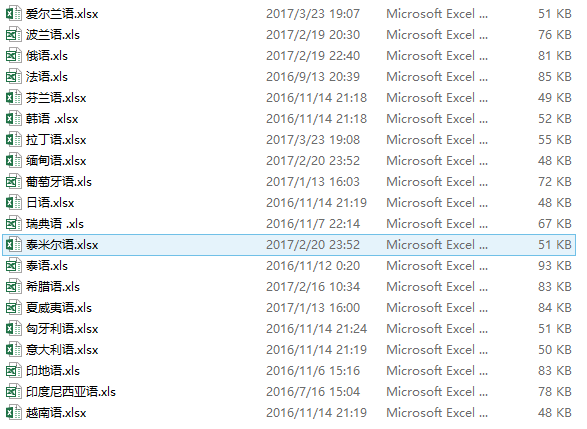
1.配置介绍：

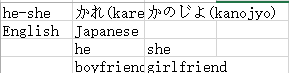
Windows8环境下Java（1.8.0\_121）+mysql+tomcat8.5.24

用的软件intellij idea2017和navicat premium

2.需求



为以上20种语言的60个英文单词（I，you，one等等）构造一个普通的网站，满足增删改查的基本功能以及对义项的一些平均、最大、最小数量求取的一些操作。

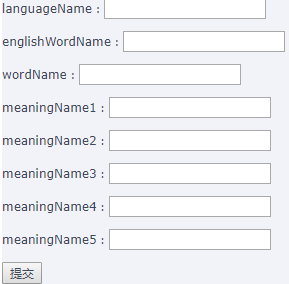


Excel表格大致为上图所示，包括的信息有英文单词、语言、该英文单词在该语言下的单词以及改语言中该单词有的所有义项（用英文表达）。

3.大致思路

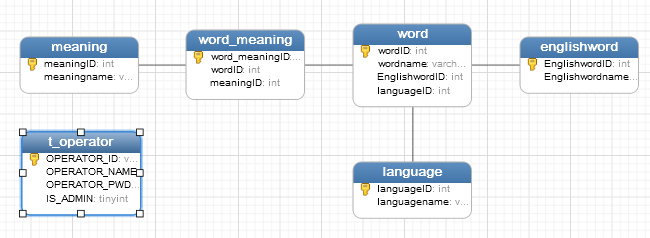
主体分为两大块，一块为数据库的建立，为了满足BCNF或者3NF的范式要求，需要对language，englishword，word，meaning这四者进行相应的数据库的使得我们的数据库不会有冗余的信息。构建完数据库的结构那么对于数据的录入（20种语言，每种语言对应60个英文单词的单词，每个单词又有好几个义项，构建的数据库一定是用各种ID值来作为主键和外键与其他值联系的，所以还算是一个很浩大的工程），大致有两个想法：

1. 把excel内容全部转成一个txt文档后（两个单词的信息之间用特殊字母（我用yao）隔开），我用python写了一套代码，实现从txt转为各种sql的insert语句的功能。有几点问题大致提一下，首先格式问题挺难办的，我得到的这一堆excel表格格式并不是完全严格，总会有一些奇怪的格式出现，导致大批量处理起来要de好的bug再对症下药；其次的问题是，因为我们是用自增的ID作为主键的，所以在插入多对多以及一对多表的时候，外键都是ID而不是具体的这些单词，然而通过python文件只能把txt处理成insert语句的value是单词的，所以这里还有一个难点在于要根据已经建立好的几张没有外键依赖的表建立出对应的ID和单词的映射关系，需要进行替换才能得到最终正确的sql语句。
2. 就是因为上述的问题，所以用方法1我只是对两种语言的所有单词用这一套处理了一遍，插入到数据库中（保证数据库有东西给java代码做测试）。那么剩下18种了？我给出的第二个想法就是，根据已有的两种语言可以把网站需要的增删改查所有功能都完成（主要必须是至少两种语言才好debug，毕竟从2到3甚至到20都是一个问题，然而从1到2则是完全不同的两个问题），然后就可以和录入进excel一样方便快捷的运行网站进行insert操作了（如下图），这样一套录入因为java代码帮你构建了复杂的逻辑去按顺序insert对应的表，所以只要把excel上的内容几乎照搬一遍即可，无需考虑逻辑问题。



