package com.twitter.product\_mixer.component\_library.candidate\_source.flexible\_injection\_pipeline

import com.twitter.inject.Logging

import com.twitter.onboarding.injections.{thriftscala => injectionsthrift}

import com.twitter.onboarding.task.service.{thriftscala => servicethrift}

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateSource

import com.twitter.product\_mixer.core.model.common.identifier.CandidateSourceIdentifier

import com.twitter.stitch.Stitch

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* Returns a list of prompts to insert into a user's timeline (inline prompt, cover modals, etc)

\* from go/flip (the prompting platform for Twitter).

\*/

@Singleton

class PromptCandidateSource @Inject() (taskService: servicethrift.TaskService.MethodPerEndpoint)

extends CandidateSource[servicethrift.GetInjectionsRequest, IntermediatePrompt]

with Logging {

override val identifier: CandidateSourceIdentifier = CandidateSourceIdentifier(

"InjectionPipelinePrompts")

override def apply(

request: servicethrift.GetInjectionsRequest

): Stitch[Seq[IntermediatePrompt]] = {

Stitch

.callFuture(taskService.getInjections(request)).map {

\_.injections.flatMap {

// The entire carousel is getting added to each IntermediatePrompt item with a

// corresponding index to be unpacked later on to populate its TimelineEntry counterpart.

case injection: injectionsthrift.Injection.TilesCarousel =>

injection.tilesCarousel.tiles.zipWithIndex.map {

case (tile: injectionsthrift.Tile, index: Int) =>

IntermediatePrompt(injection, Some(index), Some(tile))

}

case injection => Seq(IntermediatePrompt(injection, None, None))

}

}

}

}

/\*\*

\* Gives an intermediate step to help 'explosion' of tile carousel tiles due to TimelineModule

\* not being an extension of TimelineItem

\*/

case class IntermediatePrompt(

injection: injectionsthrift.Injection,

offsetInModule: Option[Int],

carouselTile: Option[injectionsthrift.Tile])