package com.twitter.product\_mixer.component\_library.scorer.param\_gated

import com.twitter.product\_mixer.component\_library.scorer.param\_gated.ParamGatedScorer.IdentifierPrefix

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

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.functional\_component.common.alert.Alert

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

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

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

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

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

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

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.Param

/\*\*

\* A [[scorer]] with [[Conditionally]] based on a [[Param]]

\*

\* @param enabledParam the param to turn this [[scorer]] on and off

\* @param scorer the underlying [[scorer]] to run when `enabledParam` is true

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

\* @tparam Result The type of the candidates

\*/

case class ParamGatedScorer[-Query <: PipelineQuery, Result <: UniversalNoun[Any]](

enabledParam: Param[Boolean],

scorer: Scorer[Query, Result])

extends Scorer[Query, Result]

with Conditionally[Query] {

override val identifier: ScorerIdentifier = ScorerIdentifier(

IdentifierPrefix + scorer.identifier.name)

override val alerts: Seq[Alert] = scorer.alerts

override val features: Set[Feature[\_, \_]] = scorer.features

override def onlyIf(query: Query): Boolean =

Conditionally.and(query, scorer, query.params(enabledParam))

override def apply(

query: Query,

candidates: Seq[CandidateWithFeatures[Result]]

): Stitch[Seq[FeatureMap]] = scorer(query, candidates)

}

object ParamGatedScorer {

val IdentifierPrefix = "ParamGated"

}