package com.twitter.frigate.pushservice.predicate.magic\_fanout

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.frigate.common.util.FeatureSwitchParams

import com.twitter.frigate.common.util.MagicFanoutTargetingPredicatesEnum

import com.twitter.frigate.common.util.MagicFanoutTargetingPredicatesEnum.MagicFanoutTargetingPredicatesEnum

import com.twitter.frigate.pushservice.model.MagicFanoutEventPushCandidate

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

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

import com.twitter.frigate.thriftscala.CommonRecommendationType

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.hermit.predicate.Predicate

import com.twitter.interests.thriftscala.UserInterests

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi.FSEnumParam

object MagicFanoutTargetingPredicateWrappersForCandidate {

/\*\*

\* Combine Prod and Experimental Targeting predicate logic

\* @return: NamedPredicate[MagicFanoutNewsEventPushCandidate]

\*/

def magicFanoutTargetingPredicate(

stats: StatsReceiver,

config: Config

): NamedPredicate[MagicFanoutEventPushCandidate] = {

val name = "magic\_fanout\_targeting\_predicate"

Predicate

.fromAsync { candidate: MagicFanoutEventPushCandidate =>

val mfTargetingPredicateParam = getTargetingPredicateParams(candidate)

val mfTargetingPredicate = MagicFanoutTargetingPredicateMapForCandidate

.apply(config)

.get(candidate.target.params(mfTargetingPredicateParam))

mfTargetingPredicate match {

case Some(predicate) =>

predicate.apply(Seq(candidate)).map(\_.head)

case None =>

throw new Exception(

s"MFTargetingPredicateMap doesnt contain value for TargetingParam: ${FeatureSwitchParams.MFTargetingPredicate}")

}

}

.withStats(stats.scope(name))

.withName(name)

}

private def getTargetingPredicateParams(

candidate: MagicFanoutEventPushCandidate

): FSEnumParam[MagicFanoutTargetingPredicatesEnum.type] = {

if (candidate.commonRecType == CommonRecommendationType.MagicFanoutSportsEvent) {

FeatureSwitchParams.MFCricketTargetingPredicate

} else FeatureSwitchParams.MFTargetingPredicate

}

/\*\*

\* SimCluster and ERG and Topic Follows Targeting Predicate

\*/

def simClusterErgTopicFollowsTargetingPredicate(

implicit stats: StatsReceiver,

interestsLookupStore: ReadableStore[InterestsLookupRequestWithContext, UserInterests]

): NamedPredicate[MagicFanoutEventPushCandidate] = {

simClusterErgTargetingPredicate

.or(MagicFanoutPredicatesForCandidate.magicFanoutTopicFollowsTargetingPredicate)

.withName("sim\_cluster\_erg\_topic\_follows\_targeting")

}

/\*\*

\* SimCluster and ERG and Topic Follows Targeting Predicate

\*/

def simClusterErgTopicFollowsUserFollowsTargetingPredicate(

implicit stats: StatsReceiver,

interestsLookupStore: ReadableStore[InterestsLookupRequestWithContext, UserInterests]

): NamedPredicate[MagicFanoutEventPushCandidate] = {

simClusterErgTopicFollowsTargetingPredicate

.or(

MagicFanoutPredicatesForCandidate.followRankThreshold(

PushFeatureSwitchParams.MagicFanoutRealgraphRankThreshold))

.withName("sim\_cluster\_erg\_topic\_follows\_user\_follows\_targeting")

}

/\*\*

\* SimCluster and ERG Targeting Predicate

\*/

def simClusterErgTargetingPredicate(

implicit stats: StatsReceiver

): NamedPredicate[MagicFanoutEventPushCandidate] = {

MagicFanoutPredicatesForCandidate.magicFanoutSimClusterTargetingPredicate

.or(MagicFanoutPredicatesForCandidate.magicFanoutErgInterestRankThresholdPredicate)

.withName("sim\_cluster\_erg\_targeting")

}

}

/\*\*

\* Object to initalze and get predicate map

\*/

object MagicFanoutTargetingPredicateMapForCandidate {

/\*\*

\* Called from the Config.scala at the time of server initialization

\* @param statsReceiver: implict stats receiver

\* @return Map[MagicFanoutTargetingPredicatesEnum, NamedPredicate[MagicFanoutNewsEventPushCandidate]]

\*/

def apply(

config: Config

): Map[MagicFanoutTargetingPredicatesEnum, NamedPredicate[MagicFanoutEventPushCandidate]] = {

Map(

MagicFanoutTargetingPredicatesEnum.SimClusterAndERGAndTopicFollows -> MagicFanoutTargetingPredicateWrappersForCandidate

.simClusterErgTopicFollowsTargetingPredicate(

config.statsReceiver,

config.interestsWithLookupContextStore),

MagicFanoutTargetingPredicatesEnum.SimClusterAndERG -> MagicFanoutTargetingPredicateWrappersForCandidate

.simClusterErgTargetingPredicate(config.statsReceiver),

MagicFanoutTargetingPredicatesEnum.SimCluster -> MagicFanoutPredicatesForCandidate

.magicFanoutSimClusterTargetingPredicate(config.statsReceiver),

MagicFanoutTargetingPredicatesEnum.ERG -> MagicFanoutPredicatesForCandidate

.magicFanoutErgInterestRankThresholdPredicate(config.statsReceiver),

MagicFanoutTargetingPredicatesEnum.TopicFollows -> MagicFanoutPredicatesForCandidate

.magicFanoutTopicFollowsTargetingPredicate(

config.statsReceiver,

config.interestsWithLookupContextStore),

MagicFanoutTargetingPredicatesEnum.UserFollows -> MagicFanoutPredicatesForCandidate

.followRankThreshold(

PushFeatureSwitchParams.MagicFanoutRealgraphRankThreshold

)(config.statsReceiver),

MagicFanoutTargetingPredicatesEnum.SimClusterAndERGAndTopicFollowsAndUserFollows ->

MagicFanoutTargetingPredicateWrappersForCandidate

.simClusterErgTopicFollowsUserFollowsTargetingPredicate(

config.statsReceiver,

config.interestsWithLookupContextStore

)

)

}

}