http://www.w3school.com.cn/

这个网站提供大量的网站开发技术知识，包括W3C的标准技术：HTML、CSS、XML 。以及其他的技术，诸如JavaScript、PHP、SQL等等。还提供了还有非常实用的在线测试功能。

# HTML

页面跳转的几种方式：

# CSS

## JQuery 实现 div弹出层示例

<%@ page contentType="text/html; charset=utf-8" language="java" errorPage="" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>jquery弹出层实例，jqueryajax.com原创</title>

<script type="text/javascript" src="http://www.jqueryajax.com/wp-content/uploads/2009/03/jquery-132min1.js"></script>

<script type="text/javascript">

$(function(){

$(".but").click(function(){

var str = "这里是信息内容，这里是信息内容！";

$(".form").html(str);

$("#faqbg").css({display:"block",height:$(document).height()});

var yscroll =document.documentElement.scrollTop;

$("#faqdiv").css("top","100px");

$("#faqdiv").css("display","block");

document.documentElement.scrollTop=0;

});

$(".close").click(function(){

$("#faqbg").css("display","none");

$("#faqdiv").css("display","none");

});

})

</script>

<style type="text/css">

/\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

冷漠夜空 2008

Email:lmyekong@163.com

QQ:135055506

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/

body,h2{margin:0 ; padding:0;}

#faqbg{background-color:#666666; position:absolute; z-index:99; left:0; top:0; display:none; width:100%; height:1000px;opacity:0.5;filter: alpha(opacity=50);-moz-opacity: 0.5;}

#faqdiv{position:absolute;width:400px; left:50%; top:50%; margin-left:-200px; height:auto; z-index:100;background-color:#fff; border:1px #8FA4F5 solid; padding:1px;}

#faqdiv h2{ height:25px; font-size:14px; background-color:#8FA4F5; position:relative; padding-left:10px; line-height:25px;}

#faqdiv h2 a{position:absolute; right:5px; font-size:12px; color:#FF0000}

#faqdiv .form{padding:10px;}

</style>

</head>

<body >

<div id="faqbg"></div>

<div id="faqdiv" style="display:none">

<h2>信息窗口<a href="#" class="close">关闭</a></h2>

<div class="form">

这里是提示信息！！

</div>

</div>

<p>&nbsp;</p>

<p align="center">

<input value="弹出" class="but" type="button" />

<input type="button" value="关闭" class="close" />

</p>

</body>

</html>

# JavaScript

## My97DatePicker时间控件详细使用说明

**onpicking事件**

实现日期选择联动：

<label class="date-title">就诊日期：</label>

<input id="d5221" class="Wdate" type="text" onFocus="var d5222=$dp.$('d5222');WdatePicker({onpicked:function(){d5222.focus();},minDate:'%y-%M-%d',maxDate:'#F{$dp.$D(\'d5222\')}'})"/>

至

<input id="d5222" class="Wdate" type="text" onFocus="WdatePicker({onpicked:doDateSearch, minDate:'#F{$dp.$D(\'d5221\')}'})"/>

选中日期后触发事件，调用doDateSearch方法：

<input id="d5222" class="Wdate" type="text" onFocus="WdatePicker({onpicked:doDateSearch, minDate:'#F{$dp.$D(\'d5221\')}'})"/>

## CKeditor所见即所得网页编辑器

CKEditor是FCKEditor的新版，CKEditor更好用，更简洁

### [CKEditor图片上传实现详细步骤(使用Struts 2)](http://blog.csdn.net/mydwr/article/details/8669594)

本人使用的CKEditor版本是3.6.3。CKEditor配置和部署我就不多说。

CKEditor的编辑器工具栏中有一项“图片域”，该工具可以贴上图片地址来在文本编辑器中加入图片，源文件一栏上可以输入图片的网络地址来引用图片，但是不能上传本地图片。

