**JAVA程序编码规范**

目 录

[1 目的 3](#_Toc427328139)

[2 执行细则 3](#_Toc427328140)

[2.1 程序目录结构 3](#_Toc427328141)

[2.2 新建接口与类时，应遵循命名规范 3](#_Toc427328142)

[2.3 单元测试 4](#_Toc427328143)

[3 相关技术： 4](#_Toc427328144)

[4 定义Entity 4](#_Toc427328145)

[5 定义Repository 7](#_Toc427328146)

[6 定义Service 8](#_Toc427328147)

[7 定义Controller 9](#_Toc427328148)

[8 定义View 11](#_Toc427328149)

[9 Ajax操作 15](#_Toc427328150)

[10 其它 16](#_Toc427328151)

[10.1 分页控制 16](#_Toc427328152)

[10.2 excel/pdf下载() 19](#_Toc427328153)

# 目的

　　规范公司内部开发人员的编码习惯，使开发人员具有一致的编码风格，从而使得代码更具有可读性与可维护性。也使得新的开发人员能够比较快速地进入开发状态，以及开发人员能够轻松地在不同项目间进行切换。

# 执行细则

## 程序目录结构

　　项目代码基于maven进行构建，采用maven标准代码结构，代码存放位置严格按照maven标准。

　　各大模块的程序划分以package名进行区分，比如com.comwave.code.rdp与com.comwave.domain.fxdm等。

## 新建接口与类时，应遵循命名规范

　　新建接口与类时，应遵循以下命名规范：

　　接口名称，以大写字母“I”开头，以已相应逻辑名称结尾，名称中含有的各单词的第一个字母应该大写，如：IOrderBatchManager。

　　接口实现类的名称，由接口名称去掉“I”的剩余部分和 “Impl”字符串组成，如：OrderBatchManagerImpl。

普通类名称，应取个有业务意义的名称，以便从类名称上就能看出它所代表的业务意义，如：OrderBatch。

其他特殊意义类，遵照相应规则定义。如Repository类名称，由Domain名称 + Repository组成；如DAO类名称，由Domain名称 + Dao组成；如Service类名称，由Domain名称 + Service组成；

* + - 接口一般放于service包下，如：com.comwave.swiftnet.fund.service。repository一般放于repository包下，如：com.comwave.swiftnet.fund.repository。
    - DAO一般放于dao包下，如：com.comwave.swiftnet.fund.dao。
    - 接口实现类一般放于service.impl包下，如：com.comwave.swiftnet.fund.service.impl。
    - 业务对象一般放于domain包下，如：com.comwave.swiftnet.fund.domain。

## 单元测试

　　必须为类中的每一个重要函数编写对应的单元测试(JUnit)。重要的处理类，在具备了足够的单元测试后，还应该编写小规模的集成测试，测试流程的处理过程是否达到预期的效果。

　　测试程序应该放在与该模块相应的test源文件夹下，包名与被测试类的包名相同。测试程序的命名为在被测试类的名称前加上“Test”前缀，如：TestOrderBatchManagerImpl。测试程序使用到的Spring配置文件或DataSet数据文件，以“springContext-”，“dataset-”作为前缀，如：springContext-orderBatchManager，dataset-orderBatchManager。测试程序使用到的配置文件皆应放于与测试程序相同的包名下，不可与正式的配置文件同目录层次。

　　在编写DAO service实现类的测试时，应结合Spring与DBUnit进行单元测试。

　　在编写其他单元测试时，如果引用到其他接口提供的方法，这时需要进行解依赖的处理，请一率使用继承已有实现类或者Dimple工具包的方法提供接口的Stub，以便解除对其他接口实现类的依赖。

