package com.twitter.product\_mixer.core.service.urt

import com.fasterxml.jackson.databind.SerializationFeature

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

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

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

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

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

import com.twitter.product\_mixer.core.product.registry.ProductPipelineRegistry

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

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.Params

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

import com.twitter.util.jackson.ScalaObjectMapper

import javax.inject.Inject

import javax.inject.Singleton

import scala.reflect.runtime.universe.TypeTag

/\*\*

\* Look up and execute products in the [[ProductPipelineRegistry]]

\*/

@Singleton

class UrtService @Inject() (productPipelineRegistry: ProductPipelineRegistry) {

def getUrtResponse[RequestType <: Request](

request: RequestType,

params: Params

)(

implicit requestTypeTag: TypeTag[RequestType]

): Stitch[urt.TimelineResponse] =

productPipelineRegistry

.getProductPipeline[RequestType, urt.TimelineResponse](request.product)

.process(ProductPipelineRequest(request, params))

.handle {

// Detect ProductDisabled and convert it to TimelineUnavailable

case pipelineFailure: PipelineFailure if pipelineFailure.category == ProductDisabled =>

UrtTransportMarshaller.unavailable("")

}

/\*\*

\* Get detailed pipeline execution as a serialized JSON String

\*/

def getPipelineExecutionResult[RequestType <: Request](

request: RequestType,

params: Params

)(

implicit requestTypeTag: TypeTag[RequestType]

): Stitch[t.PipelineExecutionResult] =

productPipelineRegistry

.getProductPipeline[RequestType, urt.TimelineResponse](request.product)

.arrow(ProductPipelineRequest(request, params)).map { detailedResult =>

val mapper = ScalaObjectMapper()

// configure so that exception is not thrown whenever case class is not serializable

mapper.underlying.configure(SerializationFeature.FAIL\_ON\_EMPTY\_BEANS, false)

val serializedJSON = mapper.writePrettyString(detailedResult)

t.PipelineExecutionResult(serializedJSON)

}

}