package com.twitter.tweetypie

package hydrator

import com.twitter.stitch.NotFound

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.\_

object MentionEntitiesHydrator {

type Type = ValueHydrator[Seq[MentionEntity], TweetCtx]

def once(h: MentionEntityHydrator.Type): Type =

TweetHydration.completeOnlyOnce(

queryFilter = queryFilter,

hydrationType = HydrationType.Mentions,

hydrator = h.liftSeq

)

def queryFilter(opts: TweetQuery.Options): Boolean =

opts.include.tweetFields.contains(Tweet.MentionsField.id)

}

object MentionEntityHydrator {

type Type = ValueHydrator[MentionEntity, TweetCtx]

val hydratedField: FieldByPath = fieldByPath(Tweet.MentionsField)

def apply(repo: UserIdentityRepository.Type): Type =

ValueHydrator[MentionEntity, TweetCtx] { (entity, \_) =>

repo(UserKey(entity.screenName)).liftToTry.map {

case Return(user) => ValueState.delta(entity, update(entity, user))

case Throw(NotFound) => ValueState.unmodified(entity)

case Throw(\_) => ValueState.partial(entity, hydratedField)

}

// only hydrate mention if userId or name is empty

}.onlyIf((entity, \_) => entity.userId.isEmpty || entity.name.isEmpty)

/\*\*

\* Updates a MentionEntity using the given user data.

\*/

def update(entity: MentionEntity, userIdent: UserIdentity): MentionEntity =

entity.copy(

screenName = userIdent.screenName,

userId = Some(userIdent.id),

name = Some(userIdent.realName)

)

}