package com.twitter.tweetypie

package federated

package promotedcontent

import com.twitter.ads.callback.thriftscala.EngagementRequest

import com.twitter.ads.internal.pcl.service.CallbackPromotedContentLogger

import com.twitter.ads.internal.pcl.strato\_adaptor.PromotedContentInputProvider

import com.twitter.ads.internal.pcl.thriftscala.PromotedContentInput

import com.twitter.adserver.thriftscala.EngagementType

import com.twitter.util.Future

object TweetPromotedContentLogger {

sealed abstract class TweetEngagementType(val engagementType: EngagementType)

case object TweetEngagement extends TweetEngagementType(EngagementType.Send)

case object ReplyEngagement extends TweetEngagementType(EngagementType.Reply)

case object RetweetEngagement extends TweetEngagementType(EngagementType.Retweet)

type Type = (EngagementRequest, TweetEngagementType, Boolean) => Future[Unit]

private[this] val TwitterContext =

com.twitter.context.TwitterContext(com.twitter.tweetypie.TwitterContextPermit)

def apply(callbackPromotedContentLogger: CallbackPromotedContentLogger): Type =

(

engagementRequest: EngagementRequest,

tweetEngagementType: TweetEngagementType,

isDark: Boolean

) => {

val pci: PromotedContentInput =

PromotedContentInputProvider(TwitterContext, engagementRequest)

// The real logging is fire-and-forget, so we can create the Future and ignore returning it.

Future.when(!isDark) {

callbackPromotedContentLogger.logNonTrendEngagement(

pci,

tweetEngagementType.engagementType,

pci.impressionId)

}

}

}