package com.twitter.tweetypie

package hydrator

import com.twitter.expandodo.thriftscala.Card

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.\_

object CardHydrator {

type Type = ValueHydrator[Option[Seq[Card]], Ctx]

case class Ctx(

urlEntities: Seq[UrlEntity],

mediaEntities: Seq[MediaEntity],

underlyingTweetCtx: TweetCtx)

extends TweetCtx.Proxy

val hydratedField: FieldByPath = fieldByPath(Tweet.CardsField)

private[this] val partialResult = ValueState.partial(None, hydratedField)

def apply(repo: CardRepository.Type): Type = {

def getCards(url: String): Stitch[Seq[Card]] =

repo(url).handle { case NotFound => Nil }

ValueHydrator[Option[Seq[Card]], Ctx] { (\_, ctx) =>

val urls = ctx.urlEntities.map(\_.url)

Stitch.traverse(urls)(getCards \_).liftToTry.map {

case Return(cards) =>

// even though we are hydrating a type of Option[Seq[Card]], we only

// ever return at most one card, and always the last one.

val res = cards.flatten.lastOption.toSeq

if (res.isEmpty) ValueState.UnmodifiedNone

else ValueState.modified(Some(res))

case \_ => partialResult

}

}.onlyIf { (curr, ctx) =>

curr.isEmpty &&

ctx.tweetFieldRequested(Tweet.CardsField) &&

!ctx.isRetweet &&

ctx.mediaEntities.isEmpty

}

}

}