package com.twitter.tweetypie

package hydrator

import com.twitter.finagle.stats.NullStatsReceiver

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.\_

/\*\*

\* Hydrates the "directedAtUser" field on the tweet. This hydrators uses one of two paths depending

\* if DirectedAtUserMetadata is present:

\*

\* 1. If DirectedAtUserMetadata exists, we use metadata.userId.

\* 2. If DirectedAtUserMetadata does not exist, we use the User screenName from the mention starting

\* at index 0 if the tweet also has a reply. Creation of a "reply to user" for

\* leading @mentions is controlled by PostTweetRequest.enableTweetToNarrowcasting

\*/

object DirectedAtHydrator {

type Type = ValueHydrator[Option[DirectedAtUser], Ctx]

case class Ctx(

mentions: Seq[MentionEntity],

metadata: Option[DirectedAtUserMetadata],

underlyingTweetCtx: TweetCtx)

extends TweetCtx.Proxy {

val directedAtScreenName: Option[String] =

mentions.headOption.filter(\_.fromIndex == 0).map(\_.screenName)

}

val hydratedField: FieldByPath =

fieldByPath(Tweet.CoreDataField, TweetCoreData.DirectedAtUserField)

def once(h: Type): Type =

TweetHydration.completeOnlyOnce(

hydrationType = HydrationType.DirectedAt,

hydrator = h

)

private val partial = ValueState.partial(None, hydratedField)

def apply(repo: UserIdentityRepository.Type, stats: StatsReceiver = NullStatsReceiver): Type = {

val withMetadata = stats.counter("with\_metadata")

val noScreenName = stats.counter("no\_screen\_name")

val withoutMetadata = stats.counter("without\_metadata")

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

ctx.metadata match {

case Some(DirectedAtUserMetadata(Some(uid))) =>

// 1a. new approach of relying exclusively on directed-at metadata if it exists and has a user id

withMetadata.incr()

repo(UserKey.byId(uid)).liftToTry.map {

case Return(u) =>

ValueState.modified(Some(DirectedAtUser(u.id, u.screenName)))

case Throw(NotFound) =>

// If user is not found, fallback to directedAtScreenName

ctx.directedAtScreenName

.map { screenName => ValueState.modified(Some(DirectedAtUser(uid, screenName))) }

.getOrElse {

// This should never happen, but let's make sure with a counter

noScreenName.incr()

ValueState.UnmodifiedNone

}

case Throw(\_) => partial

}

case Some(DirectedAtUserMetadata(None)) =>

withMetadata.incr()

// 1b. new approach of relying exclusively on directed-at metadata if it exists and has no userId

ValueState.StitchUnmodifiedNone

case None =>

// 2. when DirectedAtUserMetadata not present, look for first leading mention when has reply

withoutMetadata.incr()

val userKey = ctx.directedAtScreenName

.filter(\_ => ctx.isReply)

.map(UserKey.byScreenName)

val results = userKey.map(repo.apply).getOrElse(Stitch.NotFound)

results.liftToTry.map {

case Return(u) => ValueState.modified(Some(DirectedAtUser(u.id, u.screenName)))

case Throw(NotFound) => ValueState.UnmodifiedNone

case Throw(\_) => partial

}

}

}.onlyIf((curr, \_) => curr.isEmpty)

}

}