package com.twitter.product\_mixer.core.functional\_component.feature\_hydrator

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

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

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

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

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

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

import com.twitter.stitch.Stitch

/\*\*

\* Hydrate features about the query itself (not about the candidates)

\* e.g. features about the user who is making the request, what country the request originated from, etc.

\*

\* @note [[BaseQueryFeatureHydrator]]s populate [[Feature]]s with last-write-wins semantics for

\* duplicate [[Feature]]s, where the last hydrator to run that populates a [[Feature]] will

\* override any previously run [[BaseQueryFeatureHydrator]]s values for that [[Feature]].

\* In a [[com.twitter.product\_mixer.core.pipeline.PipelineConfig PipelineConfig]] this means

\* that the right-most [[BaseQueryFeatureHydrator]] to populate a given [[Feature]] will be

\* the value that is available to use.

\*

\* @note if you want to conditionally run a [[BaseQueryFeatureHydrator]] you can use the mixin [[com.twitter.product\_mixer.core.model.common.Conditionally]]

\* or to gate on a [[com.twitter.timelines.configapi.Param]] you can use [[com.twitter.product\_mixer.component\_library.feature\_hydrator.query.param\_gated.ParamGatedQueryFeatureHydrator]]

\*

\* @note Any exceptions that are thrown or returned as [[Stitch.exception]] will be added to the

\* [[FeatureMap]] for the [[Feature]]s that were supposed to be hydrated.

\* Accessing a failed Feature will throw if using [[FeatureMap.get]] for Features that aren't

\* [[com.twitter.product\_mixer.core.feature.FeatureWithDefaultOnFailure]]

\*/

trait BaseQueryFeatureHydrator[-Query <: PipelineQuery, FeatureType <: Feature[\_, \_]]

extends FeatureHydrator[FeatureType]

with SupportsConditionally[Query] {

override val identifier: FeatureHydratorIdentifier

/\*\* Hydrates a [[FeatureMap]] for a given [[Query]] \*/

def hydrate(query: Query): Stitch[FeatureMap]

}

trait QueryFeatureHydrator[-Query <: PipelineQuery]

extends BaseQueryFeatureHydrator[Query, Feature[\_, \_]]

/\*\*

\* When an [[AsyncHydrator]] is run it will hydrate features in the background

\* and will make them available starting at the specified point in execution.

\*

\* When `hydrateBefore` is reached, any duplicate [[Feature]]s that were already hydrated will be

\* overridden with the new value from the [[AsyncHydrator]]

\*

\* @note [[AsyncHydrator]]s have the same last-write-wins semantics for duplicate [[Feature]]s

\* as [[BaseQueryFeatureHydrator]] but with some nuance. If [[AsyncHydrator]]s for the

\* same [[Feature]] have the same `hydrateBefore` then the right-most [[AsyncHydrator]]s

\* value takes precedence. Similarly, [[AsyncHydrator]]s always hydrate after any other

\* [[BaseQueryFeatureHydrator]]. See the examples for more detail.

\* @example if [[QueryFeatureHydrator]]s that populate the same [[Feature]] are defined in a `PipelineConfig`

\* such as `[ asyncHydratorForFeatureA, normalHydratorForFeatureA ]`, where `asyncHydratorForFeatureA`

\* is an [[AsyncHydrator]], when `asyncHydratorForFeatureA` reaches it's `hydrateBefore`

\* Step in the Pipeline, the value for `FeatureA` from the `asyncHydratorForFeatureA` will override

\* the existing value from `normalHydratorForFeatureA`, even though in the initial `PipelineConfig`

\* they are ordered differently.

\* @example if [[AsyncHydrator]]s that populate the same [[Feature]] are defined in a `PipelineConfig`

\* such as `[ asyncHydratorForFeatureA1, asyncHydratorForFeatureA2 ]`, where both [[AsyncHydrator]]s

\* have the same `hydrateBefore`, when `hydrateBefore` is reached, the value for `FeatureA` from

\* `asyncHydratorForFeatureA2` will override the value from `asyncHydratorForFeatureA1`.

\*/

trait AsyncHydrator {

\_: BaseQueryFeatureHydrator[\_, \_] =>

/\*\*

\* A [[PipelineStepIdentifier]] from the [[com.twitter.product\_mixer.core.pipeline.PipelineConfig]] this is used in

\* by which the [[FeatureMap]] returned by this [[AsyncHydrator]] will be completed.

\*

\* Access to the [[Feature]]s from this [[AsyncHydrator]] prior to reaching the provided

\* [[PipelineStepIdentifier]]s will result in a [[com.twitter.product\_mixer.core.feature.featuremap.MissingFeatureException]].

\*

\* @note If [[PipelineStepIdentifier]] is a Step which is run in parallel, the [[Feature]]s will be available for all the parallel Steps.

\*/

def hydrateBefore: PipelineStepIdentifier

}