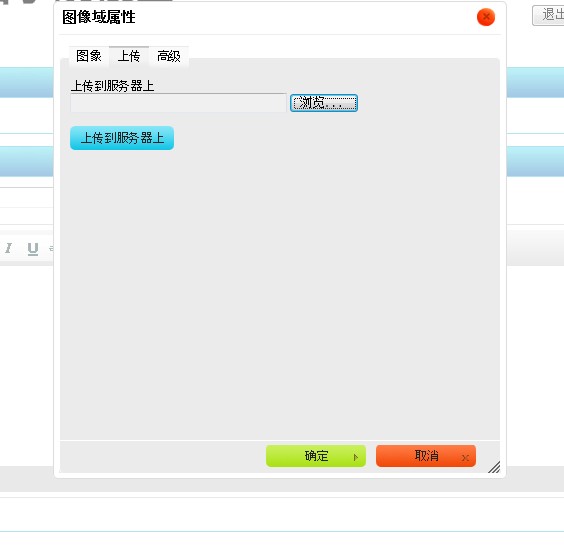
“预览”中有一大堆鸟语，看得很不爽。可以打开ckeditor/plugins/image/dialogs/image.js文件，搜索“b.config.image\_previewText”就能找到这段鸟语了，(b.config.image\_previewText||'')单引号中的内容全删了，注意别删多了。

扫除这个障碍，下面来研究图片上传。

**1.**首先，还是image.js这个文件，搜索“upload”可以找到这一段

id:'Upload',hidden:true

实际上上传功能被隐藏了，把上面的true改成false，再打开编辑器，就能找到上传功能了。



**2.**上面的只是一个上传页面。也就相当于一个HTML的form表单，要配置点击“上传到服务器上”按钮后请求的Action。可以在ckeditor/config.js中配置。

加入：

config.filebrowserUploadUrl="actions/ckeditorUpload";

var pathName = window.document.location.pathname;  
    //获取带"/"的项目名，如：/uimcardprj  
    var projectName = pathName.substring(0, pathName.substr(1).indexOf('/') + 1);  
    config.filebrowserImageUploadUrl = projectName+'/system/upload.do'; //固定路径

"ckeditorUpload"是请求的URL，也就是点击这个按钮就会post到ckeditorUpload地址进行处理，这里指向的是Struts 2的一个Action。当然，也可以用servlet或者ASP、PHP等来处理请求。

**3.**文件上传的控件相当于<input  type="file" name="**upload**" .../>，其name是”**upload**”，知道了name那么就可以在Action中获取这个文件。

private File upload;  //文件

private String uploadContentType;  //文件类型

private String uploadFileName;   //文件名

以上三个私有变量都要有set方法。如果不了解的话可以先学习一下Struts 2文件上传。

**4.**如果上传的图片格式不正确，可以在上传界面进行提示。



这个提示不是ckeditor提示的，要在Action中响应。

HttpServletResponse response =ServletActionContext.getResponse();

response.setCharacterEncoding("GBK");

PrintWriter out = response.getWriter();

if(???){

            out.print("<font color=\"red\"size=\"2\">\*文件格式不正确（必须为.jpg/.gif/.bmp/.png文件）</font>");

            return null;

}

**5.**

  InputStream is = newFileInputStream(upload);

  String uploadPath = ServletActionContext.getServletContext().getRealPath("/img/postImg");   //设置保存目录

  String fileName =java.util.UUID.randomUUID(); //采用UUID的方式随机命名

  fileName+= uploadFileName.substring(uploadFileName.length() - 4);

  File toFile = new File(uploadPath, fileName);

  OutputStream os = new FileOutputStream(toFile);

  byte[] buffer = new byte[1024];

  int length = 0;

  while ((length = is.read(buffer)) > 0) {

        os.write(buffer, 0, length);

  }

  is.close();

  os.close();

这段代码是Struts 2上传图片的核心代码，把图片上传后保存在项目的某个目录下，并随机重命名。

**6.**图片上传成功，在目录下也可以看到图片，至此图片上传成功。但是如何将图片发到编辑器中呢？

点“确定”按钮会有以下提示。



到这里，要在Action中加入一段JS。

String callback =ServletActionContext.getRequest().getParameter("CKEditorFuncNum");

out.println("<scripttype=\"text/javascript\">");

out.println("window.parent.CKEDITOR.tools.callFunction("+ callback + ",'" +"img/postImg/"+ fileName + "','')");

out.println("</script>");

