package com.twitter.product\_mixer.core.feature

/\*\*

\* A [[Feature]] is a single measurable or computable property of an entity.

\*

\* @note If a [[Feature]] is optional then the [[Value]] should be `Option[Value]`

\*

\* @note If a [[Feature]] is populated with a [[com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.FeatureHydrator]]

\* and the hydration fails, a failure will be stored for the [[Feature]].

\* If that [[Feature]] is accessed with [[com.twitter.product\_mixer.core.feature.featuremap.FeatureMap.get]]

\* then the stored exception will be thrown, essentially failing-closed.

\* You can use [[FeatureWithDefaultOnFailure]] or [[com.twitter.product\_mixer.core.feature.featuremap.FeatureMap.getOrElse]]

\* instead to avoid these issues and instead fail-open.

\* If correctly hydrating a Feature's value is essential to the request being correct,

\* then you should fail-closed on failure to hydrate it by extending [[Feature]] directly.

\*

\* This does not apply to [[Feature]]s from [[com.twitter.product\_mixer.core.functional\_component.transformer.FeatureTransformer]]

\* which throw in the calling Pipeline instead of storing their failures.

\*

\* @tparam Entity The type of entity that this feature works with. This could be a User, Tweet,

\* Query, etc.

\* @tparam Value The type of the value of this feature.

\*/

trait Feature[-Entity, Value] { self =>

override def toString: String = {

Feature.getSimpleName(self.getClass)

}

}

/\*\*

\* With a [[Feature]], if the [[com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.FeatureHydrator]] fails,

\* the failure will be caught by the platform and stored in the [[com.twitter.product\_mixer.core.feature.featuremap.FeatureMap]].

\* Accessing a failed feature via [[com.twitter.product\_mixer.core.feature.featuremap.FeatureMap.get()]]

\* will throw the exception that was caught while attempting to hydrate the feature. If there's a

\* reasonable default for a [[Feature]] to fail-open with, then throwing the exception at read time

\* can be prevented by defining a `defaultValue` via [[FeatureWithDefaultOnFailure]]. When accessing

\* a failed feature via [[com.twitter.product\_mixer.core.feature.featuremap.FeatureMap.get()]]

\* for a [[FeatureWithDefaultOnFailure]], the `defaultValue` will be returned.

\*

\*

\* @note [[com.twitter.product\_mixer.core.feature.featuremap.FeatureMap.getOrElse()]] can also be used

\* to access a failed feature without throwing the exception, by defining the default via the

\* [[com.twitter.product\_mixer.core.feature.featuremap.FeatureMap.getOrElse()]] method call

\* instead of as part of the feature declaration.

\* @note This does not apply to [[FeatureWithDefaultOnFailure]]s from [[com.twitter.product\_mixer.core.functional\_component.transformer.FeatureTransformer]]

\* which throw in the calling Pipeline instead of storing their failures.

\*

\* @tparam Entity The type of entity that this feature works with. This could be a User, Tweet,

\* Query, etc.

\* @tparam Value The type of the value of this feature.

\*/

trait FeatureWithDefaultOnFailure[Entity, Value] extends Feature[Entity, Value] {

/\*\* The default value a feature should return should it fail to be hydrated \*/

def defaultValue: Value

}

trait ModelFeatureName { self: Feature[\_, \_] =>

def featureName: String

}

object Feature {

/\*\*

\* Avoid `malformed class name` exceptions due to the presence of the `$` character

\* Also strip off trailing $ signs for readability

\*/

def getSimpleName[T](c: Class[T]): String = {

c.getName.stripSuffix("$").lastIndexOf("$") match {

case -1 => c.getSimpleName.stripSuffix("$")

case index => c.getName.substring(index + 1).stripSuffix("$")

}

}

}