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

import com.twitter.cr\_mixer.config.TimeoutConfig

import com.twitter.cr\_mixer.model.GraphSourceInfo

import com.twitter.cr\_mixer.model.ModuleNames

import com.twitter.cr\_mixer.param.RealGraphInParams

import com.twitter.cr\_mixer.source\_signal.SourceFetcher.FetcherQuery

import com.twitter.cr\_mixer.thriftscala.SourceType

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Future

import com.twitter.wtf.candidate.thriftscala.CandidateSeq

import javax.inject.Inject

import javax.inject.Named

import javax.inject.Singleton

/\*\*

\* This store fetch user recommendations from In-Network RealGraph (go/realgraph) for a given userId

\*/

@Singleton

case class RealGraphInSourceGraphFetcher @Inject() (

@Named(ModuleNames.RealGraphInStore) realGraphStoreMh: ReadableStore[UserId, CandidateSeq],

override val timeoutConfig: TimeoutConfig,

globalStats: StatsReceiver)

extends SourceGraphFetcher {

override protected val stats: StatsReceiver = globalStats.scope(identifier)

override protected val graphSourceType: SourceType = SourceType.RealGraphIn

override def isEnabled(query: FetcherQuery): Boolean = {

query.params(RealGraphInParams.EnableSourceGraphParam)

}

override def fetchAndProcess(

query: FetcherQuery,

): Future[Option[GraphSourceInfo]] = {

val rawSignals = trackPerItemStats(query)(

realGraphStoreMh.get(query.userId).map {

\_.map { candidateSeq =>

candidateSeq.candidates

.map { candidate =>

// Bundle the userId with its score

(candidate.userId, candidate.score)

}

}

}

)

rawSignals.map {

\_.map { userWithScores =>

convertGraphSourceInfo(userWithScores)

}

}

}

}