1. **JSON返回内容**
2. 增删改操作例子:

成功{status: 1; msg: “操作成功!”}

失败{status: 0; msg: “会话超时，请重新登录!”}

1. 查询例子: {status:1, data:[{},{}]}

status值: 1正常，0未登录，-1操作未被授权、被拒，-2签名验证失败，-3验证码过期，-4验证码输入有误，-5服务端程序抛异常。

总之，如果status<0直接弹出提示显示msg内容即可，0跳转至登录页面(参照现有例子)，1做正常业务逻辑。status<=0的情况都是框架拦截器做的，服务端业务代码不用关心。

**返回json时，内容必须外套{“status”:1, }，以便于和其他系统状态做统一判断。**

1. **controller方法返回值**

为了实现同一URL兼容PC浏览器、手机浏览器、AJAX、手机APP，所以禁止使用@ResponseBody，同一页面需要定义3个ftl模板文件，分别以**\_p**(PC浏览器版布局)、**\_m**(手机浏览器版布局)、**\_j**(json格式数据)为后缀，返回ModelAndView或字符串时无需关心后缀，框架拦截器会自动判断需要返回的类型并拼接后缀。前期不需要3个都定义，要么只有PC端，要么只有PDA Android端，按需定义即可。

非查询特别说明: 为了防止出现操作成功后刷新页面重新提交表单的情况，一定要按照这样的方式返回：

**return** Constant.***CLIENT\_TYPE\_J***.equals(CommonHelper.*clientType*(request))?WmsConstant.***VIEW\_NAME\_SUCCESS***:"redirect:/"+locale+"/xxx/zzz";

如果是AJAX或APP直接返回统一的成功JSON内容，如果是页面请求过来的一定要重定向。

1. **国际化**

因为该系统要实现多语言，所以禁止将文本信息直接写死在ftl、js、java代码中，要将文本以key-value的形式定义在/resource/i18n目录下的.properties文件或数据库的t\_core\_i18n\_value表中，在代码中引用key值:

1. ftl模板中引用i18n:

<@i18n key="key值">property:配置文件中的key值,i18n:国际化文件中的key值</@i18n>

1. js代码中引用i18n: $.i18n.prop("key值", "替换参数1", "替换参数2"…)
2. java代码中引用i18n:

RequestContext requestContext = **new** RequestContext(request);

requestContext.getMessage("key值")

在开发阶段可以只写中文版(zh\_CN结尾的)资源文件，其他语言后期找人翻译。

四、**静态文件**

1. 统一放在web工程的/src/main/webapp/resource目录下;

2. 引用时需要加上前缀:

ftl模板中: <@property key="prefix.resource"/>

js代码中: resourcePrefix

**五、ftl页面**

1. 统一放在web工程的/src/main/webapp/page目录下

2. 页面要求套用宏，以达到统一维护header、footer的目的，在jslist中引用自定义的js文件，csslist中引用自定义的css文件，参照现有页面。

**六、数据库操作**

纯读事务的service方法(直接被aop代理过的方法)上要加@ReadOnly注解、有多步增、删、改操作的service方法(直接被aop代理过的方法)上一定要加@Transactional注解。

**七、异常**

1. 不涉及资源关闭(如数据库连接、io流)禁止使用try-catch-finally语句块，如果涉及资源关闭，也禁止使用catch块，异常直接抛出，由框架统一处理；

2. 由框架提供的连接对象在使用完毕后不要关闭，你只关闭你创建的需要关闭的对象。

典型案例如下:

@ConnectionInjector

public int doSomethings(Connection conn) throws SQLException {

PreparedStatement stat = null;

try {

stat = conn.preparedStatement("XXX");

return stat.executeUpdate();

} finally {

if(stat!=null)

stat.close();

if(conn!=null)

conn.close();

}

}

**八、命名**

1. 类名: 名词在前，动词在后

2. 方法名: 动词在前，名词在后

3. 一定要做到简洁，如UserMapper接口中的增、删、改方法，**不要**定义成insertUser、deleteUser、updateUser，**只要**定义成insert、delete、update即可。

**九、代码提交**

禁止提交由Git和Eclipse创建的带.的文件或目录、target目录，每个子工程只允许提交src目录中的内容或pom.xml

