package com.twitter.home\_mixer.product.scored\_tweets

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

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

import com.twitter.home\_mixer.model.request.HomeMixerRequest

import com.twitter.home\_mixer.model.request.ScoredTweetsProduct

import com.twitter.home\_mixer.model.request.ScoredTweetsProductContext

import com.twitter.home\_mixer.product.scored\_tweets.model.ScoredTweetsQuery

import com.twitter.home\_mixer.product.scored\_tweets.param.ScoredTweetsParam.ServerMaxResultsParam

import com.twitter.home\_mixer.product.scored\_tweets.param.ScoredTweetsParamConfig

import com.twitter.home\_mixer.service.HomeMixerAccessPolicy.DefaultHomeMixerAccessPolicy

import com.twitter.home\_mixer.{thriftscala => t}

import com.twitter.product\_mixer.component\_library.premarshaller.cursor.UrtCursorSerializer

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

import com.twitter.product\_mixer.core.functional\_component.common.access\_policy.AccessPolicy

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

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

import com.twitter.product\_mixer.core.model.marshalling.request.Product

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

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.BadRequest

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

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

import com.twitter.product\_mixer.core.product.ProductParamConfig

import com.twitter.timelines.configapi.Params

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class ScoredTweetsProductPipelineConfig @Inject() (

scoredTweetsRecommendationPipelineConfig: ScoredTweetsRecommendationPipelineConfig,

scoredTweetsParamConfig: ScoredTweetsParamConfig)

extends ProductPipelineConfig[HomeMixerRequest, ScoredTweetsQuery, t.ScoredTweets] {

override val identifier: ProductPipelineIdentifier = ProductPipelineIdentifier("ScoredTweets")

override val product: Product = ScoredTweetsProduct

override val paramConfig: ProductParamConfig = scoredTweetsParamConfig

override def pipelineQueryTransformer(

request: HomeMixerRequest,

params: Params

): ScoredTweetsQuery = {

val context = request.productContext match {

case Some(context: ScoredTweetsProductContext) => context

case \_ => throw PipelineFailure(BadRequest, "ScoredTweetsProductContext not found")

}

val featureMap = FeatureMapBuilder()

.add(ServedTweetIdsFeature, context.servedTweetIds.getOrElse(Seq.empty))

.add(TimelineServiceTweetsFeature, context.backfillTweetIds.getOrElse(Seq.empty))

.build()

ScoredTweetsQuery(

params = params,

clientContext = request.clientContext,

pipelineCursor =

request.serializedRequestCursor.flatMap(UrtCursorSerializer.deserializeOrderedCursor),

requestedMaxResults = Some(params(ServerMaxResultsParam)),

debugOptions = request.debugParams.flatMap(\_.debugOptions),

features = Some(featureMap),

deviceContext = context.deviceContext,

seenTweetIds = context.seenTweetIds,

qualityFactorStatus = None

)

}

override val pipelines: Seq[PipelineConfig] = Seq(scoredTweetsRecommendationPipelineConfig)

override def pipelineSelector(query: ScoredTweetsQuery): ComponentIdentifier =

scoredTweetsRecommendationPipelineConfig.identifier

override val debugAccessPolicies: Set[AccessPolicy] = DefaultHomeMixerAccessPolicy

}