EMEAApiService Tutorial

# Distribution folder structure:

This application is a gradle project. Once a gradle build is completed it produced a zip file. Once extracted following folders are present.

root

|-- bin

| |-- EMEAApiService.bat

| |-- EMEAApiService.sh

|-- conf

| |-- application.properties

| |-- log4j.properties

|

|-- docs

| |-- HowTo.docx

|-- log

**Conf:**

This folder contains application.properties, log4j.properties file.

User can change server port, database connection details as necessary. Once they are changed application has to be restarted.

**log:**

By default this is the log folder, user can modify the log folder location in log4j.properties.

**Bin:**

In this folder two scripts are there, one bat file for windows and one sh file for unix environment is present to run the application.

To start the application user has to run either of these script.

# Available Features:

## RestService to display account information:

User can fire a rest api call using any rest client e.g postman. Once user specify proper account number and sort code system will display list of transactions for the account number present in the system.

If user provide invalid inputs proper message would be displayed.

### Request details:

Method : post

Url: localhost:8081/emeaapiservice/accountdetails

Body: {

"accountNumber":"1",

"sortCode":"122312"

}

Account number and sort code both are mandatory.

Both has to be numeric.

Sort code is six digit integer.

If any invalid input is provided proper message would be displayed.

Header:

Accept: application/json

Content-Type: application/json

# Sample db script to prepare data:

Following db scripts are mandatory to run and deploy the application.

CREATE TABLE user\_details (

username VARCHAR(50) NOT NULL PRIMARY KEY,

email VARCHAR(50),

password VARCHAR(500),

activated VARCHAR(50),

activationkey VARCHAR(50),

resetpasswordkey VARCHAR(50)

);

CREATE TABLE authority (

name VARCHAR(50) NOT NULL PRIMARY KEY

);

CREATE TABLE user\_authority

( username VARCHAR(50) NOT NULL,

authority VARCHAR(50) NOT NULL

);

alter table user\_authority add CONSTRAINT FK\_user FOREIGN KEY (username) REFERENCES user\_details (username) ;

alter table user\_authority add CONSTRAINT FK\_authority FOREIGN KEY (authority) REFERENCES authority (name) ;

alter table user\_authority add CONSTRAINT Uniq\_authority Unique(username,authority);

CREATE TABLE oauth\_access\_token (

token\_id VARCHAR(256) DEFAULT NULL,

token BLOB,

authentication\_id VARCHAR(256) DEFAULT NULL,

user\_name VARCHAR(256) DEFAULT NULL,

client\_id VARCHAR(256) DEFAULT NULL,

authentication BLOB,

refresh\_token VARCHAR(256) DEFAULT NULL

);

CREATE TABLE oauth\_refresh\_token (

token\_id VARCHAR(256) DEFAULT NULL,

token BLOB,

authentication BLOB

);

INSERT INTO user\_details (username,email, password, activated) VALUES ('admin', 'admin@mail.me', 'b8f57d6d6ec0a60dfe2e20182d4615b12e321cad9e2979e0b9f81e0d6eda78ad9b6dcfe53e4e22d1', 'true');

INSERT INTO user\_details (username,email, password, activated) VALUES ('user', 'user@mail.me', 'd6dfa9ff45e03b161e7f680f35d90d5ef51d243c2a8285aa7e11247bc2c92acde0c2bb626b1fac74', 'true');

INSERT INTO user\_details (username,email, password, activated) VALUES ('hasim', 'hasim@abc.com', 'test', 'true');

INSERT INTO authority (name) VALUES ('ROLE\_USER');

INSERT INTO authority (name) VALUES ('ROLE\_ADMIN');

INSERT INTO user\_authority (username,authority) VALUES ('hasim', 'ROLE\_USER');

INSERT INTO user\_authority (username,authority) VALUES ('hasim', 'ROLE\_ADMIN');

INSERT INTO user\_authority (username,authority) VALUES ('user', 'ROLE\_USER');

INSERT INTO user\_authority (username,authority) VALUES ('admin', 'ROLE\_USER');

INSERT INTO user\_authority (username,authority) VALUES ('admin', 'ROLE\_ADMIN');

select account\_info\_seq.nextVal from dual;

insert into account values (account\_info\_seq.currVal,122312);

select transaction\_seq.nextVal from dual;

insert into transaction values (transaction\_seq.currVal,'as','as',TO\_TIMESTAMP('2017-07-02 06:14:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'),'c',123.20,'we','we','we','ew','we',TO\_TIMESTAMP('2017-03-02 06:14:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'),'ew','er',234.29,'ewr','wed','we','EarMark',account\_info\_seq.currVal);

# How TO Deploy The Application

Step1: Download source code from [GitHub Link](https://github.com/hasimatsapient/EMEAApiService.git)

Step2: Extract zip file in a local folder (e.g D:\ EMEAApiService-master).

Step3: Go to folder EMEAApiService inside it.

Step4: Open command prompt and run “ gradle clean distzip “.

Step5: Go to build/distributions folder under it.

Step6: Extract files under generated distribution.

Step7: Go to folder till get bin directory.

Step8: Modify application.properties file under conf folder as necessary.

Step9: Execute sample db script present in this document for initial setup and to prepare data.

Step10: Execute EMEAApiService.bat for windows or EMEAApiService.sh for unix environment under bin folder as applicable.

Step11: Once application is started, open postman place a request with following details

Url: localhost:8085/emeaapiservice/accountdetails

Method: POST

Request Body : {

"accountNumber":"2",

"sortCode":"122312"

}

Header:

Accept – application/json

Content-type – application/json

Authorization tab select type OAuth 2.0.

And click Get New Access Token.

Fill the form with following details and click Request Token button:

Auth Url: <http://localhost:8081/oauth/authorize> (in application.properties server.port=8081 is configure)

Access Token Url: <http://localhost:8085/oauth/token>

Client Id: hasim (In sample script user id “hasim” is configured)

Client Secret: secret

Grant Type : Client Credential

Step12: Select the token and click “Use Token” button request url would look like

localhost:8085/emeaapiservice/accountdetails?access\_token=a8ac00f5-6868-4752-a72b-bd1bc8ab5cfb

Then click send button. User request would be validated and proper response would be displayed.