有了这段代码，图片上传成功后，根据这里的

"img/postImg/" + filename

相对地址，就可以使用这个图片，直接转到“图像”页面。



附：Struts 2 Action代码

**[java]** [view plaincopy](http://blog.csdn.net/xiao__gui/article/details/7684505)

1. **public** **class** CkeditorUpload **extends** ActionSupport {
3. **private** File upload;
4. **private** String uploadContentType;
5. **private** String uploadFileName;

8. **public** File getUpload() {
9. **return** upload;
10. }
12. **public** **void** setUpload(File upload) {
14. **this**.upload = upload;
15. }
17. **public** String getUploadContentType() {
18. **return** uploadContentType;
19. }
21. **public** **void** setUploadContentType(String uploadContentType) {
22. **this**.uploadContentType = uploadContentType;
23. }
25. **public** String getUploadFileName() {
26. **return** uploadFileName;
27. }
29. **public** **void** setUploadFileName(String uploadFileName) {
30. **this**.uploadFileName = uploadFileName;   }
32. **public** String execute() **throws** Exception {
33. HttpServletResponse response = ServletActionContext.getResponse();
34. response.setCharacterEncoding("GBK");
35. PrintWriter out = response.getWriter();

38. //对文件进行校验
39. **if**(upload==**null** || uploadContentType==**null** || uploadFileName==**null**){
40. out.print("<font color=\"red\" size=\"2\">\*请选择上传文件</font>");
41. **return** **null**;
42. }
44. **if** ((uploadContentType.equals("image/pjpeg") || uploadContentType.equals("image/jpeg"))
45. && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".jpg")) {
46. //IE6上传jpg图片的headimageContentType是image/pjpeg，而IE9以及火狐上传的jpg图片是image/jpeg
47. }**else** **if**(uploadContentType.equals("image/png") && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".png")){
49. }**else** **if**(uploadContentType.equals("image/gif") && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".gif")){
51. }**else** **if**(uploadContentType.equals("image/bmp") && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".bmp")){
53. }**else**{
54. out.print("<font color=\"red\" size=\"2\">\*文件格式不正确（必须为.jpg/.gif/.bmp/.png文件）</font>");
55. **return** **null**;
56. }
58. **if**(upload.length() > 600\*1024){
59. out.print("<font color=\"red\" size=\"2\">\*文件大小不得大于600k</font>");
60. **return** **null**;
61. }
63. //将文件保存到项目目录下
64. InputStream is = **new** FileInputStream(upload);
65. String uploadPath = ServletActionContext.getServletContext()
66. .getRealPath("/img/postImg");   //设置保存目录
67. String fileName = java.util.UUID.randomUUID();  //采用UUID的方式随机命名
68. fileName += uploadFileName.substring(uploadFileName.length() - 4);
69. File toFile = **new** File(uploadPath, fileName);
70. OutputStream os = **new** FileOutputStream(toFile);
71. **byte**[] buffer = **new** **byte**[1024];
72. **int** length = 0;
73. **while** ((length = is.read(buffer)) > 0) {
74. os.write(buffer, 0, length);
75. }
76. is.close();
77. os.close();

80. //设置返回“图像”选项卡
81. String callback = ServletActionContext.getRequest().getParameter("CKEditorFuncNum");
82. out.println("<script type=\"text/javascript\">");
83. out.println("window.parent.CKEDITOR.tools.callFunction(" + callback + ",'" + "img/postImg/" + fileName + "','')");
84. out.println("</script>");

87. **return** **null**;
88. }
89. }

## tabletools

## tastytable

# angularjs

## angular处理json示例

<!DOCTYPE HTML>

<html ng-app="exampleApp">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<script src="scripts/libs/angular.js"></script>

