# 1、实体类和xml，properties的解析一样

|  |
| --- |
| **public** **class** XmlEntry {    **public** String person ;  **public** Integer age;  **public** Integer sex;  **public** String name ;  **public** **double** salary; //薪水  **public** Integer classPersonNum;  **public** XmlEntry() {  }  **public** String getPerson() {  **return** person;  }  **public** **void** setPerson(String person) {  **this**.person = person;  } |

# 2、准备模板（后缀名也可以任意）

### 1、下面存放变量的标签一定要和将来map中的key（不管是不是对象产生的，但是必须有相应的名字与下面的标签匹配）对应上，否则不能够成功，而且会报错

### 2、通过1中描述，尽量让map中的和标签名字一样

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <root id="" comment="">  <person>  <person>${x.person}</person>  <age>${x.age}</age>  <name>${name}</name>  <sex>123</sex>  </person>  </root> |

# 3、将来要导入的map（这里通过方法实现的）

## 1、注意2中的name 和有x的age，可以直接存放对象，但是只要按照各自的标准放入即可

|  |
| --- |
| **private** **static** Map demo(){  XmlEntry p =**new** XmlEntry();  p.setAge(24);  p.setPerson("张宇晋");  // p.setName("HealerJean");  p.setSex(1);  Map map = **new** HashMap();  map.put("name", "zhangyujin");  map.put("x", p);  **return** map;  } |

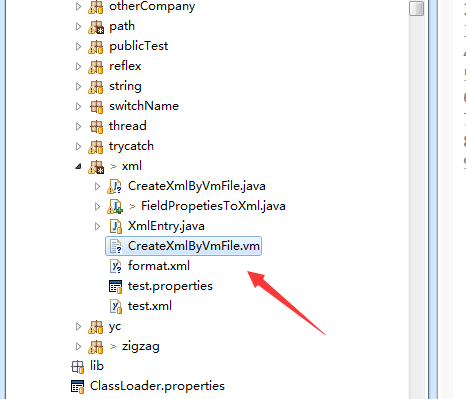
# 4、开始实现

## 1、Configuration 和Template 是最重要的一个决定位置，一个决定模板

|  |
| --- |
| /\*\*  **@param** filePath 模板的路径（目录）  \* **@param** TemplateName 模板的名字  \* **@param** dataMap 数据源  \* **@return**  \*/  **public** **static** String createXmlFile(File filePath,String TemplateName, Map dataMap) **throws** TemplateException, IOException {    //得FreeMarker配置对象  // 创建Configuration对象  Configuration cfg = **new** Configuration();  cfg.setEncoding(Locale.*CHINA*, "UTF-8");  /\*\*  设置FreeMarker的模版文件位置  1、类路径、2、文件系统以及3、Servlet Context。  \* setClassForTemplateLoading(Class clazz, String pathPrefix);  setDirectoryForTemplateLoading(File dir) throws IOException;  setServletContextForTemplateLoading(Object servletContext, String path);  \*/  // 类路径之下  // cfg.setClassForTemplateLoading(CreateXmlByVmFile.class,"");  cfg.setDirectoryForTemplateLoading(filePath);  // 创建Template对象  Template template = cfg.getTemplate(TemplateName);  template.setEncoding("utf-8");    StringWriter w =**new** StringWriter();  // 生成xml  template.process(dataMap, w);    **return** w.toString();    } |

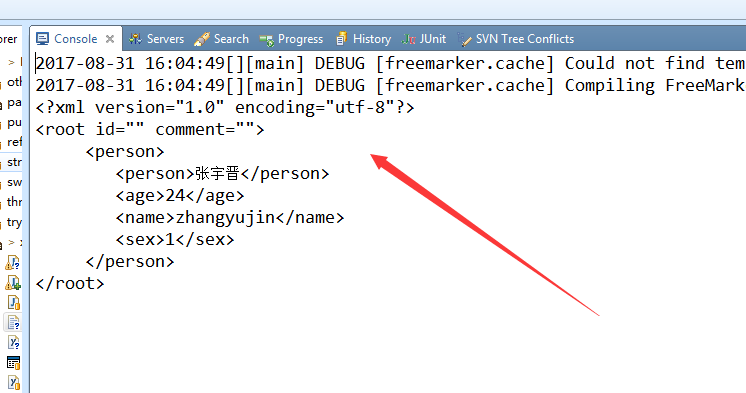
## 2、开始调用

### 1、模板的位置如图



|  |
| --- |
| **public** **static** **void** main(String args[]) **throws** TemplateException, IOException{    File file = **new** File(CreateXmlByVmFile.**class**.getResource("").getFile());  Map map =*demo*();  String xml = *createXmlFile*(file,"CreateXmlByVmFile.vm", map);  System.*out*.println(xml);    } |

# 3、执行



# 5、对象反射生成map

|  |
| --- |
| /\*\*  \* Description:  \* Date:2017/11/21 11:20  \*  \* @author cl  \*/  public class PubUtils {  /\*\*  \* 属性赋值  \*/  public static void setProperty(Map<String, Object> mailMap, Object object) throws Exception {  Class clazz = object.getClass();  Field[] fields = clazz.getDeclaredFields();  for (Field field : fields) {  field.setAccessible(true);  //得到此属性的name  String name = field.getName();  //得到此属性的值  String value = field.get(object) + "";  mailMap.put(name, value);  }  }  } |