package com.twitter.simclustersann.filters

import com.twitter.finagle.mtls.authentication.ServiceIdentifier

import com.twitter.finagle.Service

import com.twitter.finagle.SimpleFilter

import com.twitter.relevance\_platform.simclustersann.multicluster.ServiceNameMapper

import com.twitter.scrooge.Request

import com.twitter.scrooge.Response

import com.twitter.simclustersann.exceptions.InvalidRequestForSimClustersAnnVariantException

import com.twitter.simclustersann.thriftscala.SimClustersANNService

import com.twitter.util.Future

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class SimClustersAnnVariantFilter @Inject() (

serviceNameMapper: ServiceNameMapper,

serviceIdentifier: ServiceIdentifier,

) extends SimpleFilter[Request[SimClustersANNService.GetTweetCandidates.Args], Response[

SimClustersANNService.GetTweetCandidates.SuccessType

]] {

override def apply(

request: Request[SimClustersANNService.GetTweetCandidates.Args],

service: Service[Request[SimClustersANNService.GetTweetCandidates.Args], Response[

SimClustersANNService.GetTweetCandidates.SuccessType

]]

): Future[Response[SimClustersANNService.GetTweetCandidates.SuccessType]] = {

validateRequest(request)

service(request)

}

private def validateRequest(

request: Request[SimClustersANNService.GetTweetCandidates.Args]

): Unit = {

val modelVersion = request.args.query.sourceEmbeddingId.modelVersion

val embeddingType = request.args.query.config.candidateEmbeddingType

val actualServiceName = serviceIdentifier.service

val expectedServiceName = serviceNameMapper.getServiceName(modelVersion, embeddingType)

expectedServiceName match {

case Some(name) if name == actualServiceName => ()

case \_ =>

throw InvalidRequestForSimClustersAnnVariantException(

modelVersion,

embeddingType,

actualServiceName,

expectedServiceName)

}

}

}