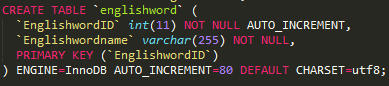
4.数据库搭建

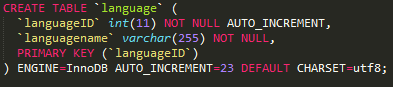
ER图：

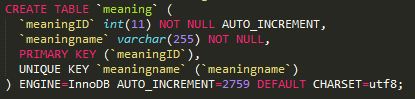


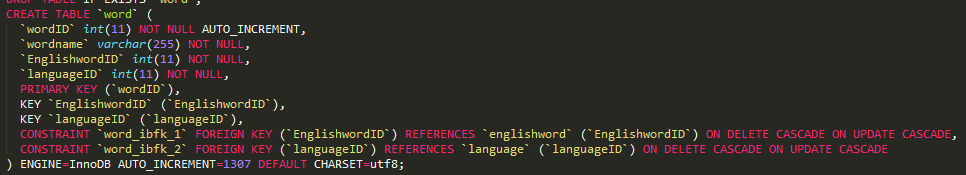
根据语言、英语单词、单词、义项之间的关系，我们可以画出数据库的ER图。其中，每个单词和它对应的语言和英语单词都是多对一的关系，所以我们将englishword和language的主键（englishwordID和languageID）作为外键添加到word表中；每个单词和各个义项是一个多对多的关系，所以我们需要建立一张中间表word\_meaning表，其中的两个外键wordID和meaningID来自word表和meaning表的主键。另外我们建了一张用户表来储存我们的用户信息。

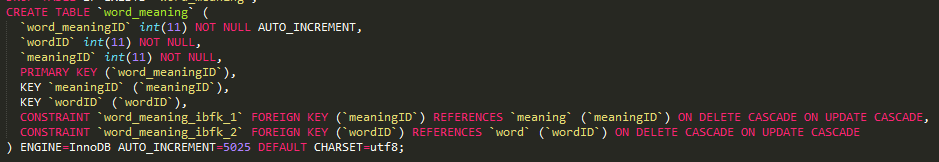
建表：

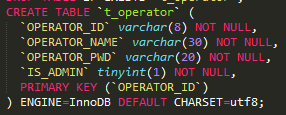












对应的五张表建立的sql语句如上图所示，需要说明的是所有的ID都被设置为自增的，也就是说作为网站的用户在增加一个新东西时（比如新的语言），只需要告诉我们的程序这种新语言的名字即可，数据库后台会自动为你分配下一个可用的ID作为主键。

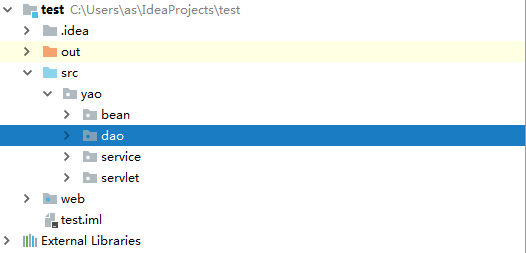
插入数据：

这部分工作就命途多舛了，曾经学习搭网站的时候这差不多是略过的一部分，随意加几条数据学到连接数据库的方法操作即可。然而在真是的数据库系统中，这其实是不比搭网站本身轻松多少的任务，特别是在你得到的是客户给你的不太规范的有着各种各样格式的零散数据的时候，真是一言难尽。所以这是python就要用起来，各种批量数据处理自己写起来，格式乱就好好先整理起来。至于相关的python代码，完全没有啥套用的可能，就自己看着手头的数据格式wiki baidu google debug去吧。好运！

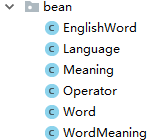
所以后台数据库就这么轻松愉快的搭完了，接下来我们转战intellij idea，看看我们的jjs框架下的java代码长什么样。

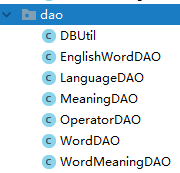
5.MVC框架搭建

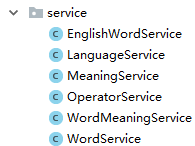
Model view controller，大名鼎鼎如雷贯耳，从此将网站的搭建规范了出来。看一下我们代码的大致组成：



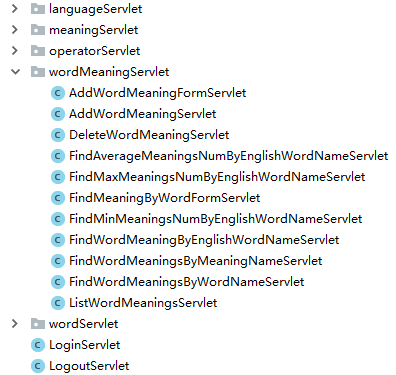
无视.idea无视out，src里面包括了许多的java文件是我们的model和controller，web里面有无数的jsp和一个很重要的xml映射文件，这些是我们的view。我们展开src里的文件夹浏览一下结构。



