APACHE Camel

spring boot

Decoded Bytes

Apache Camel w/ SpringBoot Masterclass

Course Coverage

Section 1: Build a Simple Spring-boot application, add Apache Camel capabilities

Section 2: Build a Simple File Transfer Application with record processing

Section 3: Host a REST endpoint and add JPA capabilities

Section 4: Integrate with ActiveMQ

Section 5: Exception Handling and enterprise best-practices

Components

FILE
TIMER
BEANIO
REST
JPA
ACTIVEMQ
and more...

Patterns

INJECTORS
SPLITTER
PIPELINES
WIRETAP
STATIC vs DYNAMIC
ROUTING
DIRECT vs SEDA

!! What you should now

☑ Java 8 or above

☑ Basic familiarity w/ Maven

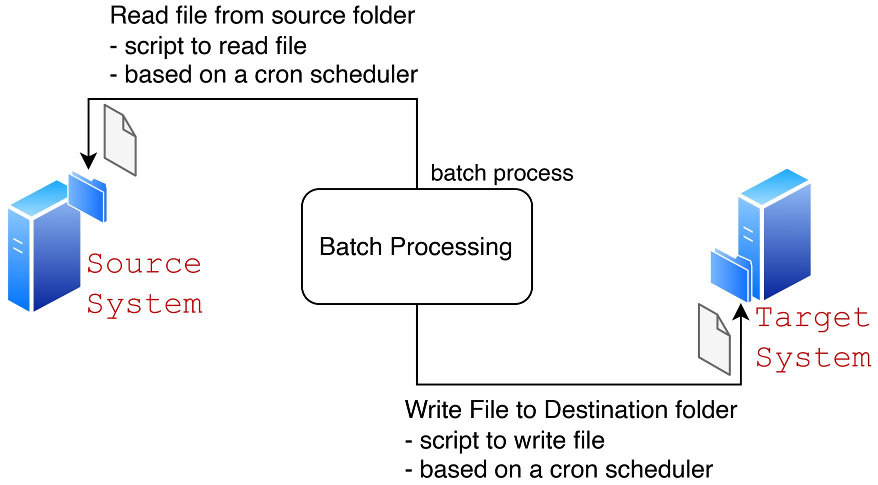
☑ Spring-boot basics

☑ Basics of REST, JPA and Messaging

Camel ToolKit

Simple and Constants, Mocking and AdviceWith etc.

Legacy File Transfer (Batch)



Read file from source folder
- script to read file
- based on a cron scheduler

batch process

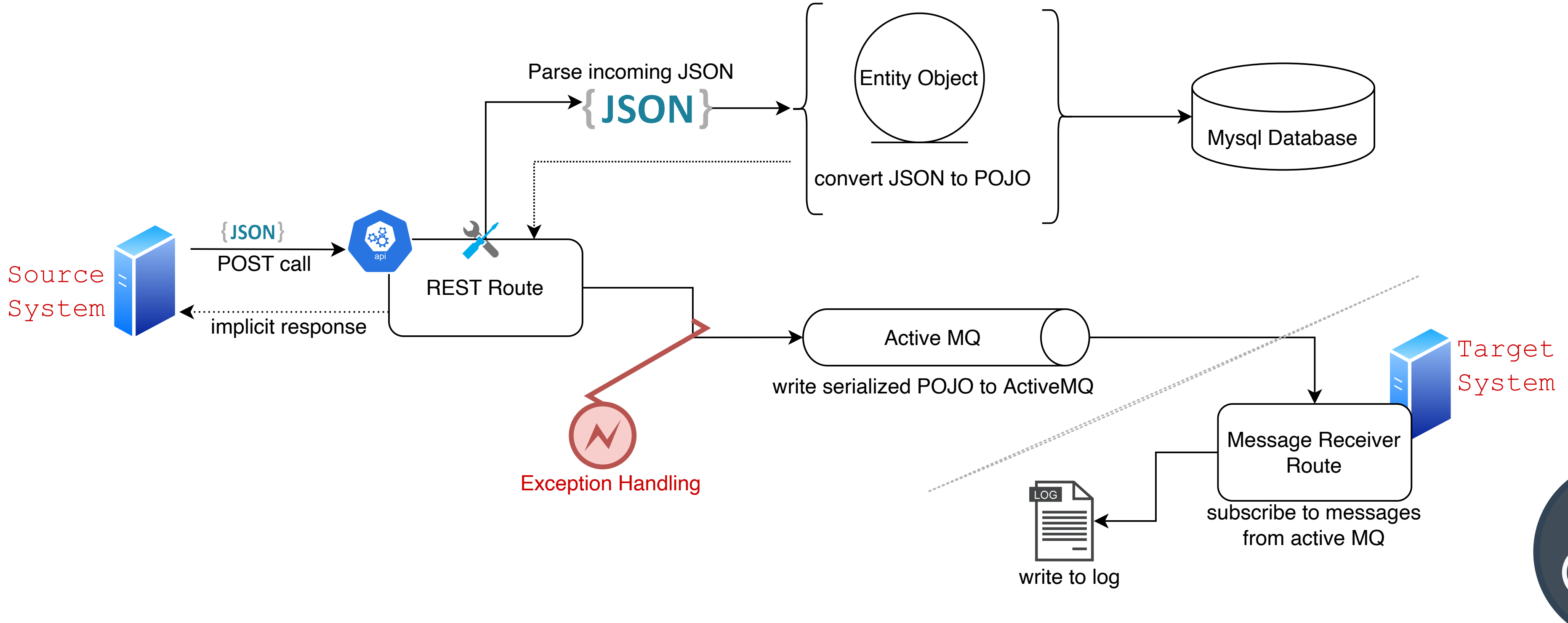
Batch Processing

Source System

Target System

Write File to Destination folder
- script to write file
- based on a cron scheduler

Updated Data Transfer (Real-time)



