package com.twitter.representationscorer.modules

import com.google.inject.Provides

import com.twitter.decider.Decider

import com.twitter.finagle.memcached.{Client => MemcachedClient}

import com.twitter.finagle.mtls.authentication.ServiceIdentifier

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finagle.thrift.ClientId

import com.twitter.hermit.store.common.ObservedReadableStore

import com.twitter.inject.TwitterModule

import com.twitter.relevance\_platform.common.readablestore.ReadableStoreWithTimeout

import com.twitter.representation\_manager.migration.LegacyRMS

import com.twitter.representationscorer.DeciderConstants

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

import com.twitter.simclusters\_v2.stores.SimClustersEmbeddingStore

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

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

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

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

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

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Timer

import javax.inject.Singleton

object EmbeddingStoreModule extends TwitterModule {

@Singleton

@Provides

def providesEmbeddingStore(

memCachedClient: MemcachedClient,

serviceIdentifier: ServiceIdentifier,

clientId: ClientId,

timer: Timer,

decider: Decider,

stats: StatsReceiver

): ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding] = {

val cacheHashKeyPrefix: String = "RMS"

val embeddingStoreClient = new LegacyRMS(

serviceIdentifier,

memCachedClient,

stats,

decider,

clientId,

timer,

cacheHashKeyPrefix

)

val underlyingStores: Map[

(EmbeddingType, ModelVersion),

ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding]

] = Map(

// Tweet Embeddings

(

LogFavBasedTweet,

Model20m145k2020) -> embeddingStoreClient.logFavBased20M145K2020TweetEmbeddingStore,

(

LogFavLongestL2EmbeddingTweet,

Model20m145k2020) -> embeddingStoreClient.logFavBasedLongestL2Tweet20M145K2020EmbeddingStore,

// InterestedIn Embeddings

(

LogFavBasedUserInterestedInFromAPE,

Model20m145k2020) -> embeddingStoreClient.LogFavBasedInterestedInFromAPE20M145K2020Store,

(

FavBasedUserInterestedIn,

Model20m145k2020) -> embeddingStoreClient.favBasedUserInterestedIn20M145K2020Store,

// Author Embeddings

(

FavBasedProducer,

Model20m145k2020) -> embeddingStoreClient.favBasedProducer20M145K2020EmbeddingStore,

// Entity Embeddings

(

LogFavBasedKgoApeTopic,

Model20m145k2020) -> embeddingStoreClient.logFavBasedApeEntity20M145K2020EmbeddingCachedStore,

(FavTfgTopic, Model20m145k2020) -> embeddingStoreClient.favBasedTfgTopicEmbedding2020Store,

)

val simClustersEmbeddingStore: ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding] = {

val underlying: ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding] =

SimClustersEmbeddingStore.buildWithDecider(

underlyingStores = underlyingStores,

decider = decider,

statsReceiver = stats.scope("simClusters\_embeddings\_store\_deciderable")

)

val underlyingWithTimeout: ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding] =

new ReadableStoreWithTimeout(

rs = underlying,

decider = decider,

enableTimeoutDeciderKey = DeciderConstants.enableSimClustersEmbeddingStoreTimeouts,

timeoutValueKey = DeciderConstants.simClustersEmbeddingStoreTimeoutValueMillis,

timer = timer,

statsReceiver = stats.scope("simClusters\_embedding\_store\_timeouts")

)

ObservedReadableStore(

store = underlyingWithTimeout

)(stats.scope("simClusters\_embeddings\_store"))

}

simClustersEmbeddingStore

}

}