Authorization Data Extraction and Validation

Design & Solution Approach Document

**(1) Business Requirement:**

- Expectation is doing the Data Extraction from the auth.csv file in the JSON format

- Validation should be aua == 650000 filter it out

- Output should be res\_state\_name

- Implementation should be with 3 layers – validation, enrichment and data processing.

(2) **Technology Stack:**

Here Scala language is used for reading the auth.csv file and doing the validation and data processing. And Http4s is used webservices used and its native Blaze server is used.

(3) **Data Model:**

Model – Layout:

case class authDtls(authCode: String,

aua: String,

sa: String,

res\_state\_name: String)

case class res\_St(res\_st\_desc: String, res\_state\_name: String)

**Data Layer:**

/\*\*

\* Trait responsible for reading/loading [[authDetails]].

\*

\* @author Xavier Alwin

\*/

trait AuthDtls {

def readAuthDtl(): IO[res\_St]

}

(4) **Solution Architecture:**

In this design, the MVC (Model, View and Controller) is used for getting the solution. For the model gateway, below mentioned validation and fetching of the result (ie. Res\_State\_name) from the scala program. [Details given in Scala layer].

For the controller, Http Layer, created an HTTP interface to interact with AuthDtls

import cats.effect.IO

import io.circe.Json

import org.http4s.HttpRoutes

import io.circe.generic.auto.\_

import org.http4s.circe.CirceEntityCodec.\_

import org.http4s.dsl.Http4sDsl

object AuthRoutes {

private def errorBody(message: Message) = Json.obj(

("message", Json.fromString(message))

)

def routes(authDtls: AuthDtls): HttpRoutes[IO] = {

val dsl = new Http4sDsl[IO]{}

import dsl.\_

HttpRoutes.of[IO] {

case \_ @ GET -> Root / "getResSt" =>

authDtls.readAuthDtl().flatMap(resSt => Ok(resSt))

}

}

}

**App Launcher:**

One step left to launch a working application. In http4s routes do not handle any requests, until you explicitly associate them with running server. So here is what we need to do in order to make books REST API live:

import cats.effect.{ExitCode, IO, IOApp}

import org.http4s.server.Router

import org.http4s.implicits.\_

import org.http4s.server.blaze.\_

import cats.implicits.\_

object Main extends IOApp {

private val authDtls: AuthDtls = new authCls

val httpRoutes = Router[IO](

"/" -> AuthRoutes.routes(authDtls)

).orNotFound

override def run(args: List[String]): IO[ExitCode] = {

BlazeServerBuilder[IO]

.bindHttp(9000, "0.0.0.0")

.withHttpApp(httpRoutes)

.serve

.compile

.drain

.as(ExitCode.Success)

}

}

In order to test how the application handles http request, we need to run the Main object. Now we can execute several test requests:

>curl -X GET localhost:9000/getResSt \

>

>[{"res\_state\_name":"Andhra Pradesh"}

{"res\_state\_name":"Andhra Pradesh"}

]

(5) **Scala Layer:**

(a) Functionality implemented in Scala:

Here reading the auth.csv file and doing below validations:

val trans\_file = Source.fromFile(fileName).getLines().drop(1)

val transactions = trans\_file.map(lines => lines.split(",")).

map(split => authDtls(split(0),split(2),split(3),split(128))).toList

val filterauth = transactions.filter(line => line.aua != "650000" && line.sa != "SERVICEMON")

val filtereqauasa = filterauth.filter(line => line.aua == line.sa)

val res\_state = filtereqauasa.map(\_.res\_state\_name)

Finally writing the result into the JSON:

val gson = new Gson

val jsonString

for (state\_nm <- res\_state)

{

val state\_name = res\_St(state\_nm.toString)

jsonString = gson.toJson(state\_name)

}

jsonString

(6) **Assumption:**

1. In the given requirement document (“The problem Statement”), it’s expecting to check with aua and asa field, but the values in the asa field is completely different wit aua, so assumed its not asa but it’s sa field from auth.csv file.

2. In the Enrichment, the condition given as “if  sa is SERVICEMON then replace sa with aua” – and in another condition for sa value should not be as SERVICEMON for fetching res\_state\_name, so both condition are contradicting, so ignored replace sa with aua condition.