# 相关技术：

Spring：Java Bean管理、事物控制

Spring MVC：MVC框架

JPA：Java持久化框架

Hibernate：JPA实现框架

Velocity：页面模版引擎

Javascript：前端页面控制脚本

# 定义Entity

1. 使用JPA注解来定义一个Entity，一个Entity对应一张表格。
2. 使用@Entity注解标识Entity为一个实体，配合@Table(name = “TableName”)[注解使用，通过name](mailto:注解和@Entity一起使用。使用name)来属性指定映像的表名。
3. 使用@Id注解来标识表中主键，配合@Column使用指定对应的字段名。
4. 普通表字段通过@Column来注解，通过name属性指定表中字段名。
5. 在注解普通表字段时不要添加length或columnDefinition属性(这些属性是给工具使用，通过Entity生成表时用到的。这里不需要定义这些属性)。
6. 当表中字段是日期类型时，使用@Column注解时还需要配合@Temporal(TemporalType)来指定日期的类型。如：是Date 还是 DateTime类型。
7. 遇到主键是复合主键类型时

7.1)对于复合主键，JPA注解提供了两种使用方式，分别是@IdClass和 @Embeddable，@EmbeddedId 。这里要求使用@IdClass注解方式，这样在Entity对象里会更加直观的看到对应的所有属性。

7.2)在使用@IdClass注解方式时，对应的主键类命名一般采用Entity名称+PK方式。另外，要使用@IdClass注解，PK类还需满足三个要求，分别是可串行化，提供无参构造函数，重写hashCode和equals方法。

1. 下面是一个带复合主键的Entity示例：

8.1) PK 定义

**public** **class** ThirdInfoScopePK **implements** Serializable {

**private** Product product;

**private** Integer thirdInfoType;

**public** Product getProduct() {

**return** product;

}

**public** **void** setProduct(Product product) {

**this**.product = product;

}

**public** Integer getThirdInfoType() {

**return** thirdInfoType;

}

**public** **void** setThirdInfoType(Integer thirdInfoType) {

**this**.thirdInfoType = thirdInfoType;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + ((thirdInfoType == **null**) ? 0 : thirdInfoType.hashCode());

result = prime \* result + ((product == **null**) ? 0 : product.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

ThirdInfoScopePK other = (ThirdInfoScopePK) obj;

**if** (thirdInfoType == **null**) {

**if** (other.thirdInfoType != **null**)

**return** **false**;

} **else** **if** (!thirdInfoType.equals(other.thirdInfoType))

**return** **false**;

**if** (product == **null**) {

**if** (other.product != **null**)

**return** **false**;

} **else** **if** (!product.equals(other.product))

**return** **false**;

**return** **true**;

}

}

8.2) Entity 定义

|  |
| --- |
| @Entity  @Table(name = "THIRD\_INFO\_SCOPE")  @IdClass(ThirdInfoScopePK.**class**)  **public** **class** ThirdInfoScope {  /\*\* 产品 \*\*/  @Id  @ManyToOne(targetEntity = Product.**class**)  @JoinColumn(name = "PROD\_CODE", foreignKey = @ForeignKey(name = "FK\_THIRD\_INFO\_SCOPE"))  **private** Product product;  /\*\* 类别 \*\*/  @Id  @Column(name = "THIRD\_INFO\_TYPE")  **private** Integer thirdInfoType;  /\*\* 最终授信日期 \*\*/  @Column(name = "CREDIT\_APRV\_DATE")  @Temporal(TemporalType.*TIMESTAMP*)  **private** Date creditAprvDate;  //getter/setter  } |