Parse incoming JSON

{JSON}

Entity Object

convert JSON to POJO

Mysql Database

Source System

{JSON}

POST call

REST Route

implicit response

Active MQ

write serialized POJO to ActiveMQ

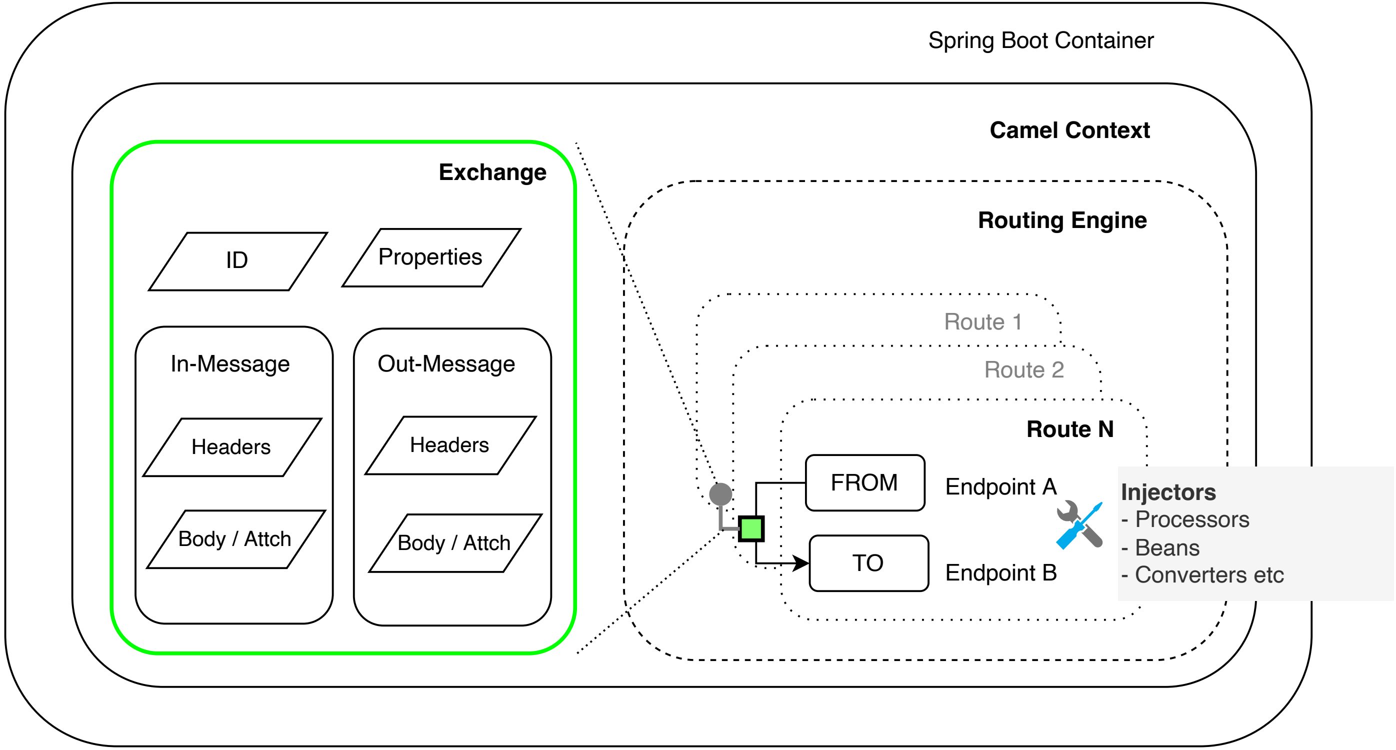
Exception Handling

Target System

Message Receiver Route

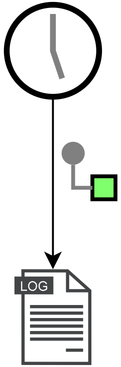
subscribe to messages from active MQ

write to log



- Route definition has a `FROM` endpoint, a `TO` endpoint
- Route definition may have Injectors like processors, Beans, or Converters
- Every Data Propagation is encapsulated in an `Exchange` object which keeps track on incoming and outgoing messages

Components to use:
- Camel Timer
- Log

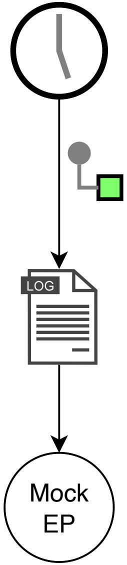


1. Run a timer with a duration i.e. create a new exchange every `x` seconds

2. Assign a body to the Out Message in the exchange

3. Read the body from In-Message in the exchange and write to log output

Components to use:
- Camel Timer
- Log
- Mock

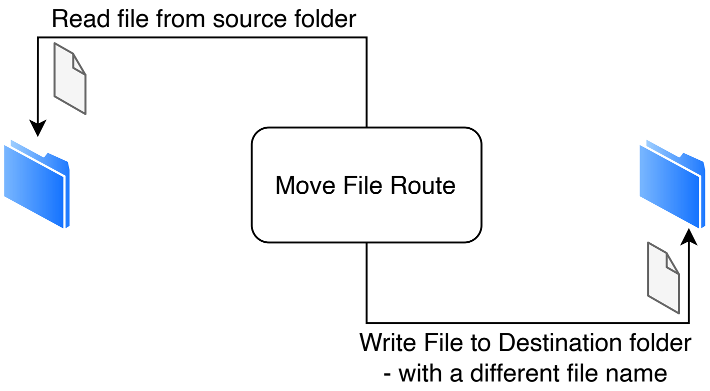


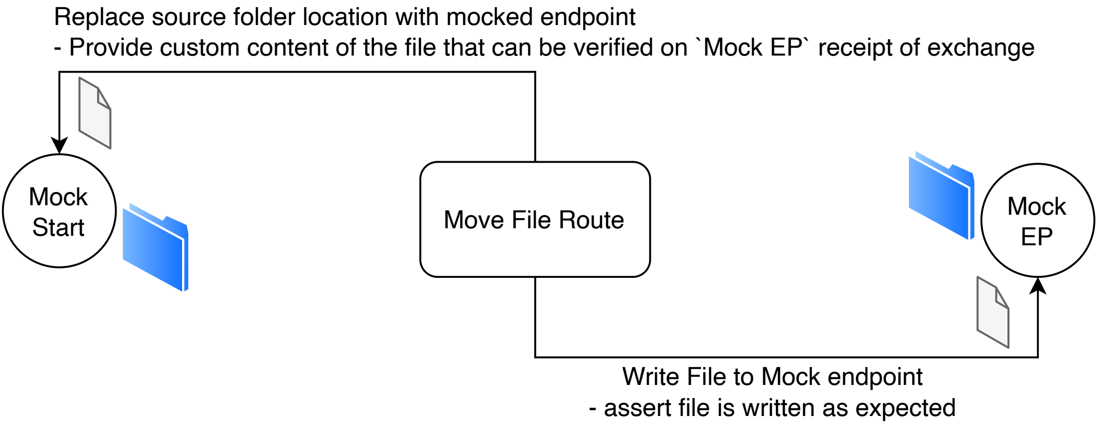
1. Run a timer with a duration i.e. create a new exchange every `x` seconds

