package com.twitter.timelineranker.model

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

import com.twitter.timelines.model.TweetId

object TweetIdRange {

val default: TweetIdRange = TweetIdRange(None, None)

val empty: TweetIdRange = TweetIdRange(Some(0L), Some(0L))

def fromThrift(range: thrift.TweetIdRange): TweetIdRange = {

TweetIdRange(fromId = range.fromId, toId = range.toId)

}

def fromTimelineRange(range: TimelineRange): TweetIdRange = {

range match {

case r: TweetIdRange => r

case \_ =>

throw new IllegalArgumentException(s"Only Tweet ID range is supported. Found: $range")

}

}

}

/\*\*

\* A range of Tweet IDs with exclusive bounds.

\*/

case class TweetIdRange(fromId: Option[TweetId] = None, toId: Option[TweetId] = None)

extends TimelineRange {

throwIfInvalid()

def throwIfInvalid(): Unit = {

(fromId, toId) match {

case (Some(fromTweetId), Some(toTweetId)) =>

require(fromTweetId <= toTweetId, "fromId must be less than or equal to toId.")

case \_ => // valid, do nothing.

}

}

def toThrift: thrift.TweetIdRange = {

thrift.TweetIdRange(fromId = fromId, toId = toId)

}

def toTimelineRangeThrift: thrift.TimelineRange = {

thrift.TimelineRange.TweetIdRange(toThrift)

}

def isEmpty: Boolean = {

(fromId, toId) match {

case (Some(fromId), Some(toId)) if fromId == toId => true

case \_ => false

}

}

}