碳烤黄蜂

功能开发指南

2023 年 8 月 18 日

Potato

0.1 编写解析器和管理器

解析器用于解析单词本文件，它将单词本文件的单词、历史记录和单词本信息分别解析为 *List<Word>*、*List<History>*和 *Info*。管理器则是用于单词本的增、删、查、改。

0.1.1 编写解析器

编写解析器，首先需要继承 *Parser* 类并实现构造器和 *parser()* 方法。

Listing 1 初始布局

```
1  import com.potato.Parser.Parser;
2  import java.io.File;
3
4  public class MyParser extends Parser
5  {
6      public MyParser(File file, String extension)
7      {
8          super(file, extension);
9      }
10
11     @Override
12     protected void parser()
13     {
14
15     }
16 }
```

由于我们自定义的解析器对应的文件类型一定是确定的，因此我们应该将构造器中的 *String extension* 换成我们解析器对应的文件类型。所有单词本文件的文件类型都定义在了 *com.potato.ToolKit.WordFileType* 中，我们以数据库为例，构造器更改如下所示：

Listing 2 构造器

```
1  public MyParser(File file)
2  {
3      super(file, WordFileType.DATABASE.type());
4  }
```

接下来编写 *parser()* 方法。这个方法是解析单词本文件的具体实现，它是一个抽象方法（因此必须实现），将直接被 *Parser* 的构造器调用。因此如果有需要在解析前初始化的对象，需要写在构造器中。

Listing 3 Parser 的构造器源码

```
/**
 * Parser用于解析储存单词本的文件
 *
 * @param file       需要解析的文件
 * @param extension 文件应有的扩展名，对于文件xxx.db，其扩展名是db
 *                  扩展名建议使用WordFileType中已定义的枚举
 */
public Parser(File file, String extension)
{
    // 使用卫语句捕捉文件类型错误
    if (!extension.equals(getExtensionName(file)))
    {
        Log.e(getClass().toString(), String.format("解析文件%s类型错误", file.getName()));
    }
    this.file = file;

    parser();
    Log.i(getClass().toString(), String.format("解析文件%s成功", file.getName()));
}
```

Parser 中定义了三个静态变量：*wordList*、*info* 和 *historyList*，它们分别储存单词列表、单词本信息和历史记录（单词本文件的详细说明和格式规范、单词的构造方法、历史记录和数据库的构造器请见 API 指南）。这三个静态变量都给出了对应的 Setter，*parser()* 方法的作用就是给这三个变量赋值。

Listing 4 parser 方法

```
@Override
protected void parser()
{
    List<Word> wordList = new ArrayList<>();
    List<History> historyList = new ArrayList<>();
    Info info = null;

    // 对以上三者赋值
    // ...
```

```
10
11      setWordList(wordList);
12      setHistoryList(historyList);
13      setInfo(info);
14  }
```

至此就完成了解析器的编写。我们在调用 *AutoParser* 时，它会从 *Config.parserMap* 中选取对应文件类型的解析器，因此我们要在这里更改解析器为我们的解析器。

Config 提供了 *setParser(WordFileType, Constructor<? extends Parser>)* 方法用于修改某文件类型对应的解析器。修改解析器建议在 *Config* 初始化时就进行，以免因为程序执行顺序错误而选用了错误了解析器。

Listing 5 修改解析器

```
1  // getConstructor()的参数是MyParser构造器中每个参数对应类型的class
2  Config.setParser(WordFileType.DATABASE, MyParser.class.getConstructor(File.class));
```