package com.twitter.cr\_mixer.module.core

import com.twitter.inject.TwitterModule

import com.google.inject.Provides

import javax.inject.Singleton

import com.twitter.util.Duration

import com.twitter.app.Flag

import com.twitter.cr\_mixer.config.TimeoutConfig

/\*\*

\* All timeout settings in CrMixer.

\* Timeout numbers are defined in source/cr-mixer/server/config/deploy.aurora

\*/

object TimeoutConfigModule extends TwitterModule {

/\*\*

\* Flag names for client timeout

\* These are used in modules extending ThriftMethodBuilderClientModule

\* which cannot accept injection of TimeoutConfig

\*/

val EarlybirdClientTimeoutFlagName = "earlybird.client.timeout"

val FrsClientTimeoutFlagName = "frsSignalFetch.client.timeout"

val QigRankerClientTimeoutFlagName = "qigRanker.client.timeout"

val TweetypieClientTimeoutFlagName = "tweetypie.client.timeout"

val UserTweetGraphClientTimeoutFlagName = "userTweetGraph.client.timeout"

val UserTweetGraphPlusClientTimeoutFlagName = "userTweetGraphPlus.client.timeout"

val UserAdGraphClientTimeoutFlagName = "userAdGraph.client.timeout"

val UserVideoGraphClientTimeoutFlagName = "userVideoGraph.client.timeout"

val UtegClientTimeoutFlagName = "uteg.client.timeout"

val NaviRequestTimeoutFlagName = "navi.client.request.timeout"

/\*\*

\* Flags for timeouts

\* These are defined and initialized only in this file

\*/

// timeout for the service

private val serviceTimeout: Flag[Duration] =

flag("service.timeout", "service total timeout")

// timeout for signal fetch

private val signalFetchTimeout: Flag[Duration] =

flag[Duration]("signalFetch.timeout", "signal fetch timeout")

// timeout for similarity engine

private val similarityEngineTimeout: Flag[Duration] =

flag[Duration]("similarityEngine.timeout", "similarity engine timeout")

private val annServiceClientTimeout: Flag[Duration] =

flag[Duration]("annService.client.timeout", "annQueryService client timeout")

// timeout for user affinities fetcher

private val userStateUnderlyingStoreTimeout: Flag[Duration] =

flag[Duration]("userStateUnderlyingStore.timeout", "user state underlying store timeout")

private val userStateStoreTimeout: Flag[Duration] =

flag[Duration]("userStateStore.timeout", "user state store timeout")

private val utegSimilarityEngineTimeout: Flag[Duration] =

flag[Duration]("uteg.similarityEngine.timeout", "uteg similarity engine timeout")

private val earlybirdServerTimeout: Flag[Duration] =

flag[Duration]("earlybird.server.timeout", "earlybird server timeout")

private val earlybirdSimilarityEngineTimeout: Flag[Duration] =

flag[Duration]("earlybird.similarityEngine.timeout", "Earlybird similarity engine timeout")

private val frsBasedTweetEndpointTimeout: Flag[Duration] =

flag[Duration](

"frsBasedTweet.endpoint.timeout",

"frsBasedTweet endpoint timeout"

)

private val topicTweetEndpointTimeout: Flag[Duration] =

flag[Duration](

"topicTweet.endpoint.timeout",

"topicTweet endpoint timeout"

)

// timeout for Navi client

private val naviRequestTimeout: Flag[Duration] =

flag[Duration](

NaviRequestTimeoutFlagName,

Duration.fromMilliseconds(2000),

"Request timeout for a single RPC Call",

)

@Provides

@Singleton

def provideTimeoutBudget(): TimeoutConfig =

TimeoutConfig(

serviceTimeout = serviceTimeout(),

signalFetchTimeout = signalFetchTimeout(),

similarityEngineTimeout = similarityEngineTimeout(),

annServiceClientTimeout = annServiceClientTimeout(),

utegSimilarityEngineTimeout = utegSimilarityEngineTimeout(),

userStateUnderlyingStoreTimeout = userStateUnderlyingStoreTimeout(),

userStateStoreTimeout = userStateStoreTimeout(),

earlybirdServerTimeout = earlybirdServerTimeout(),

earlybirdSimilarityEngineTimeout = earlybirdSimilarityEngineTimeout(),

frsBasedTweetEndpointTimeout = frsBasedTweetEndpointTimeout(),

topicTweetEndpointTimeout = topicTweetEndpointTimeout(),

naviRequestTimeout = naviRequestTimeout()

)

}