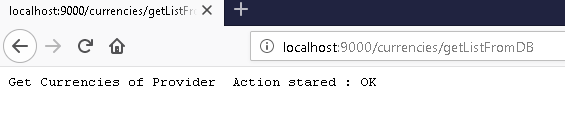
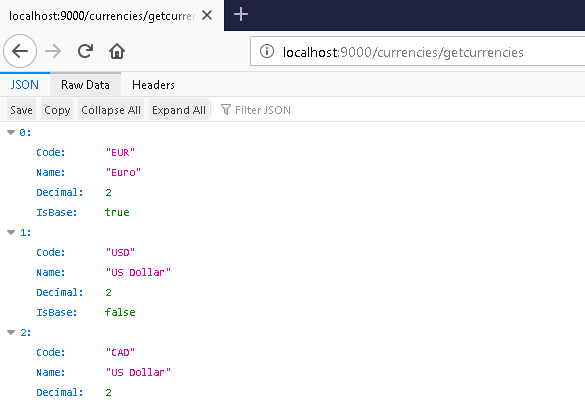
Currencies Controller Routes

**GET /currencies/getListFromDB**getListFromDB executed automatically from application timer and or manually and imports data from db to the currencies model

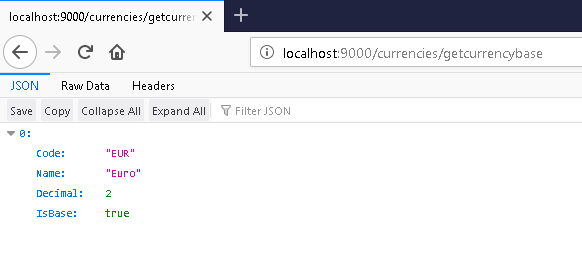


**GET /currencies/getcurrencies**

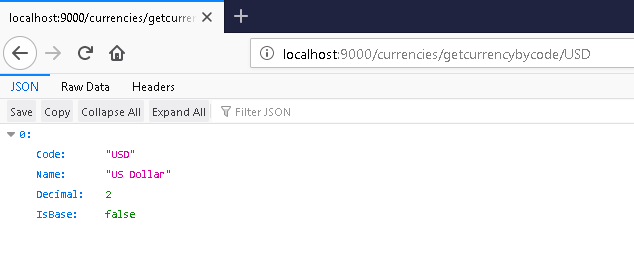
returns json of currencies returns currencies in memory of the model



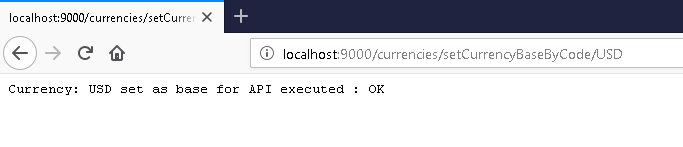
**GET /currencies/getcurrencybase**   
returns json of currency Code returns currency base from the model

  
**GET /currencies/getcurrencybycode/:code**

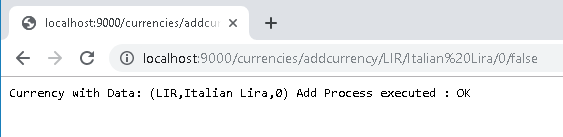
returns currency data by currency code

  
**GET /currencies/setCurrencyBaseByCode/:code**

set default currency by code set as base by currency code for the system \* and api of client not for the ratesFetcher

  
**GET /currencies/addcurrency/:currencyCode/:currencyName/:currencyDecimal/:isBase**

# insertion of a new currency inserts new currency in the model not in db to be done  
#end of currencies routes



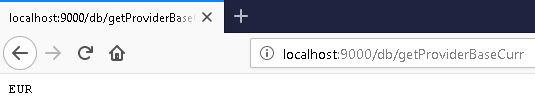
DBConnection model Routes

**GET /db/setsetDbConString**   
# sets configuration stings for the db usually form this system should be in configurations but from othe management sytems use API