<script>

var exampleApp = angular.module('exampleApp',[]);

exampleApp.controller('ShowController',['$scope', function($scope){

$scope.fromJson = function() {

var strUser = "{\"Data\":[{\"TotalNum\":\"1\",\"ParentDeptId\":\"11\",\"BookingEndTime\":\"111\"},{\"TotalNum\":\"2\",\"ParentDeptId\":\"22\",\"BookingEndTime\":\"222\"}],\"ErrorMessage\":\"\",\"Error\":\"0\"}";

var objUser = angular.fromJson(strUser);

alert(objUser.Error);

angular.forEach(objUser.Data, function(obj, key) {

alert(obj.TotalNum);

});

}

}]);

</script>

</head>

<body>

<div id="test">angular处理Json数据</div>

<div>

<div id="myCtrl" ng-controller="ShowController">

<input type="button" id="btnToJson" ng-click="fromJson()" value="objUser.firstName" />

</div>

</div>

</body>

</html>

## Angular.JS 与node.JS架构中基于token身份验证实现

# 1. 前面帖子[身份验证中Cookies与 Tokens比较](http://www.jdon.com/46054)已经说明令牌比cookie好，下面是AngularJS中的实现。 服务器后端 首先安装express-jwt: $ npm install express-jwt 配置express保护/api的每个调用： var expressJwt = require('express-jwt'); // We are going to protect /api routes with JWT app.use('/api', expressJwt({secret: secret})); app.use(express.json()); app.use(express.urlencoded()); Angular应用使用用户证书通过AJAX提交POST:

|  |
| --- |
| app.post('/authenticate', function (req, res) {  *//TODO validate req.body.username and req.body.password*  *//if is invalid, return 401*  **if** (!(req.body.username === 'john.doe' && req.body.password === 'foo[author]bar[/author]')) {  res.send(401, 'Wrong user or password');  **return**;  }   **var** profile = {  first\_name: 'John',  last\_name: 'Doe',  email: 'john@doe.com',  id: 123  };   *// We are sending the profile inside the token*  **var** token = jwt.sign(profile, secret, { expiresInMinutes: 60\*5 });   res.json({ token: token }); }); |

# 2. 下面是调用 /api/restricted获得一个资源，其证书检查由expressJwt中间件执行。

|  |
| --- |
| app.get('/api/restricted', function (req, res) {  console.log('user ' + req.user.email + ' is calling /api/restricted');  res.json({  name: 'foo'  }); }); |

# 3. 前端Angular.js 客户端首先用AngularJS 获得JWT的令牌，为了这样做我们还需要用户证书，我们将将创建一个表单让用户输入他们的用户名和密码。

|  |
| --- |
| <div ng-controller="UserCtrl">  <span></span>  <form ng-submit="submit()">  <input ng-model="user.username" type="text" name="user" placeholder="Username" />  <input ng-model="user.password" type="password" name="pass" placeholder="Password" />  <input type="submit" value="Login" />  </form> </div> |

# 4. 一个控制器处理提交的action:

|  |
| --- |
| myApp.controller('UserCtrl', function ($scope, $http, $window) {  $scope.user = {username: 'john.doe', password: 'foo[author]bar[/author]'};  $scope.message = '';  $scope.submit = function () {  $http  .post('/authenticate', $scope.user)  .success(function (data, status, headers, config) {  $window.sessionStorage.token = data.token;  $scope.message = 'Welcome';  })  .error(function (data, status, headers, config) {  *// Erase the token if the user fails to log in*  delete $window.sessionStorage.token;   *// Handle login errors here*  $scope.message = 'Error: Invalid user or password';  });  }; }); |

# 5. 现在我们已经有JWT存储在sessionStorage中，如果这个令牌被设置，我们可以为每个发出请求设置Authorization 头部，只要用$http方式即可，头部值部分使用Bearer<token>

|  |
| --- |
| myApp.factory('authInterceptor', function ($rootScope, $q, $window) {  **return** {  request: function (config) {  config.headers = config.headers || {};  **if** ($window.sessionStorage.token) {  config.headers.Authorization = 'Bearer ' + $window.sessionStorage.token;  }  **return** config;  },  response: function (response) {  **if** (response.status === 401) {  *// handle the case where the user is not authenticated*  }  **return** response || $q.when(response);  }  }; });  myApp.config(function ($httpProvider) {  $httpProvider.interceptors.push('authInterceptor'); }); |

