package com.twitter.cr\_mixer.source\_signal

import com.twitter.cr\_mixer.param.decider.CrMixerDecider

import com.twitter.cr\_mixer.param.decider.DeciderConstants

import com.twitter.cr\_mixer.source\_signal.FrsStore.Query

import com.twitter.cr\_mixer.source\_signal.FrsStore.FrsQueryResult

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.thriftscala.ClientContext

import com.twitter.follow\_recommendations.thriftscala.DisplayLocation

import com.twitter.follow\_recommendations.thriftscala.FollowRecommendationsThriftService

import com.twitter.follow\_recommendations.thriftscala.Recommendation

import com.twitter.follow\_recommendations.thriftscala.RecommendationRequest

import com.twitter.storehaus.ReadableStore

import javax.inject.Singleton

import com.twitter.simclusters\_v2.common.UserId

import com.twitter.util.Future

@Singleton

case class FrsStore(

frsClient: FollowRecommendationsThriftService.MethodPerEndpoint,

statsReceiver: StatsReceiver,

decider: CrMixerDecider)

extends ReadableStore[Query, Seq[FrsQueryResult]] {

override def get(

query: Query

): Future[Option[Seq[FrsQueryResult]]] = {

if (decider.isAvailable(DeciderConstants.enableFRSTrafficDeciderKey)) {

val recommendationRequest =

buildFollowRecommendationRequest(query)

frsClient

.getRecommendations(recommendationRequest).map { recommendationResponse =>

Some(recommendationResponse.recommendations.collect {

case recommendation: Recommendation.User =>

FrsQueryResult(

recommendation.user.userId,

recommendation.user.scoringDetails

.flatMap(\_.score).getOrElse(0.0),

recommendation.user.scoringDetails

.flatMap(\_.candidateSourceDetails.flatMap(\_.primarySource)),

recommendation.user.scoringDetails

.flatMap(\_.candidateSourceDetails.flatMap(\_.candidateSourceScores)).map(\_.toMap)

)

})

}

} else {

Future.None

}

}

private def buildFollowRecommendationRequest(

query: Query

): RecommendationRequest = {

RecommendationRequest(

clientContext = ClientContext(

userId = Some(query.userId),

countryCode = query.countryCodeOpt,

languageCode = query.languageCodeOpt),

displayLocation = query.displayLocation,

maxResults = Some(query.maxConsumerSeedsNum),

excludedIds = Some(query.excludedUserIds)

)

}

}

object FrsStore {

case class Query(

userId: UserId,

maxConsumerSeedsNum: Int,

displayLocation: DisplayLocation = DisplayLocation.ContentRecommender,

excludedUserIds: Seq[UserId] = Seq.empty,

languageCodeOpt: Option[String] = None,

countryCodeOpt: Option[String] = None)

case class FrsQueryResult(

userId: UserId,

score: Double,

primarySource: Option[Int],

sourceWithScores: Option[Map[String, Double]])

}