package com.twitter.home\_mixer.product.list\_recommended\_users

import com.twitter.conversions.DurationOps.\_

import com.twitter.home\_mixer.marshaller.timelines.RecommendedUsersCursorUnmarshaller

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

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

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

import com.twitter.home\_mixer.product.list\_recommended\_users.model.ListRecommendedUsersQuery

import com.twitter.home\_mixer.product.list\_recommended\_users.param.ListRecommendedUsersParam.ServerMaxResultsParam

import com.twitter.home\_mixer.product.list\_recommended\_users.param.ListRecommendedUsersParamConfig

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

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

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

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

import com.twitter.product\_mixer.core.functional\_component.common.alert.predicate.TriggerIfBelow

import com.twitter.product\_mixer.core.functional\_component.common.alert.predicate.TriggerIfLatencyAbove

import com.twitter.product\_mixer.core.functional\_component.common.alert.Alert

import com.twitter.product\_mixer.core.functional\_component.common.alert.LatencyAlert

import com.twitter.product\_mixer.core.functional\_component.common.alert.P99

import com.twitter.product\_mixer.core.functional\_component.common.alert.SuccessRateAlert

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

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

import com.twitter.timelines.util.RequestCursorSerializer

import com.twitter.util.Try

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class ListRecommendedUsersProductPipelineConfig @Inject() (

listRecommendedUsersMixerPipelineConfig: ListRecommendedUsersMixerPipelineConfig,

listRecommendedUsersParamConfig: ListRecommendedUsersParamConfig)

extends ProductPipelineConfig[

HomeMixerRequest,

ListRecommendedUsersQuery,

urt.TimelineResponse

] {

override val identifier: ProductPipelineIdentifier =

ProductPipelineIdentifier("ListRecommendedUsers")

override val product: request.Product = ListRecommendedUsersProduct

override val paramConfig: ProductParamConfig = listRecommendedUsersParamConfig

override def pipelineQueryTransformer(

request: HomeMixerRequest,

params: Params

): ListRecommendedUsersQuery = {

val context = request.productContext match {

case Some(context: ListRecommendedUsersProductContext) => context

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

}

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

val pipelineCursor = request.serializedRequestCursor.flatMap { cursor =>

Try(UrtCursorSerializer.deserializeUnorderedExcludeIdsCursor(cursor))

.getOrElse(RecommendedUsersCursorUnmarshaller(RequestCursorSerializer.deserialize(cursor)))

}

ListRecommendedUsersQuery(

listId = context.listId,

params = params,

clientContext = request.clientContext,

features = None,

pipelineCursor = pipelineCursor,

requestedMaxResults = Some(params(ServerMaxResultsParam)),

debugOptions = debugOptions,

selectedUserIds = context.selectedUserIds,

excludedUserIds = context.excludedUserIds,

listName = context.listName

)

}

override def pipelines: Seq[PipelineConfig] = Seq(listRecommendedUsersMixerPipelineConfig)

override def pipelineSelector(query: ListRecommendedUsersQuery): ComponentIdentifier =

listRecommendedUsersMixerPipelineConfig.identifier

override val alerts: Seq[Alert] = Seq(

SuccessRateAlert(

notificationGroup = DefaultNotificationGroup,

warnPredicate = TriggerIfBelow(99.9, 20, 30),

criticalPredicate = TriggerIfBelow(99.9, 30, 30),

),

LatencyAlert(

notificationGroup = DefaultNotificationGroup,

percentile = P99,

warnPredicate = TriggerIfLatencyAbove(1000.millis, 15, 30),

criticalPredicate = TriggerIfLatencyAbove(1500.millis, 15, 30)

)

)

override val debugAccessPolicies: Set[AccessPolicy] = DefaultHomeMixerAccessPolicy

}