2. Assign a body to the Out Message in the exchange

3. Read the body from In-Message in the exchange and write to log output

4. Add another `mock` endpoint to last of the route and assert the body received is as expected


Read file from source folder
Move File Route
Write File to Destination folder
- with a different file name



Replace source folder location with mocked endpoint
- Provide custom content of the file that can be verified on `Mock EP` receipt of exchange

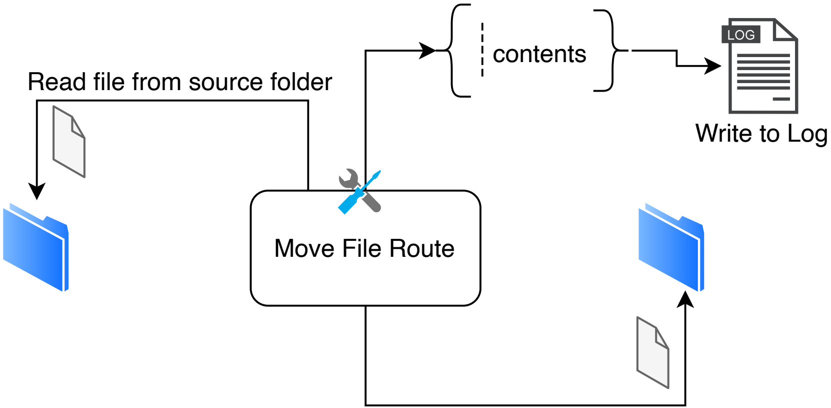
Mock Start

Move File Route

Mock EP

Write File to Mock endpoint
- assert file is written as expected

Introduction to Processors

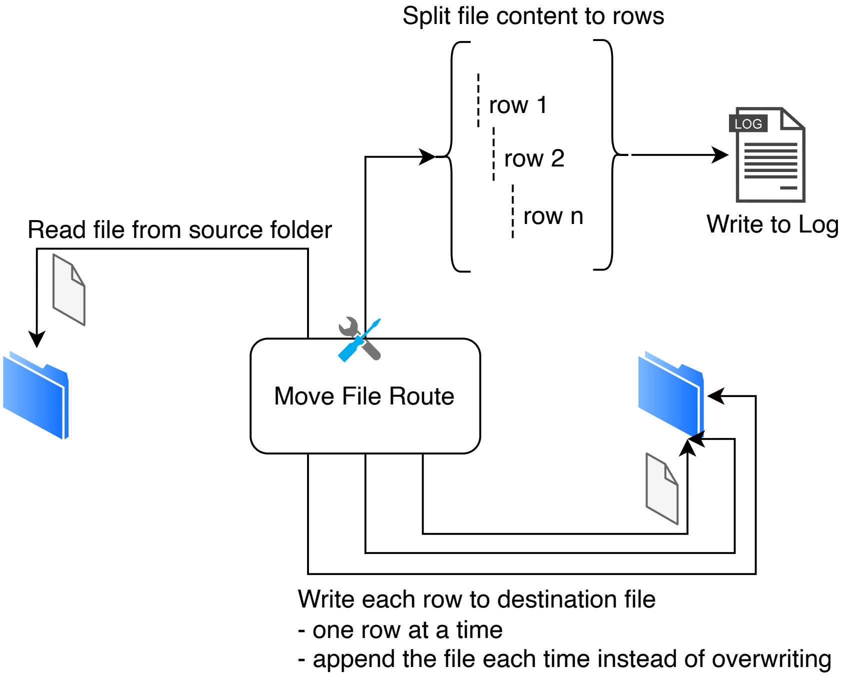


Read file from source folder

contents

Write to Log

Move File Route



Split file content to rows

row 1

row 2

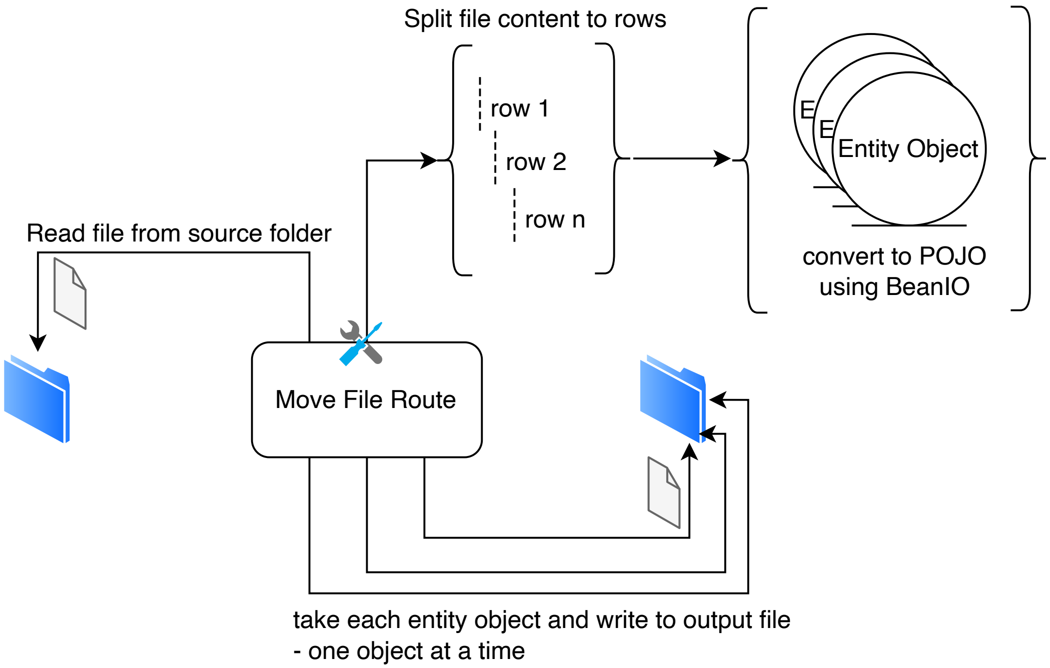
row n

Write to Log

Read file from source folder

Move File Route

Write each row to destination file
- one row at a time
- append the file each time instead of overwriting



