package com.twitter.frigate.pushservice.model

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.HydratedMagicFanoutCreatorEventCandidate

import com.twitter.frigate.common.base.MagicFanoutCreatorEventCandidate

import com.twitter.frigate.magic\_events.thriftscala.CreatorFanoutType

import com.twitter.frigate.magic\_events.thriftscala.MagicEventsReason

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

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

import com.twitter.frigate.pushservice.config.Config

import com.twitter.frigate.pushservice.ml.PushMLModelScorer

import com.twitter.frigate.pushservice.model.candidate.CopyIds

import com.twitter.frigate.pushservice.model.ibis.MagicFanoutCreatorEventIbis2Hydrator

import com.twitter.frigate.pushservice.model.ntab.MagicFanoutCreatorEventNtabRequestHydrator

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams

import com.twitter.frigate.pushservice.predicate.PredicatesForCandidate

import com.twitter.frigate.pushservice.predicate.magic\_fanout.MagicFanoutPredicatesForCandidate

import com.twitter.frigate.pushservice.predicate.ntab\_caret\_fatigue.MagicFanoutNtabCaretFatiguePredicate

import com.twitter.frigate.pushservice.take.predicates.BasicSendHandlerPredicates

import com.twitter.frigate.thriftscala.CommonRecommendationType

import com.twitter.frigate.thriftscala.FrigateNotification

import com.twitter.gizmoduck.thriftscala.User

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.client.UserId

import com.twitter.util.Future

import scala.util.control.NoStackTrace

class MagicFanoutCreatorEventPushCandidateHydratorException(private val message: String)

extends Exception(message)

with NoStackTrace

class MagicFanoutCreatorEventPushCandidate(

candidate: RawCandidate with MagicFanoutCreatorEventCandidate,

creatorUser: Option[User],

copyIds: CopyIds,

creatorTweetCountStore: ReadableStore[UserId, Int]

)(

implicit val statsScoped: StatsReceiver,

pushModelScorer: PushMLModelScorer)

extends PushCandidate

with MagicFanoutCreatorEventIbis2Hydrator

with MagicFanoutCreatorEventNtabRequestHydrator

with MagicFanoutCreatorEventCandidate

with HydratedMagicFanoutCreatorEventCandidate {

override def creatorId: Long = candidate.creatorId

override def hydratedCreator: Option[User] = creatorUser

override lazy val numberOfTweetsFut: Future[Option[Int]] =

creatorTweetCountStore.get(UserId(creatorId))

lazy val userProfile = hydratedCreator

.flatMap(\_.profile).getOrElse(

throw new MagicFanoutCreatorEventPushCandidateHydratorException(

s"Unable to get user profile to generate tapThrough for userId: $creatorId"))

override val frigateNotification: FrigateNotification = candidate.frigateNotification

override def subscriberId: Option[Long] = candidate.subscriberId

override def creatorFanoutType: CreatorFanoutType = candidate.creatorFanoutType

override def target: PushTypes.Target = candidate.target

override def pushId: Long = candidate.pushId

override def candidateMagicEventsReasons: Seq[MagicEventsReason] =

candidate.candidateMagicEventsReasons

override def statsReceiver: StatsReceiver = statsScoped

override def pushCopyId: Option[Int] = copyIds.pushCopyId

override def ntabCopyId: Option[Int] = copyIds.ntabCopyId

override def copyAggregationId: Option[String] = copyIds.aggregationId

override def commonRecType: CommonRecommendationType = candidate.commonRecType

override def weightedOpenOrNtabClickModelScorer: PushMLModelScorer = pushModelScorer

}

case class MagicFanouCreatorSubscriptionEventPushPredicates(config: Config)

extends BasicSendHandlerPredicates[MagicFanoutCreatorEventPushCandidate] {

implicit val statsReceiver: StatsReceiver = config.statsReceiver.scope(getClass.getSimpleName)

override val preCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutCreatorEventPushCandidate]

] =

List(

PredicatesForCandidate.paramPredicate(

PushFeatureSwitchParams.EnableCreatorSubscriptionPush

),

PredicatesForCandidate.isDeviceEligibleForCreatorPush,

MagicFanoutPredicatesForCandidate.creatorPushTargetIsNotCreator(),

MagicFanoutPredicatesForCandidate.duplicateCreatorPredicate,

MagicFanoutPredicatesForCandidate.magicFanoutCreatorPushFatiguePredicate(),

)

override val postCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutCreatorEventPushCandidate]

] =

List(

MagicFanoutNtabCaretFatiguePredicate(),

MagicFanoutPredicatesForCandidate.isSuperFollowingCreator()(config, statsReceiver).flip

)

}

case class MagicFanoutNewCreatorEventPushPredicates(config: Config)

extends BasicSendHandlerPredicates[MagicFanoutCreatorEventPushCandidate] {

implicit val statsReceiver: StatsReceiver = config.statsReceiver.scope(getClass.getSimpleName)

override val preCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutCreatorEventPushCandidate]

] =

List(

PredicatesForCandidate.paramPredicate(

PushFeatureSwitchParams.EnableNewCreatorPush

),

PredicatesForCandidate.isDeviceEligibleForCreatorPush,

MagicFanoutPredicatesForCandidate.duplicateCreatorPredicate,

MagicFanoutPredicatesForCandidate.magicFanoutCreatorPushFatiguePredicate,

)

override val postCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutCreatorEventPushCandidate]

] =

List(

MagicFanoutNtabCaretFatiguePredicate(),

MagicFanoutPredicatesForCandidate.isSuperFollowingCreator()(config, statsReceiver).flip

)

}