package com.twitter.product\_mixer.component\_library.module

import com.google.inject.Provides

import com.twitter.conversions.DurationOps.\_

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.inject.TwitterModule

import com.twitter.user\_session\_store.ReadOnlyUserSessionStore

import com.twitter.user\_session\_store.ReadWriteUserSessionStore

import com.twitter.user\_session\_store.UserSessionDataset

import com.twitter.user\_session\_store.UserSessionDataset.UserSessionDataset

import com.twitter.user\_session\_store.config.manhattan.UserSessionStoreManhattanConfig

import com.twitter.user\_session\_store.impl.manhattan.readonly.ReadOnlyManhattanUserSessionStoreBuilder

import com.twitter.user\_session\_store.impl.manhattan.readwrite.ReadWriteManhattanUserSessionStoreBuilder

import javax.inject.Singleton

object UserSessionStoreModule extends TwitterModule {

private val ReadWriteAppId = "timelineservice\_user\_session\_store"

private val ReadWriteStagingDataset = "tls\_user\_session\_store\_nonprod"

private val ReadWriteProdDataset = "tls\_user\_session\_store"

private val ReadOnlyAppId = "user\_session\_store"

private val ReadOnlyDataset = "user\_session\_fields"

@Provides

@Singleton

def providesReadWriteUserSessionStore(

injectedServiceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): ReadWriteUserSessionStore = {

val scopedStatsReceiver = statsReceiver.scope(injectedServiceIdentifier.service)

val dataset = injectedServiceIdentifier.environment.toLowerCase match {

case "prod" => ReadWriteProdDataset

case \_ => ReadWriteStagingDataset

}

val clientReadWriteConfig = new UserSessionStoreManhattanConfig.Prod.ReadWrite.Omega {

override val appId = ReadWriteAppId

override val defaultMaxTimeout = 400.milliseconds

override val maxRetryCount = 1

override val serviceIdentifier = injectedServiceIdentifier

override val datasetNamesById = Map[UserSessionDataset, String](

UserSessionDataset.ActiveDaysInfo -> dataset,

UserSessionDataset.NonPollingTimes -> dataset

)

}

ReadWriteManhattanUserSessionStoreBuilder

.buildReadWriteUserSessionStore(clientReadWriteConfig, statsReceiver, scopedStatsReceiver)

}

@Provides

@Singleton

def providesReadOnlyUserSessionStore(

injectedServiceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): ReadOnlyUserSessionStore = {

val scopedStatsReceiver = statsReceiver.scope(injectedServiceIdentifier.service)

val clientReadOnlyConfig = new UserSessionStoreManhattanConfig.Prod.ReadOnly.Athena {

override val appId = ReadOnlyAppId

override val defaultMaxTimeout = 400.milliseconds

override val maxRetryCount = 1

override val serviceIdentifier = injectedServiceIdentifier

override val datasetNamesById = Map[UserSessionDataset, String](

UserSessionDataset.UserHealth -> ReadOnlyDataset

)

}

ReadOnlyManhattanUserSessionStoreBuilder

.buildReadOnlyUserSessionStore(clientReadOnlyConfig, statsReceiver, scopedStatsReceiver)

}

}