Module 1: Architecture

1. What is an Enterprise?

Enterprise is a any business or organization that provides service and aims to make a profit.

1. What is Architecture?

It  a set of rules stating how **computer** software and hardware are joined together and interact to make a **computer** work. It describes the structure and behavior of the system

1. What is Enterprise Architecture?

### Enterprise architecture is the process by which organizations standardize and organize IT infrastructure to aligns with business goals. These strategies support digital transformation, IT growth and the modernization of IT as a department.

1. Why do you need a EA?

### These strategies support digital transformation, IT growth and the modernization of IT as a department.

1. Who is an Enterprise Architect?

An enterprise architect is responsible for the upkeep and maintenance of an organization’s IT networks and services. As an enterprise architect, you will be responsible for overseeing, improving and upgrading enterprise services, software and hardware. You will also need to stay on top of the latest trends and technologies and keep an eye out for any software, services or hardware that might improve business processes.

1. What is integration architecture?

It is a project specific architecture. It can be APIs project, ETL project, file transfer, event driven architecture

1. Difference between EA and Integration architecture?
2. What is a framework?
3. How are frameworks useful for EA?
4. What is TOGAF?

The TOGAF standard is an architecture framework. It provides the methods and tools for assisting in the acceptance, production, use, and maintenance of an Enterprise Architecture. It is based on an iterative process model supported by best practices and a re-usable set of existing architecture assets.

1. What is a Zachman?

Zachman Framework is a two dimensional classification scheme for descriptive representations of an Enterprise that is structured as a matrix containing 36 cells, each of them focusing on one dimension or perspective of the enterprise. Rows are often presented as different viewpoints involved in the systems development process, while columns represent different perspectives of the stakeholders involved in the organization.

The rows of Zachman Framework focus on describing the enterprise from six viewpoint perspectives of the stakeholders. These six perspectives are based on English language interrogatives 'what', 'where', 'who', 'when', 'why', and 'how' (known as W5H).

1. What is 4+1 View point?

4+1 is a view model used for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views”. The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, system engineers, and project managers. The four views of the model are logical, development, process and physical view. In addition, selected use cases or scenarios are used to illustrate the architecture serving as the 'plus one' view. Hence, the model contains 4+1 views

1. When to use TOGAF, Zachman and 4+1 view point?
2. Real time use-case of using TOGAF framework?
3. Ream time use-case of using Zachman framework?
4. Real time use-case if using 4+1 View point?
5. When will you use API Led connectivity?

When we want a faster delivery experience for it by reusing assets that are already developed

1. What is logical view, development view, process view, physical view?
   1. Logical view

* End user perspective.
* The logical architecture primarily supports the functional requirements
* It defines the services provides by the system to users
* Examples: Class Diagram, Object Diagram
  1. Development View:
* Developers perspective
* The development architecture focus on the actual software module organization on the software development environment.
* It helps us to understand how code modules work together and also understand the dependencies between modules
* Example: Component diagram, Package diagram
  1. Process View :
* System integrators perspective.
* It is closer to the working of an actual software.
* Process view consider non-functional requirements such as performance and availability
* The purpose of process view is to capture the sequence, timing and flow of inter-process information exchange
* Example: Sequence Diagram, Activity Diagram, State Diagram, Communication Diagram
  1. Physical view:
* Infrastructure perspective.
* Where application will be deployed, decide frame-work
* Physical architecture considers the non-functional requirements of the system such as availability, reliability, performance and scalability.
* Software executes on network of computers or processing nodes. Physical view illustrate which node will host which process from the process view.
* Example: Deployment diagram, Network topology

1. Explain these views with an example?
2. What tools do you use to create each of the views in 4 + 1 view points?
3. What do you achieve or what is the purpose of creating these views in 4+1 view model?
4. What is SOA?

Service-Oriented Architecture (SOA) is an architectural approach in which applications make use of services available in the network. In this architecture, services are provided to form applications, through a communication call over the internet.

