package com.twitter.timelineranker.model

import com.twitter.timelineranker.{thriftscala => thrift}

import com.twitter.timelines.model.UserId

import com.twitter.timelineservice.model.TimelineId

object TimelineQuery {

def fromThrift(query: thrift.TimelineQuery): TimelineQuery = {

val queryType = query.queryType

val id = TimelineId.fromThrift(query.timelineId)

val maxCount = query.maxCount

val range = query.range.map(TimelineRange.fromThrift)

val options = query.options.map(TimelineQueryOptions.fromThrift)

queryType match {

case thrift.TimelineQueryType.Ranked =>

val rankedOptions = getRankedOptions(options)

RankedTimelineQuery(id, maxCount, range, rankedOptions)

case thrift.TimelineQueryType.ReverseChron =>

val reverseChronOptions = getReverseChronOptions(options)

ReverseChronTimelineQuery(id, maxCount, range, reverseChronOptions)

case \_ =>

throw new IllegalArgumentException(s"Unsupported query type: $queryType")

}

}

def getRankedOptions(

options: Option[TimelineQueryOptions]

): Option[RankedTimelineQueryOptions] = {

options.map {

case o: RankedTimelineQueryOptions => o

case \_ =>

throw new IllegalArgumentException(

"Only RankedTimelineQueryOptions are supported when queryType is TimelineQueryType.Ranked"

)

}

}

def getReverseChronOptions(

options: Option[TimelineQueryOptions]

): Option[ReverseChronTimelineQueryOptions] = {

options.map {

case o: ReverseChronTimelineQueryOptions => o

case \_ =>

throw new IllegalArgumentException(

"Only ReverseChronTimelineQueryOptions are supported when queryType is TimelineQueryType.ReverseChron"

)

}

}

}

abstract class TimelineQuery(

private val queryType: thrift.TimelineQueryType,

val id: TimelineId,

val maxCount: Option[Int],

val range: Option[TimelineRange],

val options: Option[TimelineQueryOptions]) {

throwIfInvalid()

def userId: UserId = {

id.id

}

def throwIfInvalid(): Unit = {

Timeline.throwIfIdInvalid(id)

range.foreach(\_.throwIfInvalid())

options.foreach(\_.throwIfInvalid())

}

def toThrift: thrift.TimelineQuery = {

thrift.TimelineQuery(

queryType = queryType,

timelineId = id.toThrift,

maxCount = maxCount,

range = range.map(\_.toTimelineRangeThrift),

options = options.map(\_.toTimelineQueryOptionsThrift)

)

}

}