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

sealed trait Card2Key {

def toCard2Request: Card2Request

}

final case class UrlCard2Key(url: String) extends Card2Key {

override def toCard2Request: Card2Request =

Card2Request(`type` = Card2RequestType.ByUrl, url = Some(url))

}

final case class ImmediateValuesCard2Key(values: Seq[Card2ImmediateValue], tweetId: TweetId)

extends Card2Key {

override def toCard2Request: Card2Request =

Card2Request(

`type` = Card2RequestType.ByImmediateValues,

immediateValues = Some(values),

statusId = Some(tweetId)

)

}

object Card2Repository {

type Type = (Card2Key, Card2RequestOptions) => Stitch[Card2]

def apply(getCards2: Expandodo.GetCards2, maxRequestSize: Int): Type = {

case class RequestGroup(opts: Card2RequestOptions) extends SeqGroup[Card2Key, Option[Card2]] {

override def run(keys: Seq[Card2Key]): Future[Seq[Try[Option[Card2]]]] =

LegacySeqGroup.liftToSeqTry(

getCards2((keys.map(\_.toCard2Request), opts)).map { res =>

res.responsesCode match {

case Card2ResponsesCode.Ok =>

res.responses.map(\_.card)

case \_ =>

// treat all other failure cases as card-not-found

Seq.fill(keys.size)(None)

}

}

)

override def maxSize: Int = maxRequestSize

}

(card2Key, opts) =>

Stitch

.call(card2Key, RequestGroup(opts))

.lowerFromOption()

}

}