# 定义Repository

1. 定义repository时，命名使用“Entity名称”+“Repository”命名
2. 通过@Repository注解标志一个类为repository
3. 通过repository访问数据库时，如果参数较少，可直接通过方法名称来指定条件，不需要通过@Query注解来写sql语句。比如：通过名称查询数据，这时方法名称可以是findByName(String name)命名，不需要提供@Query。
4. 如果参数较多或包含嵌套对象，则需要使用@Query书写sql语句
5. 继承spring-data-jpa提供的repository接口时需要提供泛型，第一个是Entity类，第二个是主键。如果主键是复合主键，则使用Entity+PK类，其他使用基类。
6. 定义repository示例：

|  |
| --- |
| @Repository  **public** **interface** ThirdInfoScopeRepository **extends** JpaRepository<ThirdInfoScope, ThirdInfoScopePK> {  /\*\*  \* 根据产品编号获取第三方范围  \*  \* **@param** prodCode  \* 产品编号  \* **@return** 第三方范围  \*/  @Query("select m from ThirdInfoScope m where m.product.code=:prodCode")  **public** List<ThirdInfoScope> findThirdInfoScopeByProdCode(@Param("prodCode") String prodCode);  /\*\*  \* 获取第三方范围  \*  \* **@param** prodCode  \* 产品编号  \* **@return** 第三方范围  \*/  @Query("select m from ThirdInfoScope m where m.product.code=:prodCode and m.thirdInfoType=:thirdInfoType")  **public** ThirdInfoScope findThirdInfoScopeById(@Param("prodCode") String prodCode, @Param("thirdInfoType") Integer thirdInfoType);  } |

# 定义Service

1. 在Service实现类需要实现事务
2. 在Service中尽量不要有直接访问数据库的操作，通过注入repository或Dao来间接访问数据库
3. Service不需要接口，直接写类；
4. 定义Service示例：

|  |
| --- |
| @Service("thirdInfoScopeService")  @Transactional  **public** **class** ThirdInfoScopeService {  **private** **static** **final** Logger *logger* = LoggerFactory.*getLogger*(ThirdInfoScopeServiceImpl.**class**);  @Autowired  **private** ThirdInfoScopeRepository thirdInfoScopeRepository;  **public** void saveThirdInfoScopeByProdCode(ThirdInfoScope thirdInfoScope) {  …  }  } |

# 定义Controller

1. Controller类不做逻辑处理，逻辑处理放到Service中实现。Controller只是负责接收View传过来的参数给Service处理，然后将Service返回的数据封装到Model中返回给View。
2. 需要在类级别上提供@RequestMapping模块功能地址映像，然后在具体的请求处理方法上也提供@RequestMapping功能地址映像
3. 在Controller处理请求中不能够同时调用两个Service有事务改动DB的方法。（不然，如果第一个Service事务完成，这时第二个Service事务处理失败，回滚事务，这时第一个Service的事务无法实现回滚操作）
4. 定义Controller示例：

|  |
| --- |
| @Controller  @RequestMapping("/sysmanagement")  **public** **class** ThirdInfoScopeController {  **private** **static** **final** Logger *logger* = LoggerFactory.*getLogger*(ThirdInfoScopeController.**class**);  **private** **static** **final** Integer *PAGE\_SIZE* = 20;  @Autowired  **private** IThirdInfoScopeService thirdInfoScopeService;  @Autowired  **private** IProductService productService;  **public** Model init(Model model) {  model.addAttribute("products", productService.findAll());  model.addAttribute("thirdInfoTypes", ThirdInfoScope.*thirdInfoTypes*);  **return** model;  }  /\*\*  \* 删除第三方范围  \*/  @RequestMapping("/deleteThirdInfoScope")  @ResponseBody  **public** AjaxJsonResult deleteThirdInfoScope(Model model, @RequestParam String prodCode, @RequestParam String thirdInfoType) {  AjaxJsonResult result = **new** AjaxJsonResult();  **try** {  Product product = productService.findByCode(prodCode);  ThirdInfoScope thirdInfoScope = **new** ThirdInfoScope();  thirdInfoScope.setProduct(product);  thirdInfoScope.setThirdInfoType(Integer.*valueOf*(thirdInfoType));  thirdInfoScopeService.deleteThirdInfoScope(thirdInfoScope);  result.setSuccess(**true**);  result.setMessage("操作成功！");  } **catch** (Exception e) {  result.setSuccess(**false**);  result.setMessage("操作失败！");  *logger*.error("ThirdInfoScopeController.deleteThirdInfoScope failed!", e);  }  **return** result;  }  /\*\*  \* 新增第三方范围  \*/  @RequestMapping("/saveThirdInfoScope")  @ResponseBody  **public** AjaxJsonResult saveThirdInfoScope(Model model, @RequestParam String prodCode, @RequestParam String thirdInfoType) {  AjaxJsonResult result = **new** AjaxJsonResult();  **if** (StringUtils.*isBlank*(prodCode) || StringUtils.*isBlank*(thirdInfoType)) {  result.setSuccess(**false**);  result.setMessage("请选择产品或者第三方类型!");  **return** result;  }  **try** {  Integer thirdInfoTypeInt = Integer.*valueOf*(thirdInfoType);  ThirdInfoScope temp = thirdInfoScopeService.findThirdInfoScopeById(prodCode, thirdInfoTypeInt);  **if** (temp != **null**) {  result.setSuccess(**false**);  result.setMessage("已存在该记录!");  **return** result;  }  thirdInfoScopeService.saveThirdInfoScope(prodCode, thirdInfoTypeInt);  result.setSuccess(**true**);  result.setMessage("操作成功！");  } **catch** (Exception e) {  result.setSuccess(**false**);  result.setMessage("操作失败！");  *logger*.error("ThirdInfoScopeController.saveThirdInfoScope failed!", e);  }  **return** result;  }  } |

