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.FrsParams

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

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.storehaus.ReadableStore

import com.twitter.util.Future

import javax.inject.Inject

import javax.inject.Named

import javax.inject.Singleton

/\*\*\*

\* This store fetches user recommendations from FRS (go/frs) for a given userId

\*/

@Singleton

case class FrsSourceGraphFetcher @Inject() (

@Named(ModuleNames.FrsStore) frsStore: ReadableStore[FrsStore.Query, Seq[FrsQueryResult]],

override val timeoutConfig: TimeoutConfig,

globalStats: StatsReceiver)

extends SourceGraphFetcher {

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

override protected val graphSourceType: SourceType = SourceType.FollowRecommendation

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

query.params(FrsParams.EnableSourceGraphParam)

}

override def fetchAndProcess(

query: FetcherQuery,

): Future[Option[GraphSourceInfo]] = {

val rawSignals = trackPerItemStats(query)(

frsStore

.get(

FrsStore

.Query(query.userId, query.params(FrsParams.MaxConsumerSeedsNumParam))).map {

\_.map {

\_.map { v => (v.userId, v.score) }

}

}

)

rawSignals.map {

\_.map { userWithScores =>

convertGraphSourceInfo(userWithScores)

}

}

}

}