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

import com.google.inject.Provides

import com.twitter.conversions.DurationOps.\_

import com.twitter.cr\_mixer.config.TimeoutConfig

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

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

import com.twitter.cr\_mixer.param.decider.CrMixerDecider

import com.twitter.cr\_mixer.param.decider.DeciderConstants

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

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

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

import com.twitter.cr\_mixer.similarity\_engine.StandardSimilarityEngine

import com.twitter.cr\_mixer.similarity\_engine.TweetBasedUserAdGraphSimilarityEngine

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

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.hashing.KeyHasher

import com.twitter.hermit.store.common.ObservedMemcachedReadableStore

import com.twitter.inject.TwitterModule

import com.twitter.recos.user\_ad\_graph.thriftscala.UserAdGraph

import com.twitter.relevance\_platform.common.injection.LZ4Injection

import com.twitter.relevance\_platform.common.injection.SeqObjectInjection

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

import com.twitter.storehaus.ReadableStore

import com.twitter.twistly.thriftscala.TweetRecentEngagedUsers

import javax.inject.Named

import javax.inject.Singleton

object TweetBasedUserAdGraphSimilarityEngineModule extends TwitterModule {

private val keyHasher: KeyHasher = KeyHasher.FNV1A\_64

@Provides

@Singleton

@Named(ModuleNames.TweetBasedUserAdGraphSimilarityEngine)

def providesTweetBasedUserAdGraphSimilarityEngine(

userAdGraphService: UserAdGraph.MethodPerEndpoint,

tweetRecentEngagedUserStore: ReadableStore[TweetId, TweetRecentEngagedUsers],

@Named(ModuleNames.UnifiedCache) crMixerUnifiedCacheClient: MemcachedClient,

timeoutConfig: TimeoutConfig,

statsReceiver: StatsReceiver,

decider: CrMixerDecider

): StandardSimilarityEngine[

TweetBasedUserAdGraphSimilarityEngine.Query,

TweetWithScore

] = {

val underlyingStore = TweetBasedUserAdGraphSimilarityEngine(

userAdGraphService,

tweetRecentEngagedUserStore,

statsReceiver)

val memCachedStore: ReadableStore[

TweetBasedUserAdGraphSimilarityEngine.Query,

Seq[

TweetWithScore

]

] =

ObservedMemcachedReadableStore

.fromCacheClient(

backingStore = underlyingStore,

cacheClient = crMixerUnifiedCacheClient,

ttl = 10.minutes

)(

valueInjection = LZ4Injection.compose(SeqObjectInjection[TweetWithScore]()),

statsReceiver = statsReceiver.scope("tweet\_based\_user\_ad\_graph\_store\_memcache"),

keyToString = { k =>

//Example Query CRMixer:TweetBasedUTG:1234567890ABCDEF

f"CRMixer:TweetBasedUAG:${keyHasher.hashKey(k.toString.getBytes)}%X"

}

)

new StandardSimilarityEngine[

TweetBasedUserAdGraphSimilarityEngine.Query,

TweetWithScore

](

implementingStore = memCachedStore,

identifier = SimilarityEngineType.TweetBasedUserAdGraph,

globalStats = statsReceiver,

engineConfig = SimilarityEngineConfig(

timeout = timeoutConfig.similarityEngineTimeout,

gatingConfig = GatingConfig(

deciderConfig =

Some(DeciderConfig(decider, DeciderConstants.enableUserAdGraphTrafficDeciderKey)),

enableFeatureSwitch = None

)

)

)

}

}