**Insert data into database using query parameters**:

\* First we go to MySQL and click on new schema (project), in this we create a new table and

then click on finish.

\* Next we have to create a column names after we insert the rows.

\* Next we have to finish that all.

\* Fetch is nothing but to select a row by using query parameters.

Next we go to Any point Studio we can create a new project and then finish.

**The flow be like:**

HTTP Listener ->logger -> database insert -> Transform message

**HTTP listener**:

The HTTP listener is an event source that enables you to set up an HTTP server and trigger flows when HTTP requests are received.

You can choose what methods the source accepts, such as GET, POST or a list of methods, and on which path to accept requests, thereby allowing the routing of requests through different flows.

Once a request is accepted by the listener, the corresponding flow is triggered with the HTTP body as payload and the HTTP data as attributes (headers, query parameters and so on).

When the flow finishes its execution, the HTTP listener enables you to customize the HTTP response based on whether the execution was successful or not, so that different status codes can be returned.

**Transform message**:

The Transform Message component carries out transformations over the input data it receives. You can explicitly write out a transformation in Data Weave language, or you can use the UI to build it implicitly through dragging and dropping elements.

**Logger**:

This Core component helps you monitor and debug your Mule application by logging important information such as error messages, status notifications, payloads, and so on. You can add a Logger anywhere in a flow, and you can configure it to log a string that you specify, the output of a Data Weave expression you write, or any combination of strings and expressions. In Logger we have to mention what is the key we are going to use at the time of postman to perform the operation.

**Flow Explanation:**

1. Take HTTP Listener to listen the requests.

2. Take logger to print the data on console.

3. Take database insert component to insert the data into table.

4. In database insert component write the query to insert data.

**Query**:

Insert into tablename values (fields)

**Transform Message**:

%dw 2.0

output application/json

---

{

"status":200,

"message": "inserted successfully"

}

**Delete data from database using query parameters**:

**The flow be like:**

HTTP Listener ->logger -> database delete -> Transform message

**HTTP listener**:

The HTTP listener is an event source that enables you to set up an HTTP server and trigger flows when HTTP requests are received.

You can choose what methods the source accepts, such as GET, POST or a list of methods, and on which path to accept requests, thereby allowing the routing of requests through different flows.

Once a request is accepted by the listener, the corresponding flow is triggered with the HTTP body as payload and the HTTP data as attributes (headers, query parameters and so on).

When the flow finishes its execution, the HTTP listener enables you to customize the HTTP response based on whether the execution was successful or not, so that different status codes can be returned.

**Transform message**:

The Transform Message component carries out transformations over the input data it receives. You can explicitly write out a transformation in Data Weave language, or you can use the UI to build it implicitly through dragging and dropping elements.

**Logger**:

This Core component helps you monitor and debug your Mule application by logging important information such as error messages, status notifications, payloads, and so on. You can add a Logger anywhere in a flow, and you can configure it to log a string that you specify, the output of a Data Weave expression you write, or any combination of strings and expressions. In Logger we have to mention what is the key we are going to use at the time of postman to perform the operation.

**Flow Explanation:**

1. Take HTTP Listener to listen the requests.

2. Take logger to print the data on console.

3. Take database delete component to delete the data from table.

4. In database delete component write the query to delete data.

**Query**:

delete from tablename values (fields)

**Transform Message**:

%dw 2.0

output application/json

---

{

"status":200,

"message": "deleted successfully"

}

**Select data from database using query parameters**:

**The flow be like:**

HTTP Listener ->logger -> database select -> Transform message

**HTTP listener**:

The HTTP listener is an event source that enables you to set up an HTTP server and trigger flows when HTTP requests are received.

You can choose what methods the source accepts, such as GET, POST or a list of methods, and on which path to accept requests, thereby allowing the routing of requests through different flows.

Once a request is accepted by the listener, the corresponding flow is triggered with the HTTP body as payload and the HTTP data as attributes (headers, query parameters and so on).

When the flow finishes its execution, the HTTP listener enables you to customize the HTTP response based on whether the execution was successful or not, so that different status codes can be returned.

**Transform message**:

The Transform Message component carries out transformations over the input data it receives. You can explicitly write out a transformation in Data Weave language, or you can use the UI to build it implicitly through dragging and dropping elements.

**Logger**:

This Core component helps you monitor and debug your Mule application by logging important information such as error messages, status notifications, payloads, and so on. You can add a Logger anywhere in a flow, and you can configure it to log a string that you specify, the output of a Data Weave expression you write, or any combination of strings and expressions.

In Logger we have to mention what is the key we are going to use at the time of postman to perform the operation.

**Flow Explanation:**

1. Take HTTP Listener to listen the requests.

2. Take logger to print the data on console.

3. Take database select component to select the data into table.

4. In database select component write the query to select data from data base.

**Query**:

select \* from tablename values (fields)

**Transform Message**:

%dw 2.0

output application/json

---

{

"status":200,

"message": "selected successfully"

}

**Update data to database using query parameters**:

**The flow be like:**

HTTP Listener ->logger -> database Update -> Transform message

**HTTP listener**:

The HTTP listener is an event source that enables you to set up an HTTP server and trigger flows when HTTP requests are received.

You can choose what methods the source accepts, such as GET, POST or a list of methods, and on which path to accept requests, thereby allowing the routing of requests through different flows.

Once a request is accepted by the listener, the corresponding flow is triggered with the HTTP body as payload and the HTTP data as attributes (headers, query parameters and so on).

When the flow finishes its execution, the HTTP listener enables you to customize the HTTP response based on whether the execution was successful or not, so that different status codes can be returned.

**Transform message**:

The Transform Message component carries out transformations over the input data it receives. You can explicitly write out a transformation in Data Weave language, or you can use the UI to build it implicitly through dragging and dropping elements.

**Logger**:

This Core component helps you monitor and debug your Mule application by logging important information such as error messages, status notifications, payloads, and so on. You can add a Logger anywhere in a flow, and you can configure it to log a string that you specify, the output of a Data Weave expression you write, or any combination of strings and expressions.

In Logger we have to mention what is the key we are going to use at the time of postman to perform the operation.

**Flow Explanation:**

1. Take HTTP Listener to listen the requests.

2. Take logger to print the data on console.

3. Take database update component to update the data into table.

4. In database update component write the query to update data from data base.

**Query**:

update tablename values (fields) (:fields)

**Transform Message**:

%dw 2.0

output application/json

---

{

"status":200,

"message": "updated successfully"

}