package com.twitter.product\_mixer.component\_library.gate.any\_candidates\_without\_feature

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

import com.twitter.product\_mixer.core.functional\_component.common.CandidateScope

import com.twitter.product\_mixer.core.functional\_component.gate.QueryAndCandidateGate

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

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

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

import com.twitter.stitch.Stitch

/\*\*

\* A gate that enables a component only if any candidates are missing a specific feature.

\* You can restrict which candidates to check with the scope parameter.

\* This is most commonly used to do backfill scoring, where you can have one Scoring Pipeline that

\* might return a score feature "FeatureA" and another sequential pipeline that you only want to run

\* if the previous scoring pipeline fails to hydrate for all candidates.

\* @param identifier Unique identifier for this gate. Typically, AnyCandidatesWithout{YourFeature}.

\* @param scope A [[CandidateScope]] to specify which candidates to check.

\* @param missingFeature The feature that should be missing for any of the candidates for this gate to continue

\*/

case class AnyCandidatesWithoutFeatureGate(

override val identifier: GateIdentifier,

scope: CandidateScope,

missingFeature: Feature[\_, \_])

extends QueryAndCandidateGate[PipelineQuery] {

override def shouldContinue(

query: PipelineQuery,

candidates: Seq[CandidateWithDetails]

): Stitch[Boolean] =

Stitch.value(scope.partition(candidates).candidatesInScope.exists { candidateWithDetails =>

!candidateWithDetails.features.getSuccessfulFeatures.contains(missingFeature)

})

}