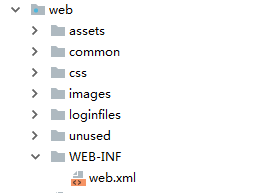


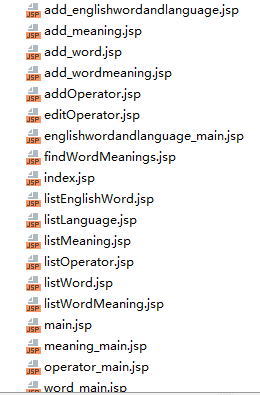






可以看出整个代码的套路简介明了，一层一层一一对应，我这一层只要负责一切前后的处理，具体的操作我丢给下一层去做，我只要给出命令和看到结果。正式这种有点像人类社会的行为让代码的思维看起来这么令人舒畅（这是所有这些框架的共同点）。接下来我们看一下view的东西：





这就是构成我们最后将会看到的页面的前端代码部分。接下来我们自顶向下（从页面到最底层model）进入我们的代码们，看看他们是怎么完成相互之间的分工合作以及沟通的。

为了让我们的说明更加简单易懂，我们以其中最为简单又不失代表性的例子——插入一种新的语言来接受各层之间的联系的，language的insert当然是最为简单的操作之一，但是那些复杂的操作（比如插入一种新的单词-义项关系或者删除某种语言），都是从这最基础的操作衍生出来的，只不过你需要为了数据库的安全性和不产生各种错误冲突进行一大堆检查或先导操作来确保用户这些复杂操作真正反映到我们的数据库中去了。

（1）Jsp



Jsp页面不做过多的介绍，基本用html语言，其中<form action="addLanguage" method="POST">这句话告诉我们，要addlanguage，拿着下面给你的信息，去web.xml查查你该找那个servelet作为controller。

（2）web.xml



查表action为addlanguage找哪啊，ok，去yao.servlet.languageServlet.AddLanguageServlet

（3）servelet

稍微介绍一下核心代码：

一开始确认一下编码方式，保证从页面的form读入的各国语言不会到这就乱码了。然后我们创建一个language并调用函数getParameters，作用就是把页面读入的语言名赋给language中的name属性，之后是validation函数用看判断输入又没什么非法错误（比如啥都没输入就点了提交就是非法的），有错误作为信息返回给页面，没有那么完事大吉，我们将创建servelet下层的service，调用service要为我servelet实现对language的add操作，我只要知道你完成没，具体怎么完成你service看着办。

request.setCharacterEncoding(**"utf-8"**);  
**language** = **new** Language();  
getParameters(request);  
HttpSession session = request.getSession();  
**if**(!validation()){  
 session.setAttribute(**"addLanguageMsg"**, **addLanguageMsg**);  
 request.getRequestDispatcher(***ADD\_LANGUAGE\_FORM***).forward(request,response);  
}  
**else**{  
 LanguageService languageService = **new** LanguageService();  
 **if**(languageService.addLanguage(**language**)){  
 request.getRequestDispatcher(***LIST\_LANGUAGE***).forward(request,response);  
 }  
 **else** {  
 session.setAttribute(**"addLanguageMsg"**, languageService.getMsg());  
 request.getRequestDispatcher(***ADD\_LANGUAGE\_FORM***).forward(request, response);  
 }  
}

**private void** getParameters(HttpServletRequest request){  
 **language**.setLanguageName(request.getParameter(**"languageName"**));  
}  
  
**private boolean** validation(){  
 **if**(**language**.getLanguageName() == **null** || **language**.getLanguageName().trim().equals(**""**)){  
 **addLanguageMsg** = **"语言名称不能为空"**;  
 **return false**;  
 }  
 **return true**;  
}

（4）service

**public boolean** addLanguage(Language language){  
 **if**(**new** LanguageDAO().findLanguageByLanguageID(language.getLanguageID()) != **null**){  
 **msg** = **"语言已存在"**;  
 **return false**;  
 } **else** {  
 **return new** LanguageDAO().addLanguage(language);  
 }  
}

service层的人一看上面来要求了，丢来了一个language（里面有language的name信息），要进行add操作，那行啊，我找我的手下languageDAO来办，DAO弟你先看看这语言库里有了吗，有的话我就给上面说不用加了已经有了，没有还劳烦DAO弟加一下这个language

