package com.twitter.tweetypie

package handler

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.CardReferenceUriExtractor

import com.twitter.tweetypie.core.NonTombstone

import com.twitter.tweetypie.core.Tombstone

import com.twitter.tweetypie.repository.CardUsersRepository

import com.twitter.tweetypie.repository.CardUsersRepository.Context

import com.twitter.tweetypie.thriftscala.CardReference

/\*\*

\* Finds a set of UserId that may be mentioned when replying to a tweet that has a card.

\*

\* Replies created without 'auto\_populate\_reply\_metadata' include both 'site' and 'author' users to

\* have a more exhaustive list of mentions to match against. This is needed because iOS and Android

\* have had different implementations client-side for years.

\*/

object CardUsersFinder {

case class Request(

cardReference: Option[CardReference],

urls: Seq[String],

perspectiveUserId: UserId) {

val uris: Seq[String] = cardReference match {

case Some(CardReferenceUriExtractor(cardUri)) =>

cardUri match {

case NonTombstone(uri) => Seq(uri)

case Tombstone => Nil

}

case \_ => urls

}

val context: CardUsersRepository.Context = Context(perspectiveUserId)

}

type Type = Request => Stitch[Set[UserId]]

/\*\*

\* From a card-related arguments in [[Request]] select the set of user ids associated with the

\* card.

\*

\* Note that this uses the same "which card do I use?" logic from Card2Hydrator which

\* prioritizes CardReferenceUri and then falls back to the last resolvable (non-None) url entity.

\*/

def apply(cardUserRepo: CardUsersRepository.Type): Type =

request =>

Stitch

.traverse(request.uris) { uri => cardUserRepo(uri, request.context) }

// select the last, non-None Set of users ids

.map(r => r.flatten.reverse.headOption.getOrElse(Set.empty))

}