package com.twitter.frigate.pushservice.target

import com.twitter.abdecider.LoggingABDecider

import com.twitter.conversions.DurationOps.\_

import com.twitter.decider.Decider

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.FeatureMap

import com.twitter.frigate.common.history.History

import com.twitter.frigate.common.history.HistoryStoreKeyContext

import com.twitter.frigate.common.history.MagicFanoutReasonHistory

import com.twitter.frigate.common.history.PushServiceHistoryStore

import com.twitter.frigate.common.history.RecItems

import com.twitter.frigate.common.store.deviceinfo.DeviceInfo

import com.twitter.frigate.common.util.ABDeciderWithOverride

import com.twitter.frigate.common.util.LanguageLocaleUtil

import com.twitter.frigate.data\_pipeline.features\_common.MrRequestContextForFeatureStore

import com.twitter.frigate.data\_pipeline.thriftscala.UserHistoryValue

import com.twitter.frigate.dau\_model.thriftscala.DauProbability

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

import com.twitter.frigate.pushservice.thriftscala.PushContext

import com.twitter.frigate.thriftscala.UserForPushTargeting

import com.twitter.gizmoduck.thriftscala.User

import com.twitter.hermit.stp.thriftscala.STPResult

import com.twitter.interests.thriftscala.InterestId

import com.twitter.notificationservice.genericfeedbackstore.FeedbackPromptValue

import com.twitter.notificationservice.thriftscala.CaretFeedbackDetails

import com.twitter.nrel.hydration.push.HydrationContext

import com.twitter.permissions\_storage.thriftscala.AppPermission

import com.twitter.service.metastore.gen.thriftscala.Location

import com.twitter.service.metastore.gen.thriftscala.UserLanguages

import com.twitter.stitch.Stitch

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.columns.frigate.logged\_out\_web\_notifications.thriftscala.LOWebNotificationMetadata

import com.twitter.timelines.configapi

import com.twitter.timelines.configapi.Params

import com.twitter.timelines.real\_graph.v1.thriftscala.RealGraphFeatures

import com.twitter.util.Duration

import com.twitter.util.Future

import com.twitter.wtf.scalding.common.thriftscala.UserFeatures

case class LoggedOutPushTargetUserBuilder(

historyStore: PushServiceHistoryStore,

inputDecider: Decider,

inputAbDecider: LoggingABDecider,

loggedOutPushInfoStore: ReadableStore[Long, LOWebNotificationMetadata]

)(

globalStatsReceiver: StatsReceiver) {

private val stats = globalStatsReceiver.scope("LORefreshForPushHandler")

private val noHistoryCounter = stats.counter("no\_logged\_out\_history")

private val historyFoundCounter = stats.counter("logged\_out\_history\_counter")

private val noLoggedOutUserCounter = stats.counter("no\_logged\_out\_user")

private val countryCodeCounter = stats.counter("country\_counter")

private val noCountryCodeCounter = stats.counter("no\_country\_counter")

private val noLanguageCodeCounter = stats.counter("no\_language\_counter")

def buildTarget(

guestId: Long,

inputPushContext: Option[PushContext]

): Future[Target] = {

val historyStoreKeyContext = HistoryStoreKeyContext(

guestId,

inputPushContext.flatMap(\_.useMemcacheForHistory).getOrElse(false)

)

if (historyStore.get(historyStoreKeyContext, Some(30.days)) == Future.None) {

noHistoryCounter.incr()

} else {

historyFoundCounter.incr()

}

if (loggedOutPushInfoStore.get(guestId) == Future.None) {

noLoggedOutUserCounter.incr()

}

Future

.join(

historyStore.get(historyStoreKeyContext, Some(30.days)),

loggedOutPushInfoStore.get(guestId)

).map {

case (loNotifHistory, loggedOutUserPushInfo) =>

new Target {

override lazy val stats: StatsReceiver = globalStatsReceiver

override val targetId: Long = guestId

override val targetGuestId = Some(guestId)

override lazy val decider: Decider = inputDecider

override lazy val loggedOutMetadata = Future.value(loggedOutUserPushInfo)

val rawLanguageFut = loggedOutMetadata.map { metadata => metadata.map(\_.language) }

override val targetLanguage: Future[Option[String]] = rawLanguageFut.map { rawLang =>

if (rawLang.isDefined) {

val lang = LanguageLocaleUtil.getStandardLanguageCode(rawLang.get)

if (lang.isEmpty) {

noLanguageCodeCounter.incr()

