package com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.heron

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.mtls.authentication.EmptyServiceIdentifier

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

import com.twitter.storehaus\_internal.memcache.ConnectionConfig

import com.twitter.storehaus\_internal.memcache.MemcacheConfig

import com.twitter.storehaus\_internal.util.KeyPrefix

import com.twitter.storehaus\_internal.util.TTL

import com.twitter.storehaus\_internal.util.ZkEndPoint

import com.twitter.summingbird\_internal.runner.store\_config.OnlineStoreOnlyConfig

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.AggregateStore

import com.twitter.util.Duration

object RealTimeAggregateStore {

val twCacheWilyPrefix = "/srv#" // s2s is only supported for wily path

def makeEndpoint(

memcacheDataSet: String,

isProd: Boolean,

twCacheWilyPrefix: String = twCacheWilyPrefix

): String = {

val env = if (isProd) "prod" else "test"

s"$twCacheWilyPrefix/$env/local/cache/$memcacheDataSet"

}

}

case class RealTimeAggregateStore(

memcacheDataSet: String,

isProd: Boolean = false,

cacheTTL: Duration = 1.day)

extends OnlineStoreOnlyConfig[MemcacheConfig]

with AggregateStore {

import RealTimeAggregateStore.\_

override val name: String = ""

val storeKeyPrefix: KeyPrefix = KeyPrefix(name)

val memcacheZkEndPoint: String = makeEndpoint(memcacheDataSet, isProd)

def online: MemcacheConfig = online(serviceIdentifier = EmptyServiceIdentifier)

def online(serviceIdentifier: ServiceIdentifier = EmptyServiceIdentifier): MemcacheConfig =

new MemcacheConfig {

val endpoint = ZkEndPoint(memcacheZkEndPoint)

override val connectionConfig =

ConnectionConfig(endpoint, serviceIdentifier = serviceIdentifier)

override val keyPrefix = storeKeyPrefix

override val ttl = TTL(Duration.fromMilliseconds(cacheTTL.inMillis))

}

}