package com.twitter.product\_mixer.component\_library.candidate\_source.people\_discovery

import com.twitter.peoplediscovery.api.{thriftscala => t}

import com.twitter.product\_mixer.component\_library.model.candidate.UserCandidate

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

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

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

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

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.UnexpectedCandidateResult

import com.twitter.stitch.Stitch

import com.twitter.util.logging.Logging

import javax.inject.Inject

import javax.inject.Singleton

object WhoToFollowModuleHeaderFeature extends Feature[UserCandidate, t.Header]

object WhoToFollowModuleDisplayOptionsFeature

extends Feature[UserCandidate, Option[t.DisplayOptions]]

object WhoToFollowModuleShowMoreFeature extends Feature[UserCandidate, Option[t.ShowMore]]

@Singleton

class PeopleDiscoveryCandidateSource @Inject() (

peopleDiscoveryService: t.ThriftPeopleDiscoveryService.MethodPerEndpoint)

extends CandidateSourceWithExtractedFeatures[t.GetModuleRequest, t.RecommendedUser]

with Logging {

override val identifier: CandidateSourceIdentifier =

CandidateSourceIdentifier(name = "PeopleDiscovery")

override def apply(

request: t.GetModuleRequest

): Stitch[CandidatesWithSourceFeatures[t.RecommendedUser]] = {

Stitch

.callFuture(peopleDiscoveryService.getModules(request))

.map { response: t.GetModuleResponse =>

// under the assumption getModules returns a maximum of one module

response.modules

.collectFirst { module =>

module.layout match {

case t.Layout.UserBioList(layout) =>

layoutToCandidatesWithSourceFeatures(

layout.userRecommendations,

layout.header,

layout.displayOptions,

layout.showMore)

case t.Layout.UserTweetCarousel(layout) =>

layoutToCandidatesWithSourceFeatures(

layout.userRecommendations,

layout.header,

layout.displayOptions,

layout.showMore)

}

}.getOrElse(throw PipelineFailure(UnexpectedCandidateResult, "unexpected missing module"))

}

}

private def layoutToCandidatesWithSourceFeatures(

userRecommendations: Seq[t.RecommendedUser],

header: t.Header,

displayOptions: Option[t.DisplayOptions],

showMore: Option[t.ShowMore],

): CandidatesWithSourceFeatures[t.RecommendedUser] = {

val features = FeatureMapBuilder()

.add(WhoToFollowModuleHeaderFeature, header)

.add(WhoToFollowModuleDisplayOptionsFeature, displayOptions)

.add(WhoToFollowModuleShowMoreFeature, showMore)

.build()

CandidatesWithSourceFeatures(userRecommendations, features)

}

}