package com.twitter.product\_mixer.component\_library.side\_effect.metrics

import com.twitter.clientapp.thriftscala.LogEvent

import com.twitter.logpipeline.client.common.EventPublisher

import com.twitter.product\_mixer.component\_library.side\_effect.ScribeClientEventSideEffect.EventNamespace

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

import com.twitter.product\_mixer.core.model.marshalling.HasMarshalling

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

/\*\*

\* Build [[ScribeClientEventMetricsSideEffect]] with extra [[EventConfig]]

\*/

case class ScribeClientEventMetricsSideEffectBuilder(

eventConfigs: Seq[EventConfig] = Seq.empty) {

/\*\*

\* Append extra [[EventConfig]] to [[ScribeClientEventMetricsSideEffectBuilder]]

\*/

def withEventConfig(

eventConfig: EventConfig

): ScribeClientEventMetricsSideEffectBuilder =

this.copy(eventConfigs = this.eventConfigs :+ eventConfig)

/\*\*

\* Build [[EventConfig]] with customized [[EventNamespace]] and customized [[CandidateMetricFunction]]

\* @param eventNamespaceOverride Override the default event namespace in [[ScribeClientEventMetricsSideEffect]]

\* @param metricFunction [[CandidateMetricFunction]]

\*/

def withEventConfig(

eventNamespaceOverride: EventNamespace,

metricFunction: CandidateMetricFunction

): ScribeClientEventMetricsSideEffectBuilder =

withEventConfig(EventConfig(eventNamespaceOverride, metricFunction))

/\*\*

\* Log served tweets events server side and build metrics in the metric center.

\* Default event name space action is "served\_tweets", default metric function is [[DefaultServedTweetsSumFunction]]

\* @param eventNamespaceOverride Override the default event namespace in [[ScribeClientEventMetricsSideEffect]]

\* @param metricFunction [[CandidateMetricFunction]]

\*/

def withServedTweets(

eventNamespaceOverride: EventNamespace = EventNamespace(action = Some("served\_tweets")),

metricFunction: CandidateMetricFunction = DefaultServedTweetsSumFunction

): ScribeClientEventMetricsSideEffectBuilder = withEventConfig(

eventNamespaceOverride = eventNamespaceOverride,

metricFunction = metricFunction)

/\*\*

\* Log served users events server side and build metrics in the metric center.

\* Default event name space action is "served\_users", default metric function is [[DefaultServedUsersSumFunction]]

\* @param eventNamespaceOverride Override the default event namespace in [[ScribeClientEventMetricsSideEffect]]

\* @param metricFunction [[CandidateMetricFunction]]

\*/

def withServedUsers(

eventNamespaceOverride: EventNamespace = EventNamespace(action = Some("served\_users")),

metricFunction: CandidateMetricFunction = DefaultServedUsersSumFunction

): ScribeClientEventMetricsSideEffectBuilder = withEventConfig(

eventNamespaceOverride = eventNamespaceOverride,

metricFunction = metricFunction)

/\*\*

\* Build [[ScribeClientEventMetricsSideEffect]]

\* @param identifier unique identifier of the side effect

\* @param defaultEventNamespace default event namespace to log

\* @param logPipelinePublisher [[EventPublisher]] to publish events

\* @param page The page which will be defined in the namespace. This is typically the service name that's scribing

\* @tparam Query [[PipelineQuery]]

\* @tparam UnmarshalledResponseType [[HasMarshalling]]

\* @return [[ScribeClientEventMetricsSideEffect]]

\*/

def build[Query <: PipelineQuery, UnmarshalledResponseType <: HasMarshalling](

identifier: SideEffectIdentifier,

defaultEventNamespace: EventNamespace,

logPipelinePublisher: EventPublisher[LogEvent],

page: String

): ScribeClientEventMetricsSideEffect[Query, UnmarshalledResponseType] = {

new ScribeClientEventMetricsSideEffect[Query, UnmarshalledResponseType](

identifier = identifier,

logPipelinePublisher = logPipelinePublisher,

defaultEventNamespace = defaultEventNamespace,

page = page,

eventConfigs = eventConfigs)

}

}