（5）DAO

到了DAO层，我们总该干点实事了。还是就以add这个操作说明。

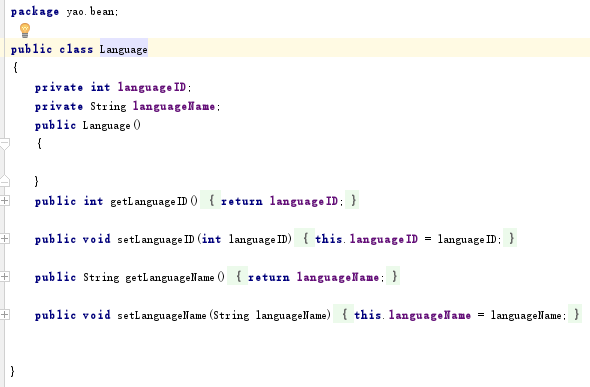
**private static final** String ***ADD\_LANGUAGE*** = **"insert into LANGUAGE values (NULL,?)"**;

**private static final** String ***FIND\_LANGUAGE\_BY\_LANGUAGENAME*** = **"select** *\** **from LANGUAGE where LANGUAGENAME = ?"**;

**public boolean** addLanguage(Language language){  
 **boolean** result = **false**;  
 **try** {  
 Connection connection = DBUtil.*getConnection*();  
 PreparedStatement pStatement = connection.prepareStatement(***ADD\_LANGUAGE***);  
 PreparedStatement pStatement1 = connection.prepareStatement(***FIND\_LANGUAGE\_BY\_LANGUAGENAME***);  
 pStatement.setString(1, language.getLanguageName());  
 pStatement1.setString(1, language.getLanguageName());  
 ResultSet resultSet = pStatement1.executeQuery();  
 **if**(resultSet.next()){  
 System.***out***.println(**"LANGUAGE已经存在了"**);  
 DBUtil.*closePreparedStatement*(pStatement);  
 DBUtil.*closeConnection*(connection);  
 **return false**;  
 }  
 **if**(pStatement.executeUpdate() == 1){  
 result = **true**;  
 }  
 DBUtil.*closePreparedStatement*(pStatement);  
 DBUtil.*closeConnection*(connection);  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 **return** result;  
}

首先我们要定义sql语句，之后连接好数据库（连接数据库的一些代码在DBUtil.java里面。接着把对应sql语句“？”的地方设置好对应参数，再用封装好的executequery执行判断是否已经存在的sql语句（select），有的话不用再插入并且传达一下“已经有了”。没有的话再执行insert的sql语句。

