package com.twitter.tweetypie.repository

import com.twitter.container.thriftscala.MaterializeAsTweetFieldsRequest

import com.twitter.container.thriftscala.MaterializeAsTweetRequest

import com.twitter.container.{thriftscala => ccs}

import com.twitter.stitch.SeqGroup

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.Return

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

import com.twitter.tweetypie.backends

import com.twitter.tweetypie.thriftscala.GetTweetFieldsResult

import com.twitter.tweetypie.thriftscala.GetTweetResult

import com.twitter.util.Future

import com.twitter.util.Try

/\*\*

\* A special kind of tweet is that, when [[tp.Tweet.underlyingCreativesContainerId]] is presented.

\* tweetypie will delegate hydration of this tweet to creatives container service.

\*/

object CreativesContainerMaterializationRepository {

type GetTweetType =

(ccs.MaterializeAsTweetRequest, Option[tp.GetTweetOptions]) => Stitch[tp.GetTweetResult]

type GetTweetFieldsType =

(

ccs.MaterializeAsTweetFieldsRequest,

tp.GetTweetFieldsOptions

) => Stitch[tp.GetTweetFieldsResult]

def apply(

materializeAsTweet: backends.CreativesContainerService.MaterializeAsTweet

): GetTweetType = {

case class RequestGroup(opts: Option[tp.GetTweetOptions])

extends SeqGroup[ccs.MaterializeAsTweetRequest, tp.GetTweetResult] {

override protected def run(

keys: Seq[MaterializeAsTweetRequest]

): Future[Seq[Try[GetTweetResult]]] =

materializeAsTweet(ccs.MaterializeAsTweetRequests(keys, opts)).map {

res: Seq[GetTweetResult] => res.map(Return(\_))

}

}

(request, options) => Stitch.call(request, RequestGroup(options))

}

def materializeAsTweetFields(

materializeAsTweetFields: backends.CreativesContainerService.MaterializeAsTweetFields

): GetTweetFieldsType = {

case class RequestGroup(opts: tp.GetTweetFieldsOptions)

extends SeqGroup[ccs.MaterializeAsTweetFieldsRequest, tp.GetTweetFieldsResult] {

override protected def run(

keys: Seq[MaterializeAsTweetFieldsRequest]

): Future[Seq[Try[GetTweetFieldsResult]]] =

materializeAsTweetFields(ccs.MaterializeAsTweetFieldsRequests(keys, opts)).map {

res: Seq[GetTweetFieldsResult] => res.map(Return(\_))

}

}

(request, options) => Stitch.call(request, RequestGroup(options))

}

}