package com.twitter.tweetypie

package repository

import com.twitter.spam.rtf.thriftscala.{SafetyLevel => ThriftSafetyLevel}

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.VisibilityResultToFilteredState.toFilteredState

import com.twitter.visibility.configapi.configs.VisibilityDeciderGates

import com.twitter.visibility.interfaces.tweets.QuotedTweetVisibilityLibrary

import com.twitter.visibility.interfaces.tweets.QuotedTweetVisibilityRequest

import com.twitter.visibility.interfaces.tweets.TweetAndAuthor

import com.twitter.visibility.models.SafetyLevel

import com.twitter.visibility.models.ViewerContext

/\*\*

\* This repository handles visibility filtering of inner quoted tweets

\* based on relationships between the inner and outer tweets. This is

\* additive to independent visibility filtering of the inner tweet.

\*/

object QuotedTweetVisibilityRepository {

type Type = Request => Stitch[Option[FilteredState]]

case class Request(

outerTweetId: TweetId,

outerAuthorId: UserId,

innerTweetId: TweetId,

innerAuthorId: UserId,

viewerId: Option[UserId],

safetyLevel: ThriftSafetyLevel)

def apply(

quotedTweetVisibilityLibrary: QuotedTweetVisibilityLibrary.Type,

visibilityDeciderGates: VisibilityDeciderGates,

): QuotedTweetVisibilityRepository.Type = { request: Request =>

quotedTweetVisibilityLibrary(

QuotedTweetVisibilityRequest(

quotedTweet = TweetAndAuthor(request.innerTweetId, request.innerAuthorId),

outerTweet = TweetAndAuthor(request.outerTweetId, request.outerAuthorId),

ViewerContext.fromContextWithViewerIdFallback(request.viewerId),

safetyLevel = SafetyLevel.fromThrift(request.safetyLevel)

)

).map(visibilityResult =>

toFilteredState(

visibilityResult = visibilityResult,

disableLegacyInterstitialFilteredReason =

visibilityDeciderGates.disableLegacyInterstitialFilteredReason()))

}

}