package com.twitter.home\_mixer.functional\_component.candidate\_source

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

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.search.earlybird.{thriftscala => t}

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Singleton

case object EarlybirdResponseTruncatedFeature

extends FeatureWithDefaultOnFailure[t.EarlybirdRequest, Boolean] {

override val defaultValue: Boolean = false

}

case object EarlybirdBottomTweetFeature

extends FeatureWithDefaultOnFailure[t.EarlybirdRequest, Option[Long]] {

override val defaultValue: Option[Long] = None

}

@Singleton

case class EarlybirdCandidateSource @Inject() (

earlybird: t.EarlybirdService.MethodPerEndpoint)

extends CandidateSourceWithExtractedFeatures[t.EarlybirdRequest, t.ThriftSearchResult] {

override val identifier = CandidateSourceIdentifier("Earlybird")

override def apply(

request: t.EarlybirdRequest

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

Stitch.callFuture(earlybird.search(request)).map { response =>

val candidates = response.searchResults.map(\_.results).getOrElse(Seq.empty)

val features = FeatureMapBuilder()

.add(EarlybirdResponseTruncatedFeature, candidates.size == request.searchQuery.numResults)

.add(EarlybirdBottomTweetFeature, candidates.lastOption.map(\_.id))

.build()

CandidatesWithSourceFeatures(candidates, features)

}

}

}