package com.twitter.cr\_mixer.similarity\_engine

import com.twitter.cr\_mixer.similarity\_engine.SimilarityEngine.MemCacheConfig

import com.twitter.cr\_mixer.similarity\_engine.SimilarityEngine.SimilarityEngineConfig

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi.Params

import com.twitter.util.Future

/\*\*

\* @tparam Query ReadableStore's input type.

\*/

case class EngineQuery[Query](

storeQuery: Query,

params: Params,

)

/\*\*

\* A straight forward SimilarityEngine implementation that wraps a ReadableStore

\*

\* @param implementingStore Provides the candidate retrieval's implementations

\* @param memCacheConfig If specified, it will wrap the underlying store with a MemCache layer

\* You should only enable this for cacheable queries, e.x. TweetIds.

\* consumer based UserIds are generally not possible to cache.

\* @tparam Query ReadableStore's input type

\* @tparam Candidate ReadableStore's return type is Seq[[[Candidate]]]

\*/

class StandardSimilarityEngine[Query, Candidate <: Serializable](

implementingStore: ReadableStore[Query, Seq[Candidate]],

override val identifier: SimilarityEngineType,

globalStats: StatsReceiver,

engineConfig: SimilarityEngineConfig,

memCacheConfig: Option[MemCacheConfig[Query]] = None)

extends SimilarityEngine[EngineQuery[Query], Candidate] {

private val scopedStats = globalStats.scope("similarityEngine", identifier.toString)

def getScopedStats: StatsReceiver = scopedStats

// Add memcache wrapper, if specified

private val store = {

memCacheConfig match {

case Some(config) =>

SimilarityEngine.addMemCache(

underlyingStore = implementingStore,

memCacheConfig = config,

statsReceiver = scopedStats

)

case \_ => implementingStore

}

}

override def getCandidates(

engineQuery: EngineQuery[Query]

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

SimilarityEngine.getFromFn(

store.get,

engineQuery.storeQuery,

engineConfig,

engineQuery.params,

scopedStats

)

}

}