package com.twitter.product\_mixer.component\_library.feature\_hydrator.query.qualityfactor\_gated

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.feature\_hydrator.QueryFeatureHydrator

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.ComponentIdentifier

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

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

import com.twitter.product\_mixer.core.quality\_factor.HasQualityFactorStatus

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.Param

object QualityFactorGatedQueryFeatureHydrator {

val IdentifierPrefix = "QfGated"

}

/\*\*

\* A [[QueryFeatureHydrator]] with [[Conditionally]] based on a qualityFactor threshold.

\* @param pipelineIdentifier identifier of the pipeline that associated with observed quality factor

\* @param qualityFactorInclusiveThreshold the threshold of the quality factor that results in the hydrator being turned off

\* @param queryFeatureHydrator the underlying [[QueryFeatureHydrator]] to run when quality factor value

\* is above the given inclusive threshold

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

\* @tparam Result The type of the candidates

\*/

case class QualityFactorGatedQueryFeatureHydrator[

-Query <: PipelineQuery with HasQualityFactorStatus,

Result <: UniversalNoun[Any]

](

pipelineIdentifier: ComponentIdentifier,

qualityFactorInclusiveThreshold: Param[Double],

queryFeatureHydrator: QueryFeatureHydrator[Query])

extends QueryFeatureHydrator[Query]

with Conditionally[Query] {

import QualityFactorGatedQueryFeatureHydrator.\_

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier(

IdentifierPrefix + queryFeatureHydrator.identifier.name)

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

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

override def onlyIf(query: Query): Boolean = Conditionally.and(

query,

queryFeatureHydrator,

query.getQualityFactorCurrentValue(pipelineIdentifier) >= query.params(

qualityFactorInclusiveThreshold))

override def hydrate(query: Query): Stitch[FeatureMap] = queryFeatureHydrator.hydrate(query)

}