# 研一三大项目

## 基于SSM的个人博客管理系统

### 总体实现：

课程设计。该项目的主要目的是学习和体验软件工程的设计过程，从而为以后设计大型工程奠定基础，从项目的任务分解，到概要设计、详细设计、编码、实现、部署的全过程由我独立负责。项目涉及到的技术有SSM框架、BootStrap、Ajax、Json、mysql。

### SSM框架：

可使用成熟的数据持久化工具

Hibernate iBATIS

关于持久化工具性能问题的考虑！

#### Spring

#### springMvc

controller

#### Mybatis

* Mybatis-config.xml //配置数据库
* 事务查询
* 逆向工程

generatorConfig\_xxxproject.xml。 //配置要生成对象所依据的表的信息

### 软件设计流程

### 技术特色-AJAX

与服务器异步交互；

浏览器页面局部刷新；

#### 为什么用到AJAX

//AJAX = Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。

//AJAX 不是新的编程语言，而是一种使用现有标准的新方法。

AJAX 最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。

AJAX 不需要任何浏览器插件，但需要用户允许JavaScript在浏览器上执行。

原因：传统的网页（不使用 Ajax）如果需要更新内容，必须重载整个网页页面。在web开发中有时有局部刷新的需求，这样做的好处是克服了页面整体刷新对网络速度受限的情况。

#### AJAX 应用

运用XHTML+CSS来表达资讯；

运用JavaScript操作DOM（Document Object Model）来执行动态效果；

运用XML和XSLT操作资料;

运用XMLHttpRequest或新的Fetch API与网页服务器进行异步资料交换；

注意：AJAX与Flash、Silverlight和Java Applet等RIA技术是有区分的。

案例：新浪微博、Google 地图、开心网等等。

#### 关键技术逻辑

onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

使用 Callback 函数

callback 函数是一种以参数形式传递给另一个函数的函数。

如果您的网站上存在多个 AJAX 任务，那么您应该为创建 XMLHttpRequest 对象编写一个标准的函数，并为每个 AJAX 任务调用该函数。

该函数调用应该包含 URL 以及发生 onreadystatechange 事件时执行的任务（每次调用可能不尽相同）：



#### XMLHttpRequest（XHR）对象

XMLHttpRequest 是 AJAX 的基础。

XMLHttpRequest 用于在后台与服务器交换数据。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

var xmlhttp;

if (window.XMLHttpRequest)

{

// IE7+, Firefox, Chrome, Opera, Safari 浏览器执行代码

xmlhttp=new XMLHttpRequest();

}

else

{

// IE6, IE5 浏览器执行代码

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

向服务器发送请求，使用 XMLHttpRequest 对象的 open() 和 send() 方法：

**xmlhttp.open("GET","ajax\_info.txt",true);**

xmlhttp.send();



* **Get还是Post**

与 POST 相比，GET 更简单也更快，并且在大部分情况下都能用。

然而，在以下情况中，请使用 POST 请求：

* 无法使用缓存文件（更新服务器上的文件或数据库）
* 向服务器发送大量数据（POST 没有数据量限制）
* 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

**为了避免得到的结果是一个缓存，请向 URL 添加一个唯一的 ID：**

xmlhttp.open("GET","/try/ajax/demo\_get.php?t=" + Math.random(),true);

**如果需要像 HTML 表单那样 POST 数据，请使用 setRequestHeader() 来添加 HTTP 头。然后在 send() 方法中规定您希望发送的数据：**

xmlhttp.open("POST","/try/ajax/demo\_post2.php",true);

xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");

xmlhttp.send("fname=Henry&lname=Ford");

关于RequestHeader的设置：<https://blog.csdn.net/runje/article/details/47720641>

当提交为单单数据时，可以使用“application/x-www-form-urlencoded”；当提交的是文件时，就需要使用“multipart/form-data”编码类型。

* **异步还是同步**

异步Async=true：通过 AJAX，JavaScript 无需等待服务器的响应，而是：在

等待服务器响应时执行其他脚本。当响应就绪后对响应进行处理

同步Async=true：不建议，JavaScript 会等到服务器响应就绪才继续执行。如果服务器繁忙或缓慢，应用程序会挂起或停止。注意：当您使用 async=false 时，请不要编写 onreadystatechange 函数 - 把代码放到 send() 语句后面即可：

