package com.twitter.home\_mixer.product.for\_you.side\_effect

import com.twitter.home\_mixer.model.HomeFeatures.CandidateSourceIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.InNetworkFeature

import com.twitter.home\_mixer.model.HomeFeatures.IsReadFromCacheFeature

import com.twitter.home\_mixer.model.HomeFeatures.PredictionRequestIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.ServedIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.ServedRequestIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.SuggestTypeFeature

import com.twitter.home\_mixer.product.for\_you.param.ForYouParam.EnableServedCandidateKafkaPublishingParam

import com.twitter.home\_mixer.service.HomeMixerAlertConfig

import com.twitter.product\_mixer.component\_library.side\_effect.KafkaPublishingSideEffect

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

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

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

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

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

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

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

import com.twitter.timelines.ml.cont\_train.common.domain.non\_scalding.ServedCandidateFeatureKeysAdapter

import com.twitter.timelines.ml.cont\_train.common.domain.non\_scalding.ServedCandidateFeatureKeysFields

import com.twitter.timelines.ml.kafka.serde.CandidateFeatureKeySerde

import com.twitter.timelines.ml.kafka.serde.TBaseSerde

import com.twitter.timelines.served\_candidates\_logging.{thriftscala => sc}

import com.twitter.timelines.suggests.common.poly\_data\_record.{thriftjava => pldr}

import com.twitter.timelineservice.suggests.{thriftscala => tls}

import org.apache.kafka.clients.producer.ProducerRecord

import org.apache.kafka.common.serialization.Serializer

import scala.collection.JavaConverters.\_

/\*\*

\* Pipeline side-effect that publishes candidate keys to a Kafka topic.

\*/

class ServedCandidateFeatureKeysKafkaSideEffect(

topic: String,

sourceIdentifiers: Set[identifier.CandidatePipelineIdentifier])

extends KafkaPublishingSideEffect[

sc.CandidateFeatureKey,

pldr.PolyDataRecord,

PipelineQuery,

Timeline

]

with PipelineResultSideEffect.Conditionally[PipelineQuery, Timeline] {

import ServedCandidateKafkaSideEffect.\_

override val identifier: SideEffectIdentifier = SideEffectIdentifier("ServedCandidateFeatureKeys")

override def onlyIf(

query: PipelineQuery,

selectedCandidates: Seq[CandidateWithDetails],

remainingCandidates: Seq[CandidateWithDetails],

droppedCandidates: Seq[CandidateWithDetails],

response: Timeline

): Boolean = query.params.getBoolean(EnableServedCandidateKafkaPublishingParam)

override val bootstrapServer: String = "/s/kafka/timeline:kafka-tls"

override val keySerde: Serializer[sc.CandidateFeatureKey] =

CandidateFeatureKeySerde().serializer()

override val valueSerde: Serializer[pldr.PolyDataRecord] =

TBaseSerde.Thrift[pldr.PolyDataRecord]().serializer

override val clientId: String = "home\_mixer\_served\_candidate\_feature\_keys\_producer"

override def buildRecords(

query: PipelineQuery,

selectedCandidates: Seq[CandidateWithDetails],

remainingCandidates: Seq[CandidateWithDetails],

droppedCandidates: Seq[CandidateWithDetails],

response: Timeline

): Seq[ProducerRecord[sc.CandidateFeatureKey, pldr.PolyDataRecord]] = {

val servedRequestIdOpt =

query.features.getOrElse(FeatureMap.empty).getOrElse(ServedRequestIdFeature, None)

extractCandidates(query, selectedCandidates, sourceIdentifiers).map { candidate =>

val isReadFromCache = candidate.features.getOrElse(IsReadFromCacheFeature, false)

val servedId = candidate.features.get(ServedIdFeature).get

val key = sc.CandidateFeatureKey(

tweetId = candidate.candidateIdLong,

viewerId = query.getRequiredUserId,

servedId = servedId)

val record =

ServedCandidateFeatureKeysAdapter

.adaptToDataRecords(

ServedCandidateFeatureKeysFields(

candidateTweetSourceId = candidate.features

.getOrElse(CandidateSourceIdFeature, None).map(\_.value.toLong).getOrElse(2L),

predictionRequestId =

candidate.features.getOrElse(PredictionRequestIdFeature, None).get,

servedRequestIdOpt = if (isReadFromCache) servedRequestIdOpt else None,

servedId = servedId,

injectionModuleName = candidate.getClass.getSimpleName,

viewerFollowsOriginalAuthor =

Some(candidate.features.getOrElse(InNetworkFeature, true)),

suggestType = candidate.features

.getOrElse(SuggestTypeFeature, None).getOrElse(tls.SuggestType.RankedOrganicTweet),

finalPositionIndex = Some(candidate.sourcePosition),

isReadFromCache = isReadFromCache

)).asScala.head

new ProducerRecord(topic, key, pldr.PolyDataRecord.dataRecord(record))

}

}

override val alerts = Seq(

HomeMixerAlertConfig.BusinessHours.defaultSuccessRateAlert(98.5)

)

}