package com.twitter.product\_mixer.core.controllers

import com.twitter.finagle.http.Request

import com.twitter.finagle.http.Response

import com.twitter.finagle.http.Status

import com.twitter.finagle.http.RouteIndex

import com.twitter.finatra.http.Controller

import com.twitter.scrooge.ThriftMethod

import com.twitter.inject.Injector

import com.twitter.inject.annotations.Flags

import com.twitter.product\_mixer.core.model.common.identifier.ProductIdentifier

import com.twitter.product\_mixer.core.module.product\_mixer\_flags.ProductMixerFlagModule.ServiceLocal

import com.twitter.product\_mixer.core.service.component\_registry.ComponentRegistry

import com.twitter.product\_mixer.core.service.component\_registry.{

RegisteredComponent => ComponentRegistryRegisteredComponent

}

import com.twitter.util.Future

import java.net.URLEncoder

/\*\*

\* Register endpoints necessary for enabling Product Mixer tooling such as alerts, dashboard

\* generation and Turntable.

\*

\* @param debugEndpoint a debug endpoint to run queries against. This feature is experimental and we

\* do not recommend that teams use it yet. Providing [[None]] will disable

\* debug queries.

\* @tparam ServiceIface a thrift service containing the [[debugEndpoint]]

\*/

case class ProductMixerController[ServiceIface](

injector: Injector,

debugEndpoint: ThriftMethod,

)(

implicit val serviceIFace: Manifest[ServiceIface])

extends Controller {

val isLocal: Boolean = injector.instance[Boolean](Flags.named(ServiceLocal))

if (!isLocal) {

prefix("/admin/product-mixer") {

val productNamesFut: Future[Seq[String]] =

injector.instance[ComponentRegistry].get.map { componentRegistry =>

componentRegistry.getAllRegisteredComponents.collect {

case ComponentRegistryRegisteredComponent(identifier: ProductIdentifier, \_, \_) =>

identifier.name

}

}

productNamesFut.map { productNames =>

productNames.foreach { productName =>

get(

route = "/debug-query/" + productName,

admin = true,

index = Some(RouteIndex(alias = "Query " + productName, group = "Feeds/Debug Query"))

) { \_: Request =>

val auroraPath =

URLEncoder.encode(System.getProperty("aurora.instanceKey", ""), "UTF-8")

// Extract service name from clientId since there isn't a specific flag for that

val serviceName = injector

.instance[String](Flags.named("thrift.clientId"))

.split("\\.")(0)

val redirectUrl =

s"https://feeds.twitter.biz/dtab/$serviceName/$productName?servicePath=$auroraPath"

val response = Response().status(Status.Found)

response.location = redirectUrl

response

}

}

}

}

}

prefix("/product-mixer") {

get(route = "/component-registry")(GetComponentRegistryHandler(injector).apply)

get(route = "/debug-configuration")(GetDebugConfigurationHandler(debugEndpoint).apply)

}

}