* SOA allows users to combine a large number of facilities from existing services to form applications.
* SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
* SOA based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

1. What is a micro service?

<https://www.edureka.co/blog/what-is-microservices/>

**Microservices** is an architectural style that structures an application as a collection of small autonomous services, modeled around a **business domain.**

1. Difference between SOA and Microservice?

<https://dzone.com/articles/microservices-vs-soa-whats-the-difference>

**Microservices**

**SOA**

**Architecture**

Designed to host services which can function independently

Designed to share resources across services

**Component sharing**

Typically does not involve component sharing

Frequently involves component sharing

**Granularity**

Fine-grained services

Larger, more modular services

**Data storage**

Each service can have an independent data storage

Involves sharing data storage between services

**Governance**

Requires collaboration between teams

Common governance protocols across teams

**Size and scope**

Better for smaller and web-based applications

Better for large scale integrations

**Communication**

Communicates through an API layer

Communicates through an ESB

**Coupling and cohesion**

Relies on bounded context for coupling

Relies on sharing resources

**Remote services**

Uses REST and JMS

Uses protocols like SOAP and AMQP

**Deployment**

Quick and easy deployment

Less flexibility in deployment

1. What is EDA? with a use-case or example?
2. What is ETL? with a use-case or example?

## Module 2: Design Centre

1. What is RESTFul API

A RESTful API is an architectural style for an application program interface ([API](https://searchapparchitecture.techtarget.com/definition/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

1. What is a Non-Restful API?

The APIs that do not follow REST architecture are called not restful APIs

1. Characteristics of a RESTFul API?

The RESFul 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?

By designing the api we can properly structure the api and also find errors at an early stage as we are following a design first approach in mulesoft while designing apis

1. Is it mandatory to design an API? Explain.
2. What ways can we design a RESTFul API?

We can desing RestFul API using RAML, OAS and APIary

1. What is RAML?

RAML stands for RESTful API Modeling Language. RAML is a [YAML](https://en.wikipedia.org/wiki/YAML)-based language for describing [RESTful](https://en.wikipedia.org/wiki/RESTful) [APIs](https://en.wikipedia.org/wiki/API). It provides a structured and clear format to describe the API. It can be used in a multitude of ways: to implement interactive API consoles, generate documentation, describing an API you are planning to build, and more.

1. What is OAS?

OAS stands for Open API. It was developed by Reverb Technologies in 2010 and 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. It 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 is a platform that uses [API Blueprint](https://apiblueprint.org/) (a markdown related syntax) to help you create the design of your API and generate the documentation, detect syntax errors, and even create a mock API, so you can start testing without having to write a single line of code

<https://developers.messagemedia.com/api-design-using-apiary/>

https://tsh.io/blog/apiary-api-documentation-tool/

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 | Apiary Editor |

1. When should you use each of these languages?
2. Example of RAML and OAS?

**Raml Example**



**Example of 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?

We can design non restful apis using following ways:

* RPC
* SOAP
* GraphQL
* Async API

1. Explain GraphQL API with example?

GraphQL is a *query language* for APIs. At its core, 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.” An API is built by defining public methods; then, the methods are called with arguments. RPC is just a bunch of functions, but in the context of an HTTP API, that entails putting the method in the URL and the arguments in the query string or body.

It use only GET and POST, with GET being used to fetch information and POST being used for everything else. 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?
2. 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 lossely 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 behavior 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?
2. What is Richardson Maturity Model? Explain?

The 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.

### **Level 0: Swamp of POX**

Level 0 uses its implementing protocol (normally HTTP, but it doesn't have to be) like a transport protocol. That is, it tunnels requests and responses through its protocol without using the protocol to indicate application state. It will use only one entry point (URI) and one kind of method (in HTTP, this normally is the POST method). Examples of these are SOAP and XML-RPC.

### **Level 1: Resources**

When your API can distinguish between different resources, it might be level 1. This level uses multiple URIs, where every URI is the entry point to a specific resource. Instead of going through <http://example.org/articles>, you actually distinguish between <http://example.org/article/1> and <http://example.org/article/2>. Still, this level uses only one single method like POST.

### **Level 2: HTTP verbs**

This level indicates that your API should use the protocol properties in order to deal with scalability and failures. Don't use a single POST method for all, but make use of GET when you are requesting resources, and use the DELETE method when you want to delete a resources. Also, use the response codes of your application protocol. Don't use [200](http://httpstatus.es/200) (OK) code when something went wrong for instance. By doing this for the HTTP application protocol, or any other application protocol you like to use, you have reached level 2.

### **Level 3: Hypermedia controls**

Level 3, the highest level, uses HATEOAS to deal with discovering the possibilities of your API towards the clients. More information about HATEOAS can be found below.

1. Explain how to use a visual editor in the design centre.
2. What are API Fragments?

API fragments are reusable component of RAML to make the design and build of a reusable

API even quicker and easier

⮚ Reusable APIs and API fragments are a critical factor in closing the IT delivery gap that arises

when there are too many IT projects that need to be delivered with too few IT resources.

Instead of starting every project from scratch, you can reuse fragments and APIs to

accelerate project delivery.