* **服务器响应**

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。



#### onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。



1xx:信息响应类，表示接收到请求并且继续处理

2xx:处理成功响应类，表示动作被成功接收、理解和接受

3xx:重定向响应类，为了完成指定的动作，必须接受进一步处理

4xx:客户端错误，客户请求包含语法错误或者是不能正确执行

5xx:服务端错误，服务器不能正确执行一个正确的请求

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

注意： onreadystatechange 事件被触发 4 次（0 - 4）, 分别是： 0-1、1-2、2-3、3-4，对应着 readyState 的每个变化。

* 使用回调函数

如果您的网站上存在多个 AJAX 任务，那么您应该为创建 XMLHttpRequest 对象编写一个标准的函数，并为每个 AJAX 任务调用该函数。

该函数调用应该包含 URL 以及发生 onreadystatechange 事件时执行的任务（每次调用可能不尽相同）

function myFunction()

{

loadXMLDoc("/try/ajax/ajax\_info.txt",function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

});

}

调用：function loadXMLDoc(url,cfunc){xmlhttp.onreadystatechange=cfunc;}

* readystate的变化过程

用 window.alert 跟踪 xmlHttp.readystate 的变化，发现于原来它运行的机制是这样的。

首先创建一个 xmlHttpRequest 的对象之后 xmlHttp.readyState 的值是 0 了，然后xmlHttp.onreadystatechange = handlestatechange 没有运行。紧接着是 open()，这个函数发生了之后 xmlHttp.readyState 的值是 1 了0-1，那么就会有一个断点在 Open() 函数处断开，保留现场，紧接着又返回到 xmlHttp.onreadystatechange = handlestatechange 运行，然后再执行 Send() 函数，这个函数发生了之后 xmlHttp.readyState 的值是 2 了1-2，接着又返回到 xmlHttp.onreadystatechange = handlestatechange 运行，以此类推。

#### AJAX的优缺点

优点：

l AJAX使用Javascript技术向服务器发送异步请求；

l AJAX无须刷新整个页面；

l 因为服务器响应内容不再是整个页面，而是页面中的局部，所以AJAX性能高；

缺点：

l AJAX并不适合所有场景，很多时候还是要使用同步交互；

l AJAX虽然提高了用户体验，但无形中向服务器发送的请求次数增多了，导致服务器压力增大；

l 因为AJAX是在浏览器中使用Javascript技术完成的，所以还需要处理浏览器兼容性问题；

### 技术特色-JSON:JavaScript Object Notation(JavaScript 对象表示法)

#### 什么是JSON?

JSON 是存储和交换文本信息的语法。类似 XML。

JSON 比 XML 更小、更快，更易解析。

JSON 独立于语言：JSON 使用 Javascript语法来描述数据对象，但是 JSON 仍然独立于语言和平台。JSON 解析器和 JSON 库支持许多不同的编程语言。 目前非常多的动态（PHP，JSP，.NET）编程语言都支持JSON。

JSON 具有自我描述性，更易理解。

例：

{

"sites": [

{ "name":"菜鸟教程" , "url":"www.runoob.com" },

{ "name":"google" , "url":"www.google.com" },

{ "name":"微博" , "url":"www.weibo.com" }

]

}

在上面的例子中，对象 "sites" 是包含三个对象的数组。每个对象代表一条关于某个网站（name、url）的记录。

JSON-转换为 JavaScript 对象

JSON文本格式在语法上与创建 JavaScript 对象的代码相同。

由于这种相似性，无需解析器，JavaScript 程序能够使用内建的 eval() 函数，用 JSON 数据来生成原生的 JavaScript 对象。

#### 为什么使用 JSON？

对于 AJAX 应用程序来说，JSON 比 XML 更快更易使用：

**使用 XML**

读取 XML 文档

使用 XML DOM 来循环遍历文档

读取值并存储在变量中

**使用 JSON**

读取 JSON 字符串

用 eval() 处理 JSON 字符串

\*\*\*\*\*

**与 XML 相同之处**

JSON 是纯文本

JSON 具有"自我描述性"（人类可读）

JSON 具有层级结构（值中存在值）

JSON 可通过 JavaScript 进行解析

JSON 数据可使用 AJAX 进行传输

**与 XML 不同之处**

没有结束标签

更短

读写的速度更快

能够使用内建的 JavaScript eval() 方法进行解析

使用数组

不使用保留字

* **常见使用：**

JSON 最常见的用法之一，是从 web 服务器上读取 JSON 数据（作为文件或作为 HttpRequest），将 JSON 数据转换为 JavaScript 对象，然后在网页中使用该数据。

