package com.twitter.frigate.pushservice.predicate.ntab\_caret\_fatigue

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.predicate.FatiguePredicate

import com.twitter.frigate.pushservice.predicate.CaretFeedbackHistoryFilter

import com.twitter.frigate.pushservice.predicate.{

TargetNtabCaretClickFatiguePredicate => CommonNtabCaretClickFatiguePredicate

}

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.frigate.pushservice.params.PushParams

import com.twitter.frigate.thriftscala.NotificationDisplayLocation

import com.twitter.frigate.thriftscala.{CommonRecommendationType => CRT}

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.hermit.predicate.Predicate

import com.twitter.notificationservice.thriftscala.CaretFeedbackDetails

import com.twitter.util.Duration

import com.twitter.util.Future

object RecTypeNtabCaretClickFatiguePredicate {

val defaultName = "RecTypeNtabCaretClickFatiguePredicateForCandidate"

private def candidateFatiguePredicate(

genericTypeCategories: Seq[String],

crts: Set[CRT]

)(

implicit stats: StatsReceiver

): NamedPredicate[

PushCandidate

] = {

val name = "f1TriggeredCRTBasedFatiguePredciate"

val scopedStats = stats.scope(s"predicate\_$name")

Predicate

.fromAsync { candidate: PushCandidate =>

if (candidate.frigateNotification.notificationDisplayLocation == NotificationDisplayLocation.PushToMobileDevice) {

if (candidate.target.params(PushParams.EnableFatigueNtabCaretClickingParam)) {

NtabCaretClickContFnFatiguePredicate

.ntabCaretClickContFnFatiguePredicates(

filterHistory = FatiguePredicate.recTypesOnlyFilter(crts),

filterCaretFeedbackHistory =

CaretFeedbackHistoryFilter.caretFeedbackHistoryFilter(genericTypeCategories),

filterInlineFeedbackHistory =

NtabCaretClickFatigueUtils.feedbackModelFilterByCRT(crts)

).apply(Seq(candidate))

.map(\_.headOption.getOrElse(false))

} else Future.True

} else {

Future.True

}

}.withStats(scopedStats)

.withName(name)

}

def apply(

genericTypeCategories: Seq[String],

crts: Set[CRT],

calculateFatiguePeriod: Seq[CaretFeedbackDetails] => Duration,

useMostRecentDislikeTime: Boolean,

name: String = defaultName

)(

implicit globalStats: StatsReceiver

): NamedPredicate[PushCandidate] = {

val scopedStats = globalStats.scope(name)

val commonNtabCaretClickFatiguePredicate = CommonNtabCaretClickFatiguePredicate(

filterCaretFeedbackHistory =

CaretFeedbackHistoryFilter.caretFeedbackHistoryFilter(genericTypeCategories),

filterHistory = FatiguePredicate.recTypesOnlyFilter(crts),

calculateFatiguePeriod = calculateFatiguePeriod,

useMostRecentDislikeTime = useMostRecentDislikeTime,

name = name

)(globalStats)

Predicate

.fromAsync { candidate: PushCandidate =>

if (candidate.frigateNotification.notificationDisplayLocation == NotificationDisplayLocation.PushToMobileDevice) {

if (candidate.target.params(PushParams.EnableFatigueNtabCaretClickingParam)) {

commonNtabCaretClickFatiguePredicate

.apply(Seq(candidate.target))

.map(\_.headOption.getOrElse(false))

} else Future.True

} else {

Future.True

}

}.andThen(candidateFatiguePredicate(genericTypeCategories, crts))

.withStats(scopedStats)

.withName(name)

}

}