package com.twitter.home\_mixer.functional\_component.selector

import com.twitter.product\_mixer.component\_library.model.presentation.urt.UrtModulePresentation

import com.twitter.product\_mixer.core.functional\_component.common.CandidateScope

import com.twitter.product\_mixer.core.functional\_component.common.CandidateScope.PartitionedCandidates

import com.twitter.product\_mixer.core.functional\_component.selector.Selector

import com.twitter.product\_mixer.core.functional\_component.selector.SelectorResult

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

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

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

/\*\*

\* This selector updates the id of the conversation modules to be the head of the module's id.

\*/

case class UpdateConversationModuleId(

override val pipelineScope: CandidateScope)

extends Selector[PipelineQuery] {

override def apply(

query: PipelineQuery,

remainingCandidates: Seq[CandidateWithDetails],

result: Seq[CandidateWithDetails]

): SelectorResult = {

val PartitionedCandidates(selectedCandidates, otherCandidates) =

pipelineScope.partition(remainingCandidates)

val updatedCandidates = selectedCandidates.map {

case module @ ModuleCandidateWithDetails(candidates, presentationOpt, \_) =>

val updatedPresentation = presentationOpt.map {

case urtModule @ UrtModulePresentation(timelineModule) =>

urtModule.copy(timelineModule =

timelineModule.copy(id = candidates.head.candidateIdLong))

}

module.copy(presentation = updatedPresentation)

case candidate => candidate

}

SelectorResult(remainingCandidates = updatedCandidates ++ otherCandidates, result = result)

}

}