package com.twitter.timelineranker.repository

import com.twitter.search.earlybird.thriftscala.EarlybirdService

import com.twitter.servo.util.Gate

import com.twitter.timelineranker.config.RuntimeConfiguration

import com.twitter.timelineranker.model.RecapQuery

import com.twitter.timelineranker.model.RecapQuery.DependencyProvider

import com.twitter.timelineranker.visibility.SgsFollowGraphDataFields

import com.twitter.timelineranker.visibility.ScopedSgsFollowGraphDataProviderFactory

import com.twitter.timelines.clients.relevance\_search.ScopedSearchClientFactory

import com.twitter.timelines.clients.relevance\_search.SearchClient

import com.twitter.timelines.clients.user\_tweet\_entity\_graph.UserTweetEntityGraphClient

import com.twitter.timelines.util.stats.RequestScope

import com.twitter.util.Duration

import com.twitter.timelineranker.config.ClientWrapperFactories

import com.twitter.timelineranker.config.UnderlyingClientConfiguration

import com.twitter.timelineranker.visibility.FollowGraphDataProvider

import com.twitter.timelines.clients.gizmoduck.GizmoduckClient

import com.twitter.timelines.clients.manhattan.ManhattanUserMetadataClient

import com.twitter.timelines.clients.tweetypie.TweetyPieClient

abstract class CandidatesRepositoryBuilder(config: RuntimeConfiguration) extends RepositoryBuilder {

def earlybirdClient(scope: String): EarlybirdService.MethodPerEndpoint

def searchProcessingTimeout: Duration

def clientSubId: String

def requestScope: RequestScope

def followGraphDataFieldsToFetch: SgsFollowGraphDataFields.ValueSet

protected lazy val clientConfig: UnderlyingClientConfiguration = config.underlyingClients

protected lazy val clientFactories: ClientWrapperFactories = config.clientWrapperFactories

protected lazy val gizmoduckClient: GizmoduckClient =

clientFactories.gizmoduckClientFactory.scope(requestScope)

protected lazy val searchClient: SearchClient = newSearchClient(clientId = clientSubId)

protected lazy val tweetyPieHighQoSClient: TweetyPieClient =

clientFactories.tweetyPieHighQoSClientFactory.scope(requestScope)

protected lazy val tweetyPieLowQoSClient: TweetyPieClient =

clientFactories.tweetyPieLowQoSClientFactory.scope(requestScope)

protected lazy val followGraphDataProvider: FollowGraphDataProvider =

new ScopedSgsFollowGraphDataProviderFactory(

clientFactories.socialGraphClientFactory,

clientFactories.visibilityProfileHydratorFactory,

followGraphDataFieldsToFetch,

config.statsReceiver

).scope(requestScope)

protected lazy val userMetadataClient: ManhattanUserMetadataClient =

clientFactories.userMetadataClientFactory.scope(requestScope)

protected lazy val userTweetEntityGraphClient: UserTweetEntityGraphClient =

new UserTweetEntityGraphClient(

config.underlyingClients.userTweetEntityGraphClient,

config.statsReceiver

)

protected lazy val perRequestSearchClientIdProvider: DependencyProvider[Option[String]] =

DependencyProvider { recapQuery: RecapQuery =>

recapQuery.searchClientSubId.map { subId =>

clientConfig.timelineRankerClientId(Some(s"$subId.$clientSubId")).name

}

}

protected lazy val perRequestSourceSearchClientIdProvider: DependencyProvider[Option[String]] =

DependencyProvider { recapQuery: RecapQuery =>

recapQuery.searchClientSubId.map { subId =>

clientConfig.timelineRankerClientId(Some(s"$subId.${clientSubId}\_source\_tweets")).name

}

}

protected def newSearchClient(clientId: String): SearchClient =

new ScopedSearchClientFactory(

searchServiceClient = earlybirdClient(clientId),

clientId = clientConfig.timelineRankerClientId(Some(clientId)).name,

processingTimeout = Some(searchProcessingTimeout),

collectConversationIdGate = Gate.True,

statsReceiver = config.statsReceiver

).scope(requestScope)

protected def newSearchClient(

earlybirdReplacementClient: String => EarlybirdService.MethodPerEndpoint,

clientId: String

): SearchClient =

new ScopedSearchClientFactory(

searchServiceClient = earlybirdReplacementClient(clientId),

clientId = clientConfig.timelineRankerClientId(Some(clientId)).name,

processingTimeout = Some(searchProcessingTimeout),

collectConversationIdGate = Gate.True,

statsReceiver = config.statsReceiver

).scope(requestScope)

}