Lab5背单词应用
(WordMaster)
修改报告

小组成员：11302010054 王可嘉

11302010056 赵砚君

11302010064 刘 芳

11307120175 陈露薇

10300700021 韩澍野

目 录

1 引言

由于背单词应用（WordMaster）的客户需求发生变化，开发人员对该软件进行了相应的修改。本文档即针对变更的模块、代码进行了详细记录，并对照原有设计结构进行了分析，总结经验。

1.1 修改部分现有功能

- 字典文件格式为txt文档dictionary.txt。背单词应用通过读取该txt文档内容导入字典数据。
- 单词词库为ABC词库。导入字典数据中的所有单词根据首字母进行词库区分，如首字母为a的单词在A词库中。用户在使用背单词应用时需选择希望背诵或查看的ABC词库。每个ABC词库分别存有相应背诵情况记录。

1.2 修改后功能

- 字典文件格式修改为xml文档dictionary.xml。背单词应用通过读取该xml文档内容导入字典数据。
- 单词词库修改为单词属性词库。导入字典数据中的所有单词根据单词词性进行词库区分，如名词(n.)词库、动词（v.）词库、形容词（adj.）词库等。用户在使用背单词应用时需选择希望背诵或查看的单词词性词库。每个单词词性词库分别存有相应背诵情况记录。

1.3 读者对象

相关项目人员。

1.4 修改术语解释

- 词库：通过单词属性划分的不同单词范围。例如，以属性为名词(n.)的单词为名词词库，单词abandonment、abbreviation都在名词词库中。其中有一些单词同时拥有多个属性，则其同时存在在多个词库中，如单词advance同时在名词词库及动词词库中。

2 需求变更影响

本节详细描述了由于用户需求变更而导致的软件模块设计上需要修改的部分。

2.1 需求变更所影响的模块

WordMaster采用MVC框架模式，总体架构将程序分成三个核心部件：Model模型、View视图和Controller控制器。本次用户需求的改动对系统整体架构不构成影响，因此MVC框架不变。下面将分别针对 Model模型、View视图和Controller控制器三个模块进行分析，说明是否需要修改以及修改部分描述。

2.1.1 View 模块

View模块负责为用户提供图形化界面。该模块主要提供两类方法：构造背单词过程中需要显示的各个界面的方法，以及获得一部分组件的方法。在本次的用户需求变更中，字典格式修改并不对用户界面造成任何影响，因此无需修改View模块。而单词词库的修改导致原词库选择界面的选项——ABC词库需修改为单词属性词库。由于 View模块不直接与Model模块进行交互， 不参与任何逻辑操作，因此仅需进行细小变更。

2.1.2 Controller 模块

Controller模块负责在View模块与Model模块之间传递数据。Controller模块从View模块中获得组件，添加监听器，在事件触发时调用Model模块或View模块的一些方法，使得图形化界面根据一些逻辑操作进行相应的切换。在本次的用户需求变更中，字典格式修改并不对导入词库后的操作造成任何影响，因此无需修改Controller模块。而单词词库的修改虽然导致原词库选择界面获取的的组件——ABC词库选择按钮变为单词属性词库选择按钮，但由于 Controller模块仅负责在View模块与Model模块之间传递数据，并不受所传数据变更的影响，因此Controller模块无需进行变更。

2.1.3 Model模块

Model模块负责背单词过程中的逻辑操作，包括读取词库存档文件、选择词库、选择起始单词、选择背诵数目、输出词库统计信息、输出本次背诵统计信息、更新存档文件等操作。 在本次的用户需求变更中，字典格式修改将导致词库读取方式修改。而单词词库的修

改导致选择词库以及相应的输出词库统计信息的逻辑需修改。 因此Model模块需要进行一定变更。

2.2 需求变更所影响的代码单元

WordMaster采用面向对象设计，因此一个类即一个代码单元。下面将针对需要修改的类进行分析，说明修改原因及修改部分描述。

2.2.1 View模块类

- WordStorePanel

该类是选择词库以及功能的视图类，其中提供的词库选择栏需由原来的26个首字母的词库类型，修改为不同单词属性的词库类型。

2.2.2 Model模块类

- ProcessModel

该类控制frame中panel及button的变换。由于单词词库修改，新增initWordBase和addToType方法。当用户选择词库后frame需显示相应Panel前，调用initWordBase遍历xml字典数据，取出每个单词释义，调用addToType进行字符串解析，将字典数据中所有出现的单词属性加入wordType链表。

- WordBaseModel

该类在数据库中存取词库信息、用户背单词记录并处理数据逻辑。由于字典文件格式修改，该类读取字典数据的文件来源需修改。由于单词词库修改，新增isTheType函数，并新增对读取到的字典中所有单词的分类功能。即当给定单词属性后，调用新增的isTheType函数，从字典数据中读取单词并判断是否含有该单词属性，若有，则存入相应词库中。