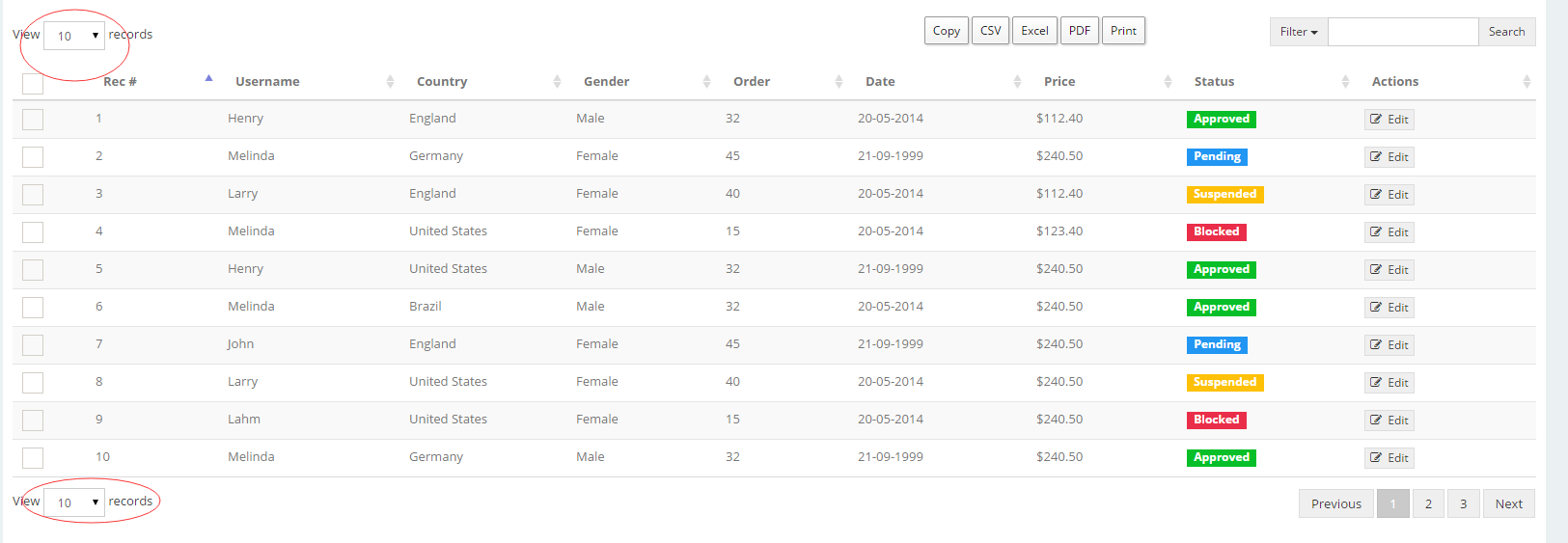
# 6. 尽管sessionStorage并不是所有浏览器支持，你可以使用 polyfill替代cookie保存在localStorage，这个数据能够持久到浏览器标签页关闭。 最后，我们能够向一个授权资源发出请求：

|  |
| --- |
| $http({url: '/api/restricted', method: 'GET'}) .success(function (data, status, headers, config) {  console.log(data.name); *// Should log 'foo'* }); |

# 7. 服务器后端控制台会显示： user foo@[bar](http://www.jdon.com/blog/bar).com is calling /api/restricted

jquery.dataTables.js文件

\_fnFeatureHtmlLength 方法



# jquery

## jQuery 语法

jQuery 语法是为 HTML 元素的选取编制的，可以对元素执行某些操作。

基础语法是：$(selector).action()

美元符号定义 jQuery

选择符（selector）“查询”和“查找” HTML 元素

jQuery 的 action() 执行对元素的操作

示例

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有段落

$(".test").hide() - 隐藏所有 class="test" 的所有元素（注意selector中不能有空格）

$("#test").hide() - 隐藏所有 id="test" 的元素

提示：jQuery 使用的语法是 XPath 与 CSS 选择器语法的组合。

**文档就绪函数：**

您也许已经注意到在我们的实例中的所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){

--- jQuery functions go here ----

});

