1、Java源文件

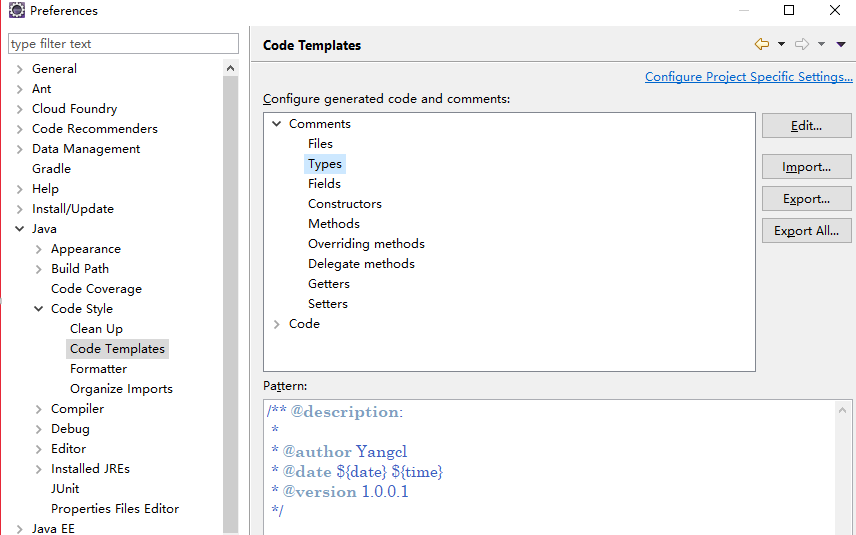
1.1、命名风格

Java文件以大写字母开头，全部使用英文，禁止使用中英结合的命名方式。这其中包括你的类名称、方法名称、变量名称和数据库字段等等。

1.2、类与方法上的注释

1.2.1、类注释

类开头必须有注释标识，不允许为空，不允许拷贝他人注释且不做修改。Eclipse添加自定义注释的方式如下：



/\*\* **@description**:

\*

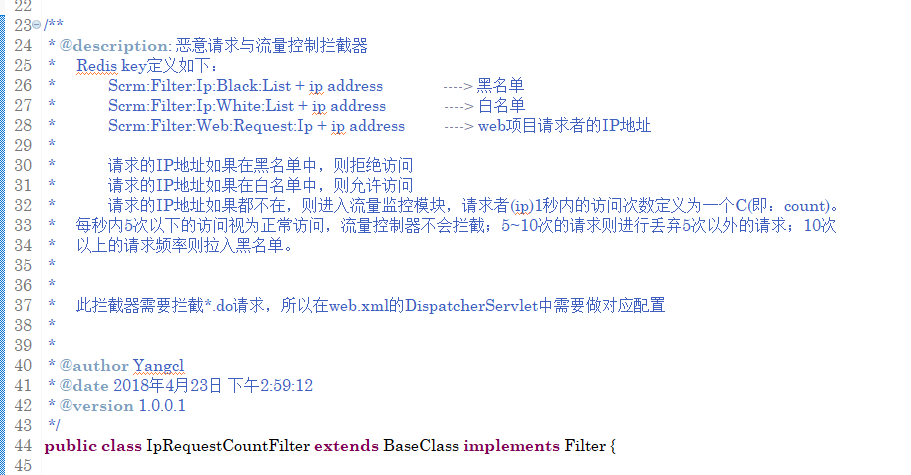
\* **@author** 此处填写你的名字，如：ShaoHua.Ma

\* **@date** ${date} ${time}

\* **@version** 1.0.0.1

\*/

Comments下的Types代表的是类注释，将会出现在你类的开头部分，配置好的效果如下：



务必详细描述这个类是做什么的，不要让接手你代码的人去猜。

1.2.2、方法注释

Comments下的Constructors代表类的构造函数，Methods代表类中的方法，Overriding methods代表你实现接口中的方法，这三个地方的配置如下：

/\*\* **@description**:

\*

\* ${tags}

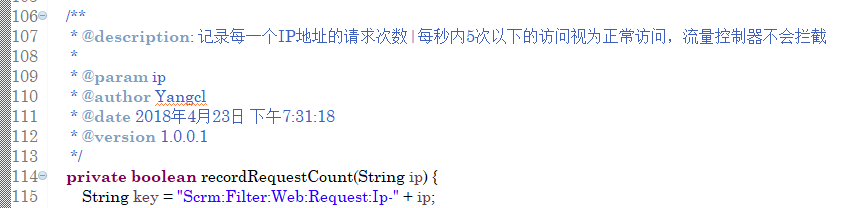
\* **@author** 此处填写你的名字，如：ShaoHua.Ma

\* **@date** ${date} ${time}

\* **@version** 1.0.0.1

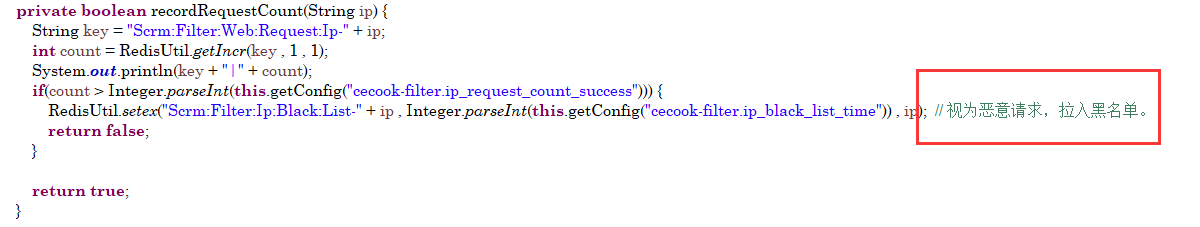
\*/

配置好的效果如下：



1.2.3、行注释

针对方法中具体的单行注释，根据具体情况在关键点处加以说明即可，组织好语言，让别人能看懂。比如：



1.3、声明

1.3.1、行变量声明

推荐一行一个声明，因为这样以利于写注释。即

private int level; *// indentation level*

private int size; *// size of table*

要优于

private int level, size;

不要将不同类型变量的声明放在同一行，例如：

private int foo, fooarray[]; *//WRONG!*

注意：上面的例子中，在类型和标识符之间放了一个空格，另一种被允许的替代方式是使用制表符：

private int level; *// indentation level*

private int size; *// size of table*

private Object currentEntry; *// currently selected table entry*

1.3.2、接口方法声明

针对SpringMvc接口类，第一个字母需要加I，比如：ISocialSmsService.java 而不是 SocialSmsService.java。需要做到表意明确；其次接口要添加public关键字。

**public** **interface** ISocialSmsService **extends** IBaseService<UserDemo, Integer> **{ // 花括号要在这里，不要放到下一行**

**public** JSONObject addInfo(UserDemo entity , HttpSession session);

**public** JSONObject deleteOne(UserDemo entity);

**public** JSONObject ajaxUploadFileCfile(String type , HttpServletRequest request);

**public** JSONObject apiHttpClientTest(HttpServletRequest request);

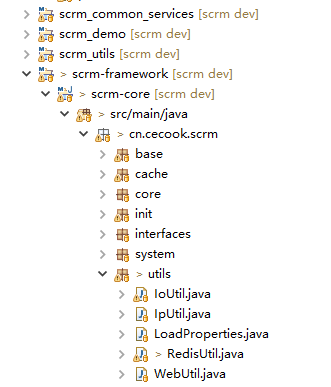
**public** JSONObject apiProcessorTest(ApiExampleDto dto);

}

针对Dubbo的RPC接口类，需要在第上述的基础上，添加Rpc作为标识，比如：ISocialSmsRpcService.java或SocialSmsRpcServiceImpl.java。需要做到表意明确。