3 代码修改情况

本节详细描述了在新的需求变更（v2）之后，代码具体的修改情况。包括变更代码所在的包、模块、类、函数等具体位置，以及改动的原因，结果等。

3.1 修改模块

本次需求变更后影响的代码设计模块包含Model，View，Controller。其中Model模块为主要变更模块，变更的功能为导入的词典格式由txt到xml的转变，以及词库类型由首字母到词性的转变。View模块中相应的变动为，将原先词库显示的首字母改为词性。Controller模块中的变动为将原先的首字母词库监听器替换为词性词库的监听器。

3.2 修改代码类

在本次项目中，需求变更影响到的类有三个：

- ProcessModel.java
- WordBaseModel.java
- WordStorePanel.java

具体的代码修改情况如下表。包含方法签名，所属类，所属模块，原代码，变更代码，新增代码，以及备注。

表1 代码修改情况（共6项）

方法签名：changeModel(int a)	总编号：1-01
所属类：ProcessModel.java	编号：01
所属模块：Model	
原代码：无	
变更代码：无	

新增代码：
`if (a==1)`
`{`
`  initWordBase();`
`}`	

备注：在进入第二个窗口的之前这一状态，调用initWordBase开始读XML词典数据库	
方法签名：initWordBase()	总编号：1-02
所属类：ProcessModel.java	编号：02

表1 代码修改情况（共6项）

所属模块：Model

原代码：无

变更代码：无

新增代码：

```
public void initWordBase() throws DocumentException
{
	String fileName;
	fileName =ReadDat.filePath + ReadDat.fileName + ".xml";

	File inputXml=new File(fileName);
	SAXReader saxReader = new SAXReader();
	Document document = saxReader.read(inputXml);
	Element words=document.getRootElement();
	for(Iterator i = words.elementIterator(); i.hasNext();){
		...
	}
}
```

备注：将整个XML词典数据库遍历一遍，取出每一个单词的中文释义，

调用addToType函数对中文释义进行字符串解析，找出词典中存在的

所有的词性，将词性加入wordtype链表当中。

方法签名：addToType(String chinMeaning,String english) 总编号：1-03

所属类：ProcessModel.java 编号：03

所属模块：Model

原代码：无

变更代码：无

表1 代码修改情况（共6项）

新增代码：

```
public void addToType(String chinMeaning,String english)
{
     if (chinMeaning.lastIndexOf('.')<0)
     {
       return;
     }
             ...
              ...
     if (j==ReadDat.wordtype.size())
       {
              ReadDat.wordtype.add(singleMean);
       }
}
```

备注：把XML每一个中文释义的字符串取出，解析每一个单词的中文释义中的词性，若链表wordtype中未包含该词性，则将该词性存入；若已包含，则忽略这条信息。

方法签名：inputFromDictionary(String letter) 总编号：1-04

所属类：WordBaseModel.java 编号：04

所属模块：Model

原代码：

```
fileName =ReadDat.filePath + ReadDat.fileName + ".txt";
```

变更代码：

```
fileName =ReadDat.filePath + ReadDat.fileName + ".xml";
```

表1 代码修改情况（共6项）

新增代码：

```
File inputXml=new File(fileName);
SAXReader saxReader = new SAXReader();
Document document = saxReader.read(inputXml);
Element words=document.getRootElement();
for(Iterator i = words.elementIterator(); i.hasNext();)
{
                    ...
}
unrecitednum=sumnum;
```

备注：方法参数为某一给定的词性，此方法的功能为从文档里读含有该词性的中英文解释，并存入到该词性所对应的词库数据库中(wordlist全局变量)。

方法签名： isTheType(String chinMeaning) 总编号：1-05

所属类：WordBaseModel.java 编号：05

所属模块：Model

原代码：无

变更代码：无

表1 代码修改情况（共6项）

新增代码：

```
public boolean isTheType(String chinMeaning)
{
       if (chinMeaning.lastIndexOf('.')<0)
       {
        return false;
       }
       ...
       int i;
       for (i=0;i<types.length();i++)
       {
            ...
       }
       ...
       return false;
}
```

备注：方法参数为某一单词的中文释义，此方法的功能为解析此中文释义，判断是否包含给定的词性（ReadDat类中的全局变量），返回值为布尔值。

方法签名： setCountModel() 总编号：1-06

所属类：WordStorePanel.java 编号：06

所属模块：View

原代码：

```
for(int i = 0; i < 26;i++)
{
       name[0] = (char)('A'+i )+ ".词库";
       model2.insertRow(i,name);
}
```

表1 代码修改情况（共6项）

变更代码：

```
for(int i = 0; i < ReadDat.wordtype.size();i++)
{
        name[0] = ReadDat.wordtype.get(i)+".词库";
        model2.insertRow(i,name);
}
```

