of API fragment
* Like a function
* Used to define common set of codes for all Http methods
* Is keyword is used



1. Explain Library with example?

* Type of API fragment
* Provide an ability to include data-types, resource Types, traits in a namespace environment
* When we want to define more than one datatype in a single file so that easy data retrieval and editing can be done. In such scenarios we can go for Libraries.



1. Explain Examples with examples?

Example fragment is used to define the example response of some operations we are doing in the RAML spec.

Example fragment is used to define the NamedExample

1. Difference between an example and examples?

Example is for a single example and examples for a set of examples(array)

1. What is data-modelling?

Data modelling is the process of creating a data model for the data to be stored in a database. This data model is a conceptual representation of Data objects, the associations between different data objects, and the rules.

1. What are the different approaches to create data-models?

* Canonical data model
* mirror a backend system
* bounded context

1. What is a canonical data model?

* Aims organization to create a common definition of entire data units
* Create a canonical data model which is understood by entire organization
* It is easy to maintain
* Allows for smoother integration between systems
* It is time consuming to create a common model is that understandable to all systems

1. What is Enterprise wide data-model?

An Enterprise Data Model is an integrated view of the data produced and consumed across an entire organization. It incorporates an appropriate industry perspective. The model unites, formalizes and represents the things important to an organization, as well as the rules governing them.

1. What are the advantages and disadvantages of an enterprise wide data-model?

Advantages

* Standardize concepts across applications.
* Standardize definitions across applications.
* Add to organizational memory.
* Drive applications towards greater uniformity.
* Seed data models for new applications.
* Reach beyond the current data to the anticipated future data.
* Provide a grand data model as a basis for application integration and data exchange.
* Support data warehouse efforts.
* Enable better IT purchase decisions and suggest how to interface vendor products more effectively.

Disadvantages

* Even smaller change made in structure require modification in the entire application.
* To develop Data model one should know physical data stored characteristics.
* This is a navigational system produces complex application development, management. Thus, it requires a knowledge of the biographical truth.

1. What is a bounded context data model?

* Based on Domain Driven Design
* There will be large domain which is using a common language, it is divided into sub domains based on context
* Domains will be independent of each other

1. What is mirroring a data-model in a backend system?

* Mirroring is creating an exact copy of file/folder
* When an update in source, it will get updated in mirrored backup