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

如果在文档没有完全加载之前就运行函数，操作可能失败。

## [jQuery取得select选中的值](http://www.cnblogs.com/greatverve/archive/2010/02/03/1662565.html)

获取选中的值：$("#select1").val();

获取选中的文本：$("#select1  option:selected").text();

## jQuery更改超链接的href

$(document).ready(function(){

$(".js-btn").click(function(){

var hospitalId = $(".js-hospital").val();

if(hospitalId == "" ){

alert("请选择医院");

return false;

}

this.href = this.href + "?hospitalId=" + hospitalId;

});

});

提交超链接是HTML本身就会自己完成的事情，JavaScript只是在提交之前更改一些属性的值，如href的值。另外，JavaScript也可以通过return false 来中断提交。

## jquery获取li列表中当前点击的元素下的各种值

**列表的HTML代码如下：**

<ul>

<li>

<a href="#1" class="case-info J\_CaseInfo js-meeting" >

<div class="expert">

<div class="date J\_Date">

<p>2015-01-26</p>

<p>星期一</p>

<input type="hidden" name="deptId" value="2001">

<input type="hidden" name="doctorId" value="5001">

</div>

<i class="icon pm">aa</i>

</div>

</a>

</li>

<li>

<a href="#2" class="case-info J\_CaseInfo js-meeting" deptid="aaa">

<div class="expert">

<div class="date J\_Date">

<p>2015-01-27</p>

<p>星期二</p>

<input type="hidden" name="deptId" value="2002">

<input type="hidden" name="doctorId" value="5002">

</div>

<i class="icon pm">bb</i>

</div>

</a>

</li>

</ul>

**jquery获取各个元素的代码示例如下：**

<script type="text/javascript">

$(document).ready(function(){

$(".js-meeting").click(function(){

alert($(this).attr('href')); //#1 (获取某个属性的值)

alert(this.href); //http://localhost:8080/opc/hospital/hospitalschedule?hospitalId=2000001#1

alert($(this).html()); //返回html文档： <div class="expert">…… </div>

alert($(this).find('div i').text()); //aa

alert($(this).find('div div p').text()); //2015-01-26星期一

alert($(this).find('div div p').eq(0).text()); //2015-01-26

alert($(this).find('div div p').eq(1).text()); //星期一

alert($(this).find("input[name='doctorId']").val()); //5001

});

});

</script>



## jquery与动态弹出层

## jquery与Json

以下例子演示jquery如何处理服务端返回的Json数据：

$(document).ready(function(){

$(".js-meeting").click(function(){

var hisPlanId = $(this).find("div div input").val();

var url = "${urlPath}" + "/opc/resource/getResourceByPlanId?hospitalId=${hospitalId}&hisPlanId=" + hisPlanId;

$.get(url, function(data, status){

//接收到的数据转化为JQuery对象，由JQuery为我们处理

var jsonData = eval(data);

if(jsonData.success == true){

$("tbody").html("<tr><th width=\"40%\">时间段</th><th width=\"20%\">就诊序号</th><th width=\"20%\">剩余号源</th><th width=\"20%\">操作</th></tr>");

//遍历对象数组，index是数组的索引号，objVal是遍历的一个对象。

$.each(jsonData.obj, function(index, objVal) {

//在table上追加一行

$("tbody").append("<tr><th height=\"40px\">"+objVal.startTime+"</th><th>"+objVal.orderNum+"</th><th>"+objVal.leaveNums+"</th><th><b>"+state+"</b></th></tr>");

});

}

});

});

});

public class Json {

private boolean success = false;

private String msg = "";

private Object obj = null; //实体列表，如 List<ResourceDto>

public boolean isSuccess() {

return success;

}

public void setSuccess(boolean success) {

this.success = success;

}

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

public Object getObj() {

return obj;

}

public void setObj(Object obj) {

this.obj = obj;

}

}

服务端的接口代码如下：

@RequestMapping("/getResourceByPlanId")

