package com.twitter.tweetypie

package federated.columns

import com.twitter.stitch.Stitch

import com.twitter.strato.catalog.OpMetadata

import com.twitter.strato.config.ContactInfo

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.opcontext.OpContext

import com.twitter.strato.thrift.ScroogeConv

import com.twitter.tweetypie.federated.context.GetRequestContext

import com.twitter.tweetypie.federated.context.RequestContext

import com.twitter.tweetypie.thriftscala.{graphql => gql}

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

class UnretweetColumn(

unretweet: thrift.UnretweetRequest => Future[thrift.UnretweetResult],

getRequestContext: GetRequestContext,

) extends StratoFed.Column("tweetypie/unretweet.Tweet")

with StratoFed.Execute.StitchWithContext

with StratoFed.HandleDarkRequests {

override val policy: Policy = AccessPolicy.TweetMutationCommonAccessPolicies

// It's acceptable to retry or reapply an unretweet operation,

// as multiple calls result in the same end state.

override val isIdempotent: Boolean = true

override type Arg = gql.UnretweetRequest

override type Result = gql.UnretweetResponseWithSubqueryPrefetchItems

override val argConv: Conv[Arg] = ScroogeConv.fromStruct

override val resultConv: Conv[Result] = ScroogeConv.fromStruct

override val contactInfo: ContactInfo = TweetypieContactInfo

override val metadata: OpMetadata =

OpMetadata(

Some(Production),

Some(PlainText("Removes any retweets by the calling user of the given source tweet.")))

override def execute(gqlRequest: Arg, opContext: OpContext): Stitch[Result] = {

val ctx: RequestContext = getRequestContext(opContext)

val req = thrift.UnretweetRequest(

ctx.twitterUserId,

gqlRequest.sourceTweetId,

)

val stitchUnretweet = handleDarkRequest(opContext)(

light = Stitch.callFuture(unretweet(req)),

// For dark requests, we don't want to send traffic to tweetypie.

// Since the response is the same regardless of the request, we take a no-op

// action instead.

dark = Stitch.value(thrift.UnretweetResult(state = thrift.TweetDeleteState.Ok))

)

stitchUnretweet.map { \_ =>

gql.UnretweetResponseWithSubqueryPrefetchItems(

data = Some(gql.UnretweetResponse(Some(gqlRequest.sourceTweetId)))

)

}

}

}

object UnretweetColumn {

val Path = "tweetypie/unretweet.Tweet"

}