|  |
| --- |
| 成都微视联软件技术有限公司 |
| WisdomReport 6000文档生成引擎简单设计 |
| 智慧辅助录入系统关键模块设计 |
|  |
| **易高辉** |
| **2018/7/9** |

|  |
| --- |
|  |

**目录**

[一、 概要 3](#_Toc518998016)

[二、 总体设计思想 3](#_Toc518998017)

[三、 0层设计 4](#_Toc518998018)

[四、 1层设计 5](#_Toc518998019)

[1. 核心思路 5](#_Toc518998020)

[2. 核心流程 6](#_Toc518998021)

[（1） 通过TempletTool创建文档模板 6](#_Toc518998022)

[（2） 为文档增加1个动态数据 6](#_Toc518998023)

[（3） 文档自动关联运算，增加新的动态数据 6](#_Toc518998024)

[五、 接口和数据结构 7](#_Toc518998025)

[1. 原始数据 7](#_Toc518998026)

[六、 Story建议 7](#_Toc518998027)

# 概要

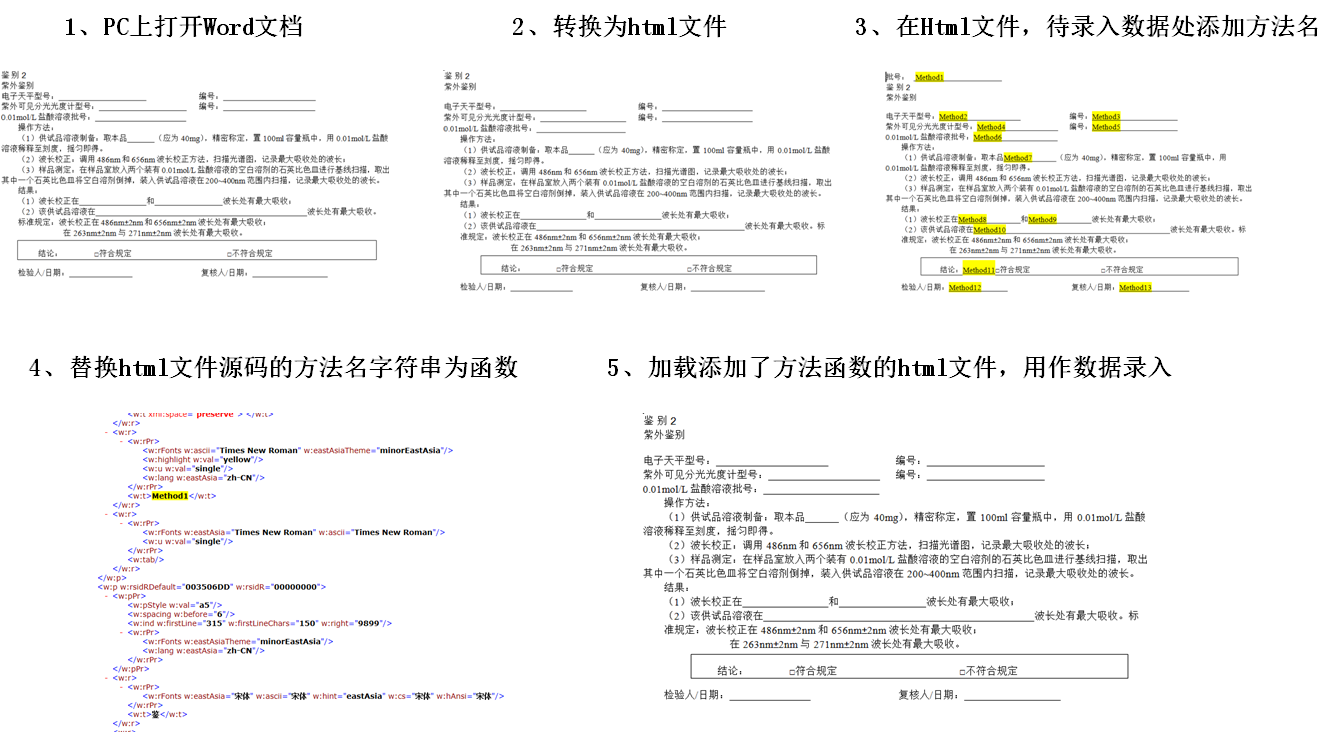
通过配置文件驱动，动态生成客户原始记录和报告，满足快速、低成本、大规模定制客户原始记录和报告需求。

本文为动态生成文档的设计说明书。

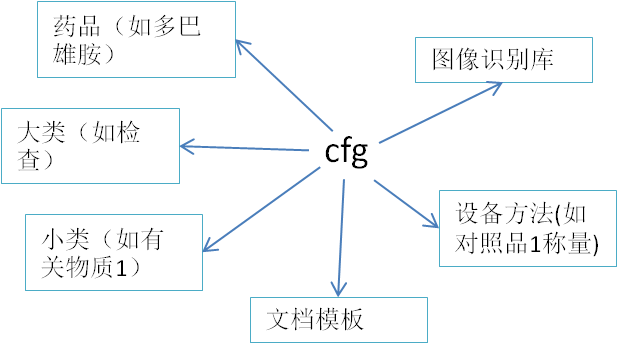
# 总体设计思想

**1、配置文件定义模板，通过模板引擎生成文档：**

**模板文档生成过程：**

****

**2、配置文件包含信息：**



# 0层设计

*帮助理解上下文*



1、**文档加载**：

（1）LoadActivity从Cfg获得“药品-大类-小类”目录层次结构， 并通过DataCheckIn加载为目录按钮，客户选择某个原始记录做操作，由DocFactory创建文档对象（Document）。

