**Java Web编程规范**

**（版本：1.0）**

目录

[1 引言 1](#_Toc315694219)

[2 总体目录结构 1](#_Toc315694220)

[3 源代码 1](#_Toc315694221)

[3.1 包名命名规则 2](#_Toc315694222)

[3.2 源代码注释 2](#_Toc315694223)

[3.3 SQL语句 3](#_Toc315694224)

[3.4 异常 3](#_Toc315694225)

[4 Web目录 3](#_Toc315694226)

[4.1 WEB-INF 3](#_Toc315694227)

[4.2 view 4](#_Toc315694228)

# 引言

本文档不介绍编程技巧，仅仅是一个编程规范。

请您意识到以下两点：

* ***不良的编码和命名习惯将会浪费开发团队大量宝贵的时间。***

本文档在遵循《Java编程规范》基础之上，对web编程的细节规范做了详细的阐述。

# 总体目录结构

目录结构：

Documents 工程文档

config 配置文件

src 源代码

web Web目录

web\WEB-INF\lib 库文件目录

# 源代码

目录结构：

src

dev. \*

handler 业务类：类名以“Handler”结尾

domain 实体类

dao 数据访问类：类名以“DAO” 结尾

service 业务逻辑类：类名以“Service”结尾

system 系统类

tag 自定义标签类：类名以“Tag”结尾

util 工具类：类名以“Util”结尾

## 包名命名规则

命名规则：dev.子项目名称.\*。

例如：

dev.management.\*

dev.institution.\*

* service包：

业务类的命名规则：实体类名（或业务名称）+Service，例如：

MarketOrderPaymentService

* system包：

这个包里通常就放两个类：

WebappListener：Web应用程序监听类

SystemEnvironment：系统环境变量类

## 源代码注释

 代码注释，注释需要与代码对齐

* **类注释**

类注释，新增类需要注释类说明，创建人、创建时间；修改类需要注释修改人、修改时间、修改内容。

采用如下格式：

/\*\*

\* 类说明

\* <pre>

\* Modify Information:

\* Author Date Description

\* ============ =========== ============================

\* aaaa 2015-06-11 Create this file

\* bbbb 2015-06-13 增加xxxx方法

\* cccc 2015-07-05 修改xxxx方法

\* </pre>

\*/

* **方法注释**

方法注释，方法说明、入参说明、出参说明、异常信息，复杂算法需要增加算法说明。

采用如下格式：

/\*\*

\* 方法说明

\*/

## SQL语句

SQL语句的关键字需要大写， 非关键字如FieldName与TableName等首字母大写，如果由多个单词组成，那么每个单词的首字母均需要大写。

例如：

SELECT FieldNameList FROM TableName WHERE ... ORDER BY ... DESC

UPDATE TableName SET FieldName=value WHERE ...