# 定义View

1. 一个页面模版代码不要太大，一般在200行左右，不要超过300行。可提取出一些公共的或页面功能块模版，然后通过引入的方式来实现减小页面篇幅。
2. 一些公共的页面，比如头部，尾部，可以使用宏定义。
3. 定义View示例：

|  |
| --- |
| #cw\_htmlheader("system.ThirdInfoScope.configuration",["js/common/calendar.js","js/sysmanagement/viewThirdInfoScope.js"])  #set($path = ${request.contextPath})  <div class="main" width="80%">  <div>  <p class="title01">#springMessage("system.ThirdInfoScope.configuration")</p>  <form id="form\_searchThirdInfoScope" action="#springUrl("/sysmanagement/searchThirdInfoScope")" method="post" >  <input type="hidden" name="curPage" id="curPage"/>  <p class="ts">  <span>产品名称: </span>  <select name="prodCode" id="prodCode" style="width:200px;">  <option selected="selected" value="">-请选择-</option>  #foreach($product in $products)  <option #if($!{proCode} == $!{product.code}) selected="selected" #end value="$!product.code">$!{product.name}</option>  #end  </select>  #if(${userUtils.hasFunctionCode("THIRD\_INFO\_SCOPE\_QUERY")})  <input type="submit" value="查询 "/>  #end  </p>  </p>  <p class="trgt">  #if(${userUtils.hasFunctionCode("THIRD\_INFO\_SCOPE\_SAVE")})  <input type="button" onclick="javascript:openDialog('div\_thirdInfoScope','新增第三方范围','550','150')" value="新增"/>  #end  </p>  </form>  <table class="tb01 mgt" >  <tbody>  <tr>  <th width="5%">序号</th>  <th width="40%">产品名称</th>  <th width="40%">第三方信息类型</th>  <th>操作</th>  </tr>  #if($thirdInfoScopes && $thirdInfoScopes.size()==0)  <tr>  <td colspan="4">#springMessage("title.noRecords")</td>  </tr>  #end  #foreach($record in $thirdInfoScopes)  <tr>  <td>$!{velocityCount}</td>  <td>$!{record.product.name}</td>  <td>$!{record.thirdInfoTypeStr}</td>  <td>  #if(${userUtils.hasFunctionCode("THIRD\_INFO\_SCOPE\_DELETE")})  <a class="up" onclick="javascript:deleteThirdInfoScope(this)" href="#" prodCode="$!{record.product.code}" thirdInfoType="$!{record.thirdInfoType}"> <img src="${path}/images/del.png">删除</a>  #end &nbsp;  </td>  </tr>  #end  </tbody>  </table>  </div>  <div id="div\_thirdInfoScope" class="popupbox1" style="display:none;">  <form id="form\_saveThirdInfoScope" class="simplevalidator" method="post" action="/sysmanagement/saveThirdInfoScope">  <table class="tb02">  <tr>  <th>产品名称</th>  <td>  <select name="prodeCode" id="form\_saveThirdInfoScope\_prodeCode" style="width:200px;" class="required">  <option selected="selected" value="">-请选择-</option>  #foreach($product in $products)  <option value="$!product.code">$!{product.name}</option>  #end  </select>  </td>  <th>第三方类型</th>  <td>  <select name="thirdInfoType" id="form\_saveThirdInfoScope\_thirdInfoType" class="required">  <option selected="selected" value="">-请选择-</option>  #foreach($thirdInfoType in $thirdInfoTypes.entrySet()))  <option value="$!thirdInfoType.key">$!{thirdInfoType.value}</option>  #end  </select>  </td>  </tr>  </tbody>  </table>  <p class="conct pad">  <input type="button" value=" 确认 " onclick="saveThirdInfoScope()"/>  <input type="button" value=" 取消 " onclick="closeDialog('div\_thirdInfoScope')" />  </p>  </form>  </div>  <p class="page">  #cw\_pageable($page)  </p>  </div>  #cw\_commonfooter()  #cw\_htmlfooter() |

