package com.twitter.product\_mixer.core.pipeline.scoring

import com.twitter.product\_mixer.core.functional\_component.scorer.ScoredCandidateResult

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

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

import com.twitter.product\_mixer.core.model.common.presentation.ItemCandidateWithDetails

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

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

import com.twitter.stitch.Arrow

/\*\*

\* A Scoring Pipeline

\*

\* This is an abstract class, as we only construct these via the [[ScoringPipelineBuilder]].

\*

\* A [[ScoringPipeline]] is capable of pre-filtering candidates for scoring, performing the scoring

\* then running selection heuristics (ranking, dropping, etc) based off of the score.

\* @tparam Query the domain model for the query or request

\* @tparam Candidate the domain model for the candidate being scored

\*/

abstract class ScoringPipeline[-Query <: PipelineQuery, Candidate <: UniversalNoun[Any]]

extends Pipeline[ScoringPipeline.Inputs[Query], Seq[ScoredCandidateResult[Candidate]]] {

override private[core] val config: ScoringPipelineConfig[Query, Candidate]

override val arrow: Arrow[ScoringPipeline.Inputs[Query], ScoringPipelineResult[Candidate]]

override val identifier: ScoringPipelineIdentifier

}

object ScoringPipeline {

case class Inputs[+Query <: PipelineQuery](

query: Query,

candidates: Seq[ItemCandidateWithDetails])

}