package com.twitter.unified\_user\_actions.enricher.hydrator

import com.twitter.dynmap.DynMap

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.graphql.thriftscala.AuthHeaders

import com.twitter.graphql.thriftscala.Authentication

import com.twitter.graphql.thriftscala.Document

import com.twitter.graphql.thriftscala.GraphQlRequest

import com.twitter.graphql.thriftscala.GraphqlExecutionService

import com.twitter.graphql.thriftscala.Variables

import com.twitter.unified\_user\_actions.enricher.ImplementationException

import com.twitter.unified\_user\_actions.enricher.graphql.GraphqlRspParser

import com.twitter.unified\_user\_actions.enricher.hcache.LocalCache

import com.twitter.unified\_user\_actions.enricher.internal.thriftscala.EnrichmentEnvelop

import com.twitter.unified\_user\_actions.enricher.internal.thriftscala.EnrichmentIdType

import com.twitter.unified\_user\_actions.enricher.internal.thriftscala.EnrichmentInstruction

import com.twitter.unified\_user\_actions.enricher.internal.thriftscala.EnrichmentKey

import com.twitter.unified\_user\_actions.thriftscala.AuthorInfo

import com.twitter.unified\_user\_actions.thriftscala.Item

import com.twitter.util.Future

class DefaultHydrator(

cache: LocalCache[EnrichmentKey, DynMap],

graphqlClient: GraphqlExecutionService.FinagledClient,

scopedStatsReceiver: StatsReceiver = NullStatsReceiver)

extends AbstractHydrator(scopedStatsReceiver) {

private def constructGraphqlReq(

enrichmentKey: EnrichmentKey

): GraphQlRequest =

enrichmentKey.keyType match {

case EnrichmentIdType.TweetId =>

GraphQlRequest(

// see go/graphiql/M5sHxua-RDiRtTn48CAhng

document = Document.DocumentId("M5sHxua-RDiRtTn48CAhng"),

operationName = Some("TweetHydration"),

variables = Some(

Variables.JsonEncodedVariables(s"""{"rest\_id": "${enrichmentKey.id}"}""")

),

authentication = Authentication.AuthHeaders(

AuthHeaders()

)

)

case \_ =>

throw new ImplementationException(

s"Missing implementation for hydration of type ${enrichmentKey.keyType}")

}

private def hydrateAuthorInfo(item: Item.TweetInfo, authorId: Option[Long]): Item.TweetInfo = {

item.tweetInfo.actionTweetAuthorInfo match {

case Some(\_) => item

case \_ =>

item.copy(tweetInfo = item.tweetInfo.copy(

actionTweetAuthorInfo = Some(AuthorInfo(authorId = authorId))

))

}

}

override protected def safelyHydrate(

instruction: EnrichmentInstruction,

key: EnrichmentKey,

envelop: EnrichmentEnvelop

): Future[EnrichmentEnvelop] = {

instruction match {

case EnrichmentInstruction.TweetEnrichment =>

val dynMapFuture = cache.getOrElseUpdate(key) {

graphqlClient

.graphql(constructGraphqlReq(enrichmentKey = key))

.map { body =>

body.response.flatMap { r =>

GraphqlRspParser.toDynMapOpt(r)

}.get

}

}

dynMapFuture.map(map => {

val authorIdOpt =

map.getLongOpt("data.tweet\_result\_by\_rest\_id.result.core.user.legacy.id\_str")

val hydratedEnvelop = envelop.uua.item match {

case item: Item.TweetInfo =>

envelop.copy(uua = envelop.uua.copy(item = hydrateAuthorInfo(item, authorIdOpt)))

case \_ => envelop

}

hydratedEnvelop

})

case \_ => Future.value(envelop)

}

}

}