package com.twitter.simclustersann.modules

import com.twitter.finatra.mtls.thriftmux.modules.MtlsThriftWebFormsModule

import com.twitter.finatra.thrift.ThriftServer

import com.twitter.simclusters\_v2.thriftscala.EmbeddingType

import com.twitter.simclusters\_v2.thriftscala.InternalId

import com.twitter.simclusters\_v2.thriftscala.ModelVersion

import com.twitter.simclusters\_v2.thriftscala.SimClustersEmbeddingId

import com.twitter.thriftwebforms.MethodOptions

import com.twitter.thriftwebforms.view.ServiceResponseView

import com.twitter.util.Future

import com.twitter.simclustersann.thriftscala.SimClustersANNTweetCandidate

import com.twitter.simclustersann.thriftscala.Query

import com.twitter.simclustersann.thriftscala.SimClustersANNConfig

import com.twitter.simclustersann.thriftscala.ScoringAlgorithm

import com.twitter.thriftwebforms.MethodOptions.Access

import scala.reflect.ClassTag

import com.twitter.simclustersann.thriftscala.SimClustersANNService

import scala.collection.mutable

class CustomMtlsThriftWebFormsModule[T: ClassTag](server: ThriftServer)

extends MtlsThriftWebFormsModule[T](server: ThriftServer) {

private val Nbsp = "&nbsp;"

private val LdapGroups = Seq("recosplat-sensitive-data-medium", "simclusters-ann-admins")

override protected def methodOptions: Map[String, MethodOptions] = {

val tweetId = 1568796529690902529L

val sannDefaultQuery = SimClustersANNService.GetTweetCandidates.Args(

query = Query(

sourceEmbeddingId = SimClustersEmbeddingId(

embeddingType = EmbeddingType.LogFavLongestL2EmbeddingTweet,

modelVersion = ModelVersion.Model20m145k2020,

internalId = InternalId.TweetId(tweetId)

),

config = SimClustersANNConfig(

maxNumResults = 10,

minScore = 0.0,

candidateEmbeddingType = EmbeddingType.LogFavBasedTweet,

maxTopTweetsPerCluster = 400,

maxScanClusters = 50,

maxTweetCandidateAgeHours = 24,

minTweetCandidateAgeHours = 0,

annAlgorithm = ScoringAlgorithm.CosineSimilarity

)

))

Seq("getTweetCandidates")

.map(

\_ -> MethodOptions(

defaultRequestValue = Some(sannDefaultQuery),

responseRenderers = Seq(renderTimeline),

allowedAccessOverride = Some(Access.ByLdapGroup(LdapGroups))

)).toMap

}

val FullAccessLdapGroups: Seq[String] =

Seq(

"recosplat-sensitive-data-medium",

"simclusters-ann-admins",

"recos-platform-admins"

)

override protected def defaultMethodAccess: MethodOptions.Access = {

MethodOptions.Access.ByLdapGroup(FullAccessLdapGroups)

}

def renderTimeline(r: AnyRef): Future[ServiceResponseView] = {

val simClustersANNTweetCandidates = r match {

case response: Iterable[\_] =>

response.map(x => x.asInstanceOf[SimClustersANNTweetCandidate]).toSeq

case \_ => Seq()

}

renderTweets(simClustersANNTweetCandidates)

}

private def renderTweets(

simClustersANNTweetCandidates: Seq[SimClustersANNTweetCandidate]

): Future[ServiceResponseView] = {

val htmlSb = new mutable.StringBuilder()

val headerHtml = s"""<h3>Tweet Candidates</h3>"""

val tweetsHtml = simClustersANNTweetCandidates.map { simClustersANNTweetCandidate =>

val tweetId = simClustersANNTweetCandidate.tweetId

val score = simClustersANNTweetCandidate.score

s"""<blockquote class="twitter-tweet"><a href="https://twitter.com/tweet/statuses/$tweetId"></a></blockquote> <b>score:</b> $score <br><br>"""

}.mkString

htmlSb ++= headerHtml

htmlSb ++= Nbsp

htmlSb ++= tweetsHtml

Future.value(

ServiceResponseView(

"SimClusters ANN Tweet Candidates",

htmlSb.toString(),

Seq("//platform.twitter.com/widgets.js")

)

)

}

}