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

/\*\*

\* A mixin trait that can be added to a [[Component]] that's marked with [[SupportsConditionally]]

\* A [[Component]] with [[SupportsConditionally]] and [[Conditionally]] will only be run when `onlyIf` returns true

\* if `onlyIf` returns false, the [[Component]] is skipped and no stats are recorded for it.

\*

\* @note if an exception is thrown when evaluating `onlyIf`, it will bubble up to the containing `Pipeline`,

\* however the [[Component]]'s stats will not be incremented. Because of this `onlyIf` should never throw.

\*

\* @note each [[Component]] that [[SupportsConditionally]] has an implementation with in the

\* component library that will conditionally run the component based on a [[com.twitter.timelines.configapi.Param]]

\*

\* @note [[Conditionally]] functionality is wired into the Component's Executor.

\*

\* @tparam Input the input that is used to gate a component on or off

\*/

trait Conditionally[-Input] { \_: SupportsConditionally[Input] =>

/\*\*

\* if `onlyIf` returns true, the underling [[Component]] is run, otherwise it's skipped

\* @note must not throw

\*/

def onlyIf(query: Input): Boolean

}

/\*\*

\* Marker trait added to the base type for each [[Component]] which supports the [[Conditionally]] mixin

\*

\* @note this is `private[core]` because it can only be added to the base implementation of components by the Product Mixer team

\*

\* @tparam Input the input that is used to gate a component on or off if [[Conditionally]] is mixed in

\*/

private[core] trait SupportsConditionally[-Input] { \_: Component => }

object Conditionally {

/\*\*

\* Helper method for combining the [[Conditionally.onlyIf]] of an underlying [[Component]] with an additional predicate

\*/

def and[ComponentType <: Component, Input](

query: Input,

component: ComponentType with SupportsConditionally[Input],

onlyIf: Boolean

): Boolean =

onlyIf && {

component match {

// @unchecked is safe here because the type parameter is guaranteed by

// the `SupportsConditionally[Input]` type parameter

case underlying: Conditionally[Input @unchecked] =>

underlying.onlyIf(query)

case \_ =>

true

}

}

}