package com.twitter.tweetypie

package config

import com.twitter.flockdb.client.StatusGraph

import com.twitter.servo.util.FutureArrow

import com.twitter.stitch.timelineservice.TimelineService.GetPerspectives

import com.twitter.tweetypie.client\_id.ClientIdHelper

import com.twitter.tweetypie.repository.DeviceSourceRepository.Type

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.serverutil.\_

import com.twitter.visibility.common.tflock.UserIsInvitedToConversationRepository

/\*\*

\* Tweetypie's read path composes results from many data sources. This

\* trait is a collection of repositories for external data access.

\* These repositories should not have (within-Tweetypie) caches,

\* deciders, etc. applied to them, since that is done when the

\* repositories are composed together. They should be the minimal

\* wrapping of the external clients in order to expose an Arrow-based

\* interface.

\*/

trait ExternalRepositories {

def card2Repo: Card2Repository.Type

def cardRepo: CardRepository.Type

def cardUsersRepo: CardUsersRepository.Type

def conversationIdRepo: ConversationIdRepository.Type

def containerAsTweetRepo: CreativesContainerMaterializationRepository.GetTweetType

def containerAsTweetFieldsRepo: CreativesContainerMaterializationRepository.GetTweetFieldsType

def deviceSourceRepo: DeviceSourceRepository.Type

def escherbirdAnnotationRepo: EscherbirdAnnotationRepository.Type

def stratoSafetyLabelsRepo: StratoSafetyLabelsRepository.Type

def stratoCommunityMembershipRepo: StratoCommunityMembershipRepository.Type

def stratoCommunityAccessRepo: StratoCommunityAccessRepository.Type

def stratoPromotedTweetRepo: StratoPromotedTweetRepository.Type

def stratoSuperFollowEligibleRepo: StratoSuperFollowEligibleRepository.Type

def stratoSuperFollowRelationsRepo: StratoSuperFollowRelationsRepository.Type

def stratoSubscriptionVerificationRepo: StratoSubscriptionVerificationRepository.Type

def unmentionedEntitiesRepo: UnmentionedEntitiesRepository.Type

def geoScrubTimestampRepo: GeoScrubTimestampRepository.Type

def mediaMetadataRepo: MediaMetadataRepository.Type

def perspectiveRepo: PerspectiveRepository.Type

def placeRepo: PlaceRepository.Type

def profileGeoRepo: ProfileGeoRepository.Type

def quoterHasAlreadyQuotedRepo: QuoterHasAlreadyQuotedRepository.Type

def lastQuoteOfQuoterRepo: LastQuoteOfQuoterRepository.Type

def relationshipRepo: RelationshipRepository.Type

def retweetSpamCheckRepo: RetweetSpamCheckRepository.Type

def tweetCountsRepo: TweetCountsRepository.Type

def tweetResultRepo: TweetResultRepository.Type

def tweetSpamCheckRepo: TweetSpamCheckRepository.Type

def urlRepo: UrlRepository.Type

def userIsInvitedToConversationRepo: UserIsInvitedToConversationRepository.Type

def userRepo: UserRepository.Type

}

class ExternalServiceRepositories(

clients: BackendClients,

statsReceiver: StatsReceiver,

settings: TweetServiceSettings,

clientIdHelper: ClientIdHelper)

extends ExternalRepositories {

lazy val card2Repo: Card2Repository.Type =

Card2Repository(clients.expandodo.getCards2, maxRequestSize = 5)

lazy val cardRepo: CardRepository.Type =

CardRepository(clients.expandodo.getCards, maxRequestSize = 5)

lazy val cardUsersRepo: CardUsersRepository.Type =

CardUsersRepository(clients.expandodo.getCardUsers)

lazy val conversationIdRepo: ConversationIdRepository.Type =

ConversationIdRepository(clients.tflockReadClient.multiSelectOne)

lazy val containerAsTweetRepo: CreativesContainerMaterializationRepository.GetTweetType =

CreativesContainerMaterializationRepository(

clients.creativesContainerService.materializeAsTweet)

lazy val containerAsTweetFieldsRepo: CreativesContainerMaterializationRepository.GetTweetFieldsType =

CreativesContainerMaterializationRepository.materializeAsTweetFields(

clients.creativesContainerService.materializeAsTweetFields)

lazy val deviceSourceRepo: Type = {

DeviceSourceRepository(

DeviceSourceParser.parseAppId,

FutureArrow(clients.passbirdClient.getClientApplications(\_))