Split file content to rows

row 1

row 2

row n

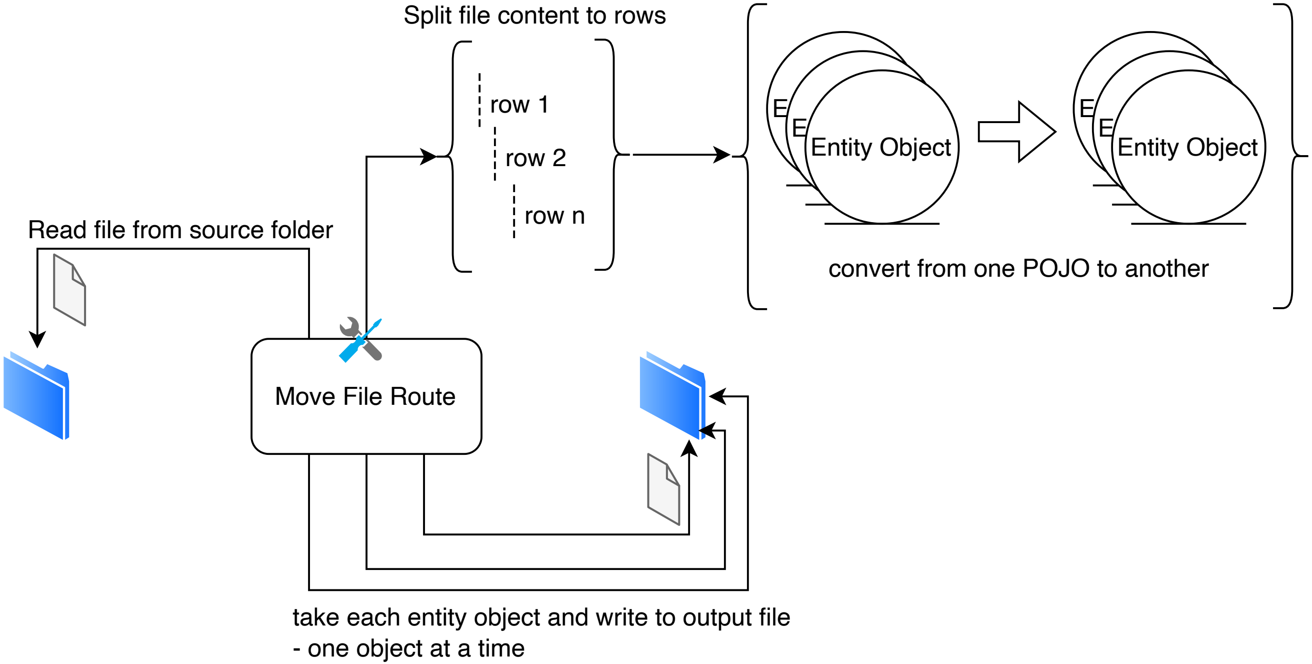
Entity Object

convert to POJO
using BeanIO

Read file from source folder

Move File Route

take each entity object and write to output file
- one object at a time



Split file content to rows

row 1

row 2

row n

Entity Object

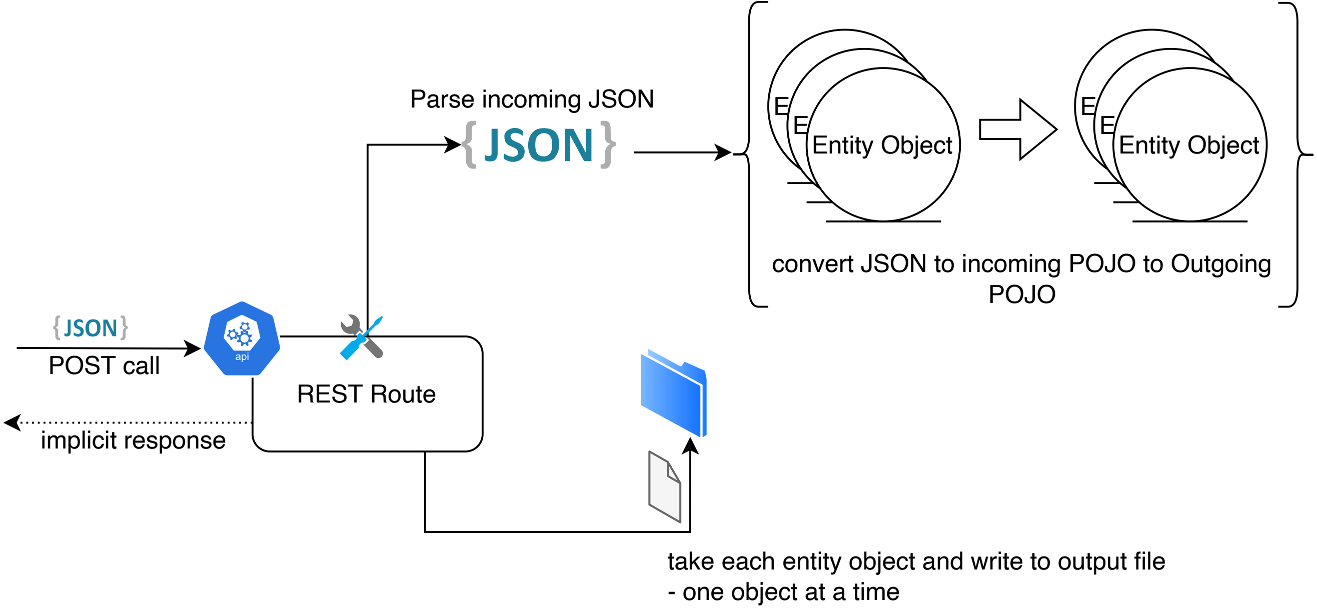
Entity Object

convert from one POJO to another

Read file from source folder

Move File Route

take each entity object and write to output file
- one object at a time



Parse incoming JSON

convert JSON to incoming POJO to Outgoing POJO

POST call

REST Route

implicit response

take each entity object and write to output file - one object at a time

Postman

1. Browse to postman.com/downloads/

2. Install Postman

MySQL

For MacOS

1. Open Terminal

2. command to run:
```
brew install mysql
```

3. command to run:
```
mysql_secure_installation
```

Start/Stop MySQL

```
mysql.server start/stop
```

For Windows

1. Download MySQL from https://dev.mysql.com/downloads/installer/

2. Additional configuration available on below link https://dev.mysql.com/doc/mysql-installation-excerpt/8.0/en/windows-installation.html

