package com.twitter.tweetypie

package hydrator

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.\_

/\*\*

\* Ensures that the tweet's author and source tweet's author (if retweet) are visible to the

\* viewing user - ctx.opts.forUserId - when enforceVisibilityFiltering is true.

\* If either of these users is not visible then a FilteredState.Suppress will be returned.

\*

\* Note: blocking relationship is NOT checked here, this means if viewing user `forUserId` is blocked

\* by either the tweet's author or source tweet's author, this will not filter out the tweet.

\*/

object TweetAuthorVisibilityHydrator {

type Type = ValueHydrator[Unit, TweetCtx]

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

ValueHydrator[Unit, TweetCtx] { (\_, ctx) =>

val ids = Seq(ctx.userId) ++ ctx.sourceUserId

val keys = ids.map(id => toRepoQuery(id, ctx))

Stitch

.traverse(keys)(repo.apply).flatMap { responses =>

val fs: Option[FilteredState.Unavailable] = responses.flatten.headOption

fs match {

case Some(fs: FilteredState.Unavailable) => Stitch.exception(fs)

case None => ValueState.StitchUnmodifiedUnit

}

}

}.onlyIf((\_, ctx) => ctx.opts.enforceVisibilityFiltering)

private def toRepoQuery(userId: UserId, ctx: TweetCtx) =

UserVisibilityRepository.Query(

UserKey(userId),

ctx.opts.forUserId,

ctx.tweetId,

ctx.isRetweet,

ctx.opts.isInnerQuotedTweet,

Some(ctx.opts.safetyLevel))

}