package com.twitter.servo.util

import com.twitter.finagle.stats.{NullStatsReceiver, StatsReceiver}

import com.twitter.logging.\_

import com.twitter.scrooge.{BinaryThriftStructSerializer, ThriftStruct, ThriftStructCodec}

import com.twitter.util.Future

object Scribe {

/\*\*

\* Returns a new FutureEffect for scribing text to the specified category.

\*/

def apply(

category: String,

statsReceiver: StatsReceiver = NullStatsReceiver

): FutureEffect[String] =

Scribe(loggingHandler(category = category, statsReceiver = statsReceiver))

/\*\*

\* Returns a new FutureEffect for scribing text to the specified logging handler.

\*/

def apply(handler: Handler): FutureEffect[String] =

FutureEffect[String] { msg =>

handler.publish(new LogRecord(handler.getLevel, msg))

Future.Unit

}

/\*\*

\* Returns a new FutureEffect for scribing thrift objects to the specified category.

\* The thrift object will be serialized to binary then converted to Base64.

\*/

def apply[T <: ThriftStruct](

codec: ThriftStructCodec[T],

category: String

): FutureEffect[T] =

Scribe(codec, Scribe(category = category))

/\*\*

\* Returns a new FutureEffect for scribing thrift objects to the specified category.

\* The thrift object will be serialized to binary then converted to Base64.

\*/

def apply[T <: ThriftStruct](

codec: ThriftStructCodec[T],

category: String,

statsReceiver: StatsReceiver

): FutureEffect[T] =

Scribe(codec, Scribe(category = category, statsReceiver = statsReceiver))

/\*\*

\* Returns a new FutureEffect for scribing thrift objects to the underlying scribe effect.

\* The thrift object will be serialized to binary then converted to Base64.

\*/

def apply[T <: ThriftStruct](

codec: ThriftStructCodec[T],

underlying: FutureEffect[String]

): FutureEffect[T] =

underlying contramap serialize(codec)

/\*\*

\* Builds a logging Handler that scribes log messages, wrapped with a QueueingHandler.

\*/

def loggingHandler(

category: String,

formatter: Formatter = BareFormatter,

maxQueueSize: Int = 5000,

statsReceiver: StatsReceiver = NullStatsReceiver

): Handler =

new QueueingHandler(

ScribeHandler(category = category, formatter = formatter, statsReceiver = statsReceiver)(),

maxQueueSize = maxQueueSize

)

/\*\*

\* Returns a function that serializes thrift structs to Base64.

\*/

def serialize[T <: ThriftStruct](c: ThriftStructCodec[T]): T => String = {

val serializer = BinaryThriftStructSerializer(c)

t => serializer.toString(t)

}

}