Not Tested ☺

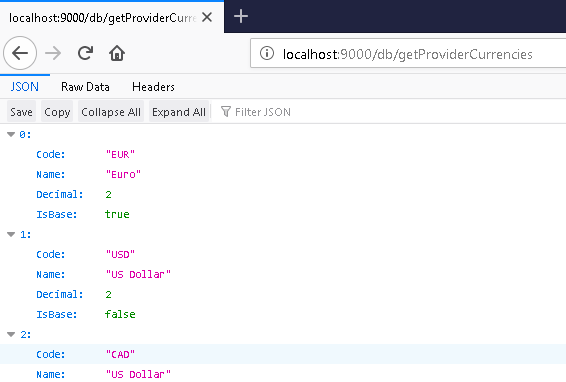
**GET /db/getProviderBaseCurr**

def getProviderBaseCurr(): String returns the base currency used from the provider

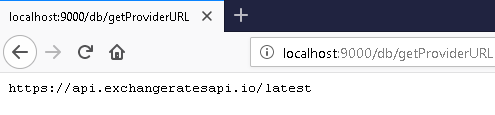


GET /db/getProviderCurrencies

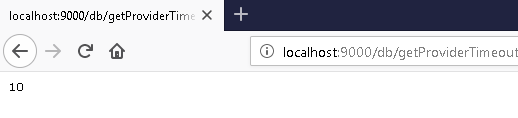
returns list of currencies offered by the provider from code momentarely should be from db read

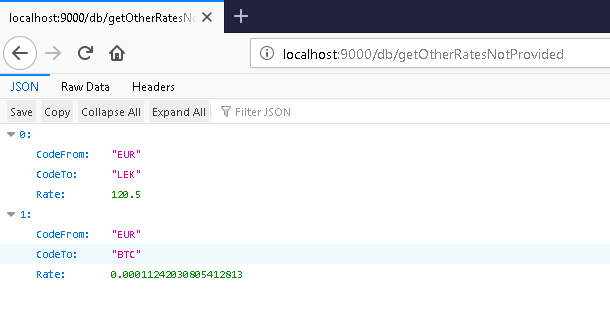


**GET /db/getProviderURL**   
returns the url of the provider (from code momentarely ) should be from db read and with other params as key of authorisation, for use from the fetcher model



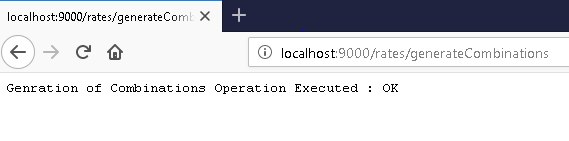
**GET /db/getProviderTimeout**   
# returns the timeout of the provider for use form the featcher as timeout for the request



**GET /db/getOtherRatesNotProvided**   
# def getOtherRatesNotProvided:List[Rate] return from db the currencies not offered from the provider  
  
for use form the calculator module in order to calculate the missing rates and not given combinations from the provider

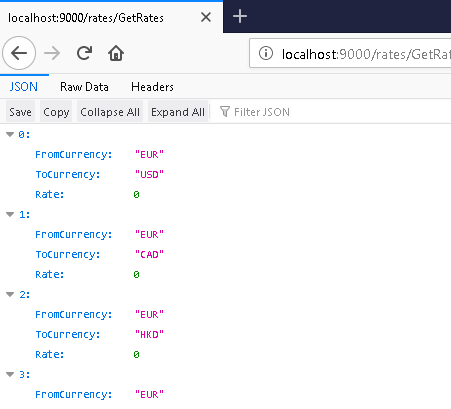
Rates Model Routes

**GET /rates/generateCombinations**   
gets the currencies from the currency controller and tests the generation of combinations

