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

case class LookupEngineQuery[Query](

storeQuery: Query, // the actual Query type of the underlying store

lookupKey: String,

params: Params,

)

/\*\*

\* This Engine provides a map interface for looking up different model implementations.

\* It provides modelId level monitoring for free.

\*

\* Example use cases include OfflineSimClusters lookup

\*

\*

\* @param versionedStoreMap A mapping from a modelId to a corresponding implementation

\* @param memCacheConfigOpt 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.

\*/

class LookupSimilarityEngine[Query, Candidate <: Serializable](

versionedStoreMap: Map[String, ReadableStore[Query, Seq[Candidate]]], // key = modelId

override val identifier: SimilarityEngineType,

globalStats: StatsReceiver,

engineConfig: SimilarityEngineConfig,

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

extends SimilarityEngine[LookupEngineQuery[Query], Candidate] {

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

private val underlyingLookupMap = {

memCacheConfigOpt match {

case Some(config) =>

versionedStoreMap.map {

case (modelId, store) =>

(

modelId,

SimilarityEngine.addMemCache(

underlyingStore = store,

memCacheConfig = config,

keyPrefix = Some(modelId),

statsReceiver = scopedStats

)

)

}

case \_ => versionedStoreMap

}

}

override def getCandidates(

engineQuery: LookupEngineQuery[Query]

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

val versionedStore =

underlyingLookupMap

.getOrElse(

engineQuery.lookupKey,

throw new IllegalArgumentException(

s"${this.getClass.getSimpleName} ${identifier.toString}: ModelId ${engineQuery.lookupKey} does not exist"

)

)

SimilarityEngine.getFromFn(

fn = versionedStore.get,

storeQuery = engineQuery.storeQuery,

engineConfig = engineConfig,

params = engineQuery.params,

scopedStats = scopedStats.scope(engineQuery.lookupKey)

)

}

}