package com.twitter.product\_mixer.component\_library.feature\_hydrator.query.param\_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.FeatureHydratorIdentifier

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

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.Param

/\*\*

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

\*

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

\* @param queryFeatureHydrator the underlying [[QueryFeatureHydrator]] 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 ParamGatedQueryFeatureHydrator[-Query <: PipelineQuery, Result <: UniversalNoun[Any]](

enabledParam: Param[Boolean],

queryFeatureHydrator: QueryFeatureHydrator[Query])

extends QueryFeatureHydrator[Query]

with Conditionally[Query] {

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier(

"ParamGated" + 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.params(enabledParam))

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

}