ActiveMQ

1. Browse to https://activemq.apache.org/components/classic/download/

2. Download Active MQ zip file

3. Extract the zip file into location of your choice

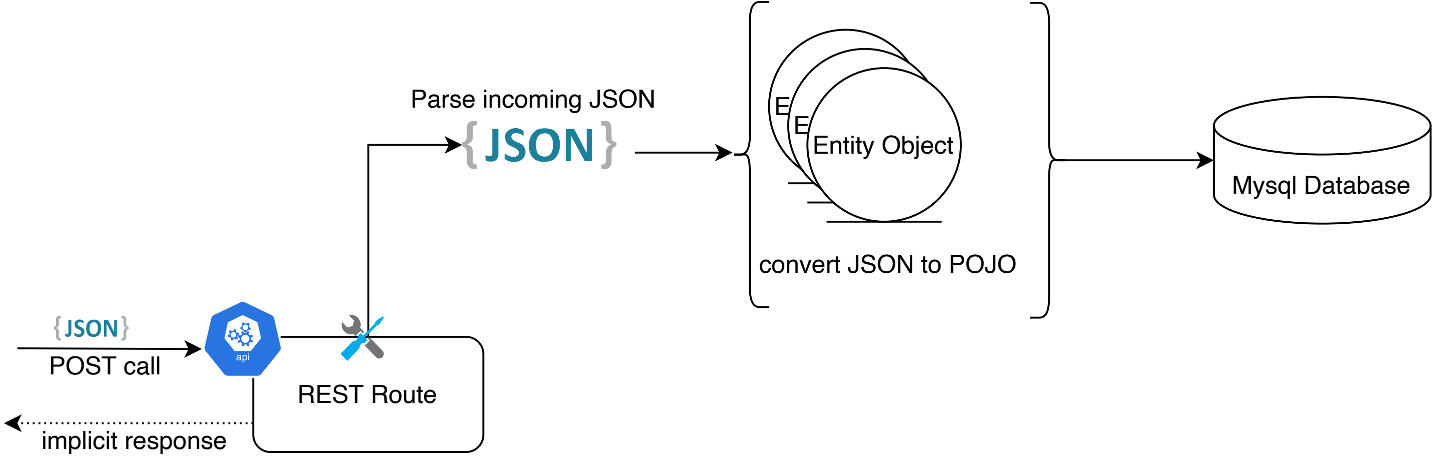
4. Browse to apache-activemq-{version}/bin

5. Run ./activemq start

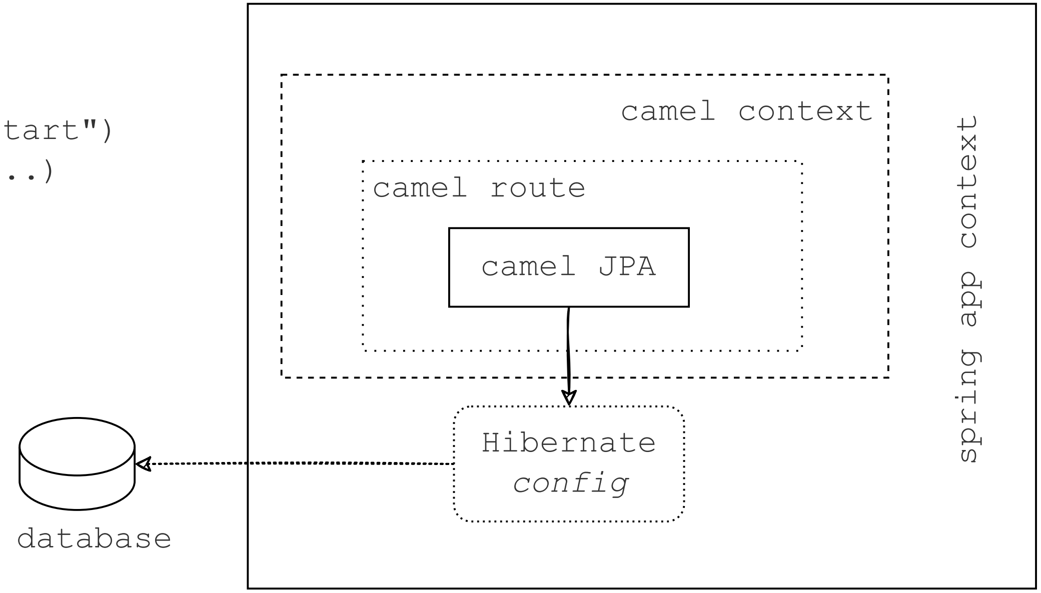
6. Open a browser and access ActiveMQ console http://localhost:8161/index.html