1.3.3、包路径声明

所有的业务、工具等子包均放入cn.cecook.scrm下，如下图：



1.3.4、DTO|Entity|View

DTO：即数据传输模型，是一个查询操作的字段封装类。比如：UserInfoDto.java

Entity：实体类。是和数据库表中的字段一一对应的类。Entity需要保持纯净，禁止添加任何与数据库表无关的字段！比如：UserInfo.java

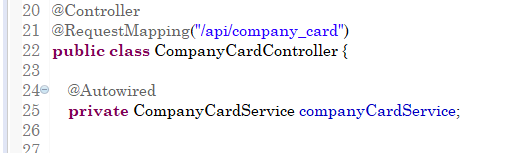
View：返回视图。view的概念来自于MySQL中的视图，他可以是多张表进行联合查询后的结果。在Mybatis的查询中，

以实体类作为返回视图往往无法满足需求，通常需要一个视图来作为返回值的承载者，比如：UserInfoView.java

1.4、项目现有问题整理

1.4.1、Controller层与Service层的命名规范

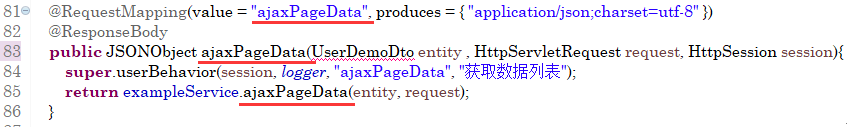
代码如下图：



RequestMapping(“/api/company\_card”) 尽量不要采用这种命名风格，不方便IDE进行全局搜索和定位。比如可以这样写：RequestMapping(“/companyCard”)；

依据是你Controller层的前两个单词：company 和 card进行驼峰组合。**这样写的好处是：**在Eclipse中使用Ctrl+Shift+R 就可以快速定位到CompanyCardController.java。

对于Controller层中的方法，通常这样写：



上图中的方法是获取一个列表页的分页数据信息。

@RequestMapping(value = "ajaxPageData", produces = { "application/json;charset=utf-8" })

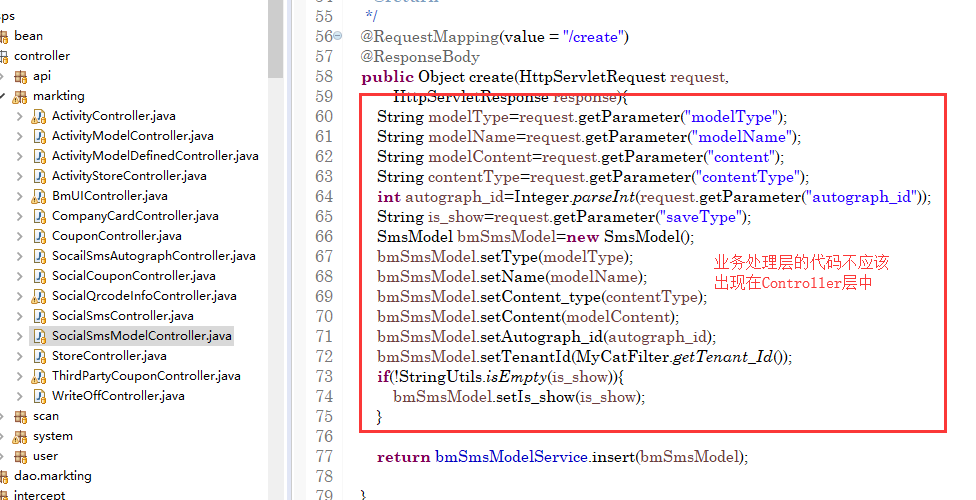
ajax(代表这个方法是一个异步请求)Page(页面)Data(数据)，针对异步的ajax请求全部以ajax开头。produces = { "application/json;charset=utf-8" }，即：向页面相应json数据，encoding为utf-8。

其中的UserDemoDto为数据传输模型。重申！公司约定禁止在实体类中(即：entity)添加与数据库表无关的字段的作为查询的辅助条件。

Controller层与Service层的命名规范：上图中三处画红线的地方都是ajaxPageData，从方法上的RequestMapping到方法名到Service层的方法名全部一致即可。Service层的方法可以自由组合Dubbo的RPC服务。