1. Why do you need Fragments?

Let’s imagine you were building a customer API. You’ll notice that some of the snippets or

“fragments” of the API spec have a lot of potential for reuse. This can be a security scheme

or a data type. For example, a customer data type can consist of first name, last name, and

address. Most likely, the way you describe a customer stays consistent across your APIs,

which makes this data type, or fragment, reusable. If you build a library of these fragments,

you then have many assets for reuse that can accelerate the speed of building your API spec.

Reusable APIs and API fragments are a critical factor in closing the IT delivery gap that arises

when there are too many IT projects that need to be delivered with too few IT resources.

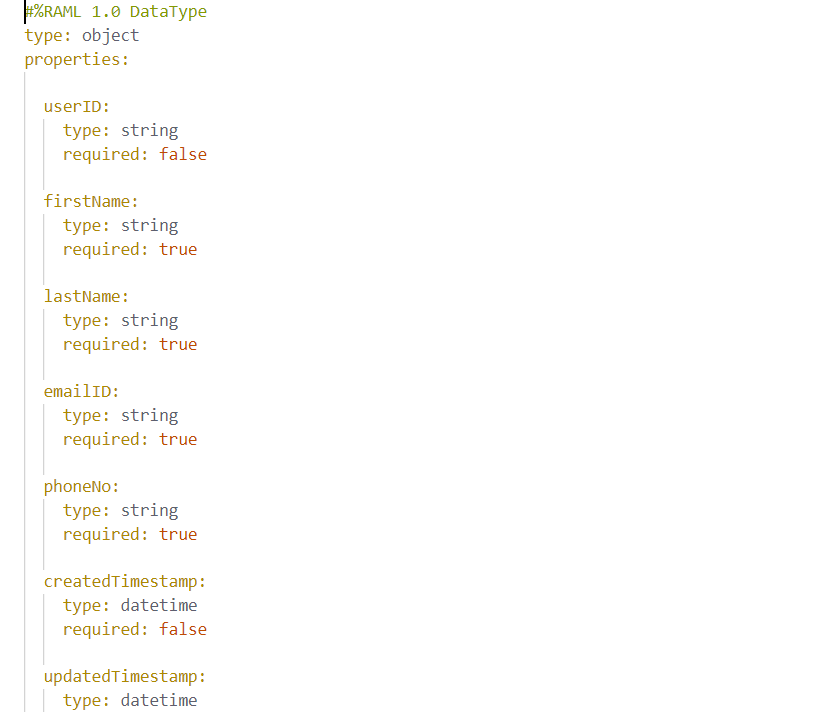
Instead of starting every project from scratch, you can reuse fragments and APIs to

accelerate project delivery.

1. Explain data type with example?

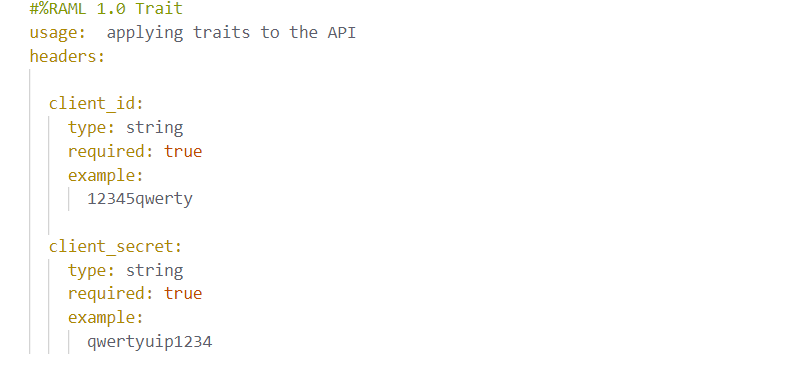
Data Types, a concise and versatile way to describe and validate data inside your API

definition.