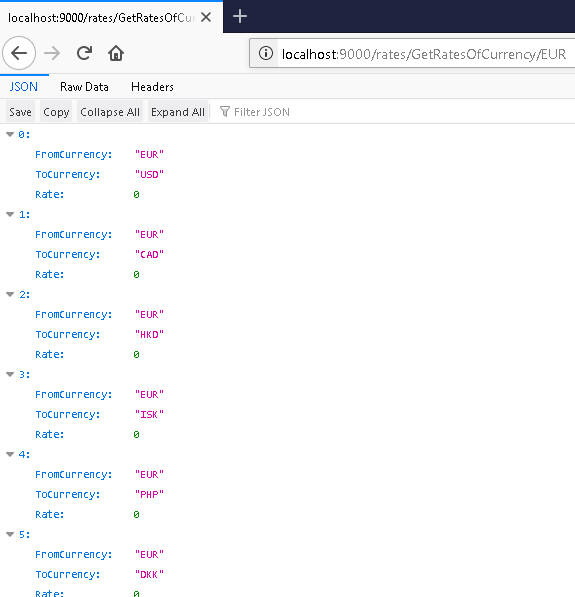


**GET /rates/GetRates**

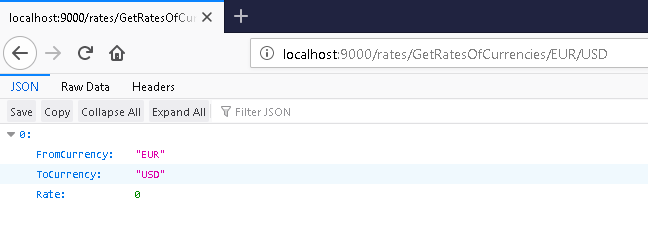
returns the test of generate



**GET /rates/GetRatesOfCurrency/:Code**   
returns filtering from generate



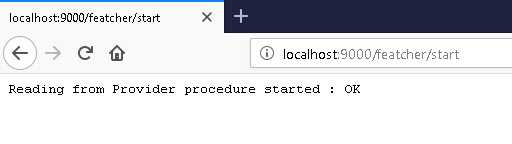
**GET /rates/GetRatesOfCurrencies/:codeFrom/:codeTo**   
returns specific from generate



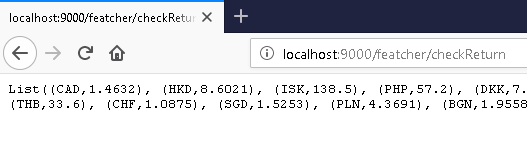
**GET /rates/AddRate/:codeFrom/:codeTo/:rate**   
add another rate from api to model  
Not Tested with screenshot ☺

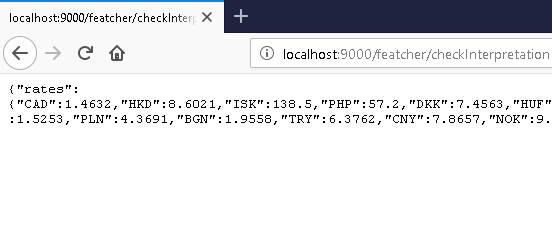
Rates Featcher Model Routes

GET /featcher/start   
# function used for starting the async process of fetch data from provider sarted in timer but can be executed manually

  
**GET /featcher/checkReturn**

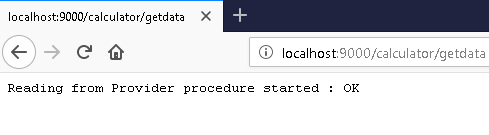
function to test the return of the provider if has return or what return it has



**GET /featcher/checkInterpretation**   
# checkes if the return could be parsed to model if enpty list than failed  
#end of ratesfetcher routes  


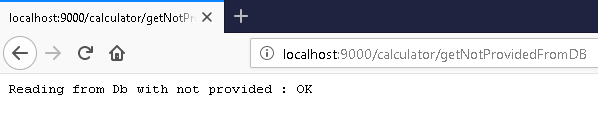
Rates Calculator Module   
**GET /calculator/getdata**

tryes to get data from the interpretation of featcher and set is to liss in combination with the provider base

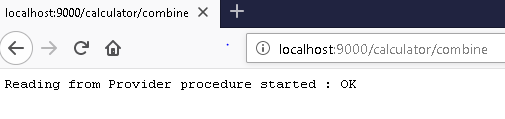


**GET /calculator/getNotProvidedFromDB**

tryes to get the rates not provided by the provider should be reading from db / sample in code

