package com.twitter.product\_mixer.core.feature.featuremap

import com.fasterxml.jackson.core.JsonGenerator

import com.fasterxml.jackson.databind.JsonSerializer

import com.fasterxml.jackson.databind.SerializerProvider

import com.twitter.product\_mixer.core.feature.featurestorev1.featurevalue.FeatureStoreV1Response

import com.twitter.product\_mixer.core.feature.featurestorev1.featurevalue.FeatureStoreV1ResponseFeature

import com.twitter.util.Return

/\*\*

\* Rendering feature maps is dangerous because we don't control all the data that's stored in them.

\* This can result failed requests, as we might try to render a recursive structure, very large

\* structure, etc. Create a simple map using toString, this mostly works and is better than failing

\* the request.

\*

\* @note changes to serialization logic can have serious performance implications given how hot the

\* serialization path is. Consider benchmarking changes with [[com.twitter.product\_mixer.core.benchmark.CandidatePipelineResultSerializationBenchmark]]

\*/

private[featuremap] class FeatureMapSerializer() extends JsonSerializer[FeatureMap] {

override def serialize(

featureMap: FeatureMap,

gen: JsonGenerator,

serializers: SerializerProvider

): Unit = {

gen.writeStartObject()

featureMap.underlyingMap.foreach {

case (FeatureStoreV1ResponseFeature, Return(value)) =>

// We know that value has to be [[FeatureStoreV1Response]] but its type has been erased,

// preventing us from pattern-matching.

val featureStoreResponse = value.asInstanceOf[FeatureStoreV1Response]

val featuresIterator = featureStoreResponse.richDataRecord.allFeaturesIterator()

while (featuresIterator.moveNext()) {

gen.writeStringField(

featuresIterator.getFeature.getFeatureName,

s"${featuresIterator.getFeatureType.name}(${truncateString(

featuresIterator.getFeatureValue.toString)})")

}

featureStoreResponse.failedFeatures.foreach {

case (failedFeature, failureReasons) =>

gen.writeStringField(

failedFeature.toString,

s"Failed(${truncateString(failureReasons.toString)})")

}

case (name, Return(value)) =>

gen.writeStringField(name.toString, truncateString(value.toString))

case (name, error) =>

// Note: we don't match on Throw(error) because we want to keep it for the toString

gen.writeStringField(name.toString, truncateString(error.toString))

}

gen.writeEndObject()

}

// Some features can be very large when stringified, for example when a dependant candidate

// pipeline is used, the entire previous candidate pipeline result is serialized into a feature.

// This causes significant performance issues when the result is later sent over the wire.

private def truncateString(input: String): String =

if (input.length > 1000) input.take(1000) + "..." else input

}