package com.twitter.graph\_feature\_service.server.controllers

import com.twitter.discovery.common.stats.DiscoveryStatsFilter

import com.twitter.finagle.Service

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.finatra.thrift.Controller

import com.twitter.graph\_feature\_service.server.handlers.ServerGetIntersectionHandler.GetIntersectionRequest

import com.twitter.graph\_feature\_service.server.handlers.ServerGetIntersectionHandler

import com.twitter.graph\_feature\_service.thriftscala

import com.twitter.graph\_feature\_service.thriftscala.Server.GetIntersection

import com.twitter.graph\_feature\_service.thriftscala.Server.GetPresetIntersection

import com.twitter.graph\_feature\_service.thriftscala.\_

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class ServerController @Inject() (

serverGetIntersectionHandler: ServerGetIntersectionHandler

)(

implicit statsReceiver: StatsReceiver)

extends Controller(thriftscala.Server) {

private val getIntersectionService: Service[GetIntersectionRequest, GfsIntersectionResponse] =

new DiscoveryStatsFilter(statsReceiver.scope("srv").scope("get\_intersection"))

.andThen(Service.mk(serverGetIntersectionHandler))

val getIntersection: Service[GetIntersection.Args, GfsIntersectionResponse] = { args =>

// TODO: Disable updateCache after HTL switch to use PresetIntersection endpoint.

getIntersectionService(

GetIntersectionRequest.fromGfsIntersectionRequest(args.request, cacheable = true))

}

handle(GetIntersection) { getIntersection }

def getPresetIntersection: Service[

GetPresetIntersection.Args,

GfsIntersectionResponse

] = { args =>

// TODO: Refactor after HTL switch to PresetIntersection

val cacheable = args.request.presetFeatureTypes == PresetFeatureTypes.HtlTwoHop

getIntersectionService(

GetIntersectionRequest.fromGfsPresetIntersectionRequest(args.request, cacheable))

}

handle(GetPresetIntersection) { getPresetIntersection }

}