# Ajax操作

1. 关于屏蔽层。当用户发送一个请求时，需要提供屏蔽层。屏蔽层的添加可以在js开始时添加。取消则可以在Ajax请求的成功，失败或完成时取消。
2. Ajax示例：

|  |
| --- |
| **function** submitForm(status) {  isSubmit = status;  // 表单字段验证  …  …  // 其他处理  **if** (isSubmit) {  confirmInfo\_1("提示", "您确认要提交？", **function**() {  bodymask("请稍候..."); // 添加屏蔽层  // 其他操作处理  $.ajax({  url : context\_path + "/process/loan/saveStepReleaseApply",  type : "POST",  data : $("#form1").serialize(),  dataType : "json",  success : **function**(data) {  bodyunmask(); // 取消屏蔽层  alertInfo('提示', data.message, **function**() {  **if** (data.success) {  window.location.href = context\_path + "/workList/viewWorkList";  }  });  },  complete : **function**() {  // bodyunmask();  }  });  });  } **else** {  bodymask("请稍候..."); // 添加屏蔽层  // 其他处理  $.ajax({  url : context\_path + "/process/loan/saveStepReleaseApply",  type : "POST",  data : $("#form1").serialize(),  dataType : "json",  success : **function**(data) {  bodyunmask(); // 取消屏蔽层  alertInfo('提示', data.message, **function**() {  **if** (data.success) {  window.location.href = context\_path + "/workList/viewWorkList";  }  });  },  error: **function**(e) {  bodyunmask(); // 取消屏蔽层  alertInfo('发生系统错误，请稍后再试!');  }  });  }  } |

# 其它

## 分页控制

PS : 这里DAO只是一个关于分页实现的示例：

|  |
| --- |
| @Service("thirdInfoScopeDao")  @Transactional  **public** **class** ThirdInfoScopeDao **extends** BaseJpaDAO **implements** IThirdInfoScopeService {  **private** **static** **final** Logger *logger* = LoggerFactory.*getLogger*(ThirdInfoScopeDao.**class**);  @Autowired  **private** ThirdInfoScopeRepository thirdInfoScopeRepository;  @SuppressWarnings("unchecked")  **public** QueryPage<Object[]> findThirdInfoScopeByPage(String prodCode, QueryPage<Object[]> page) {  StringBuffer jql = **new** StringBuffer("select t from ThirdInfoScope t ");  jql.append(" where 1=1 ");  List<Object> params = **new** ArrayList<Object>();  **if** (StringUtils.*isNotBlank*(prodCode)) {  jql.append(" and t.product.code = ? ");  params.add(prodCode);  }  jql.append(" order by t.product.code");  **return** findByJQL(jql, page, params);  }  } |

