package com.twitter.tweetypie

package repository

import com.twitter.expandodo.thriftscala.\_

import com.twitter.stitch.SeqGroup

import com.twitter.stitch.Stitch

import com.twitter.stitch.compat.LegacySeqGroup

import com.twitter.tweetypie.backends.Expandodo

object CardUsersRepository {

type CardUri = String

type Type = (CardUri, Context) => Stitch[Option[Set[UserId]]]

case class Context(perspectiveUserId: UserId) extends AnyVal

case class GetUsersGroup(perspectiveId: UserId, getCardUsers: Expandodo.GetCardUsers)

extends SeqGroup[CardUri, GetCardUsersResponse] {

protected override def run(keys: Seq[CardUri]): Future[Seq[Try[GetCardUsersResponse]]] =

LegacySeqGroup.liftToSeqTry(

getCardUsers(

GetCardUsersRequests(

requests = keys.map(k => GetCardUsersRequest(k)),

perspectiveUserId = Some(perspectiveId)

)

).map(\_.responses)

)

}

def apply(getCardUsers: Expandodo.GetCardUsers): Type =

(cardUri, ctx) =>

Stitch.call(cardUri, GetUsersGroup(ctx.perspectiveUserId, getCardUsers)).map { resp =>

val authorUserIds = resp.authorUserIds.map(\_.toSet)

val siteUserIds = resp.siteUserIds.map(\_.toSet)

if (authorUserIds.isEmpty) {

siteUserIds

} else if (siteUserIds.isEmpty) {

authorUserIds

} else {

Some(authorUserIds.get ++ siteUserIds.get)

}

}

}