package com.twitter.timelineranker.entity\_tweets

import com.twitter.conversions.DurationOps.\_

import com.twitter.timelineranker.config.RequestScopes

import com.twitter.timelineranker.config.RuntimeConfiguration

import com.twitter.timelineranker.parameters.ConfigBuilder

import com.twitter.timelineranker.repository.CandidatesRepositoryBuilder

import com.twitter.timelineranker.visibility.SgsFollowGraphDataFields

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

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

import com.twitter.util.Duration

class EntityTweetsRepositoryBuilder(config: RuntimeConfiguration, configBuilder: ConfigBuilder)

extends CandidatesRepositoryBuilder(config) {

// Default client id for this repository if the upstream requests doesn't indicate one.

override val clientSubId = "community"

override val requestScope: RequestScope = RequestScopes.EntityTweetsSource

override val followGraphDataFieldsToFetch: SgsFollowGraphDataFields.ValueSet =

SgsFollowGraphDataFields.ValueSet(

SgsFollowGraphDataFields.FollowedUserIds,

SgsFollowGraphDataFields.MutuallyFollowingUserIds,

SgsFollowGraphDataFields.MutedUserIds

)

/\*\*

\* [1] timeout is derived from p9999 TLR <-> Earlybird latency and shall be less than

\* request timeout of timelineranker client within downstream timelinemixer, which is

\* 1s now

\*

\* [2] processing timeout is less than request timeout [1] with 100ms space for networking and

\* other times such as gc

\*/

override val searchProcessingTimeout: Duration = 550.milliseconds // [2]

override def earlybirdClient(scope: String): EarlybirdService.MethodPerEndpoint =

config.underlyingClients.createEarlybirdClient(

scope = scope,

requestTimeout = 650.milliseconds, // [1]

timeout = 900.milliseconds, // [1]

retryPolicy = config.underlyingClients.DefaultRetryPolicy

)

def apply(): EntityTweetsRepository = {

val entityTweetsSource = new EntityTweetsSource(

gizmoduckClient,

searchClient,

tweetyPieHighQoSClient,

userMetadataClient,

followGraphDataProvider,

clientFactories.visibilityEnforcerFactory.apply(

VisibilityRules,

RequestScopes.EntityTweetsSource

),

config.underlyingClients.contentFeaturesCache,

config.statsReceiver

)

new EntityTweetsRepository(entityTweetsSource)

}

}