（6）bean



Bean就是model，也就是和数据库里ID，name匹配的在代码里的变量。再加上set get函数就可以啦。

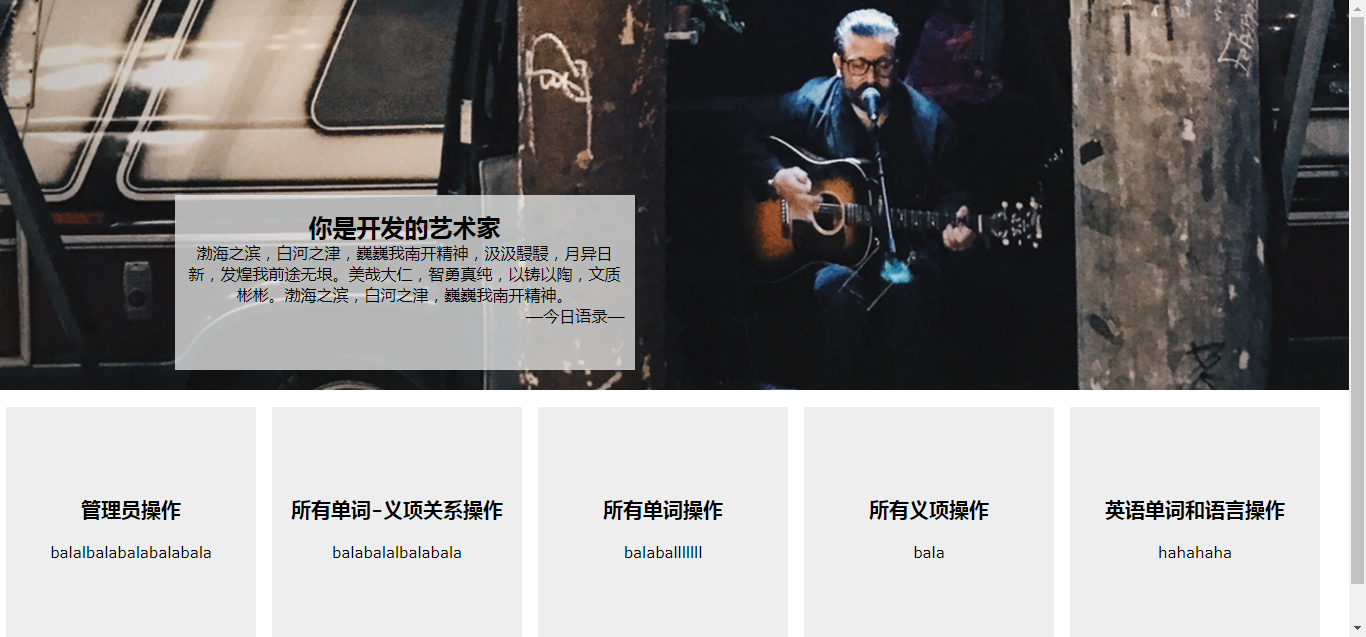
代码总结：

以上就是自顶向下的一套操作，很好的告诉我们从你输入一种语言名称点击“插入”按钮后发生在前端后端的各种操作。这就达到我们的目的了，主语其他一堆复杂的操作，都是建立在刚刚这一套的思想上的，有兴趣可以看我的github相关内容（包含所有代码）。

6.效果展示



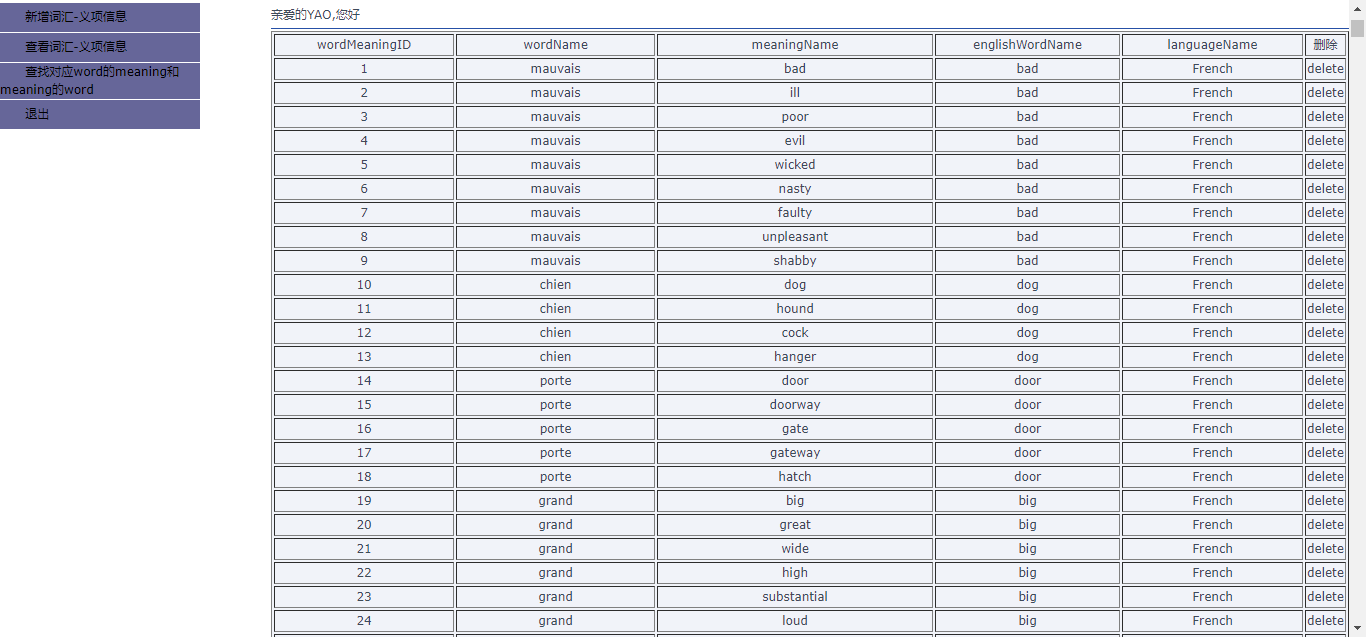
登录界面



登录后的主界面



增加界面

查看操作



要求的一些其他操作