@ResponseBody

@Description("按条件查询排班")

public Json getResourceByPlanId(Long hospitalId,String hisPlanId){

Json j = new Json();

j.setSuccess(false);

try{

List<ResourceDto> resourceDtoList = infoService.getPlanResource(0, hospitalId, hisPlanId);

j.setSuccess(true);

j.setObj(resourceDtoList);

return j;

}

catch(Exception e){

e.printStackTrace();

}

return j;

}

## $.ajax()方法详解

虽然get()和post()函数非常简洁易用，但是对于更复杂的一些设计需求还是无法实现，比如在ajax发送的不同时段做出不同的动作等。jQuery提供一个更为具体的函数：ajax()。

jquery中的ajax方法参数总是记不住，这里记录一下。

**1.url**:   
要求为String类型的参数，（默认为当前页地址）发送请求的地址。

**2.type**:   
要求为String类型的参数，请求方式（post或get）默认为get。注意其他http请求方法，例如put和delete也可以使用，但仅部分浏览器支持。

**3.timeout**:   
要求为Number类型的参数，设置请求超时时间（毫秒）。此设置将覆盖$.ajaxSetup()方法的全局设置。

**4.async**:   
要求为Boolean类型的参数，默认设置为true，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为false。注意，同步请求将锁住浏览器，用户其他操作必须等待请求完成才可以执行。

**5.cache**:   
要求为Boolean类型的参数，默认为true（当dataType为script时，默认为false），设置为false将不会从浏览器缓存中加载请求信息。

**6.data**:   
要求为Object或String类型的参数，发送到服务器的数据。如果已经不是字符串，将自动转换为字符串格式。get请求中将附加在url后。防止这种自动转换，可以查看　　processData选项。对象必须为key/value格式，例如{foo1:"bar1",foo2:"bar2"}转换为&foo1=bar1&foo2=bar2。如果是数组，JQuery将自动为不同值对应同一个名称。例如{foo:["bar1","bar2"]}转换为&foo=bar1&foo=bar2。

**7.dataType**:   
要求为String类型的参数，预期服务器返回的数据类型。如果不指定，JQuery将自动根据http包mime信息返回responseXML或responseText，并作为回调函数参数传递。可用的类型如下：  
xml：返回XML文档，可用JQuery处理。  
html：返回纯文本HTML信息；包含的script标签会在插入DOM时执行。  
script：返回纯文本JavaScript代码。不会自动缓存结果。除非设置了cache参数。注意在远程请求时（不在同一个域下），所有post请求都将转为get请求。  
json：返回JSON数据。  
jsonp：JSONP格式。使用SONP形式调用函数时，例如myurl?callback=?，JQuery将自动替换后一个“?”为正确的函数名，以执行回调函数。  
text：返回纯文本字符串。

**8.beforeSend**：  
要求为Function类型的参数，发送请求前可以修改XMLHttpRequest对象的函数，例如添加自定义HTTP头。在beforeSend中如果返回false可以取消本次ajax请求。XMLHttpRequest对象是惟一的参数。  
            function(XMLHttpRequest){  
               this;   //调用本次ajax请求时传递的options参数  
            }  
**9.complete**：  
要求为Function类型的参数，请求完成后调用的回调函数（请求成功或失败时均调用）。参数：XMLHttpRequest对象和一个描述成功请求类型的字符串。  
          function(XMLHttpRequest, textStatus){  
             this;    //调用本次ajax请求时传递的options参数  
          }

**10.success**：要求为Function类型的参数，请求成功后调用的回调函数，有两个参数。  
         (1)由服务器返回，并根据dataType参数进行处理后的数据。  
         (2)描述状态的字符串。  
         function(data, textStatus){  
            //data可能是xmlDoc、jsonObj、html、text等等  
            this;  //调用本次ajax请求时传递的options参数  
         }

**11.error**:  
要求为Function类型的参数，请求失败时被调用的函数。该函数有3个参数，即XMLHttpRequest对象、错误信息、捕获的错误对象(可选)。ajax事件函数如下：  
       function(XMLHttpRequest, textStatus, errorThrown){  
          //通常情况下textStatus和errorThrown只有其中一个包含信息  
          this;   //调用本次ajax请求时传递的options参数  
       }

