package com.twitter.home\_mixer.util

import com.twitter.bijection.Injection

import com.twitter.io.Buf

import com.twitter.servo.util.Transformer

import com.twitter.storage.client.manhattan.bijections.Bijections

import com.twitter.util.Return

import com.twitter.util.Try

import java.nio.ByteBuffer

object InjectionTransformerImplicits {

implicit class ByteArrayInjectionToByteBufferTransformer[A](baInj: Injection[A, Array[Byte]]) {

private val bbInj: Injection[A, ByteBuffer] = baInj

.andThen(Bijections.byteArray2Buf)

.andThen(Bijections.byteBuffer2Buf.inverse)

def toByteBufferTransformer(): Transformer[A, ByteBuffer] = new InjectionTransformer(bbInj)

def toByteArrayTransformer(): Transformer[A, Array[Byte]] = new InjectionTransformer(baInj)

}

implicit class BufInjectionToByteBufferTransformer[A](bufInj: Injection[A, Buf]) {

private val bbInj: Injection[A, ByteBuffer] = bufInj.andThen(Bijections.byteBuffer2Buf.inverse)

private val baInj: Injection[A, Array[Byte]] = bufInj.andThen(Bijections.byteArray2Buf.inverse)

def toByteBufferTransformer(): Transformer[A, ByteBuffer] = new InjectionTransformer(bbInj)

def toByteArrayTransformer(): Transformer[A, Array[Byte]] = new InjectionTransformer(baInj)

}

implicit class ByteBufferInjectionToByteBufferTransformer[A](bbInj: Injection[A, ByteBuffer]) {

private val baInj: Injection[A, Array[Byte]] = bbInj.andThen(Bijections.bb2ba)

def toByteBufferTransformer(): Transformer[A, ByteBuffer] = new InjectionTransformer(bbInj)

def toByteArrayTransformer(): Transformer[A, Array[Byte]] = new InjectionTransformer(baInj)

}

}

class InjectionTransformer[A, B](inj: Injection[A, B]) extends Transformer[A, B] {

override def to(a: A): Try[B] = Return(inj(a))

override def from(b: B): Try[A] = Try.fromScala(inj.invert(b))

}