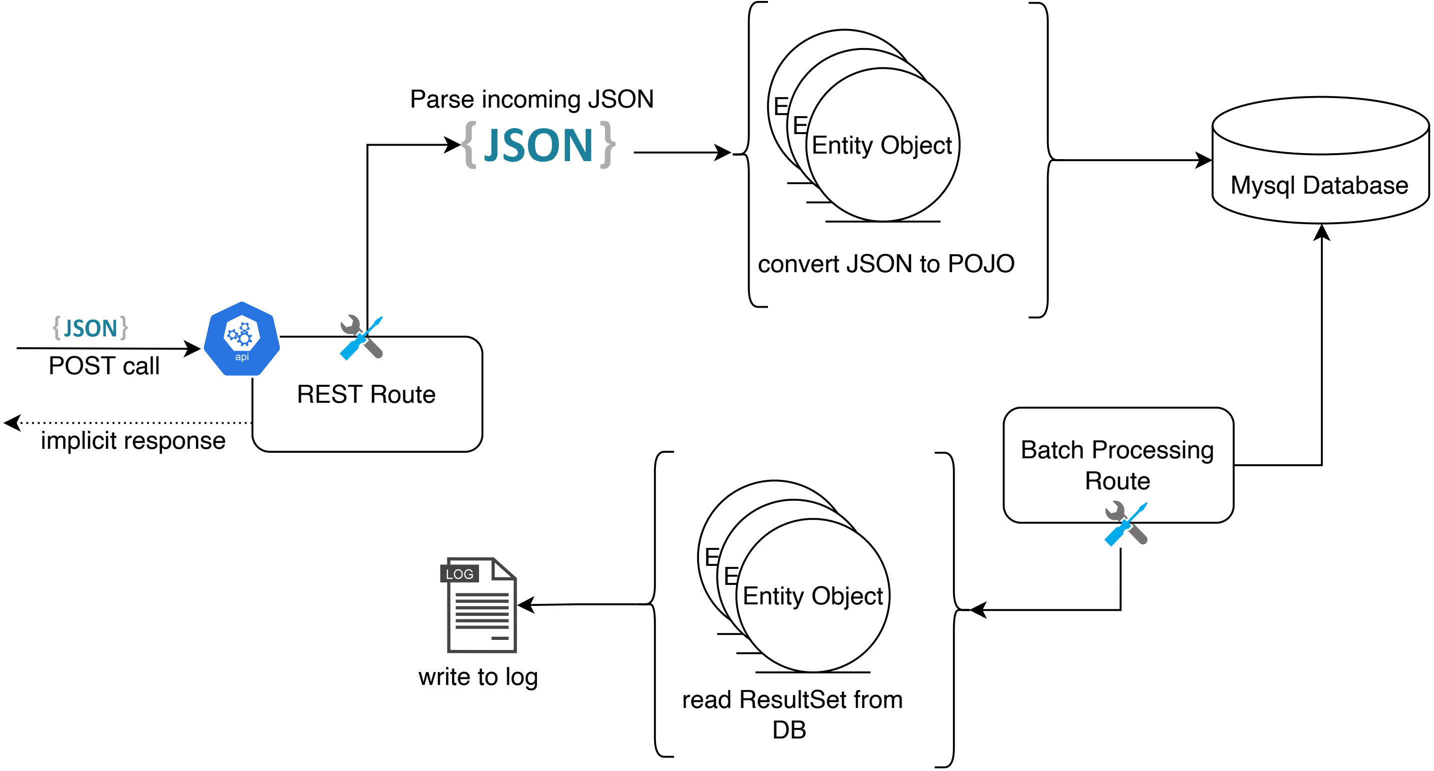
user: admin
password: admin

7. To Stop activeMQ run ./activemq stop



```
from("direct:start")
    .to("jpa:....)
```





Parse incoming JSON

{JSON}

Entity Object

convert JSON to POJO

Mysql Database

{JSON}

POST call

api

REST Route

implicit response

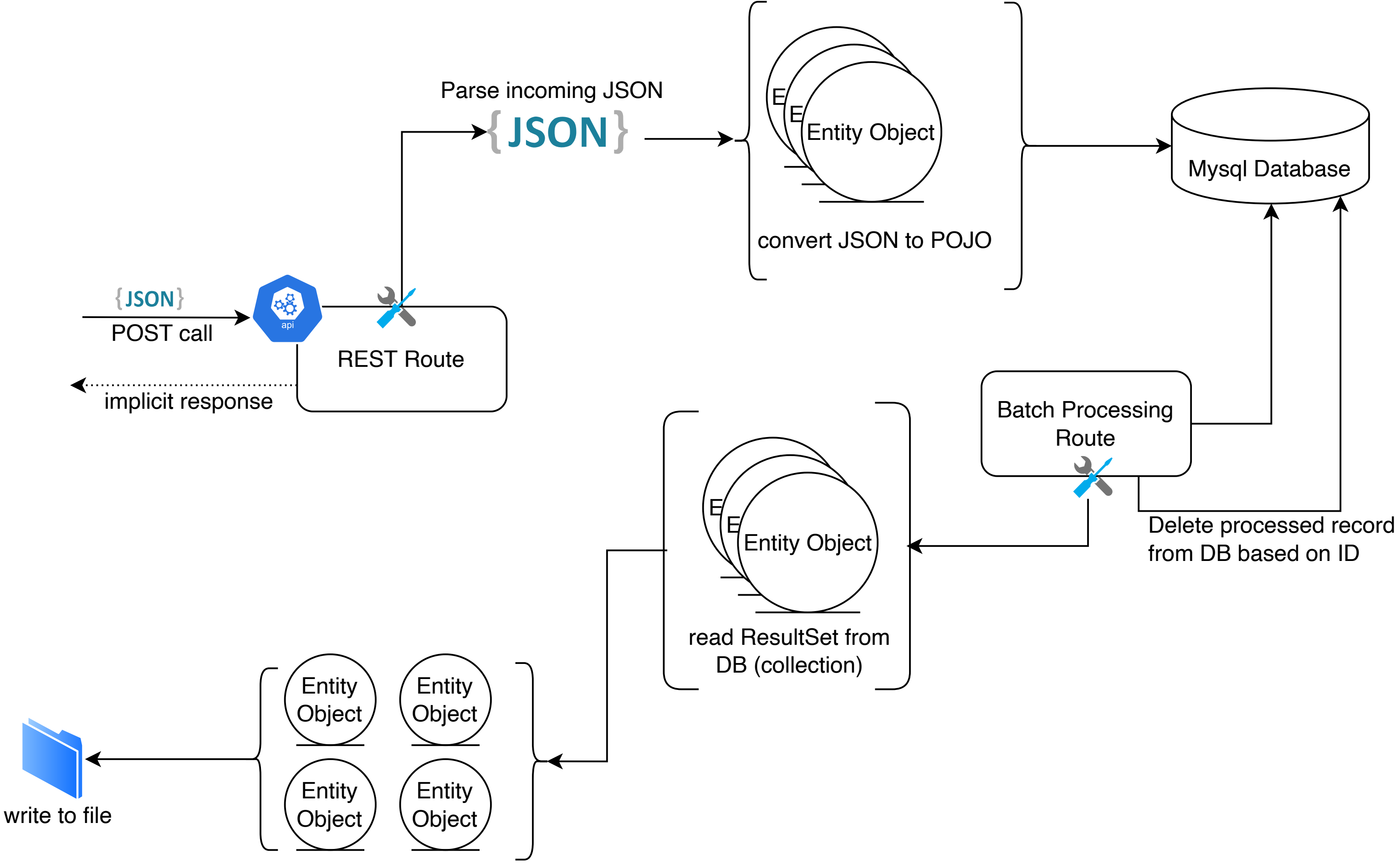
Batch Processing Route

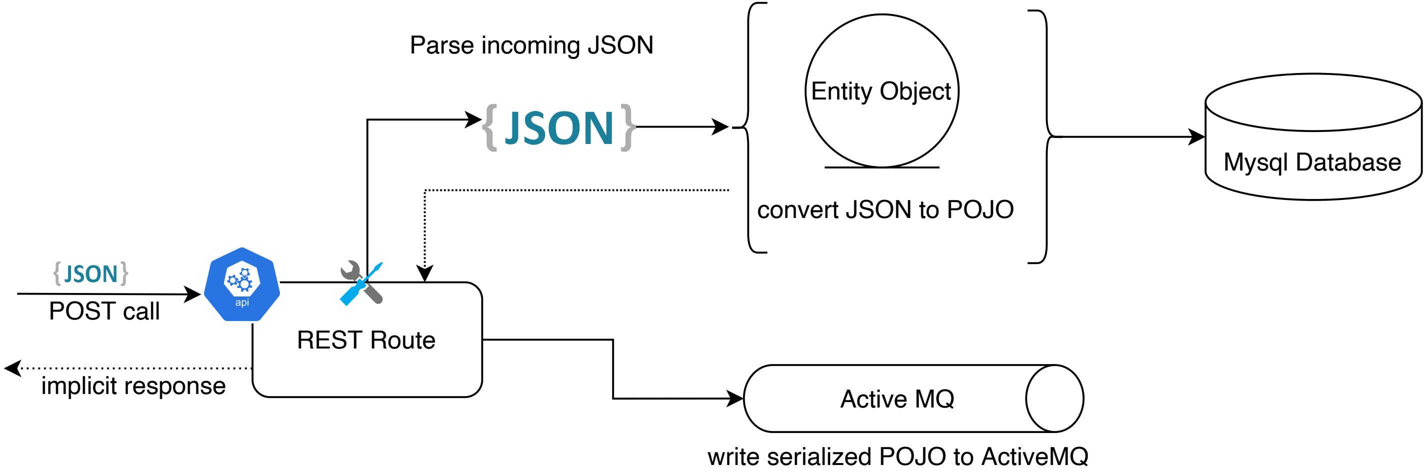
Entity Object

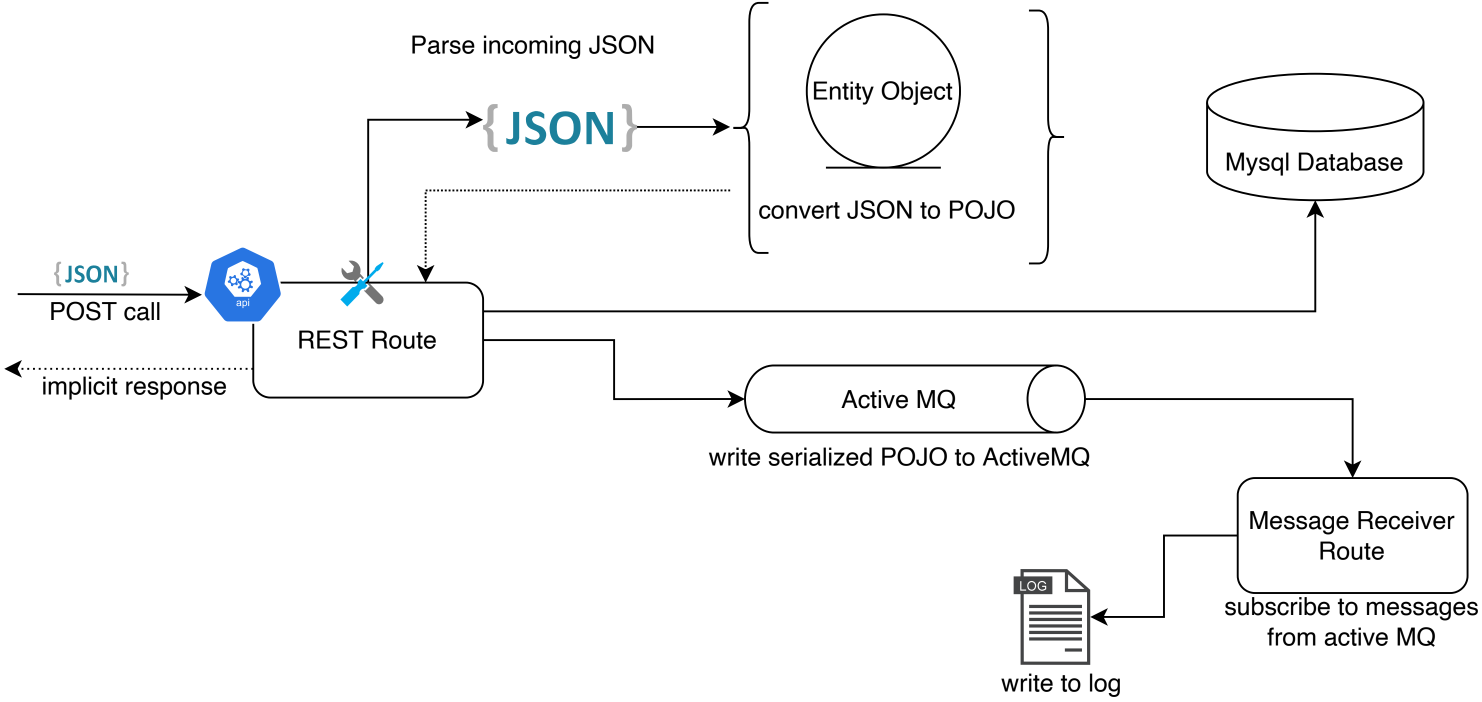
read ResultSet from DB

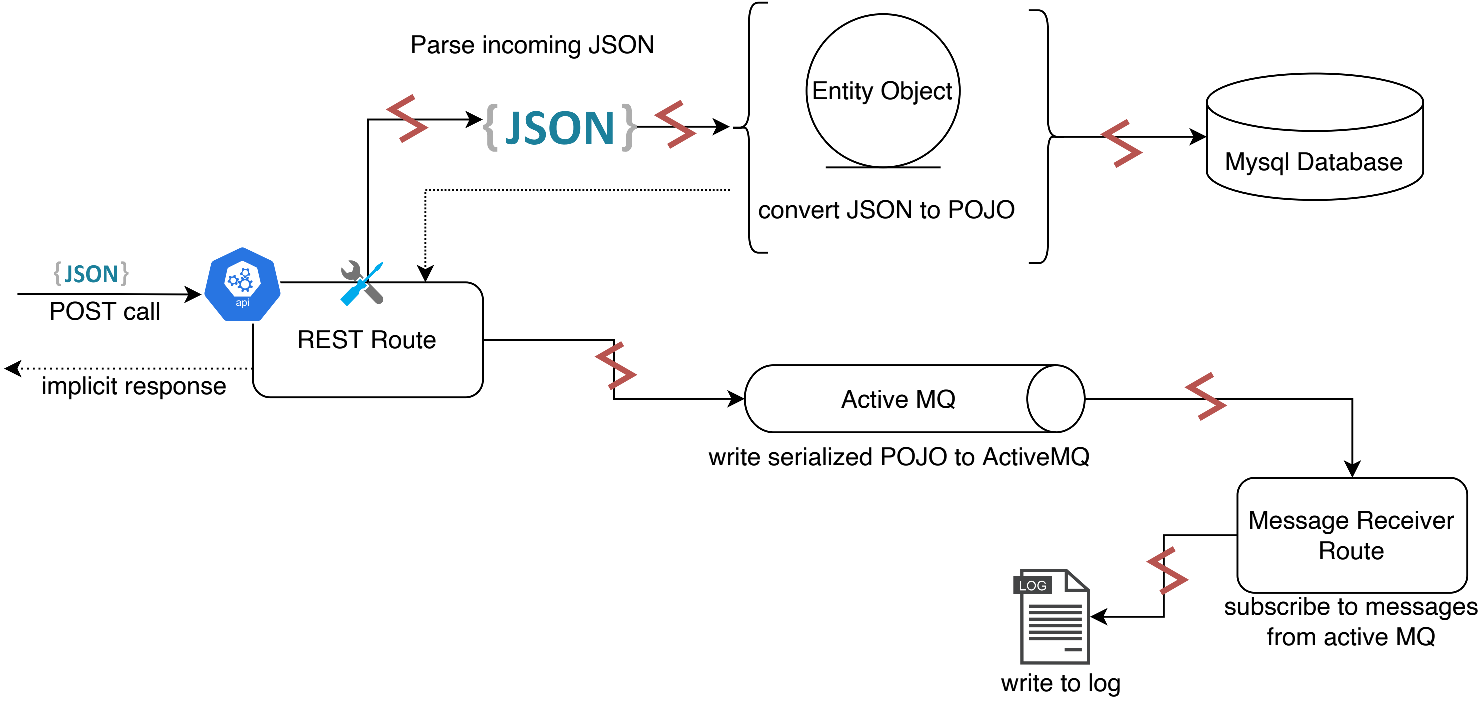
LOG

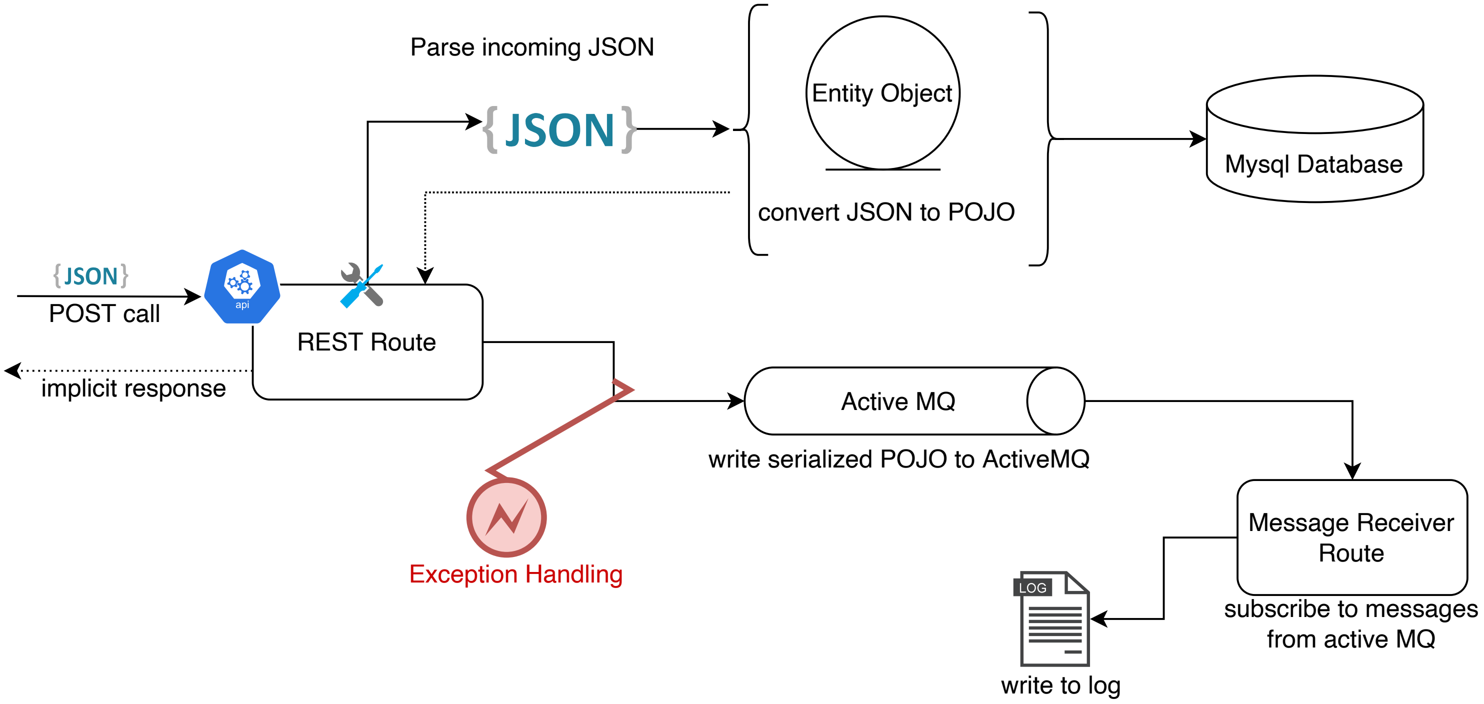
write to log


{JSON}
POST call
api
