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

import com.twitter.product\_mixer.component\_library.model.candidate.BaseTweetCandidate

import com.twitter.product\_mixer.component\_library.model.candidate.BaseUserCandidate

import com.twitter.product\_mixer.component\_library.side\_effect.metrics.CandidateMetricFunction.getCountForType

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.product\_mixer.core.model.common.presentation.ItemCandidateWithDetails

/\*\*

\* Function to extract numerical metric value from [[CandidateWithDetails]].

\* This CandidateMetricFunction will be applied on all [[CandidateWithDetails]] instances in the

\* candidateSelection from the RecommendationPipeline.

\*/

trait CandidateMetricFunction {

def apply(candidateWithDetails: CandidateWithDetails): Long

}

object CandidateMetricFunction {

private val defaultCountOnePf: PartialFunction[CandidateWithDetails, Long] = {

case \_ => 0L

}

/\*\*

\* Count the occurrences of a certain candidate type from [[CandidateWithDetails]].

\*/

def getCountForType(

candidateWithDetails: CandidateWithDetails,

countOnePf: PartialFunction[CandidateWithDetails, Long]

): Long = {

(countOnePf orElse defaultCountOnePf)(candidateWithDetails)

}

}

object DefaultServedTweetsSumFunction extends CandidateMetricFunction {

override def apply(candidateWithDetails: CandidateWithDetails): Long =

getCountForType(

candidateWithDetails,

{

case item: ItemCandidateWithDetails =>

item.candidate match {

case \_: BaseTweetCandidate => 1L

case \_ => 0L

}

})

}

object DefaultServedUsersSumFunction extends CandidateMetricFunction {

override def apply(candidateWithDetails: CandidateWithDetails): Long =

getCountForType(

candidateWithDetails,

{

case item: ItemCandidateWithDetails =>

item.candidate match {

case \_: BaseUserCandidate => 1L

case \_ => 0L

}

})

}