package com.twitter.home\_mixer.product.subscribed

import com.twitter.clientapp.{thriftscala => ca}

import com.twitter.home\_mixer.candidate\_pipeline.ConversationServiceCandidatePipelineConfigBuilder

import com.twitter.home\_mixer.candidate\_pipeline.EditedTweetsCandidatePipelineConfig

import com.twitter.home\_mixer.candidate\_pipeline.NewTweetsPillCandidatePipelineConfig

import com.twitter.home\_mixer.functional\_component.decorator.HomeConversationServiceCandidateDecorator

import com.twitter.home\_mixer.functional\_component.decorator.urt.builder.HomeFeedbackActionInfoBuilder

import com.twitter.home\_mixer.functional\_component.feature\_hydrator.\_

import com.twitter.home\_mixer.functional\_component.selector.UpdateHomeClientEventDetails

import com.twitter.home\_mixer.functional\_component.selector.UpdateNewTweetsPillDecoration

import com.twitter.home\_mixer.functional\_component.side\_effect.\_

import com.twitter.home\_mixer.model.GapIncludeInstruction

import com.twitter.home\_mixer.param.HomeGlobalParams.MaxNumberReplaceInstructionsParam

import com.twitter.home\_mixer.param.HomeMixerFlagName.ScribeClientEventsFlag

import com.twitter.home\_mixer.product.following.model.HomeMixerExternalStrings

import com.twitter.home\_mixer.product.subscribed.model.SubscribedQuery

import com.twitter.home\_mixer.product.subscribed.param.SubscribedParam.ServerMaxResultsParam

import com.twitter.home\_mixer.util.CandidatesUtil

import com.twitter.inject.annotations.Flag

import com.twitter.logpipeline.client.common.EventPublisher

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.async.AsyncQueryFeatureHydrator

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.impressed\_tweets.ImpressedTweetsQueryFeatureHydrator

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.social\_graph.SGSFollowedUsersQueryFeatureHydrator

import com.twitter.product\_mixer.component\_library.feature\_hydrator.query.social\_graph.SGSSubscribedUsersQueryFeatureHydrator

import com.twitter.product\_mixer.component\_library.gate.NonEmptyCandidatesGate

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

import com.twitter.product\_mixer.component\_library.premarshaller.urt.UrtDomainMarshaller

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.AddEntriesWithReplaceAndShowAlertInstructionBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.OrderedBottomCursorBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.OrderedGapCursorBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.OrderedTopCursorBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.ReplaceAllEntries

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.ReplaceEntryInstructionBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.ShowAlertInstructionBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.StaticTimelineScribeConfigBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.UrtMetadataBuilder

import com.twitter.product\_mixer.component\_library.selector.DropMaxCandidates

import com.twitter.product\_mixer.component\_library.selector.InsertAppendResults

import com.twitter.product\_mixer.component\_library.selector.SelectConditionally

import com.twitter.product\_mixer.component\_library.selector.UpdateSortCandidates

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

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

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.QueryFeatureHydrator

import com.twitter.product\_mixer.core.functional\_component.marshaller.TransportMarshaller

import com.twitter.product\_mixer.core.functional\_component.marshaller.response.urt.UrtTransportMarshaller

import com.twitter.product\_mixer.core.functional\_component.premarshaller.DomainMarshaller

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

import com.twitter.product\_mixer.core.functional\_component.side\_effect.PipelineResultSideEffect

import com.twitter.product\_mixer.core.model.common.UniversalNoun

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.Timeline

import com.twitter.product\_mixer.core.model.marshalling.response.urt.TimelineModule

import com.twitter.product\_mixer.core.model.marshalling.response.urt.TimelineScribeConfig

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.tweet.TweetItem

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

import com.twitter.product\_mixer.core.pipeline.candidate.CandidatePipelineConfig

import com.twitter.product\_mixer.core.pipeline.candidate.DependentCandidatePipelineConfig

import com.twitter.product\_mixer.core.pipeline.mixer.MixerPipelineConfig

import com.twitter.product\_mixer.core.product.guice.scope.ProductScoped

import com.twitter.stringcenter.client.StringCenter

import com.twitter.timelines.render.{thriftscala => urt}

import javax.inject.Inject

import javax.inject.Provider

import javax.inject.Singleton

@Singleton

class SubscribedMixerPipelineConfig @Inject() (

subscribedEarlybirdCandidatePipelineConfig: SubscribedEarlybirdCandidatePipelineConfig,

conversationServiceCandidatePipelineConfigBuilder: ConversationServiceCandidatePipelineConfigBuilder[

