package com.twitter.tweetypie.federated.columns

import com.twitter.stitch.Stitch

import com.twitter.strato.access.Access.LdapGroup

import com.twitter.strato.catalog.Fetch

import com.twitter.strato.catalog.OpMetadata

import com.twitter.strato.config.AnyOf

import com.twitter.strato.config.ContactInfo

import com.twitter.strato.config.FromColumns

import com.twitter.strato.config.Has

import com.twitter.strato.config.Path

import com.twitter.strato.config.Policy

import com.twitter.strato.data.Conv

import com.twitter.strato.data.Description.PlainText

import com.twitter.strato.data.Lifecycle.Production

import com.twitter.strato.fed.StratoFed

import com.twitter.strato.response.Err

import com.twitter.strato.thrift.ScroogeConv

import com.twitter.tweetypie.UserId

import com.twitter.tweetypie.thriftscala.federated.GetStoredTweetsByUserView

import com.twitter.tweetypie.thriftscala.federated.GetStoredTweetsByUserResponse

import com.twitter.tweetypie.{thriftscala => thrift}

import com.twitter.util.Future

class GetStoredTweetsByUserColumn(

handler: thrift.GetStoredTweetsByUserRequest => Future[thrift.GetStoredTweetsByUserResult])

extends StratoFed.Column(GetStoredTweetsByUserColumn.Path)

with StratoFed.Fetch.Stitch {

override val contactInfo: ContactInfo = TweetypieContactInfo

override val metadata: OpMetadata = OpMetadata(

lifecycle = Some(Production),

description =

Some(PlainText("Fetches hydrated Tweets for a particular User regardless of Tweet state."))

)

override val policy: Policy = AnyOf(

Seq(

FromColumns(Set(Path("tweetypie/data-provider/storedTweets.User"))),

Has(LdapGroup("tweetypie-team"))

))

override type Key = UserId

override type View = GetStoredTweetsByUserView

override type Value = GetStoredTweetsByUserResponse

override val keyConv: Conv[Key] = Conv.ofType

override val viewConv: Conv[View] = ScroogeConv.fromStruct[GetStoredTweetsByUserView]

override val valueConv: Conv[Value] = ScroogeConv.fromStruct[GetStoredTweetsByUserResponse]

override def fetch(key: Key, view: View): Stitch[Result[Value]] = {

val request = thrift.GetStoredTweetsByUserRequest(

userId = key,

options = Some(

thrift.GetStoredTweetsByUserOptions(

bypassVisibilityFiltering = view.bypassVisibilityFiltering,

setForUserId = view.setForUserId,

startTimeMsec = view.startTimeMsec,

endTimeMsec = view.endTimeMsec,

cursor = view.cursor,

startFromOldest = view.startFromOldest,

additionalFieldIds = view.additionalFieldIds

))

)

Stitch

.callFuture(handler(request))

.map { result =>

Fetch.Result.found(

GetStoredTweetsByUserResponse(

storedTweets = result.storedTweets,

cursor = result.cursor

))

}

.rescue {

case \_ => Stitch.exception(Err(Err.Internal))

}

}

}

object GetStoredTweetsByUserColumn {

val Path = "tweetypie/internal/getStoredTweets.User"

}