服务器端开发规范：

1、所有客户端的异步请求都走ActionServlet。

Action为最小的业务逻辑处理单元

IAction需要实现的方法DoAction(UserInfo user)

user为当前的用户。

返回为JSON字符串，自己定义，客户端自己处理，你要是愿意使用PlainText也可以。

首先创建一个自己的Action类，然后写自己的业务逻辑。

最后把这个Action注册到Base\_Actions表里，目前是手工插入。

为了让你的Action可以被系统调用，还要在Base\_Log里面定义改操作的业务日志，没有业务日志的请求一律不处理。

举个例子：

客户端有个请求R1需要处理action1，参数为p1=1,p2=2

客户端请求为AJ {action:"action1",p1:1,p2:2}

服务器端：

class Action1{

public IDTO doAction(UserInfo user){

// get Alert List

return new XXXDTO();

}

}

客户端处理这个返回的DTO的JSON。

IDTO定义了一个说明接口，用来约束返回值。

所有的DTO对象都要继承IDTO并且以DTO结尾。

2、所有模块的HTML请求都走TemplateServlet。

Template是每个模块的标准模板。

IModuleTemplate为模板输出的逻辑定义。

包含一个方法html(UserInfo user)，user用来验证权限并且格式化输出的模板。

验证权限的过程是解析字符串的过程，所以提交的模板越规范，验证的过程越简单。同时在实现的时候可以使用缓存技术来按角色缓存一些模板来提高效率。

3、所有的数据访问都由王辉和吴剑负责。

4、包结构说明

com.xunku.actions下面定义的业务Action的接口和实现。

com.xunku.actions.\*下面写自己模块的action的实现。

com.xunku.constant常量定义

com.xunku.controller 控制器实现

com.xunku.dao 数据库访问定义

com.xunku.daoImpl数据库访问实现

com.xunku.dto 数据传输接口定义

com.xunku.dto.\*具体业务的数据传输对象定义

com.xunku.module.controller模块的控制器定义

com.xunku.module.template 模块的模板定义

com.xunku.ObjectModel微博系统的对象模型定义

com.xunku.portal.ActionServlet action的servlet

com.xunku.portal.TemplateServlet 模板的servlet

com.xunku.portal.UploadifyServlet上传组件的servlet