SubscribedQuery

],

homeFeedbackActionInfoBuilder: HomeFeedbackActionInfoBuilder,

editedTweetsCandidatePipelineConfig: EditedTweetsCandidatePipelineConfig,

newTweetsPillCandidatePipelineConfig: NewTweetsPillCandidatePipelineConfig[SubscribedQuery],

dismissInfoQueryFeatureHydrator: DismissInfoQueryFeatureHydrator,

gizmoduckUserQueryFeatureHydrator: GizmoduckUserQueryFeatureHydrator,

requestQueryFeatureHydrator: RequestQueryFeatureHydrator[SubscribedQuery],

sgsFollowedUsersQueryFeatureHydrator: SGSFollowedUsersQueryFeatureHydrator,

sgsSubscribedUsersQueryFeatureHydrator: SGSSubscribedUsersQueryFeatureHydrator,

manhattanTweetImpressionsQueryFeatureHydrator: TweetImpressionsQueryFeatureHydrator[

SubscribedQuery

],

memcacheTweetImpressionsQueryFeatureHydrator: ImpressedTweetsQueryFeatureHydrator,

publishClientSentImpressionsEventBusSideEffect: PublishClientSentImpressionsEventBusSideEffect,

publishClientSentImpressionsManhattanSideEffect: PublishClientSentImpressionsManhattanSideEffect,

homeTimelineServedCandidatesSideEffect: HomeScribeServedCandidatesSideEffect,

clientEventsScribeEventPublisher: EventPublisher[ca.LogEvent],

externalStrings: HomeMixerExternalStrings,

@ProductScoped stringCenterProvider: Provider[StringCenter],

urtTransportMarshaller: UrtTransportMarshaller,

@Flag(ScribeClientEventsFlag) enableScribeClientEvents: Boolean)

extends MixerPipelineConfig[SubscribedQuery, Timeline, urt.TimelineResponse] {

override val identifier: MixerPipelineIdentifier = MixerPipelineIdentifier("Subscribed")

private val dependentCandidatesStep = MixerPipelineConfig.dependentCandidatePipelinesStep

private val resultSelectorsStep = MixerPipelineConfig.resultSelectorsStep

override val fetchQueryFeatures: Seq[QueryFeatureHydrator[SubscribedQuery]] = Seq(

requestQueryFeatureHydrator,

sgsFollowedUsersQueryFeatureHydrator,

sgsSubscribedUsersQueryFeatureHydrator,

AsyncQueryFeatureHydrator(dependentCandidatesStep, dismissInfoQueryFeatureHydrator),

AsyncQueryFeatureHydrator(dependentCandidatesStep, gizmoduckUserQueryFeatureHydrator),

AsyncQueryFeatureHydrator(resultSelectorsStep, manhattanTweetImpressionsQueryFeatureHydrator),

AsyncQueryFeatureHydrator(resultSelectorsStep, memcacheTweetImpressionsQueryFeatureHydrator)

)

private val earlybirdCandidatePipelineScope =

SpecificPipeline(subscribedEarlybirdCandidatePipelineConfig.identifier)

private val conversationServiceCandidatePipelineConfig =

conversationServiceCandidatePipelineConfigBuilder.build(

Seq(NonEmptyCandidatesGate(earlybirdCandidatePipelineScope)),

HomeConversationServiceCandidateDecorator(homeFeedbackActionInfoBuilder)

)

override val candidatePipelines: Seq[CandidatePipelineConfig[SubscribedQuery, \_, \_, \_]] =

Seq(subscribedEarlybirdCandidatePipelineConfig)

override val dependentCandidatePipelines: Seq[

DependentCandidatePipelineConfig[SubscribedQuery, \_, \_, \_]

