package com.twitter.product\_mixer.core.model.common.identifier

import com.fasterxml.jackson.databind.annotation.JsonSerialize

import com.twitter.conversions.StringOps

import scala.util.matching.Regex

/\*\*

\* Component Identifiers are a type of identifier used in product mixer to identify

\* unique components - products, pipelines, candidate sources.

\*

\* Each identifier has two parts - a type and a name. Subclasses of [[ComponentIdentifier]]

\* should hardcode the `componentType`, and be declared in this file.

\*

\* For example, a [[ProductPipelineIdentifier]] has the type "ProductPipeline".

\*

\* Component identifiers are used in:

\* - Logs

\* - Tooling

\* - Metrics

\* - Feature Switches

\*

\* A component identifier name is restricted to:

\* - 3 to 80 characters to ensure reasonable length

\* - A-Z, a-z, and Digits

\* - Must start with A-Z

\* - Digits only on the ends of "words"

\* - Examples include "AlphaSample" and "UsersLikeMe"

\* - and "SimsV2" or "Test6"

\*

\* Avoid including types like "Pipeline", "MixerPipeline" etc in your identifier. these

\* can be implied by the type itself, and will automatically be used where appropriate (logs etc).

\*/

@JsonSerialize(using = classOf[ComponentIdentifierSerializer])

abstract class ComponentIdentifier(

val componentType: String,

val name: String)

extends Equals {

val file: sourcecode.File = ""

override val toString: String = s"$name$componentType"

val snakeCase: String = StringOps.toSnakeCase(toString)

val toScopes: Seq[String] = Seq(componentType, name)

}

object ComponentIdentifier {

// Allows for CamelCase and CamelCaseVer3 styles

val AllowedCharacters: Regex = "([A-Z][A-Za-z]\*[0-9]\*)+".r

val MinLength = 3

val MaxLength = 80

/\*\*

\* When a [[ComponentIdentifier.name]] is [[BasedOnParentComponent]]

\* then when operations that depend on the [[ComponentIdentifier]]

\* are performed, like registering and stats, we will perform that

\* operation by substituting the [[ComponentIdentifier.name]] with

\* the parent component's [[ComponentIdentifier.name]].

\*/

private[core] val BasedOnParentComponent = "BasedOnParentComponent"

def isValidName(name: String): Boolean = {

name match {

case n if n.length < MinLength =>

false

case n if n.length > MaxLength =>

false

case AllowedCharacters(\_\*) =>

true

case \_ =>

false

}

}

implicit val ordering: Ordering[ComponentIdentifier] =

Ordering.by { component =>

val componentTypeRank = component match {

case \_: ProductIdentifier => 0

case \_: ProductPipelineIdentifier => 1

case \_: MixerPipelineIdentifier => 2

case \_: RecommendationPipelineIdentifier => 3

case \_: ScoringPipelineIdentifier => 4

case \_: CandidatePipelineIdentifier => 5

case \_: PipelineStepIdentifier => 6

case \_: CandidateSourceIdentifier => 7

case \_: FeatureHydratorIdentifier => 8

case \_: GateIdentifier => 9

case \_: FilterIdentifier => 10

case \_: TransformerIdentifier => 11

case \_: ScorerIdentifier => 12

case \_: DecoratorIdentifier => 13

case \_: DomainMarshallerIdentifier => 14

case \_: TransportMarshallerIdentifier => 15

case \_: SideEffectIdentifier => 16

case \_: PlatformIdentifier => 17

case \_: SelectorIdentifier => 18

case \_ => Int.MaxValue

}

// First rank by type, then by name for equivalent types for overall order stability

(componentTypeRank, component.name)

}

}

/\*\*

\* HasComponentIdentifier indicates that component has a [[ComponentIdentifier]]

\*/

trait HasComponentIdentifier {

val identifier: ComponentIdentifier

}