（2）Document从Cfg获得文档内容、排版信息，通过模板引擎加载为模板。然后，从Server端加载已经录入的数据到模板，形成文档呈现给客户。

2、**文档更新**：

手工输入数据，通过Document完成保存，手工输入过程，本身提供了输入数据在文档的位置信息。

自动识别数据，由DeviceRecognize从cfg获得数据在文档的位置信息，由ImageRecognize从Cfg获得图像识别库，完成内容识别。识别结果由Document完成保存。

# 1层设计

## 核心思路

由模板生成工具（TempletTool）创建生成文档资源（html文件），并打成cfg包。

Cfg作为核心数据结构，封装文档的静态模板、动态数据在文档的位置信息，以及动态数据关联运算的函数名。对Cfg的访问，做抽象封装，保证接口稳定。

Document表示整个药品的原始记录或报告，由多个子文档（小类对应的原始记录）构成。通过TempletEngin合成子文档。

TempletEngin获得Cfg的html文件，通过merge动作，将动态数据（OriginData）合并到模板文档中，并执行flush操作，完成关联运算，产生/刷新得到新的动态数据。

OriginData封装一个实验数据，包含该数据的值、在文档和数据库的定位信息，以及修改历史。

Utility是cfg的文档合成函数的实现库，通过“函数名：{函数实现， [数据源, …]} 字典方式封装。



## 核心流程

### 通过TempletTool创建文档模板

### 模板引擎加载药品xml配置文件MedicineProperty.xml，得到药品，大类，小类返回给app显示，通过配置文件中知名的url地址得到小类对应的html空白页面，app请求服务器数据进行同步通过回调将数据给模板引擎，模板引擎进行数据绑定返回绑定了数据的html给app显示，通过MedicineProperty.xml配置文件的method标签得到html文档中需要输入的部分的配置信息，用于后面的html的输入和刷新

### app为文档增加1个动态数据

### app调用计算接口，根据第二步得到的输入部分的配置信息，模板引擎可以通过method中的formula属性查找到与method唯一对应的计算公式，模板引擎解析公式，app请求服务器数据进行同步，app将同步后的数据给模板引擎，模板引擎进行数据绑定和对其他相关联的位置进行计算并刷新html，返回给app端

### app调用保存接口，模板引擎提取所有需要输入的数据（包括已经输入的数据和未输入的数据）返回给app部分，app部分调用后台的保存服务，将数据传入后台保存

### app点击自动录入按钮，app扫描设备二维码，访问模板引擎，将设备信息传给模板引擎，模板引擎得到当前小类下面所有需要自动录入的method中的所有设备属性与当前设备相同的method返回给app，app部分弹出列表供用户选择要录入的是哪一个method，app记录下来用户选择的method和扫描得到的结果一起传入给模板引擎。模板引擎进行数据绑定并且返回html给app进行显示

# 接口和数据结构

## 原始数据

**原始数据结构：**

class OriginData:

{

某个原始数据在文档的位置信息和值

动态数据在数据库的位置索引，文档加载时，用此作为索引，整体获得某个小类下所有的数据

int 药品编号

int 批次编号

int 大类编号

int 小类编号

int method 在小类的位置信息（实验方法）

string value

int version

int status

}

Class DocumentOriginData{

String className;

List<OriginData> originData;

}

**原始数据表设计：**

每个原始记录对应一张表

## 模板文档

**配置文件文档数据结构：**

* 1. 药品配置文件：

文件名：MedicineProperty.xml

数据结构参照文档结构

* 1. 设备配置文件：

文件名：DeviceProperty.xml

数据结构参照文档结构

## 模板引擎

得到所有药品名的接口：

Public Map<Integer ,String> getAllMedicines ()

{

返回药品Id，和名字的键值对

}

根据药品名得到checkClass集合的接口

Public Map<Integer,String> getCheckClassNameByMedicineName(String name)

{

返回CheckClassId，和名字的键值对

}

根据药品名和大类名得到checkAction的集合的接口

Public Map<Integer,String> getCheckActions (String MedicineName,String checkClassName)

{

返回checkActionId，和名字的键值对

}

数据合并接口：

String Merge(DocumentData dataList)

{

Foreach data in dataList

Find by data.method

调用计算接口

得到Medicines

将值保存到medicine对象中

数据绑定

返回静态Html

}

计算接口：

Public String CalculateAll(String CheckActionName)

{

Find medicines.getMedicine.getCheckClass.getCheckActions(CheckName)

Parse medicineProperty.xml

getDataFromHtmlTemplate

calculate()

}

通过设备信息得到当前checkAction下面需要用到该设备的methodName

Public List<String> getMethodNameByDeviceId(Integer deviceId)

{

返回所有method中deviceId属性=参数的method的名字列表

}

# Story建议

Story ID： DBE-Story-001

Story name： x

Story描述： x