package com.twitter.frigate.pushservice.predicate

import com.twitter.frigate.common.base.TargetUser

import com.twitter.frigate.common.candidate.CaretFeedbackHistory

import com.twitter.frigate.common.candidate.TargetABDecider

import com.twitter.frigate.common.util.MrNtabCopyObjects

import com.twitter.notificationservice.thriftscala.CaretFeedbackDetails

import com.twitter.notificationservice.thriftscala.GenericNotificationMetadata

import com.twitter.notificationservice.thriftscala.GenericType

object CaretFeedbackHistoryFilter {

def caretFeedbackHistoryFilter(

categories: Seq[String]

): TargetUser with TargetABDecider with CaretFeedbackHistory => Seq[CaretFeedbackDetails] => Seq[

CaretFeedbackDetails

] = { target => caretFeedbackDetailsSeq =>

caretFeedbackDetailsSeq.filter { caretFeedbackDetails =>

caretFeedbackDetails.genericNotificationMetadata match {

case Some(genericNotificationMetadata) =>

isFeedbackSupportedGenericType(genericNotificationMetadata)

case None => false

}

}

}

private def filterCriteria(

caretFeedbackDetails: CaretFeedbackDetails,

genericTypes: Seq[GenericType]

): Boolean = {

caretFeedbackDetails.genericNotificationMetadata match {

case Some(genericNotificationMetadata) =>

genericTypes.contains(genericNotificationMetadata.genericType)

case None => false

}

}

def caretFeedbackHistoryFilterByGenericType(

genericTypes: Seq[GenericType]

): TargetUser with TargetABDecider with CaretFeedbackHistory => Seq[CaretFeedbackDetails] => Seq[

CaretFeedbackDetails

] = { target => caretFeedbackDetailsSeq =>

caretFeedbackDetailsSeq.filter { caretFeedbackDetails =>

filterCriteria(caretFeedbackDetails, genericTypes)

}

}

def caretFeedbackHistoryFilterByGenericTypeDenyList(

genericTypes: Seq[GenericType]

): TargetUser with TargetABDecider with CaretFeedbackHistory => Seq[CaretFeedbackDetails] => Seq[

CaretFeedbackDetails

] = { target => caretFeedbackDetailsSeq =>

caretFeedbackDetailsSeq.filterNot { caretFeedbackDetails =>

filterCriteria(caretFeedbackDetails, genericTypes)

}

}

def caretFeedbackHistoryFilterByRefreshableType(

refreshableTypes: Set[Option[String]]

): TargetUser with TargetABDecider with CaretFeedbackHistory => Seq[CaretFeedbackDetails] => Seq[

CaretFeedbackDetails

] = { target => caretFeedbackDetailsSeq =>

caretFeedbackDetailsSeq.filter { caretFeedbackDetails =>

caretFeedbackDetails.genericNotificationMetadata match {

case Some(genericNotificationMetadata) =>

refreshableTypes.contains(genericNotificationMetadata.refreshableType)

case None => false

}

}

}

def caretFeedbackHistoryFilterByRefreshableTypeDenyList(

refreshableTypes: Set[Option[String]]

): TargetUser with TargetABDecider with CaretFeedbackHistory => Seq[CaretFeedbackDetails] => Seq[

CaretFeedbackDetails

] = { target => caretFeedbackDetailsSeq =>

caretFeedbackDetailsSeq.filter { caretFeedbackDetails =>

caretFeedbackDetails.genericNotificationMetadata match {

case Some(genericNotificationMetadata) =>

!refreshableTypes.contains(genericNotificationMetadata.refreshableType)

case None => true

}

}

}

private def isFeedbackSupportedGenericType(

notificationMetadata: GenericNotificationMetadata

): Boolean = {

val genericNotificationTypeName =

(notificationMetadata.genericType, notificationMetadata.refreshableType) match {

case (GenericType.RefreshableNotification, Some(refreshableType)) => refreshableType

case \_ => notificationMetadata.genericType.name

}

MrNtabCopyObjects.AllNtabCopyTypes

.flatMap(\_.refreshableType)

.contains(genericNotificationTypeName)

}

}