1. Difference between Mule projects and Domain projects?

* **Mule project** is a MuleSoft project which have **all the dependencies defined in itself**, it doesn’t share its resources or use shared resources. **Domain project** is **used to share** resources between mule application
* Domain project cannot be **deployed to Cloudhub** whereas Mule projects can be deloyed to Cloudhub
* There is no message flow in domain project. Has only global elements and configuration.xml
* **Configuration.xml** file starts with **domain tag** in domain project whereas it starts with **mule tag** in mule project
* Pom file packaging is also different. In domain project it is **domain package** and in mule project it **is mule-application** package

1. What is logging? What is the relation of log4j2 with logging?

Logging helps you to monitor and troubleshoot your application and server.

There are many ways to log some are log4j, slf4j, JSON Logger. Mule internally use slf4j

logging.

Log4j is a java logging utility.

The log4j. properties file is a log4j configuration file which keeps properties

in key-value pairs.

1. What do you mean by Appenders?

**Log4j2 File** uses appenders to write a log. Appenders specify how you want to write a log. **Example** if you want to write a log in a text file, or write a log in a db or write a log in a separate file. You can customize it

1. Differentiate between App logs and Runtime logs.

**App logs**:

The log contains information about any errors raised in the app (unless you have app logic to handle those errors). It also contains anything you want to explicitly log, if you build the logic in the app. You can view an app log as follows:

* If you’re running an app from Anypoint Studio, the output from the app log is visible in Anypoint Studio’s console window.
* If you’re using Mule from the command line to run an app, the app log is visible in your OS console.

**Runtime logs:**

The runtime log (mule\_ee.log) contains information about app and lifecycle events. For example, the log records an entry when a Mule service or app starts, deploys, stops, or undeploys. The runtime log configuration is located in the log4j2.xml file, in

the /conf directory. You can customize this file when running the server in

standalone mode.

1. What are other debug options for a Mule application outside of Anypoint studio?

Log files can be externalized using system like SPLUNK, ELK

1. How can one share Mule applications with other developers?
2. Differentiate between Message Id and Correlation Id?

Correlation Id

* When you perform logging it is important to maintain the correlation ID
* used to uniquely track a request from one system to another
* It keeps the log information of all the APIs that are used to complete a request
* Used mainly for tracking the user activity
* It is recommended to send a correlation ID to keep track activities

**Message ID**

* Every message has an ID to uniquely identify the message

1. How can the Correlation Id be maintained even after making an external HTTP request?
2. What are the standards to be followed when versioning of assets in Exchange? Explain various versioning models? What is the impact of versioning to the consumers using the API?

* Semantic Versioning is a industry standard which is followed to do versioning of software system.
* Format: **MAJOR.MINOR.PATCH**
  + Example: 4.3.0
* It is having four components: Major release number, minor release number, patch

release number and an optional service pack number.

* We should make all the changes backward compatible
* If there are backward incompatible change. Then change **major version. Example** Mule Major version changes from 3 to 4 because there are several functionalities introduced which were not compatible
* If there are backward compatible change. Then change **minor version**
* Bug fixer are **patch version**
* Versioning should be done only through url – **add only the major version**
  + **Example:** http:// localhost:8081/api/**v1**/employees

1. Discuss various control flows in Mule 4?
2. Choice Router

The Choice router dynamically routes messages through a flow according to a set of DataWeave expressions that evaluate message content.

The Choice router enables [content-based routing](http://www.enterpriseintegrationpatterns.com/ContentBasedRouter.html), which is a common way to introduce routing logic based on content of the current message.

1. First Successful

The First Successful router iterates through a list of configured processing routes until one of the routes executes successfully. If any processing route fails execution (throws an error), the router executes the next configured route. If none of the configured routes execute successfully, the First Successful router throws an error.

1. Round Robin

A round robin is a router which iterates through two or more routes in order. During execution it routes to only one route. It maintain the track of the previously selected route and thus avoid the simultaneous execution of the same route .

1. Scatter Gather

The routing message processor Scatter-Gather sends a request message to multiple targets concurrently. It collects the responses from all routes, and aggregates them into a single message.

Error in scatter gather:  Mule: COMPOSITE\_ERROR

This error will be thrown by the S-G component only after every route either fails or completes

To handle this error type, a try scope can be used in each route of Scatter-Gather component. If the error is successfully handled by try scope, then the route will be able to generate a Mule event, for sure.

1. How can one introduce content based routing in a flow?

The Choice router dynamically routes messages through a flow according to a set of DataWeave expressions that evaluate message content

1. Differentiate between mapObject, map and flatMap functions?

**mapObject**

Iterates over an object using a mapper that acts on keys, values, or indices of that object.

Syntax: mapObject({ (K)?: V }, (value: V, key: K, index: Number) -> Object): Object

Example:

Input

%dw 2.0

output application/json

---

{"a":"b","c":"d"} mapObject (value,key,index) -> { (index) : { (value):key} }

Output

{ "0": { "b": "a" }, "1": { "d": "c" } }

**Map**

Iterates over items in an array and outputs the results into a new array.

Syntax:   
map(Array<T>, (item: T, index: Number) -> R): Array<R>

Example

Input

%dw 2.0

output application/json

---

["jose", "pedro", "mateo"] map (value, index) -> { (index) : value}

OutPut:

[ { "0": "jose" }, { "1": "pedro" }, { "2": "mateo" } ]

**flatMAp:**

Iterates over each item in an array and flattens the results.

Syntax:   
flatMap(Array<T>, (item: T, index: Number) -> Array<R>): Array<R>

Example

Input

%dw 2.0

output application/json

---

[ [3,5], [0.9,5.5] ] flatMap (value, index) -> value

Output

[ 3, 5, 0.9, 5.5]

1. How to convert an Object into Array?

Use Pluck

Useful for mapping an object into an array, pluck iterates over an object and returns an array of keys, values, or indices from the object.

1. How to convert a nested array into a normal array?

Use Flatten

Turns a set of subarrays (such as [ [1,2,3], [4,5,[6]], [], [null] ]) into a single, flattened array (such as [ 1, 2, 3, 4, 5, [6], null ]).

Note that it flattens only the first level of subarrays and omits empty subarrays.

1. Difference between flatten and flatMap?

Instead of returning an array of arrays (as map does when you iterate over the values within an input like [ [1,2], [3,4] ]), flatMap returns a flattened array that looks like this: [1, 2, 3, 4]. flatMap is similar to flatten, but flatten only acts on the values of the arrays, while flatMap can act on values and indices of items in the array.

1. Describe the various Error handling mechanisms and when to use them?

**On-Error Propagate**

* + Roll back previous transaction
  + processes the error message and re-throws the error to its parent flow.
  + The HTTP listener returns an error response.

**On-Error Continue**

* Commit previous transaction
* catches the error and does not report it as an error
* The HTTP listener returns a successful response

1. What are the available Transaction Types in Mulesoft? Explain citing examples?

Transactions are operations in a Mule app for which the result cannot remain indeterminate. When a series of steps in a flow must succeed or fail as one unit, Mule uses a transaction to demarcate that unit.

Two transaction types

* Single Resource (Local, the default)
* Extended Architecture

https://docs.mulesoft.com/mule-runtime/4.3/transaction-management