package com.twitter.follow\_recommendations.services

import com.twitter.finagle.thrift.ClientId

import com.twitter.finatra.thrift.routing.ThriftWarmup

import com.twitter.follow\_recommendations.thriftscala.FollowRecommendationsThriftService.GetRecommendations

import com.twitter.follow\_recommendations.thriftscala.ClientContext

import com.twitter.follow\_recommendations.thriftscala.DebugParams

import com.twitter.follow\_recommendations.thriftscala.DisplayContext

import com.twitter.follow\_recommendations.thriftscala.DisplayLocation

import com.twitter.follow\_recommendations.thriftscala.Profile

import com.twitter.follow\_recommendations.thriftscala.RecommendationRequest

import com.twitter.inject.Logging

import com.twitter.inject.utils.Handler

import com.twitter.scrooge.Request

import com.twitter.scrooge.Response

import com.twitter.util.Return

import com.twitter.util.Throw

import com.twitter.util.Try

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class FollowRecommendationsServiceWarmupHandler @Inject() (warmup: ThriftWarmup)

extends Handler

with Logging {

private val clientId = ClientId("thrift-warmup-client")

override def handle(): Unit = {

val testIds = Seq(1L)

def warmupQuery(userId: Long, displayLocation: DisplayLocation): RecommendationRequest = {

val clientContext = ClientContext(

userId = Some(userId),

guestId = None,

appId = Some(258901L),

ipAddress = Some("0.0.0.0"),

userAgent = Some("FAKE\_USER\_AGENT\_FOR\_WARMUPS"),

countryCode = Some("US"),

languageCode = Some("en"),

isTwoffice = None,

userRoles = None,

deviceId = Some("FAKE\_DEVICE\_ID\_FOR\_WARMUPS")

)

RecommendationRequest(

clientContext = clientContext,

displayLocation = displayLocation,

displayContext = None,

maxResults = Some(3),

fetchPromotedContent = Some(false),

debugParams = Some(DebugParams(doNotLog = Some(true)))

)

}

// Add FRS display locations here if they should be targeted for warm-up

// when FRS is starting from a fresh state after a deploy

val displayLocationsToWarmUp: Seq[DisplayLocation] = Seq(

DisplayLocation.HomeTimeline,

DisplayLocation.HomeTimelineReverseChron,

DisplayLocation.ProfileSidebar,

DisplayLocation.NuxInterests,

DisplayLocation.NuxPymk

)

try {

clientId.asCurrent {

// Iterate over each user ID created for testing

testIds foreach { id =>

// Iterate over each display location targeted for warm-up

displayLocationsToWarmUp foreach { displayLocation =>

val warmupReq = warmupQuery(id, displayLocation)

info(s"Sending warm-up request to service with query: $warmupReq")

warmup.sendRequest(

method = GetRecommendations,

req = Request(GetRecommendations.Args(warmupReq)))(assertWarmupResponse)

// send the request one more time so that it goes through cache hits

warmup.sendRequest(

method = GetRecommendations,

req = Request(GetRecommendations.Args(warmupReq)))(assertWarmupResponse)

}

}

}

} catch {

case e: Throwable =>

// we don't want a warmup failure to prevent start-up

error(e.getMessage, e)

}

info("Warm-up done.")

}

/\* Private \*/

private def assertWarmupResponse(result: Try[Response[GetRecommendations.SuccessType]]): Unit = {

// we collect and log any exceptions from the result.

result match {

case Return(\_) => // ok

case Throw(exception) =>

warn()

error(s"Error performing warm-up request: ${exception.getMessage}", exception)

}

}

}