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

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

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

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

import com.twitter.frigate.common.util.StatsUtil

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

import com.twitter.util.Future

/\*\*\*

\* A SourceGraphFetcher is a trait that extends from `SourceFetcher`

\* and is specialized in tackling User Graph (eg., RealGraphOon, FRS) fetch.

\*

\* The [[ResultType]] of a SourceGraphFetcher is a `GraphSourceInfo` which contains a userSeedSet.

\* When we pass in userId, the underlying store returns one GraphSourceInfo.

\*/

trait SourceGraphFetcher extends SourceFetcher[GraphSourceInfo] {

protected final val DefaultSeedScore = 1.0

protected def graphSourceType: SourceType

/\*\*\*

\* RawDataType contains a consumers seed UserId and a score (weight)

\*/

protected type RawDataType = (UserId, Double)

def trackStats(

query: FetcherQuery

)(

func: => Future[Option[GraphSourceInfo]]

): Future[Option[GraphSourceInfo]] = {

val productScopedStats = stats.scope(query.product.originalName)

val productUserStateScopedStats = productScopedStats.scope(query.userState.toString)

StatsUtil

.trackOptionStats(productScopedStats) {

StatsUtil

.trackOptionStats(productUserStateScopedStats) {

func

}

}

}

// Track per item stats on the fetched graph results

def trackPerItemStats(

query: FetcherQuery

)(

func: => Future[Option[Seq[RawDataType]]]

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

val productScopedStats = stats.scope(query.product.originalName)

val productUserStateScopedStats = productScopedStats.scope(query.userState.toString)

StatsUtil.trackOptionItemsStats(productScopedStats) {

StatsUtil.trackOptionItemsStats(productUserStateScopedStats) {

func

}

}

}

/\*\*\*

\* Convert Seq[RawDataType] into GraphSourceInfo

\*/

protected final def convertGraphSourceInfo(

userWithScores: Seq[RawDataType]

): GraphSourceInfo = {

GraphSourceInfo(

sourceType = graphSourceType,

seedWithScores = userWithScores.map { userWithScore =>

userWithScore.\_1 -> userWithScore.\_2

}.toMap

)

}

}