package com.twitter.servo.cache

import com.twitter.util.{Throw, Return, Try}

import java.io.{DataOutputStream, ByteArrayOutputStream}

import java.nio.ByteBuffer

import scala.collection.mutable

import scala.util.control.NonFatal

object IterableSerializer {

// Serialized format for version 0:

// Header:

// 1 byte - Version

// 4 byte - number of items

// Data, 1 per item:

// 4 bytes - item length in bytes (n)

// n bytes - item data

val FormatVersion = 0

}

/\*\*

\* A `Serializer` for `Iterable[T]`s.

\*

\* @param itemSerializer a Serializer for the individual elements.

\* @param itemSizeEstimate estimated size in bytes of individual elements

\*/

class IterableSerializer[T, C <: Iterable[T]](

newBuilder: () => mutable.Builder[T, C],

itemSerializer: Serializer[T],

itemSizeEstimate: Int = 8)

extends Serializer[C] {

import IterableSerializer.FormatVersion

if (itemSizeEstimate <= 0) {

throw new IllegalArgumentException(

"Item size estimate must be positive. Invalid estimate provided: " + itemSizeEstimate

)

}

override def to(iterable: C): Try[Array[Byte]] = Try {

assert(iterable.hasDefiniteSize, "Must have a definite size: %s".format(iterable))

val numItems = iterable.size

val baos = new ByteArrayOutputStream(1 + 4 + (numItems \* (4 + itemSizeEstimate)))

val output = new DataOutputStream(baos)

// Write serialization version format and set length.

output.writeByte(FormatVersion)

output.writeInt(numItems)

iterable.foreach { item =>

val itemBytes = itemSerializer.to(item).get()

output.writeInt(itemBytes.length)

output.write(itemBytes)

}

output.flush()

baos.toByteArray()

}

override def from(bytes: Array[Byte]): Try[C] = {

try {

val buf = ByteBuffer.wrap(bytes)

val formatVersion = buf.get()

if (formatVersion < 0 || formatVersion > FormatVersion) {

Throw(new IllegalArgumentException("Invalid serialization format: " + formatVersion))

} else {

val numItems = buf.getInt()

val builder = newBuilder()

builder.sizeHint(numItems)

var i = 0

while (i < numItems) {

val itemBytes = new Array[Byte](buf.getInt())

buf.get(itemBytes)

val item = itemSerializer.from(itemBytes).get()

builder += item

i += 1

}

Return(builder.result())

}

} catch {

case NonFatal(e) => Throw(e)

}

}

}