**Controller：部分范例如下，详细信息可参考**ThirdInfoScopeController类：

|  |
| --- |
| @Controller  @RequestMapping("/sysmanagement")  **public** **class** ThirdInfoScopeController {  **private** **static** **final** Logger *logger* = LoggerFactory.*getLogger*(ThirdInfoScopeController.**class**);  **private** **static** **final** Integer *PAGE\_SIZE* = 20;  @Autowired  **private** IThirdInfoScopeService thirdInfoScopeService;  @Autowired  **private** IProductService productService;  **public** Model init(Model model) {  model.addAttribute("products", productService.findAll());  model.addAttribute("thirdInfoTypes", ThirdInfoScope.*thirdInfoTypes*);  **return** model;  }  /\*\*  \* 查询第三方范围  \*/  @RequestMapping("/searchThirdInfoScope")  **public** String searchThirdInfoScope(Model model, @RequestParam(required = **false**) String prodCode, @RequestParam(required = **false**, defaultValue = "1") Integer curPage) {  QueryPage<Object[]> page = thirdInfoScopeService.findThirdInfoScopeByPage(prodCode, **new** QueryPage<Object[]>(curPage, *PAGE\_SIZE*));  model.addAttribute("page", page);  model.addAttribute("thirdInfoScopes", page.getContent());  model.addAttribute("proCode", prodCode);  init(model);  **return** "/sysmanagement/viewThirdInfoScope";  }  } |

Velocity：

1. 可通过page对象判断是否要查到记录；
2. 通过page.content引用查询记录列表；
3. 通过#cw\_pageable($page)加装分页列表；
4. 自定义gotoPage javascript方法；

Page宏定义：

##-----------------------------------------page---------------------------------------------------------

## ----------------------------------------

## bulid pageable

## ----------------------------------------

#macro(cw\_pageable $page)

#if($page && ${page.totalPages} != 0)

<input type="hidden" name="curPage" id="curPage" value="$!{page.number}"/>

#springMessage("totals")$!{page.totalElements}#springMessage("pages")

#if(${page.number}>1)

#set($last=${page.number}- 1)

<input type="button" value="上一页" onclick="gotoPageByNumber($!{last})"/>

#else

,

#end

#springMessage("index")

#set($curLen = (${page.totalPages}))

<select id="page\_select" onchange="gotoPage(this)">

#foreach($i in [1..$curLen])

<option value='$i' #if($i == ${page.number})selected="selected"#end>$i</option>

#end

</select>

#springMessage("page")

#if(${page.number}<$!{page.totalPages})

#set($next=${page.number}+1)

<input type="button" value="下一页" onclick="gotoPageByNumber($!{next})"/>

#else

/

#end

#springMessage("total")$!{page.totalPages}#springMessage("page")

#end

#end

gotoPage Jsfunction定义：

**function** gotoPage(obj) {

$("#curPage").val(obj.value);

$("#form\_searchThirdInfoScope").submit();

}

vm文件中引用宏定义：

<p class="page">

#cw\_pageable($page)

</p>

## excel/pdf下载()

controller

|  |
| --- |
| @Controller  @RequestMapping("/download")  **public** **class** DownloadFile {  @Autowired  **private** IUserService userService;  @RequestMapping("/viewDownLoad")  **public** String viewDownLoad() {  **return** "/download/viewDownLoad";  }  @RequestMapping("/downloadExcel")  **public** String downloadExcel(Model model) {  model.addAttribute("userList", userService.findAll());  **return** "excelView";  }  @RequestMapping("/downloadPdf")  **public** String downloadPdf(Model model) {  model.addAttribute("userList", userService.findAll());  **return** "pdfView";  }  } |