None

} else {

Option(lang)

}

} else None

}

val country = loggedOutMetadata.map(\_.map(\_.countryCode))

if (country.isDefined) {

countryCodeCounter.incr()

} else {

noCountryCodeCounter.incr()

}

if (loNotifHistory == null) {

noHistoryCounter.incr()

} else {

historyFoundCounter.incr()

}

override lazy val location: Future[Option[Location]] = country.map {

case Some(code) =>

Some(

Location(

city = "",

region = "",

countryCode = code,

confidence = 0.0,

lat = None,

lon = None,

metro = None,

placeIds = None,

weightedLocations = None,

createdAtMsec = None,

ip = None,

isSignupIp = None,

placeMap = None

))

case \_ => None

}

override lazy val pushContext: Option[PushContext] = inputPushContext

override lazy val history: Future[History] = Future.value(loNotifHistory)

override lazy val magicFanoutReasonHistory30Days: Future[MagicFanoutReasonHistory] =

Future.value(null)

override lazy val globalStats: StatsReceiver = globalStatsReceiver

override lazy val pushTargeting: Future[Option[UserForPushTargeting]] = Future.None

override lazy val appPermissions: Future[Option[AppPermission]] = Future.None

override lazy val lastHTLVisitTimestamp: Future[Option[Long]] = Future.None

override lazy val pushRecItems: Future[RecItems] = Future.value(null)

override lazy val isNewSignup: Boolean = false

override lazy val metastoreLanguages: Future[Option[UserLanguages]] = Future.None

override lazy val optOutUserInterests: Future[Option[Seq[InterestId]]] = Future.None

override lazy val mrRequestContextForFeatureStore: MrRequestContextForFeatureStore =

null

override lazy val targetUser: Future[Option[User]] = Future.None

override lazy val notificationFeedbacks: Future[Option[Seq[FeedbackPromptValue]]] =

Future.None

override lazy val promptFeedbacks: Stitch[Seq[FeedbackPromptValue]] = null

override lazy val seedsWithWeight: Future[Option[Map[Long, Double]]] = Future.None

override lazy val tweetImpressionResults: Future[Seq[Long]] = Future.Nil

override lazy val params: configapi.Params = Params.Empty

override lazy val deviceInfo: Future[Option[DeviceInfo]] = Future.None

override lazy val userFeatures: Future[Option[UserFeatures]] = Future.None

override lazy val isOpenAppExperimentUser: Future[Boolean] = Future.False

override lazy val featureMap: Future[FeatureMap] = Future.value(null)

override lazy val dauProbability: Future[Option[DauProbability]] = Future.None

override lazy val labeledPushRecsHydrated: Future[Option[UserHistoryValue]] =

Future.None

override lazy val onlineLabeledPushRecs: Future[Option[UserHistoryValue]] = Future.None

override lazy val realGraphFeatures: Future[Option[RealGraphFeatures]] = Future.None

override lazy val stpResult: Future[Option[STPResult]] = Future.None

override lazy val globalOptoutProbabilities: Seq[Future[Option[Double]]] = Seq.empty

override lazy val bucketOptoutProbability: Future[Option[Double]] = Future.None

override lazy val utcOffset: Future[Option[Duration]] = Future.None

override lazy val abDecider: ABDeciderWithOverride =

ABDeciderWithOverride(inputAbDecider, ddgOverrideOption)(globalStatsReceiver)

override lazy val resurrectionDate: Future[Option[String]] = Future.None

override lazy val isResurrectedUser: Boolean = false

override lazy val timeSinceResurrection: Option[Duration] = None

override lazy val inlineActionHistory: Future[Seq[(Long, String)]] = Future.Nil

override lazy val caretFeedbacks: Future[Option[Seq[CaretFeedbackDetails]]] =

Future.None

override def targetHydrationContext: Future[HydrationContext] = Future.value(null)

override def isBlueVerified: Future[Option[Boolean]] = Future.None

override def isVerified: Future[Option[Boolean]] = Future.None

override def isSuperFollowCreator: Future[Option[Boolean]] = Future.None

}

}

}

}