] = Seq(

conversationServiceCandidatePipelineConfig,

editedTweetsCandidatePipelineConfig,

newTweetsPillCandidatePipelineConfig

)

override val failOpenPolicies: Map[CandidatePipelineIdentifier, FailOpenPolicy] = Map(

editedTweetsCandidatePipelineConfig.identifier -> FailOpenPolicy.Always,

newTweetsPillCandidatePipelineConfig.identifier -> FailOpenPolicy.Always,

)

override val resultSelectors: Seq[Selector[SubscribedQuery]] = Seq(

UpdateSortCandidates(

ordering = CandidatesUtil.reverseChronTweetsOrdering,

candidatePipeline = conversationServiceCandidatePipelineConfig.identifier

),

DropMaxCandidates(

candidatePipeline = editedTweetsCandidatePipelineConfig.identifier,

maxSelectionsParam = MaxNumberReplaceInstructionsParam

),

DropMaxCandidates(

candidatePipeline = conversationServiceCandidatePipelineConfig.identifier,

maxSelectionsParam = ServerMaxResultsParam

),

InsertAppendResults(candidatePipeline = conversationServiceCandidatePipelineConfig.identifier),

// This selector must come after the tweets are inserted into the results

UpdateNewTweetsPillDecoration(

pipelineScope = SpecificPipelines(

conversationServiceCandidatePipelineConfig.identifier,

newTweetsPillCandidatePipelineConfig.identifier

),

stringCenter = stringCenterProvider.get(),

seeNewTweetsString = externalStrings.seeNewTweetsString,

tweetedString = externalStrings.tweetedString

),

InsertAppendResults(candidatePipeline = editedTweetsCandidatePipelineConfig.identifier),

SelectConditionally(

selector =

InsertAppendResults(candidatePipeline = newTweetsPillCandidatePipelineConfig.identifier),

includeSelector = (\_, \_, results) => CandidatesUtil.containsType[TweetCandidate](results)

),

UpdateHomeClientEventDetails(

candidatePipelines = Set(conversationServiceCandidatePipelineConfig.identifier)

),

)

private val homeScribeClientEventSideEffect = HomeScribeClientEventSideEffect(

enableScribeClientEvents = enableScribeClientEvents,

logPipelinePublisher = clientEventsScribeEventPublisher,

injectedTweetsCandidatePipelineIdentifiers =

Seq(conversationServiceCandidatePipelineConfig.identifier),

)

override val resultSideEffects: Seq[PipelineResultSideEffect[SubscribedQuery, Timeline]] = Seq(

homeScribeClientEventSideEffect,

homeTimelineServedCandidatesSideEffect,

publishClientSentImpressionsEventBusSideEffect,

publishClientSentImpressionsManhattanSideEffect

)

override val domainMarshaller: DomainMarshaller[SubscribedQuery, Timeline] = {

val instructionBuilders = Seq(

ReplaceEntryInstructionBuilder(ReplaceAllEntries),

AddEntriesWithReplaceAndShowAlertInstructionBuilder(),

ShowAlertInstructionBuilder(),

)

val idSelector: PartialFunction[UniversalNoun[\_], Long] = {

// exclude ads while determining tweet cursor values

case item: TweetItem if item.promotedMetadata.isEmpty => item.id

case module: TimelineModule

if module.items.headOption.exists(\_.item.isInstanceOf[TweetItem]) =>

module.items.last.item match { case item: TweetItem => item.id }

}

val topCursorBuilder = OrderedTopCursorBuilder(idSelector)

val bottomCursorBuilder =

OrderedBottomCursorBuilder(idSelector, GapIncludeInstruction.inverse())

val gapCursorBuilder = OrderedGapCursorBuilder(idSelector, GapIncludeInstruction)

val metadataBuilder = UrtMetadataBuilder(

title = None,

scribeConfigBuilder = Some(

StaticTimelineScribeConfigBuilder(

TimelineScribeConfig(page = Some("subscribed"), section = None, entityToken = None)))

)

UrtDomainMarshaller(

instructionBuilders = instructionBuilders,

metadataBuilder = Some(metadataBuilder),

cursorBuilders = Seq(topCursorBuilder, bottomCursorBuilder, gapCursorBuilder)

)

}

override val transportMarshaller: TransportMarshaller[Timeline, urt.TimelineResponse] =

urtTransportMarshaller

}