package com.twitter.representation\_manager.config

import com.twitter.simclusters\_v2.thriftscala.EmbeddingType

import com.twitter.simclusters\_v2.thriftscala.ModelVersion

import com.twitter.util.Duration

/\*

\* --------------------------------------------

\* PLEASE NOTE:

\* Having in-memory cache is not necessarily a free performance win, anyone considering it should

\* investigate rather than blindly enabling it

\* --------------------------------------------

\* \*/

sealed trait InMemoryCacheParams

/\*

\* This holds params that is required to set up a in-mem cache for a single embedding store

\*/

case class EnabledInMemoryCacheParams(

ttl: Duration,

maxKeys: Int,

cacheName: String)

extends InMemoryCacheParams

object DisabledInMemoryCacheParams extends InMemoryCacheParams

/\*

\* This is the class for the in-memory cache config. Client could pass in their own cacheParamsMap to

\* create a new InMemoryCacheConfig instead of using the DefaultInMemoryCacheConfig object below

\* \*/

class InMemoryCacheConfig(

cacheParamsMap: Map[

(EmbeddingType, ModelVersion),

InMemoryCacheParams

] = Map.empty) {

def getCacheSetup(

embeddingType: EmbeddingType,

modelVersion: ModelVersion

): InMemoryCacheParams = {

// When requested embedding type doesn't exist, we return DisabledInMemoryCacheParams

cacheParamsMap.getOrElse((embeddingType, modelVersion), DisabledInMemoryCacheParams)

}

}

/\*

\* Default config for the in-memory cache

\* Clients can directly import and use this one if they don't want to set up a customised config

\* \*/

object DefaultInMemoryCacheConfig extends InMemoryCacheConfig {

// set default to no in-memory caching

val cacheParamsMap = Map.empty

}