 SQL禁用COUNT（\*）统计记录数，使用COUNT（1）代替。

## 异常

数据访问层直接抛出原始异常。

业务层需要自定义异常，将原始异常封装为最终用户可读的CodeException异常提示。

CodeException必须包含异常编码及异常信息，异常编码命名遵循

《Payment-CodeMessage.xlsx》

所有异常必须记logger日志。如：“logger.error("", e);”。

# Web目录

目录结构

web

css 样式文件目录

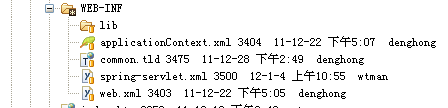
images 图片文件目录

js 脚本文件目录

view 页面文件目录

WEB-INF

## WEB-INF



* spring.xml：数据源及组件配置文件
* spring-mvc.xml：Action映射文件。

其中映射地址的每个单词的首字母大写。例如：

<property name="funcID" value="2704" />

这个value值对应的是权限编码及系统菜单节点表中的节点编号（NodeID）

## view

其中common目录存放公共页面，通常会有Frames.jsp框架页面，Common.jsp全局变量、参数等定义页面。

其他目录命名以权限编号命名，如2101，

* 页面文件名：每个单词的首字母大写。

（习惯命名：查询页面：Input.jsp；

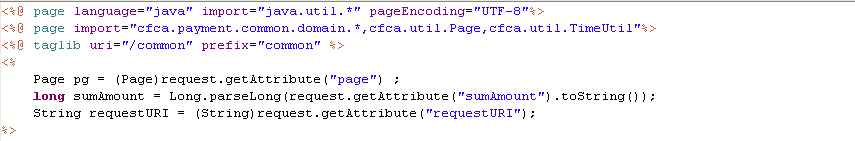
列表页面：List.jsp；

详情页面：Detail.jsp；

添加页面：Add.jsp；

修改页面：Edit.jsp）

* page import：有多个包需要引入时尽量放在一个import里面。
* 标签：缩进要统一美观。

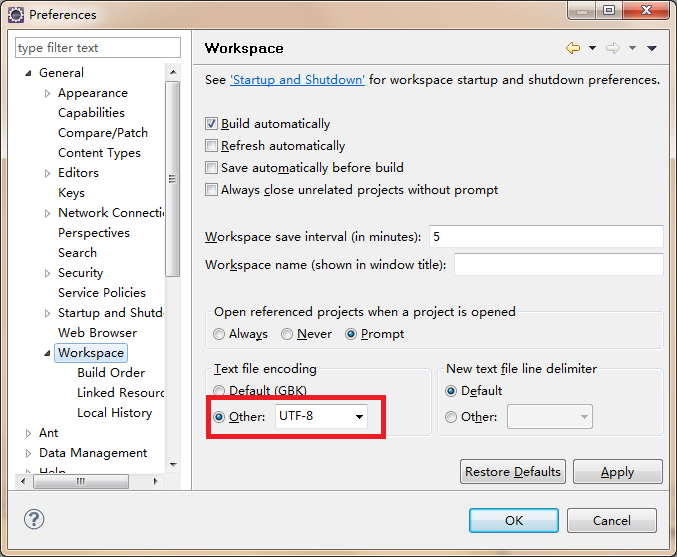


* 页面中的java语句：变量定义，对象的获取等统一放在最上面。
* Js代码：对没有位置要求的代码统一放在<head>标签里。例如：

<script type="text/javascript" >…</script>

**注：所有命名不得用简写，另外ID等特殊单词全部字母都要大写**

**源代码文件采用UTF-8编码。**



* **包名命名规则：dev.项目名称.\* 小写**

例如：

dev.common.\*

dev.gateway.\*

* **类命名规则：首字母大写的驼峰命名法**

示例：

SignAgreementService

* **避免在代码中使用无意义的数字，用静态常量代替**
* **常量命名规则：全大写以“\_”分隔单词**

示例：

INSTITUTION\_WHITELIST\_DATAVERSION\_PARAMETER\_CODE

* **方法命名规则：以动词开头首字母小写的驼峰命名法**

示例：getQuickBankList

* **变量命名规则：首字母小写的驼峰命名法**

示例：guaranteePaymentAccountNumber

* **Import语句规范：每个Import语句独立成行；禁用通配符。**

示例：

**import** dev.tools.util.GUID;

**import** dev.tools.util.IPUtil;

* **源代码尽量一行写完。**
* **SQL语句关键词大写，字段名和表名大小写与数据字典保持一致。**

例如：

UPDATE AuthTxToBank SET StatusManager=?,Status=15 WHERE TxNo=? SELECT \* FROM Corporation WHERE CorporationID = ?

* **SQL禁用COUNT（\*）统计记录数，使用COUNT（1）代替。**
* **方法相似相近原则：命名或功能类似的方法放在一起。**

示例：

findPaymentAccountByAccountNumber(String accountNumber);

findPaymentAccountByIndividualID(String individualID);

* **类职责明确**

DAO实现单表操作（增、删、改），尽量不要在DAO中有复杂的多表操作，逻辑判断等；

Query实现数据库表的查询类操作；

* **事务处理**

事务处理使用Spring注解方式；

同一个类内部没有事务的方法调用有事务的方法，事务无效；

**事务内部不能出现文件读写、通讯、大量查询操作，要抽离到事务之外，事务操作必须保证原子性。举例实现：**

@Component("BO1")

**public** **class** BO1 **extends** BaseBO {

// do方法上不能有事务标记

**public** **void** **do**(){

// 1.查询List

query1.getList();

// 2.数据库事务操作

transcationBO.doBusiness();

// 3.发送通知

sendNotification();

}

**private** **void** sendNotification(){...};

}

@Component("TranscationBO")

**public** **class** TranscationBO **extends** BaseBO {

@Transactional(propagation = Propagation.*REQUIRED*)

**public** **void** doBusiness(){

dao1.insert();

dao1.update();

dao2.update();

}

}

* **方法功能单一原则**

方法的职责尽量单一，不在方法中使用分支判断逻辑实现多重功能；

* **资源释放**

IO操作等需要使用结束close()的对象必须在try -catch-finally 的finally中 close()。

* **日志文件打印**

正常日志：记录方法入参、关键操作的结果、输出结果（info或debug）；

示例：

*logger*.info("XXXX");

异常日志：堆栈信息在需要转换日志的地方打印（如

Exception—>CodeException）（error）；

**try** {

……

} **catch** (Exception e) {

*logger*.error("", e);

String errorCode = "XXXX";

String errorMessage = "XXXX";

**throw** **new** CodeException(errorCode, errorMessage);

}

直接抛出CodeException的地方需要打印业务检验结果及其他必要的信息（error）。

最外层调用打印堆栈信息。（error）

* **异常处理**

底层方法Exception直接抛出到业务层处理，由业务层封装为CodeException，尽量不要在底层方法封装CodeException；

外层捕获异常时，分层捕获，先捕获CodeException，再捕获Exception。

示例：

**try** {

……

} **catch** (CodeException ce) {

**throw** ce;

} **catch** (Exception e) {

*logger*.error("", e);

String errorCode = "2001";

String errorMessage = "系统内部错误。";

**throw** **new** CodeException(errorCode, errorMessage);

}

* **代码注释，注释需要与代码对齐**

类注释，新增类需要注释类说明，创建人、创建时间；修改类需要注释修改人、修改时间、修改内容。

方法注释，方法说明、入参说明、出参说明、异常信息，复杂算法需要增加算法说明。

对方法体内分支及容易引起误解的地方，增加说明。

/\*\*

\* 类说明

\* <pre>

\* Modify Information:

\* Author Date Description

\* ============ =========== ============================

\* aaaa 2015-06-11 Create this file

\* bbbb 2015-06-13 增加xxxx方法

\* cccc 2015-07-05 修改xxxx方法

\* </pre>

\*/

/\*\*

\* 校验登录密码

\* **@param** phoneNumber 用户登录手机号

\* **@param** loginPassword 用户登录密码

\* **@param** ip 用户IP

\* **@param** sourceSystem 来源系统

\* **@param** privateKey 私钥证书

\* **@return** 校验结果，OK代表校验通过，其他代表校验失败原因

\* **@throws** Exception

\*/

//密码校验失败，登录失败次数加1

timesLoginFailed = corporationOperator.getTimesLoginFailed() + 1;

# 3目录结构

目录结构：

Documents 项目文档

config 配置文件

lib 库文件

src 源代码

# 4其它

本文未作说明的地方以个人经验和其它事实规范为准。