package com.twitter.frigate.pushservice.model

import com.twitter.escherbird.metadata.thriftscala.EntityMegadata

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.MagicFanoutNewsEventCandidate

import com.twitter.frigate.common.store.interests.InterestsLookupRequestWithContext

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

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.MagicFanoutNewsEventIbis2Hydrator

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

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

import com.twitter.frigate.pushservice.predicate.event.EventPredicatesForCandidate

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

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

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

import com.twitter.frigate.pushservice.store.EventRequest

import com.twitter.frigate.pushservice.store.UttEntityHydrationStore

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

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.hermit.store.semantic\_core.SemanticEntityForQuery

import com.twitter.interests.thriftscala.UserInterests

import com.twitter.livevideo.timeline.domain.v2.Event

import com.twitter.simclusters\_v2.thriftscala.SimClustersInferredEntities

import com.twitter.storehaus.ReadableStore

class MagicFanoutNewsEventPushCandidate(

candidate: RawCandidate with MagicFanoutNewsEventCandidate,

copyIds: CopyIds,

override val fanoutEvent: Option[FanoutEvent],

override val semanticEntityResults: Map[SemanticEntityForQuery, Option[EntityMegadata]],

simClusterToEntities: Map[Int, Option[SimClustersInferredEntities]],

lexServiceStore: ReadableStore[EventRequest, Event],

interestsLookupStore: ReadableStore[InterestsLookupRequestWithContext, UserInterests],

uttEntityHydrationStore: UttEntityHydrationStore

)(

implicit statsScoped: StatsReceiver,

pushModelScorer: PushMLModelScorer)

extends MagicFanoutEventPushCandidate(

candidate,

copyIds,

fanoutEvent,

semanticEntityResults,

simClusterToEntities,

lexServiceStore,

interestsLookupStore,

uttEntityHydrationStore

)(statsScoped, pushModelScorer)

with MagicFanoutNewsEventCandidate

with MagicFanoutNewsEventIbis2Hydrator

with MagicFanoutNewsEventNTabRequestHydrator {

override lazy val stats: StatsReceiver = statsScoped.scope("MagicFanoutNewsEventPushCandidate")

override val statsReceiver: StatsReceiver = statsScoped.scope("MagicFanoutNewsEventPushCandidate")

}

case class MagicFanoutNewsEventCandidatePredicates(config: Config)

extends BasicSendHandlerPredicates[MagicFanoutNewsEventPushCandidate] {

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

override val preCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutNewsEventPushCandidate]

] =

List(

EventPredicatesForCandidate.accountCountryPredicateWithAllowlist,

PredicatesForCandidate.isDeviceEligibleForNewsOrSports,

MagicFanoutPredicatesForCandidate.inferredUserDeviceLanguagePredicate,

PredicatesForCandidate.secondaryDormantAccountPredicate(statsReceiver),

MagicFanoutPredicatesForCandidate.highPriorityNewsEventExceptedPredicate(

MagicFanoutTargetingPredicateWrappersForCandidate

.magicFanoutTargetingPredicate(statsReceiver, config)

)(config),

MagicFanoutPredicatesForCandidate.geoOptOutPredicate(config.safeUserStore),

EventPredicatesForCandidate.isNotDuplicateWithEventId,

MagicFanoutPredicatesForCandidate.highPriorityNewsEventExceptedPredicate(

MagicFanoutPredicatesForCandidate.newsNotificationFatigue()

)(config),

MagicFanoutPredicatesForCandidate.highPriorityNewsEventExceptedPredicate(

MagicFanoutNtabCaretFatiguePredicate()

)(config),

MagicFanoutPredicatesForCandidate.escherbirdMagicfanoutEventParam()(statsReceiver),

MagicFanoutPredicatesForCandidate.hasCustomTargetingForNewsEventsParam(

statsReceiver

)

)

override val postCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutNewsEventPushCandidate]

] =

List(

MagicFanoutPredicatesForCandidate.magicFanoutNoOptoutInterestPredicate,

MagicFanoutPredicatesForCandidate.geoTargetingHoldback(),

MagicFanoutPredicatesForCandidate.userGeneratedEventsPredicate,

EventPredicatesForCandidate.hasTitle,

)

}