)

}

lazy val escherbirdAnnotationRepo: EscherbirdAnnotationRepository.Type =

EscherbirdAnnotationRepository(clients.escherbird.annotate)

lazy val quoterHasAlreadyQuotedRepo: QuoterHasAlreadyQuotedRepository.Type =

QuoterHasAlreadyQuotedRepository(clients.tflockReadClient)

lazy val lastQuoteOfQuoterRepo: LastQuoteOfQuoterRepository.Type =

LastQuoteOfQuoterRepository(clients.tflockReadClient)

lazy val stratoSafetyLabelsRepo: StratoSafetyLabelsRepository.Type =

StratoSafetyLabelsRepository(clients.stratoserverClient)

lazy val stratoCommunityMembershipRepo: StratoCommunityMembershipRepository.Type =

StratoCommunityMembershipRepository(clients.stratoserverClient)

lazy val stratoCommunityAccessRepo: StratoCommunityAccessRepository.Type =

StratoCommunityAccessRepository(clients.stratoserverClient)

lazy val stratoSuperFollowEligibleRepo: StratoSuperFollowEligibleRepository.Type =

StratoSuperFollowEligibleRepository(clients.stratoserverClient)

lazy val stratoSuperFollowRelationsRepo: StratoSuperFollowRelationsRepository.Type =

StratoSuperFollowRelationsRepository(clients.stratoserverClient)

lazy val stratoPromotedTweetRepo: StratoPromotedTweetRepository.Type =

StratoPromotedTweetRepository(clients.stratoserverClient)

lazy val stratoSubscriptionVerificationRepo: StratoSubscriptionVerificationRepository.Type =

StratoSubscriptionVerificationRepository(clients.stratoserverClient)

lazy val geoScrubTimestampRepo: GeoScrubTimestampRepository.Type =

GeoScrubTimestampRepository(clients.geoScrubEventStore.getGeoScrubTimestamp)

lazy val mediaMetadataRepo: MediaMetadataRepository.Type =

MediaMetadataRepository(clients.mediaClient.getMediaMetadata)

lazy val perspectiveRepo: GetPerspectives =

GetPerspectives(clients.timelineService.getPerspectives)

lazy val placeRepo: PlaceRepository.Type =

GeoduckPlaceRepository(clients.geoHydrationLocate)

lazy val profileGeoRepo: ProfileGeoRepository.Type =

ProfileGeoRepository(clients.gnipEnricherator.hydrateProfileGeo)

lazy val relationshipRepo: RelationshipRepository.Type =

RelationshipRepository(clients.socialGraphService.exists, maxRequestSize = 6)

lazy val retweetSpamCheckRepo: RetweetSpamCheckRepository.Type =

RetweetSpamCheckRepository(clients.scarecrow.checkRetweet)

lazy val tweetCountsRepo: TweetCountsRepository.Type =

TweetCountsRepository(

clients.tflockReadClient,

maxRequestSize = settings.tweetCountsRepoChunkSize

)

lazy val tweetResultRepo: TweetResultRepository.Type =

ManhattanTweetRepository(

clients.tweetStorageClient.getTweet,

clients.tweetStorageClient.getStoredTweet,

settings.shortCircuitLikelyPartialTweetReads,

statsReceiver.scope("manhattan\_tweet\_repo"),

clientIdHelper,

)

lazy val tweetSpamCheckRepo: TweetSpamCheckRepository.Type =

TweetSpamCheckRepository(clients.scarecrow.checkTweet2)

lazy val unmentionedEntitiesRepo: UnmentionedEntitiesRepository.Type =

UnmentionedEntitiesRepository(clients.stratoserverClient)

lazy val urlRepo: UrlRepository.Type =

UrlRepository(

clients.talon.expand,

settings.thriftClientId.name,

statsReceiver.scope("talon\_url\_repo"),

clientIdHelper,

)

lazy val userRepo: UserRepository.Type =

GizmoduckUserRepository(

clients.gizmoduck.getById,

clients.gizmoduck.getByScreenName,

maxRequestSize = 100

)

lazy val userIsInvitedToConversationRepo: UserIsInvitedToConversationRepository.Type =

UserIsInvitedToConversationRepository(

FutureArrow(clients.tflockReadClient.multiSelectOne(\_)),

FutureArrow((clients.tflockReadClient.contains(\_: StatusGraph, \_: Long, \_: Long)).tupled))

}