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.BaseGameScore

import com.twitter.frigate.common.base.MagicFanoutSportsEventCandidate

import com.twitter.frigate.common.base.MagicFanoutSportsScoreInformation

import com.twitter.frigate.common.base.TeamInfo

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.MagicFanoutSportsEventIbis2Hydrator

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

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.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.livevideo.timeline.domain.v2.HydrationOptions

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

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

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

class MagicFanoutSportsPushCandidate(

candidate: RawCandidate

with MagicFanoutSportsEventCandidate

with MagicFanoutSportsScoreInformation,

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 MagicFanoutSportsEventCandidate

with MagicFanoutSportsScoreInformation

with MagicFanoutSportsEventNTabRequestHydrator

with MagicFanoutSportsEventIbis2Hydrator {

override val isScoreUpdate: Boolean = candidate.isScoreUpdate

override val gameScores: Future[Option[BaseGameScore]] = candidate.gameScores

override val homeTeamInfo: Future[Option[TeamInfo]] = candidate.homeTeamInfo

override val awayTeamInfo: Future[Option[TeamInfo]] = candidate.awayTeamInfo

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

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

override lazy val eventRequestFut: Future[Option[EventRequest]] = {

Future.join(target.inferredUserDeviceLanguage, target.accountCountryCode).map {

case (inferredUserDeviceLanguage, accountCountryCode) =>

Some(

EventRequest(

eventId,

lookupContext = LookupContext(

hydrationOptions = HydrationOptions(

includeSquareImage = true,

includePrimaryImage = true

),

language = inferredUserDeviceLanguage,

countryCode = accountCountryCode

)

))

}

}

}

case class MagicFanoutSportsEventCandidatePredicates(config: Config)

extends BasicSendHandlerPredicates[MagicFanoutSportsPushCandidate] {

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

override val preCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutSportsPushCandidate]

] =

List(

PredicatesForCandidate.paramPredicate(PushFeatureSwitchParams.EnableScoreFanoutNotification)

)

override val postCandidateSpecificPredicates: List[

NamedPredicate[MagicFanoutSportsPushCandidate]

] =

List(

PredicatesForCandidate.isDeviceEligibleForNewsOrSports,

MagicFanoutPredicatesForCandidate.inferredUserDeviceLanguagePredicate,

MagicFanoutPredicatesForCandidate.highPriorityEventExceptedPredicate(

MagicFanoutTargetingPredicateWrappersForCandidate

.magicFanoutTargetingPredicate(statsReceiver, config)

)(config),

PredicatesForCandidate.secondaryDormantAccountPredicate(

statsReceiver

),

MagicFanoutPredicatesForCandidate.highPriorityEventExceptedPredicate(

MagicFanoutNtabCaretFatiguePredicate()

)(config),

)

}