package com.twitter.timelineranker.common

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.IndividualRequestTimeoutException

import com.twitter.search.earlybird.thriftscala.ThriftSearchResult

import com.twitter.servo.util.FutureArrow

import com.twitter.timelineranker.core.CandidateEnvelope

import com.twitter.timelineranker.core.HydratedTweets

import com.twitter.timelineranker.model.PartiallyHydratedTweet

import com.twitter.timelines.model.tweet.HydratedTweet

import com.twitter.util.Future

object TweetHydrationTransform {

val EmptyHydratedTweets: HydratedTweets =

HydratedTweets(Seq.empty[HydratedTweet], Seq.empty[HydratedTweet])

val EmptyHydratedTweetsFuture: Future[HydratedTweets] = Future.value(EmptyHydratedTweets)

}

object CandidateTweetHydrationTransform extends TweetHydrationTransform {

override def apply(envelope: CandidateEnvelope): Future[CandidateEnvelope] = {

hydrate(

searchResults = envelope.searchResults,

envelope = envelope

).map { tweets => envelope.copy(hydratedTweets = tweets) }

}

}

object SourceTweetHydrationTransform extends TweetHydrationTransform {

override def apply(envelope: CandidateEnvelope): Future[CandidateEnvelope] = {

hydrate(

searchResults = envelope.sourceSearchResults,

envelope = envelope

).map { tweets => envelope.copy(sourceHydratedTweets = tweets) }

}

}

// Static IRTE to indicate timeout in tweet hydrator. Placeholder timeout duration of 0 millis is used

// since we are only concerned with the source of the exception.

object TweetHydrationTimeoutException extends IndividualRequestTimeoutException(0.millis) {

serviceName = "tweetHydrator"

}

/\*\*

\* Transform which hydrates tweets in the CandidateEnvelope

\*\*/

trait TweetHydrationTransform extends FutureArrow[CandidateEnvelope, CandidateEnvelope] {

import TweetHydrationTransform.\_

protected def hydrate(

searchResults: Seq[ThriftSearchResult],

envelope: CandidateEnvelope

): Future[HydratedTweets] = {

if (searchResults.nonEmpty) {

Future.value(

HydratedTweets(searchResults.map(PartiallyHydratedTweet.fromSearchResult))

)

} else {

EmptyHydratedTweetsFuture

}

}

}