由于 JSON 语法是 JavaScript 语法的子集，JavaScript 函数 eval() 可用于将 JSON 文本转换为 JavaScript 对象。

eval() 函数使用的是 JavaScript 编译器，可解析 JSON 文本，然后生成 JavaScript 对象。必须把文本包围在括号中，这样才能避免语法错误：

var obj = eval ("(" + txt + ")");

#### 基本语法

JSON 对象使用在大括号({})中书写。

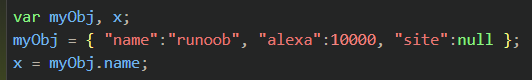
对象可以包含多个 key/value（键/值）对。

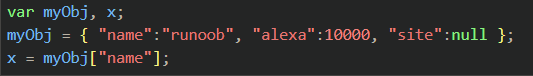
key 必须是字符串，value 可以是合法的 JSON 数据类型（字符串, 数字, 对象, 数组, 布尔值或 null）。

key 和 value 中使用冒号(:)分割。

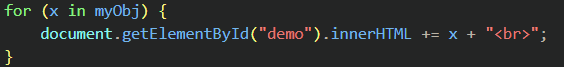
每个 key/value 对使用逗号(,)分割。

访问：可以使用点号（.）来访问对象的值，也可以使用中括号（[]）来访问对象的值。

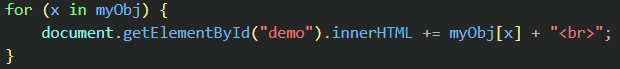




遍历：可以使用 for-in 来循环对象的属性：

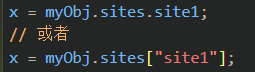


在 for-in 循环对象的属性时，使用中括号（[]）来访问属性的值：



**嵌套：JSON 对象中可以包含另外一个 JSON 对象**

可以使用点号(.)或者中括号([])来访问嵌套的 JSON 对象。



删除：我们可以使用 delete 关键字来删除 JSON 对象的属性

delete myObj.sites.site1;

elete myObj.sites["site1"]

删除数值的元素，数组的大小不变，delete 运算符并不是彻底删除元素，而是删除它的值，但仍会保留空间。

运算符 delete 只是将该值置为 undefined，而不会影响数组长度，即将其变为稀疏数组

#### JSON.parse()

JSON 通常用于与服务端交换数据。

在接收服务器数据时一般是字符串。

我们可以使用 JSON.parse() 方法将数据转换为 JavaScript 对象。

JSON.parse(text[, reviver])

text:必需， 一个有效的 JSON 字符串。

reviver: 可选，一个转换结果的函数， 将为对象的每个成员调用此函数。

\*解析前要确保你的数据是标准的 JSON 格式，否则会解析出错。

在线工具检测：https://c.runoob.com/front-end/53。

例如我们从服务器接收了以下数据：

{ "name":"runoob", "alexa":10000, "site":"www.runoob.com" }

我们使用 JSON.parse() 方法处理以上数据，将其转换为 JavaScript 对象：

var obj = JSON.parse('{ "name":"runoob", "alexa":10000, "site":"www.runoob.com" }');

我们可以使用 AJAX 从服务器请求 JSON 数据，并解析为 JavaScript 对象。

如果从服务端接收的是数组的 JSON 数据，则 JSON.parse 会将其转换为 JavaScript 数组

可以启用 JSON.parse 的第二个参数 reviver，一个转换结果的函数，对象的每个成员调用此函数。

var text = '{ "name":"Runoob", "initDate":"2013-12-14", "site":"www.runoob.com"}';

var obj = JSON.parse(text, **function (key, value) {**

**if (key == "initDate") {**

**return new Date(value);**

**} else { //使用 JSON.parse 的 reviver 函数时一定要注意遍历到最后的顶层对象 key 为 ""，需要返回 value。**

**return value;**

**}**});

#### JSON.stringify()

JSON 通常用于与服务端交换数据。

在向服务器发送数据时一般是字符串。

我们可以使用 JSON.stringify() 方法将 JavaScript 对象转换为字符串。

我们也可以将 JavaScript 数组转换为 JSON 字符串。

JSON.stringify(value[, replacer[, space]])

value:

必需， 要转换的 JavaScript 值（通常为对象或数组）。

replacer:

可选。用于转换结果的函数或数组。

如果 replacer 为函数，则 JSON.stringify 将调用该函数，并传入每个成员的键和值。使用返回值而不是原始值。如果此函数返回 undefined，则排除成员。根对象的键是一个空字符串：""。

如果 replacer 是一个数组，则仅转换该数组中具有键值的成员。成员的转换顺序与键在数组中的顺序一样。当 value 参数也为数组时，将忽略 replacer 数组。

space:

可选，文本添加缩进、空格和换行符，如果 space 是一个数字，则返回值文本在每个级别缩进指定数目的空格，如果 space 大于 10，则文本缩进 10 个空格。space 也可以使用非数字，如：\t。

* 解析数据

JSON 不能存储 Date 对象。

JSON.stringify() 会将所有日期转换为字符串。

var obj = { "name":"Runoob", "initDate":new Date(), "site":"www.runoob.com"};

var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;

* 解析函数

JSON 不允许包含函数，JSON.stringify() 会删除 JavaScript 对象的函数，包括 key 和 value。

我们可以在执行 JSON.stringify() 函数前将函数转换为字符串来避免以上问题的发生

var obj = { "name":"Runoob", "alexa":function () {return 10000;}, "site":"www.runoob.com"};

obj.alexa = obj.alexa.toString();

var myJSON = JSON.stringify(obj);

#### eval()函数

eval(string):函数可计算某个字符串，并执行其中的的 JavaScript 代码。

val("var a=1"); // 声明一个变量a并赋值1。

eval("2+3"); // 执行加运算，并返回运算值。

eval("mytest()"); // 执行mytest()函数。

eval("{b:2}"); // 声明一个对象。

对于服务器返回的JSON字符串，如果 jQuery 异步请求没做类型说明，或者以字符串方式接受，那么需要做一次对象化处理，方式不是太麻烦，就是将该字符串放于 eval()中执行一次。这种方式也适合以普通 javascipt 方式获取 json 对象，以下举例说明：

var u = eval('('+user+')');

**为什么要 eval 这里要添加 ('('+user+')') 呢？**

原因在于：eval 本身的问题。 由于 json 是以 {} 的方式来开始以及结束的，在 js 中，它会被当成一个语句块来处理，所以必须强制性的将它转换成一种表达式。

加上圆括号的目的是迫使 eval 函数在处理 JavaScript 代码的时候强制将括号内的表达式（expression）转化为对象，而不是作为语句（statement）来执行。举一个例子，例如对象字面量 {}，如若不加外层的括号，那么 eval 会将大括号识别为 javascript 代码块的开始和结束标记，那么{}将会被认为是执行了一句空语句。

**测试实例**

var user = '{name:"张三",age:23,'+

'address:{city:"青岛",zip:"266071"},'+ 'email:"iteacher@haiersoft.com.cn",'+

'showInfo:function(){'+

'document.write("姓名："+this.name+"<br/>");'+

'document.write("年龄："+this.age+"<br/>");'+

'document.write("地址："+this.address.city+"<br/>");'+

'document.write("邮编："+this.address.zip+"<br/>");'+

'document.write("E-mail："+this.email+"<br/>");} }';

var u = eval('('+user+')');

u.showInfo();

### 技术特色Ajax+Json的常见应用场景

#### ajax处理登录验证信息 解析json格式的String类型对象和ajax获得服务端的响应数据的方式request.responseText

#### 评论处理，用ajax将表单的数据提交给服务端，客户端解析从服务端传过来的json格式的List类型的数据：

用ajax来获取服务器端的数据，如果从服务器传到前端的数据都是JSON的字符串形式，在前端要读取数据，需要先将字符串转化成json格式的对象。

js中的eval（）函数就是将json格式的字符串转化为json格式的对象。

#### jquery中的ajax+json（项目中用的最多的）

#### 用Ajax进行数据验证、按需取数据、自动更新页面

#### 文本框中的应用

搜索下拉条：当我们在百度中输入一个“传”字后，会马上出现一个下拉列表！列表中显示的是包含“传”字的10个关键字。

其实这里就使用了AJAX技术！当文件框发生了输入变化时，浏览器会使用AJAX技术向服务器发送一个请求，查询包含“传”字的前10个关键字，然后服务器会把查询到的结果响应给浏览器，最后浏览器把这10个关键字显示在下拉列表中。

l 整个过程中页面没有刷新，只是刷新页面中的局部位置而已！

l 当请求发出后，浏览器还可以进行其他操作，无需等待服务器的响应！

校验用户名：当输入用户名后，把光标移动到其他表单项上时，浏览器会使用AJAX技术向服务器发出请求，服务器会查询名为zhangSan的用户是否存在，最终服务器返回true表示名为zhangSan的用户已经存在了，浏览器在得到结果后显示“用户名已被注册！”。

l 整个过程中页面没有刷新，只是局部刷新了；

l 在请求发出后，浏览器不用等待服务器响应结果就可以进行其他操作；