package com.twitter.product\_mixer.component\_library.filter

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

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

import com.twitter.product\_mixer.core.functional\_component.filter.Filter

import com.twitter.product\_mixer.core.functional\_component.filter.FilterResult

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

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

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.stitch.Stitch

/\*\*

\* A [[filter]] that filters candidates based on a country code feature

\*

\* @param countryCodeFeature the feature to filter candidates on

\*/

case class TweetAuthorCountryFilter[Candidate <: BaseTweetCandidate](

countryCodeFeature: Feature[Candidate, Option[String]])

extends Filter[PipelineQuery, Candidate] {

override val identifier: FilterIdentifier = FilterIdentifier("TweetAuthorCountry")

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val userCountry = query.getCountryCode

val (keptCandidates, removedCandidates) = candidates.partition { filteredCandidate =>

val authorCountry = filteredCandidate.features.get(countryCodeFeature)

(authorCountry, userCountry) match {

case (Some(authorCountryCode), Some(userCountryCode)) =>

authorCountryCode.equalsIgnoreCase(userCountryCode)

case \_ => true

}

}

Stitch.value(

FilterResult(

kept = keptCandidates.map(\_.candidate),

removed = removedCandidates.map(\_.candidate)

)

)

}

}