配置文件有两种方式

|  |
| --- |
| <!-- download file  1.classpath:views.properties  2./WEB-INF/views.xml  -->  <!--  <bean id="downloadResolver" class="org.springframework.web.servlet.view.ResourceBundleViewResolver">  <property name="order" value="1" />  <property name="basename" value="views" />  </bean>  -->  <bean id=*"downloadResolver"* class=*"org.springframework.web.servlet.view.XmlViewResolver"*>  <property name=*"order"* value=*"1"*/>  <property name=*"location"* value=*"/WEB-INF/views.xml"*/>  </bean> |

views.properties

|  |
| --- |
| excelView.(class)=com.comwave.smvc.demo.web.builder.ExcelBuilder  pdfView.(class)=com.comwave.smvc.demo.web.builder.PdfBuilder |

ExcelBuilder需继承AbstractExcelView

|  |
| --- |
| **public** **class** ExcelBuilder **extends** AbstractExcelView {  /\*  \* (non-Javadoc)  \* @see org.springframework.web.servlet.view.document.AbstractExcelView#buildExcelDocument(java.util.Map,  \* org.apache.poi.hssf.usermodel.HSSFWorkbook, javax.servlet.http.HttpServletRequest,  \* javax.servlet.http.HttpServletResponse)  \*/  @SuppressWarnings("unchecked")  @Override  **protected** **void** buildExcelDocument(Map<String, Object> model, HSSFWorkbook workbook, HttpServletRequest request, HttpServletResponse response)  **throws** Exception {  // 设置文件名  response.setHeader("Content-disposition", "attachment;filename=" + System.*currentTimeMillis*() + ".xls");  List<User> userList = (List<User>) model.get("userList");  HSSFSheet sheet = workbook.createSheet("Sheet1");  sheet.setDefaultColumnWidth(20);  // Excel Style  // Create Font Style  HSSFCellStyle simpleStyle = workbook.createCellStyle();  simpleStyle.setBorderRight(HSSFBorderFormatting.*BORDER\_THIN*);  simpleStyle.setBorderBottom(HSSFBorderFormatting.*BORDER\_THIN*);  HSSFFont simpleFont = workbook.createFont();  simpleFont.setFontHeightInPoints((**short**) 12);  simpleFont.setBoldweight(HSSFFont.*BOLDWEIGHT\_NORMAL*);  simpleFont.setFontName("Calibri");  simpleStyle.setFont(simpleFont);  HSSFRow header = sheet.createRow(0);  header.createCell(0).setCellValue("USER ID");  header.getCell(0).setCellStyle(simpleStyle);  header.createCell(1).setCellValue("USER NAME");  header.getCell(1).setCellStyle(simpleStyle);  header.createCell(2).setCellValue("FIRST NAME");  header.getCell(2).setCellStyle(simpleStyle);  header.createCell(3).setCellValue("LAST NAME");  header.getCell(3).setCellStyle(simpleStyle);  header.createCell(4).setCellValue("NOTE");  header.getCell(4).setCellStyle(simpleStyle);  header.createCell(5).setCellValue("USER STATUS");  header.getCell(5).setCellStyle(simpleStyle);  // create data rows  **int** rowCount = 1;  **for** (User user : userList) {  HSSFRow aRow = sheet.createRow(rowCount++);  aRow.createCell(0).setCellValue(user.getUserId());  aRow.createCell(1).setCellValue(user.getUserName());  aRow.createCell(2).setCellValue(user.getFirstName());  aRow.createCell(3).setCellValue(user.getLastName());  aRow.createCell(4).setCellValue(user.getNote());  aRow.createCell(5).setCellValue(user.getCNState());  }  }  } |

PdfBuilder需继承AbstractPdfView用法和下载excel类似