1.4.2、DTO传输模型规范

SPS项目中发现了下图中的代码：



这段代码中有2处比较严重的问题

1. 接收参数时缺少DTO数据传输模型来封装参数。
2. 业务处理写在了Controller层。

针对问题1，通常是程序开发人员对页面开发认识不够深导致，不知道Html标签的 id|class|name 该如何使用。事实上在一个form表单中，可以这样来提交页面数据：

页面HTML



JavaScript代码

**var** ajaxs = {

/\*\*

\* **@descriptions** 封装同步ajax请求

\*

\* **@param** type\_ 必填post or get，一般默认填写post

\* **@param** url\_ 所请求的后台路径 var url\_ = "login/login.do";

\* **@param** data\_ 请求参数 {name:name\_ , password:'0001'} or $("#the-form-id").serializeArray()，一般选择form序列化的数组

\* **@returns** json 后台向页面返回的json字符串

\*

\* **@date** 2016年5月23日上午10:34:27

\* **@author** Yangcl

\* **@version** 1.0.0.1

\*/

sendAjax:**function**(type\_, url\_ , data\_){

**var** msg\_ = **null**;

$.ajax({

dataType : "text",

type :type\_,

url : url\_,

data : data\_,

async : **false**,

success : **function**(msg) {

msg\_ = msg;

},

error: **function**(msg) {

msg\_ = msg;

}

});

**return** msg\_;

}

}

// ajax的方法提交要保存的数据

**function** addInfo() {

**var** type\_ = 'post';

**var** url\_ = '${basePath}example/ajaxAddInfo.do';

**var** data\_ = $("#form-example").serializeArray();

**var** obj = JSON.parse(ajaxs.sendAjax(type\_, url\_, data\_));

**if** (obj.status) {

}

malert(obj.msg, 'title');

}

Java代码

@RequestMapping(value = "ajaxAddInfo", produces = { "application/json;charset=utf-8" })

@ResponseBody

**public** JSONObject addInfo(UserDemoDto dto , HttpSession session){

**super**.userBehavior(session, *logger*, "ajax\_add\_info", "添加一条信息到数据库");

**return** exampleService.addInfo(dto , session);

}

**public** **class** UserDemoDto {

**private** Integer id;

**private** String userName;

**private** String password;

**private** String idNumber;

**private** Integer sex;

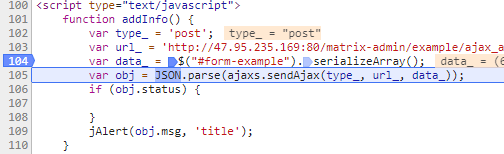
**private** String mobile;

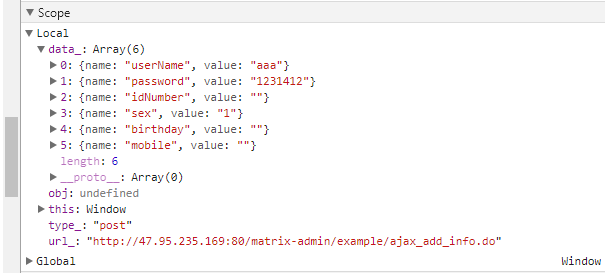
……

}

Jquery的serializeArray()方法可以将包裹在form标签体内的input/checkbox/radio/textarea/select等等标签序列化成对象数组，但前提是他们要有name属性。

驼峰格式的name属性值就可以直接转换成DTO中的字段，如果在谷歌浏览器的调试控制台进行断点调试，则会显示如下：





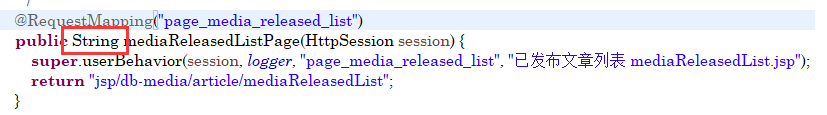
问题1也就迎刃而解了。针对问题二，我们约定如下：**禁止在Controller层书写任何与业务有关的代码！**业务处理全部要写到Service层！

