package com.twitter.ann.file\_store

import com.twitter.ann.common.IndexOutputFile

import com.twitter.ann.common.thriftscala.FileBasedIndexIdStore

import com.twitter.bijection.Injection

import com.twitter.mediaservices.commons.codec.ArrayByteBufferCodec

import com.twitter.mediaservices.commons.codec.ThriftByteBufferCodec

import com.twitter.storehaus.Store

import com.twitter.util.Future

import java.util.concurrent.{ConcurrentHashMap => JConcurrentHashMap}

import scala.collection.JavaConverters.\_

object WritableIndexIdFileStore {

/\*\*

\* @param injection: Injection to convert typed Id to bytes.

\* @tparam V: Type of Id

\* @return File based Writable Store

\*/

def apply[V](

injection: Injection[V, Array[Byte]]

): WritableIndexIdFileStore[V] = {

new WritableIndexIdFileStore[V](

new JConcurrentHashMap[Long, Option[V]],

injection

)

}

}

class WritableIndexIdFileStore[V] private (

map: JConcurrentHashMap[Long, Option[V]],

injection: Injection[V, Array[Byte]])

extends Store[Long, V] {

private[this] val store = Store.fromJMap(map)

override def get(k: Long): Future[Option[V]] = {

store.get(k)

}

override def put(kv: (Long, Option[V])): Future[Unit] = {

store.put(kv)

}

/\*\*

\* Serialize and store the mapping in thrift format

\* @param file : File path to store serialized long indexId <-> Id mapping

\*/

def save(file: IndexOutputFile): Unit = {

saveThrift(toThrift(), file)

}

def getInjection: Injection[V, Array[Byte]] = injection

private[this] def toThrift(): FileBasedIndexIdStore = {

val indexIdMap = map.asScala

.collect {

case (key, Some(value)) => (key, ArrayByteBufferCodec.encode(injection.apply(value)))

}

FileBasedIndexIdStore(Some(indexIdMap))

}

private[this] def saveThrift(thriftObj: FileBasedIndexIdStore, file: IndexOutputFile): Unit = {

val codec = new ThriftByteBufferCodec(FileBasedIndexIdStore)

val bytes = ArrayByteBufferCodec.decode(codec.encode(thriftObj))

val outputStream = file.getOutputStream()

outputStream.write(bytes)

outputStream.close()

}

}