package com.twitter.cr\_mixer.ranker

import com.twitter.cr\_mixer.model.BlendedCandidate

import com.twitter.cr\_mixer.model.CrCandidateGeneratorQuery

import com.twitter.cr\_mixer.model.RankedCandidate

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.util.Future

import com.twitter.util.JavaTimer

import com.twitter.util.Time

import com.twitter.util.Timer

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* CR-Mixer internal ranker

\*/

@Singleton

class SwitchRanker @Inject() (

defaultRanker: DefaultRanker,

globalStats: StatsReceiver) {

private val stats: StatsReceiver = globalStats.scope(this.getClass.getCanonicalName)

implicit val timer: Timer = new JavaTimer(true)

def rank(

query: CrCandidateGeneratorQuery,

candidates: Seq[BlendedCandidate],

): Future[Seq[RankedCandidate]] = {

defaultRanker.rank(candidates)

}

}

object SwitchRanker {

/\*\* Prefers candidates generated from sources with the latest timestamps.

\* The newer the source signal, the higher a candidate ranks.

\* This ordering biases against consumer-based candidates because their timestamp defaults to 0

\*/

val TimestampOrder: Ordering[RankedCandidate] =

math.Ordering

.by[RankedCandidate, Time](

\_.reasonChosen.sourceInfoOpt

.flatMap(\_.sourceEventTime)

.getOrElse(Time.fromMilliseconds(0L)))

.reverse

}