1.4.3、去除Gson

项目还存在有使用谷歌的Gson 情况，所有使用的地方，全部替换成阿里巴巴的fastjson。

1.4.4、Controller层与页面跳转请求

页面跳转请求不同于Ajax的请求，只返回页面路径即可。



1.4.5、请求路径地址问题

所有对应的请求需要添加 .do 后缀。比如：user/userPageList.do

2、dubbo规范

太多了。。。请参考如下链接：

编码约定

<http://dubbo.apache.org/books/dubbo-dev-book/coding.html>

服务化最佳实践

<http://dubbo.apache.org/books/dubbo-user-book/best-practice.html>

缓存扩展

<http://dubbo.apache.org/books/dubbo-dev-book/impls/cache.html>

3、JavaScript源文件

3.1、对象封装



在处理页面的js脚本，约定使用JavaScript对象封装，禁止在JSP页面中乱写脚本，针对特定的业务逻辑要求开发人员对其进行统一封装。将使用的局部变量封装在对象内，方便复用、增强代码的可维护性、易读性、复用性。

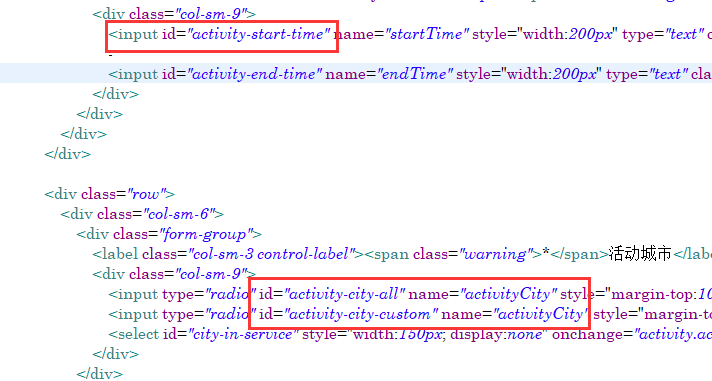
**注意！**

禁止使用全局变量！！

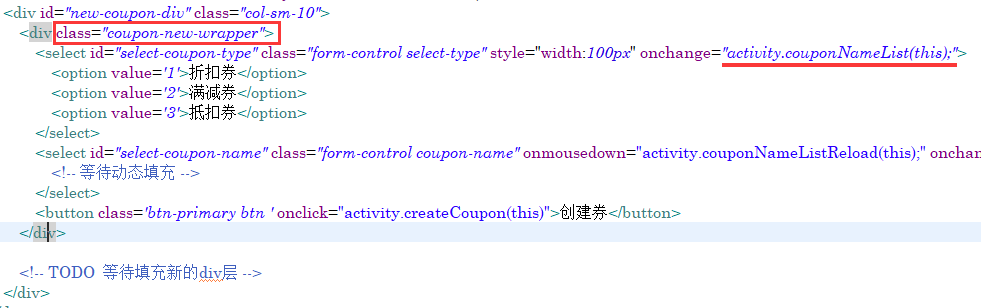
最大限度减少在JSP页面中暴露JS脚本！！

3.2、Html标签的 id|class|name 的使用说明

id与name：



class的使用：



在Html元素的定义中，id代表唯一性标识，与数据库的id很相似，一般不会重复。name属性则用于序列化form表单中的输入框类型的标签，比如checkbox、radio、input、select、textarea、select这几种。

你几乎看不到div、span、p这样的标签有name属性

举个id与name结合使用的列子：$("#the-form-id").serializeArray()

form标签中包含了需要输入的文本框(如：input)；serializeArray()方法则会遍历出所有带有name属性的html标签，并形成一个对象数组，具体请百度一下。

class则代表一类有规律的标签，他们可能有相同的样式或相同的行为属性。

比如在动态追加div标签的时候，就需要使用class 而不是 id