新增代码：无

备注：将第二个窗口显示出的词库类型，修改之前为26个首字母的词库类型，更改后变为不同词性的词库类型。

4 经验与教训

在本次需求变更的实现过程中，我们体会到了MVC建模的诸多优点，有效地提高了我们代码重构的效率，实现起来也十分轻松。这次Lab我们组选取的是第二种需求变更要求，即，将原先的txt格式的词典换为XML词典，选择词库时将原先的26个首字母词库替换为单词词性词库。由于在Lab4中我们对于本项目的体系结构设计采用的是MVC开发模式，对于软件的各个部件按照MVC的要求进行了精确的划分，在后期实现上也是完全按照设计的方案来实现，所以在拿到需求变更后，我们很快的找到了需求变更所对应的代码结构，清晰的列出了需要变动的模块、类、方法等，为后期的实现打好了非常扎实的基础。回顾一下我们Lab4中每个模块的功能设计：

1. View模块负责为用户提供图形化界面。该模块主要提供两类方法：构造背单词过程中需要显示的各个界面的方法，以及获得一部分组件的方法。用途是供Controller模块调用。所有需要输入数据动态生成的界面都依赖于Controller模块调用时传入的数据。View模块不直接与Model模块进行交互，不参与任何与背单词相关的逻辑操作。

2. Controller模块负责在View模块与Model模块之间传递数据。Controller模块从View模块中获得组件，添加监听器，在事件触发时调用用Model模块或View模块的一些方法，使得图形化界面根据一些逻辑操作进行相应的切换。

3. Model模块负责背单词过程中的逻辑操作，包括读取词库存档文件、选择词库、选择起始单词、选择背诵数目、输出词库统计信息、输出本次背诵统计信息、更新存档文件等操作。所有读入的输入都由Controller模块提供, Model模块不直接与用户进行交互。

依照这样的设计模式和功能划分，我们将此次的需求变更需要变动的部分聚焦在了Model模块和View模块，其中Model模块是重点。由于在之前的设计中，Controller模块负责在View模块与Model模块之间传递数据。即Controller模块在整个系统中担当接口的角色，在本次需求变更中并不需要发生改动。

首先从XML格式的词典入手，我们需要一个函数来解析XML格式的数据，于是引入dom4j.jar包。对于词库的类别数目由于在需求中并未明确给出，所以我们需要遍历一遍词典，解析每个单词的中文释义以获取到所有的词性，从而将不同的词性存入新加入的全局变量wordtype中。这些都是顺理成章能想到的变更方法，得益于之前松耦合的设计，部分代码的改动并不会影响其他功能的实现。除此之外的一些界面显示等细节变动都顺应着主要功能的改变，比如View模块中显示选择词库的界面要相应的有之前的首字母词库变为词性词库。

然而在实现的过程中，我们也遇到了意想不到的问题。在解析词典数据库中的每个单词的中文释义时，发现有的词性标注的并不规范。例如：‹chinese›]n.星期四 ‹/chinese›，‹chinese›v,,n.呻吟‹/chinese›等，其中红字标注部分为不规范的词性标注形式，这对我们标准化的解析字符串带来了很多繁琐的工作量，处理过程也因此增加了诸多困难。经过小组内的商讨，我们决定与提供词典数据库的客户沟通(王海TA)。因为毕竟数据库是由客户提供，在不经过客户允许的情况下我们无权对其进行修改，也不应随意猜测客户的本意。想的极端一点儿，组内有同学提出，说不定客户就是认为“]n”算是一种词性也说不准，我们作为软件开发者，应该严格依照客户的意愿来实现代码功能，不能只按照自己的意愿去妄加揣测。所幸，在与客户交流时，我们提出了我们的难处，同时也提出了我们愿意接受的解决方

案--即手动将XML词典数据库中的不规范表述更正，使得我们在解析时可以用同一的方法进行处理。客户同意了我们的方案，这个问题得到了圆满的解决。

除此之外，我们也总结了一些本次Lab中的教训。就局部模块划分来说，我们的设计仍有一定欠缺，尤其是Model模块中的四个类，功能不够明确。ProcessModel类的主要功能为控制frame中panel及button的变换，但除此之外也包含了几个其他的函数，处理panel变换前的准备工作。这些函数并不与ProcessModel的主要函数同时被复用，而是在特殊情况下才被调用，因此根据软件构件级设计中的共同复用原则，这些处理不同情况的函数应当分离，而不能打包为一类。虽然在本次需求变更的代码修改过程中，由于仅仅是增加了函数来记录字典数据中出现的单词属性，而非更改，且增加功能单一，因此这一问题并未造成很大的影响。但若出现客户需求变动需要对之前的方法进行修改，且更多更为分散的情况，那么这一不合理的函数打包会使得开发者在明确了ProcessModel类需要修改后却仍然无法快速地找到具体的代码修改部分。在日后的项目开发过程中若遇到类似问题，应当将这些函数单独根据处理的事件进行分类，以使后续的代码维护工作更有效率。