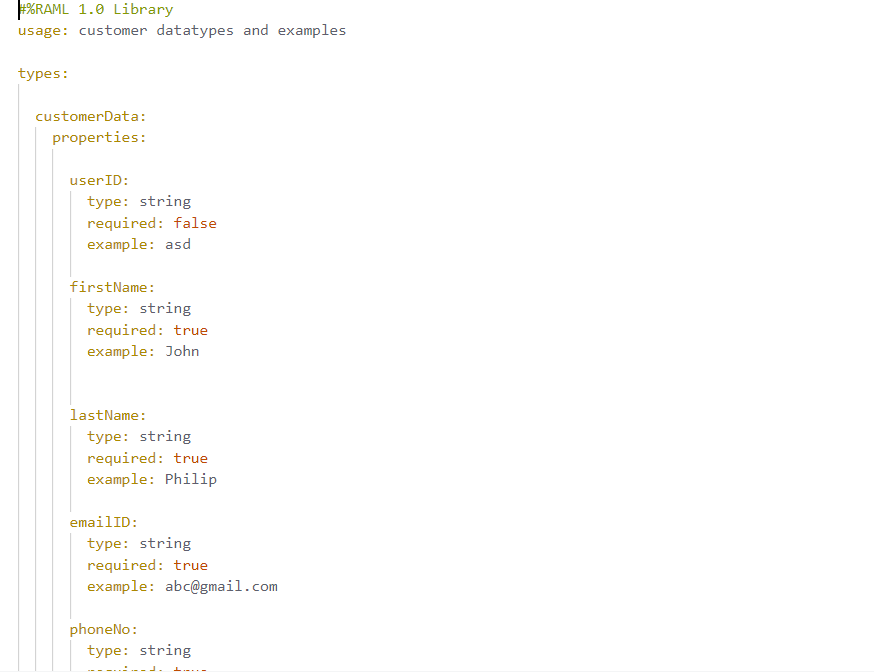
1. Explain Traits with examples?

* Traits is like function and is used to define common attributes for HTTP method (GET, PUT, POST, PATCH, DELETE, etc) such as whether or not they are filterable, searchable, or pageable.
* Traits can be declared in the same RAML file or you can create different files for creating the traits



1. Explain Library with example?

* Libraries provide the ability to bring in pre-defined sets of data-types, resourceTypes, traits, security schemas, and reusable assets - all in a namespaced 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.
* And as the definition says, its not only used for datatypes for all other traits like examples, traits etc but widely used for datatypes.



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.
* Some of the practical examples are given below:



1. Difference between an example and examples?

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

1. What is data-modelling?

**Data modeling (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. Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data

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?

* Canonical data models are a type of data model [that aims](https://www.techopedia.com/definition/30598/canonical-data-model-cdm) to present data entities and relationships in the simplest possible form in order to integrate processes across various systems and databases. More often than not, the data exchanged across various systems rely on different languages, syntax, and protocols. A CDM is also known as a common data model.

1. What is Enterprise wide data-model?
2. What are the advantages and disadvantages of an enterprise wide data-model?
3. What is a bounded context data model?

The bounded context concept originated in Domain-Driven Design (DDD) circles. It is a way of modeling a large domain as independent sub-domains. It promotes an object-model-first approach to a service, defining a data model that a service is responsible for and is "bound to." DDD deals with large models by dividing them into different Bounded Contexts(BC) and being explicit about their interrelationships

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

A mirror backup is an exact copy of the selected folders and files from the source being backed up. Mirror backups are unique in that when you delete a file from the source, that file will eventually be deleted on the mirror backup

## Module 3: Exchange

1. What are connectors?

Connector is a software that provides connection between a mule flow and an external resource. Connectors facilitate easy integration with third-party applications

1. What are the different types of connectors?

Select

Premium

Mulesoft Certified

Community Connector

1. What is a REST connector?

REST Connectors are connector assets with REST endpoints. These connectors are accessed by sending an HTTP request to the host defined in the connector configuration.

Anypoint’s REST Connect and Mule 4 facilitate the conversion of an API-specification to a connector when published to Anypoint Exchange

1. What is a SELECT Connector?

MuleSoft maintains Select connectors. Connectors included in the open source Mule distribution can be used by everyone,

Eg: **MongoDB Connector - Mule 4**, **Web Service Consumer Connector - Mule 4**,

**SharePoint Connector-Mule-04:**

1. What are premium connectors?

MuleSoft maintains Premium connectors. We must purchase Premium connectors as add-ons to our subscription.

Developed by Mulesoft Organisation and certified by the Mulesoft partner.

Eg: **AS2, FTPS**

1. What are mulesoft certified connectors?

MuleSoft Certified connectors are developed by MuleSoft’s partners and developer community and are reviewed and certified by MuleSoft. For support, customers should contact the MuleSoft partner that created the MuleSoft Certified connector.

This type of connector does not require an additional fee to use with Mulesoft Enterprise Edition

Eg: Google analytics connector, atlassian jira rest connector.

1. What are community connectors?

MuleSoft or members of the MuleSoft community write and maintain the Community connectors Connectors built by the community or MuleSoft are generally open-source, although each package may vary.

Community Connectors are generally developed by the developer community.

Eg: slack connector, splunk connector-mule3, google calendar connector

1. What is a custom connector?

Custom connectors are the connectors created by the mulesoft developers community.

We can develop our own connector using the new Mule SDK platform for Mule Runtime 4.

Mule SDK replaces Devkit for developing Custom Connectors.

1. How to create a custom connector?