<!-- csv文件解析依赖 -->

<dependency>

<groupId>com.opencsv</groupId>

<artifactId>opencsv</artifactId>

<version>4.3.2</version>

</dependency>

package com.lxk.utils;

import com.opencsv.bean.CsvToBean;

import com.opencsv.bean.CsvToBeanBuilder;

import com.opencsv.bean.HeaderColumnNameMappingStrategy;

import lombok.extern.slf4j.Slf4j;

import org.springframework.web.multipart.MultipartFile;

import java.io.InputStreamReader;

import java.util.List;

@Slf4j

public class CsvUtil {

/\*\*

\* 解析csv文件并转成bean

\* @param file csv文件

\* @param clazz 类

\* @param <T> 泛型

\* @return 泛型bean集合

\*/

public static <T> List<T> getCsvData(MultipartFile file, Class<T> clazz) {

InputStreamReader in = null;

try {

in = new InputStreamReader(file.getInputStream(), "gbk");

} catch (Exception e) {

log.error("读取csv文件失败！");

}

HeaderColumnNameMappingStrategy<T> strategy = new HeaderColumnNameMappingStrategy<>();

strategy.setType(clazz);

CsvToBean<T> csvToBean = new CsvToBeanBuilder<T>(in)

.withSeparator(',')

.withQuoteChar('\'')

.withMappingStrategy(strategy).build();

return csvToBean.parse();

}

}

public class DataAndTypeCsv {

/\*\*

\* 字典代码

\*/

@CsvBindByName(column = "code")

private String code;

/\*\*

\* 简写

\*/

@CsvBindByName(column = "short\_name")

private String shortName;

/\*\*

\* 名称

\*/

@CsvBindByName(column = "name", required = true) //是否可以为null 否

private String name;

/\*\*

\* 拼音或英文描述

\*/

@CsvBindByName(column = "remark")

private String remark;

/\*\*

\* 父类型id

\*/

@CsvBindByName(column = "parent\_id")

private Integer parentId;

/\*\*

\* 类型名称

\*/

@CsvBindByName(column = "type\_name", required = true) //是否可以为null 否

private String typeName;

/\*\*

\* 类型id

\*/

@CsvBindByName(column = "type\_id", required = true) //是否可以为null 否

private Integer typeId;

……

get、set方法

……

位置！！！！

ColumnPositionMappingStrategy<Object> strategy = new ColumnPositionMappingStrategy<>();

@CsvBindByPosition(position=1)

private String address;

@CsvBindByPosition(position=2)

private String manageIpaddress;

@CsvBindByPosition(position=3)

private String eqpTypeId;

@CsvBindByPosition(position=4)

private String regionId;

<dependencies>

<dependency>

<groupId>org.apache.avro</groupId>

<artifactId>avro</artifactId>

<version>1.8.1</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.avro</groupId>

<artifactId>avro-maven-plugin</artifactId>

<version>1.8.1</version>

<executions>

<execution>

<phase>generate-sources</phase>

<goals>

<goal>schema</goal>

</goals>

<configuration>

<sourceDirectory>${project.basedir}/src/main/avro/</sourceDirectory>

<outputDirectory>${project.basedir}/src/main/java/</outputDirectory>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

</plugins>

</build>

@Test

public void test() throws IOException {

User user1 = new User();

user1.setAge(11);

user1.setName("小明");

User user2 = new User("小红", 12);

// Serialize user1, user2 and user3 to disk

DatumWriter<User> userDatumWriter = new SpecificDatumWriter<User>(User.class);

DataFileWriter<User> dataFileWriter = new DataFileWriter<User>(userDatumWriter);

dataFileWriter.create(user1.getSchema(), new File("users.avro"));

dataFileWriter.append(user1);

dataFileWriter.append(user2);

dataFileWriter.close();

}

@Test

public void testDeserialize() throws IOException {

DatumReader<User> userDatumReader = new SpecificDatumReader<User>(User.class);

DataFileReader<User> dataFileReader = new DataFileReader<User>(new File("users.avro"), userDatumReader);

User user = null;

while (dataFileReader.hasNext()) {

// Reuse user object by passing it to next(). This saves us from

// allocating and garbage collecting many objects for files with

// many items.

user = dataFileReader.next(user);

System.out.println(user);

}

}