package com.twitter.tsp.service

import com.twitter.abdecider.ABDeciderFactory

import com.twitter.abdecider.LoggingABDecider

import com.twitter.tsp.thriftscala.TspTweetInfo

import com.twitter.discovery.common.configapi.FeatureContextBuilder

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.gizmoduck.thriftscala.LookupContext

import com.twitter.gizmoduck.thriftscala.QueryFields

import com.twitter.gizmoduck.thriftscala.User

import com.twitter.gizmoduck.thriftscala.UserService

import com.twitter.hermit.store.gizmoduck.GizmoduckUserStore

import com.twitter.logging.Logger

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

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

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

import com.twitter.spam.rtf.thriftscala.SafetyLevel

import com.twitter.stitch.storehaus.StitchOfReadableStore

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.client.{Client => StratoClient}

import com.twitter.timelines.configapi

import com.twitter.timelines.configapi.CompositeConfig

import com.twitter.tsp.common.FeatureSwitchConfig

import com.twitter.tsp.common.FeatureSwitchesBuilder

import com.twitter.tsp.common.LoadShedder

import com.twitter.tsp.common.ParamsBuilder

import com.twitter.tsp.common.RecTargetFactory

import com.twitter.tsp.common.TopicSocialProofDecider

import com.twitter.tsp.handlers.TopicSocialProofHandler

import com.twitter.tsp.stores.LocalizedUttRecommendableTopicsStore

import com.twitter.tsp.stores.LocalizedUttTopicNameRequest

import com.twitter.tsp.stores.TopicResponses

import com.twitter.tsp.stores.TopicSocialProofStore

import com.twitter.tsp.stores.TopicSocialProofStore.TopicSocialProof

import com.twitter.tsp.stores.TopicStore

import com.twitter.tsp.stores.UttTopicFilterStore

import com.twitter.tsp.thriftscala.TopicSocialProofRequest

import com.twitter.tsp.thriftscala.TopicSocialProofResponse

import com.twitter.util.JavaTimer

import com.twitter.util.Timer

import javax.inject.Inject

import javax.inject.Singleton

import com.twitter.topiclisting.TopicListing

import com.twitter.topiclisting.utt.UttLocalization

@Singleton

class TopicSocialProofService @Inject() (

topicSocialProofStore: ReadableStore[TopicSocialProofStore.Query, Seq[TopicSocialProof]],

tweetInfoStore: ReadableStore[TweetId, TspTweetInfo],

serviceIdentifier: ServiceIdentifier,

stratoClient: StratoClient,

gizmoduck: UserService.MethodPerEndpoint,

topicListing: TopicListing,

uttLocalization: UttLocalization,

decider: TopicSocialProofDecider,

loadShedder: LoadShedder,

stats: StatsReceiver) {

import TopicSocialProofService.\_

private val statsReceiver = stats.scope("topic-social-proof-management")

private val isProd: Boolean = serviceIdentifier.environment == "prod"

private val optOutStratoStorePath: String =

if (isProd) "interests/optOutInterests" else "interests/staging/optOutInterests"

private val notInterestedInStorePath: String =

if (isProd) "interests/notInterestedTopicsGetter"

else "interests/staging/notInterestedTopicsGetter"

private val userOptOutTopicsStore: ReadableStore[UserId, TopicResponses] =

TopicStore.userOptOutTopicStore(stratoClient, optOutStratoStorePath)(

statsReceiver.scope("ints\_interests\_opt\_out\_store"))

private val explicitFollowingTopicsStore: ReadableStore[UserId, TopicResponses] =

TopicStore.explicitFollowingTopicStore(stratoClient)(

statsReceiver.scope("ints\_explicit\_following\_interests\_store"))

private val userNotInterestedInTopicsStore: ReadableStore[UserId, TopicResponses] =

TopicStore.notInterestedInTopicsStore(stratoClient, notInterestedInStorePath)(

statsReceiver.scope("ints\_not\_interested\_in\_store"))

private lazy val localizedUttRecommendableTopicsStore: ReadableStore[

LocalizedUttTopicNameRequest,

Set[

SemanticCoreEntityId

]

] = new LocalizedUttRecommendableTopicsStore(uttLocalization)

implicit val timer: Timer = new JavaTimer(true)

private lazy val uttTopicFilterStore = new UttTopicFilterStore(

topicListing = topicListing,

userOptOutTopicsStore = userOptOutTopicsStore,

explicitFollowingTopicsStore = explicitFollowingTopicsStore,

notInterestedTopicsStore = userNotInterestedInTopicsStore,

localizedUttRecommendableTopicsStore = localizedUttRecommendableTopicsStore,

timer = timer,

stats = statsReceiver.scope("UttTopicFilterStore")

)

private lazy val scribeLogger: Option[Logger] = Some(Logger.get("client\_event"))

private lazy val abDecider: LoggingABDecider =

ABDeciderFactory(

abDeciderYmlPath = configRepoDirectory + "/abdecider/abdecider.yml",

scribeLogger = scribeLogger,

decider = None,

environment = Some("production"),

).buildWithLogging()

private val builder: FeatureSwitchesBuilder = FeatureSwitchesBuilder(

statsReceiver = statsReceiver.scope("featureswitches-v2"),

abDecider = abDecider,

featuresDirectory = "features/topic-social-proof/main",

configRepoDirectory = configRepoDirectory,

addServiceDetailsFromAurora = !serviceIdentifier.isLocal,

fastRefresh = !isProd

)

private lazy val overridesConfig: configapi.Config = {

new CompositeConfig(

Seq(

FeatureSwitchConfig.config

)

)

}

private val featureContextBuilder: FeatureContextBuilder = FeatureContextBuilder(builder.build())

private val paramsBuilder: ParamsBuilder = ParamsBuilder(

featureContextBuilder,

abDecider,

overridesConfig,

statsReceiver.scope("params")

)

private val userStore: ReadableStore[UserId, User] = {

val queryFields: Set[QueryFields] = Set(

QueryFields.Profile,

QueryFields.Account,

QueryFields.Roles,

QueryFields.Discoverability,

QueryFields.Safety,

QueryFields.Takedowns

)

val context: LookupContext = LookupContext(safetyLevel = Some(SafetyLevel.Recommendations))

GizmoduckUserStore(

client = gizmoduck,

queryFields = queryFields,

context = context,

statsReceiver = statsReceiver.scope("gizmoduck")

)

}

private val recTargetFactory: RecTargetFactory = RecTargetFactory(

abDecider,

userStore,

paramsBuilder,

statsReceiver

)

private val topicSocialProofHandler =

new TopicSocialProofHandler(

topicSocialProofStore,

tweetInfoStore,

uttTopicFilterStore,

recTargetFactory,

decider,

statsReceiver.scope("TopicSocialProofHandler"),

loadShedder,

timer)

val topicSocialProofHandlerStoreStitch: TopicSocialProofRequest => com.twitter.stitch.Stitch[

TopicSocialProofResponse

] = StitchOfReadableStore(topicSocialProofHandler.toReadableStore)

}

object TopicSocialProofService {

private val configRepoDirectory = "/usr/local/config"

}