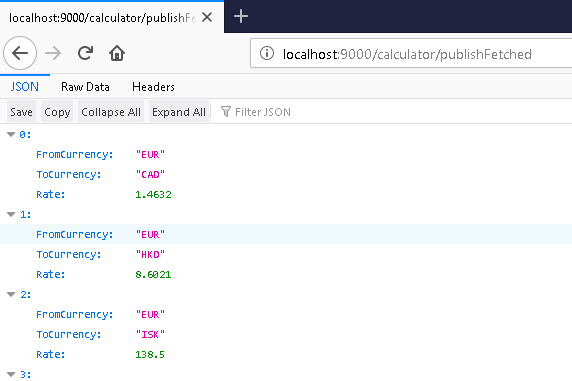


**GET /calculator/combine**   
combines the rates from provider and rates from db and calculates all rates for the combinations between currencies if a-b = x and c-d = y than b-d = y/x

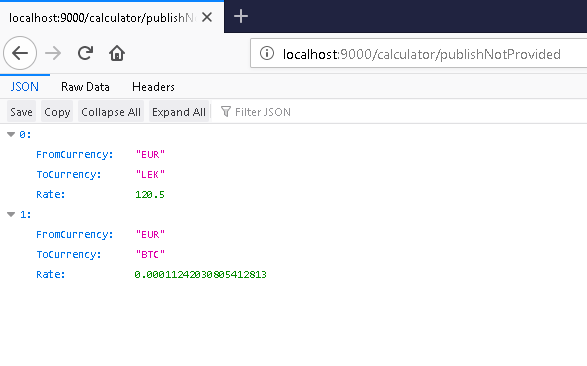


**GET /calculator/publishFetched**

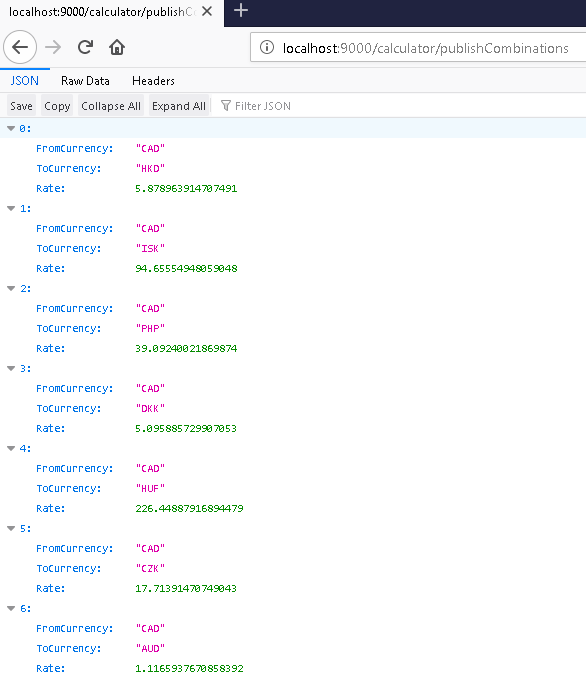
returns the fetched rates from the fetcher model



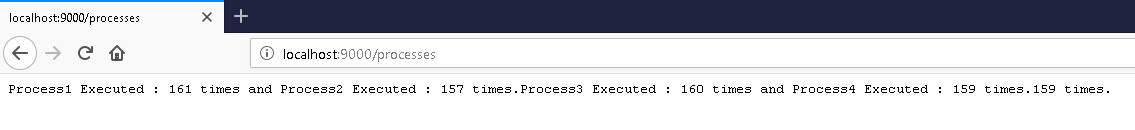
GET /calculator/publishNotProvided

returns the rates not offered from the provider but form db  


**GET /calculator/publishCombinations**

returns the combinations calculated from the combine method  
  


ProcessController routes  
 **GET /processes**

#monitors the processes executing   


**Totally in 5 processe executed automatically from the ApplicationTimed**

Client API Routes

**GET /api/SetRatesFromCalculator**

Gets the data form the memory of calculator controller and sets it in the memory of Client API Controller. Also get executed periodically from the application timed

**GET /api/latestValues**

#returns to the client the lates with the system base currency if not with EUR def currency

**GET /api/latestWithBase/:currCode**

returns to teh client the rates with a specific base offered by the client

**GET /api/latestForCouple/:currCodeFrom/:currCodeTo**

returns the specific rate of two currencies specified.

**Same as above but from the data of api model in ram.**