REST Route
implicit response
Parse incoming JSON
{JSON}
E
E
Entity Object
convert JSON to POJO
Mysql Database
Batch Processing Route
Delete processed record from DB based on ID
E
E
Entity Object
read ResultSet from DB (collection)
Entity Object
Entity Object
Entity Object
Entity Object
write to file


Parse incoming JSON
{JSON}
Entity Object
convert JSON to POJO
Mysql Database
{JSON}
POST call
api
REST Route
implicit response
Active MQ
write serialized POJO to ActiveMQ


Parse incoming JSON
{JSON}
Entity Object
convert JSON to POJO
Mysql Database
{JSON}
POST call
api
REST Route
implicit response
Active MQ
write serialized POJO to ActiveMQ
Message Receiver Route
subscribe to messages from active MQ
LOG
write to log


Parse incoming JSON
{JSON}
Entity Object
convert JSON to POJO
Mysql Database
{JSON}
POST call
REST Route
implicit response
Active MQ
write serialized POJO to ActiveMQ
Message Receiver Route
subscribe to messages from active MQ
LOG
write to log


Parse incoming JSON
{JSON}
Entity Object
convert JSON to POJO
Mysql Database
{JSON}
POST call
api
REST Route
implicit response
Active MQ
write serialized POJO to ActiveMQ
Exception Handling
Message Receiver Route
subscribe to messages from active MQ
LOG
write to log