package com.twitter.home\_mixer.product.list\_recommended\_users.candidate\_source

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateSource

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

import com.twitter.search.adaptive.adaptive\_results.thriftscala.AdaptiveSearchResultData

import com.twitter.search.adaptive.adaptive\_results.thriftscala.Result

import com.twitter.search.adaptive.adaptive\_results.thriftscala.ResultData

import com.twitter.search.blender.adaptive\_search.thriftscala.AdaptiveSearchResponse

import com.twitter.search.blender.adaptive\_search.thriftscala.Container

import com.twitter.search.blender.thriftscala.BlenderService

import com.twitter.search.blender.thriftscala.ThriftBlenderRequest

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class BlenderUsersCandidateSource @Inject() (

blenderClient: BlenderService.MethodPerEndpoint)

extends CandidateSource[ThriftBlenderRequest, Long] {

override val identifier: CandidateSourceIdentifier = CandidateSourceIdentifier("BlenderUsers")

override def apply(request: ThriftBlenderRequest): Stitch[Seq[Long]] = {

Stitch.callFuture(

blenderClient.serveV2(request).map { response =>

val userIdsOpt =

response.adaptiveSearchResponse.map(extractUserIdsFromAdaptiveSearchResponse)

userIdsOpt.getOrElse(Seq.empty)

}

)

}

private def extractUserIdsFromAdaptiveSearchResponse(

response: AdaptiveSearchResponse

): Seq[Long] = {

response match {

case AdaptiveSearchResponse(Some(Seq(Container(Some(results), \_))), \_, \_) =>

results.map(\_.data).collect {

case AdaptiveSearchResultData.Result(Result(ResultData.User(user), \_)) =>

user.id

}

case \_ => Seq.empty

}

}

}