package com.twitter.product\_mixer.core.functional\_component.candidate\_source

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

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

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

import com.twitter.stitch.Stitch

/\*\*

\* Retrieve Candidates from the Query

\*/

trait CandidateExtractor[-Request, +Candidate] {

def apply(query: Request): Seq[Candidate]

}

/\*\*

\* Identity extractor for returning the Request as a Seq of candidates

\*/

case class IdentityCandidateExtractor[Request]() extends CandidateExtractor[Request, Request] {

def apply(candidate: Request): Seq[Request] = Seq(candidate)

}

/\*\*

\* Retrieve Candidates from a [[Feature]] on the [[PipelineQuery]]'s FeatureMap. This extractor

\* supports a transform if the Feature value and the Seq of [[Candidate]] types do not match

\*/

trait QueryFeatureCandidateExtractor[-Query <: PipelineQuery, FeatureValue, +Candidate]

extends CandidateExtractor[Query, Candidate] {

def feature: Feature[Query, FeatureValue]

override def apply(query: Query): Seq[Candidate] =

query.features.map(featureMap => transform(featureMap.get(feature))).getOrElse(Seq.empty)

def transform(featureValue: FeatureValue): Seq[Candidate]

}

/\*\*

\* Retrieve Candidates from a [[Feature]] on the [[PipelineQuery]]'s FeatureMap. This extractor can

\* be used with a single [[Feature]] if the Feature value and the Seq of [[Candidate]] types match.

\*/

case class CandidateQueryFeatureCandidateExtractor[-Query <: PipelineQuery, Candidate](

override val feature: Feature[Query, Seq[Candidate]])

extends QueryFeatureCandidateExtractor[Query, Seq[Candidate], Candidate] {

override def transform(featureValue: Seq[Candidate]): Seq[Candidate] = featureValue

}

/\*\*

\* A [[CandidateSource]] that retrieves candidates from the Request via a [[CandidateExtractor]]

\*/

case class PassthroughCandidateSource[-Request, +Candidate](

override val identifier: CandidateSourceIdentifier,

candidateExtractor: CandidateExtractor[Request, Candidate])

extends CandidateSource[Request, Candidate] {

def apply(query: Request): Stitch[Seq[Candidate]] = Stitch.value(candidateExtractor(query))

}