package com.twitter.graph\_feature\_service.scalding

import com.twitter.scalding.\_

import com.twitter.scalding\_internal.job.TwitterExecutionApp

import com.twitter.scalding\_internal.job.analytics\_batch.{

AnalyticsBatchExecution,

AnalyticsBatchExecutionArgs,

BatchDescription,

BatchFirstTime,

BatchIncrement,

TwitterScheduledExecutionApp

}

import java.util.TimeZone

/\*\*

\* Each job only needs to implement this runOnDateRange() function. It makes it easier for testing.

\*/

trait GraphFeatureServiceBaseJob {

implicit val timeZone: TimeZone = DateOps.UTC

implicit val dateParser: DateParser = DateParser.default

def runOnDateRange(

enableValueGraphs: Option[Boolean] = None,

enableKeyGraphs: Option[Boolean] = None

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit]

/\*\*

\* Print customized counters in the log

\*/

def printerCounters[T](execution: Execution[T]): Execution[Unit] = {

execution.getCounters

.flatMap {

case (\_, counters) =>

counters.toMap.toSeq

.sortBy(e => (e.\_1.group, e.\_1.counter))

.foreach {

case (statKey, value) =>

println(s"${statKey.group}\t${statKey.counter}\t$value")

}

Execution.unit

}

}

}

/\*\*

\* Trait that wraps things about adhoc jobs.

\*/

trait GraphFeatureServiceAdhocBaseApp extends TwitterExecutionApp with GraphFeatureServiceBaseJob {

override def job: Execution[Unit] = Execution.withId { implicit uniqueId =>

Execution.getArgs.flatMap { args: Args =>

implicit val dateRange: DateRange = DateRange.parse(args.list("date"))(timeZone, dateParser)

printerCounters(runOnDateRange())

}

}

}

/\*\*

\* Trait that wraps things about scheduled jobs.

\*

\* A new daily app only needs to declare the starting date.

\*/

trait GraphFeatureServiceScheduledBaseApp

extends TwitterScheduledExecutionApp

with GraphFeatureServiceBaseJob {

def firstTime: RichDate // for example: RichDate("2018-02-21")

def batchIncrement: Duration = Days(1)

override def scheduledJob: Execution[Unit] = Execution.withId { implicit uniqueId =>

val analyticsArgs = AnalyticsBatchExecutionArgs(

batchDesc = BatchDescription(getClass.getName),

firstTime = BatchFirstTime(firstTime),

batchIncrement = BatchIncrement(batchIncrement)

)

AnalyticsBatchExecution(analyticsArgs) { implicit dateRange =>

printerCounters(runOnDateRange())

}

}

}