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

case class TweetLanguageFilter[Candidate <: BaseTweetCandidate](

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

extends Filter[PipelineQuery, Candidate] {

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

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[Candidate]]

): Stitch[FilterResult[Candidate]] = {

val userAppLanguage = query.getLanguageCode

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

val tweetLanguage = filterCandidate.features.get(languageCodeFeature)

(tweetLanguage, userAppLanguage) match {

case (Some(tweetLanguageCode), Some(userAppLanguageCode)) =>

tweetLanguageCode.equalsIgnoreCase(userAppLanguageCode)

case \_ => true

}

}

Stitch.value(

FilterResult(

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

removed = removedCandidates.map(\_.candidate)))

}

}