**12.contentType**：  
要求为String类型的参数，当发送信息至服务器时，内容编码类型默认为"application/x-www-form-urlencoded"。该默认值适合大多数应用场合。

**13.dataFilter**：  
要求为Function类型的参数，给Ajax返回的原始数据进行预处理的函数。提供data和type两个参数。data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的dataType参数。函数返回的值将由jQuery进一步处理。  
            function(data, type){  
                //返回处理后的数据  
                return data;  
            }

**14.dataFilter**：  
要求为Function类型的参数，给Ajax返回的原始数据进行预处理的函数。提供data和type两个参数。data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的dataType参数。函数返回的值将由jQuery进一步处理。  
            function(data, type){  
                //返回处理后的数据  
                return data;  
            }

**15.global**：  
要求为Boolean类型的参数，默认为true。表示是否触发全局ajax事件。设置为false将不会触发全局ajax事件，ajaxStart或ajaxStop可用于控制各种ajax事件。

**16.ifModified**：  
要求为Boolean类型的参数，默认为false。仅在服务器数据改变时获取新数据。服务器数据改变判断的依据是Last-Modified头信息。默认值是false，即忽略头信息。

**17.jsonp**：  
要求为String类型的参数，在一个jsonp请求中重写回调函数的名字。该值用来替代在"callback=?"这种GET或POST请求中URL参数里的"callback"部分，例如{jsonp:'onJsonPLoad'}会导致将"onJsonPLoad=?"传给服务器。

**18.username**：  
要求为String类型的参数，用于响应HTTP访问认证请求的用户名。

**19.password**：  
要求为String类型的参数，用于响应HTTP访问认证请求的密码。

**20.processData**：  
要求为Boolean类型的参数，默认为true。默认情况下，发送的数据将被转换为对象（从技术角度来讲并非字符串）以配合默认内容类型"application/x-www-form-urlencoded"。如果要发送DOM树信息或者其他不希望转换的信息，请设置为false。

**21.scriptCharset**：  
要求为String类型的参数，只有当请求时dataType为"jsonp"或者"script"，并且type是GET时才会用于强制修改字符集(charset)。通常在本地和远程的内容编码不同时使用。

案例代码：

[复制代码](javascript:void(0);)

$(function(){

$('#send').click(function(){

$.ajax({

type: "GET",

url: "test.json",

data: {username:$("#username").val(), content:$("#content").val()},

dataType: "json",

success: function(data){

$('#resText').empty(); //清空resText里面的所有内容

var html = '';

$.each(data, function(commentIndex, comment){

html += '<div class="comment"><h6>' + comment['username']

+ ':</h6><p class="para"' + comment['content']

+ '</p></div>';

});

$('#resText').html(html);

}

});

});

});

[复制代码](javascript:void(0);)

**22.顺便说一下$.each()函数**:  
$.each()函数不同于JQuery对象的each()方法，它是一个全局函数，不操作JQuery对象，而是以一个数组或者对象作为第1个参数，以一个回调函数作为第2个参数。回调函数拥有两个参数：第1个为对象的成员或数组的索引，第2个为对应变量或内容。

另外可以参考：http://www.w3school.com.cn/jquery/ajax\_ajax.asp

# note.js

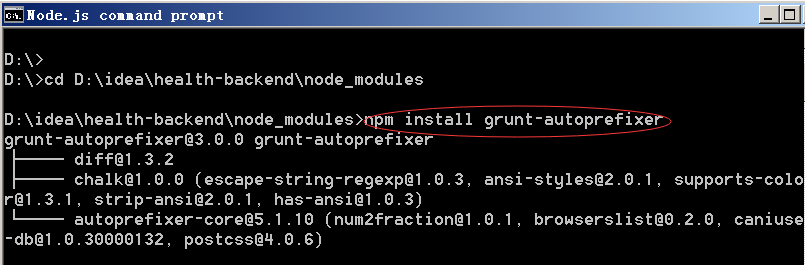
## 安装node.js模块

1、运行node.js的命令行工具

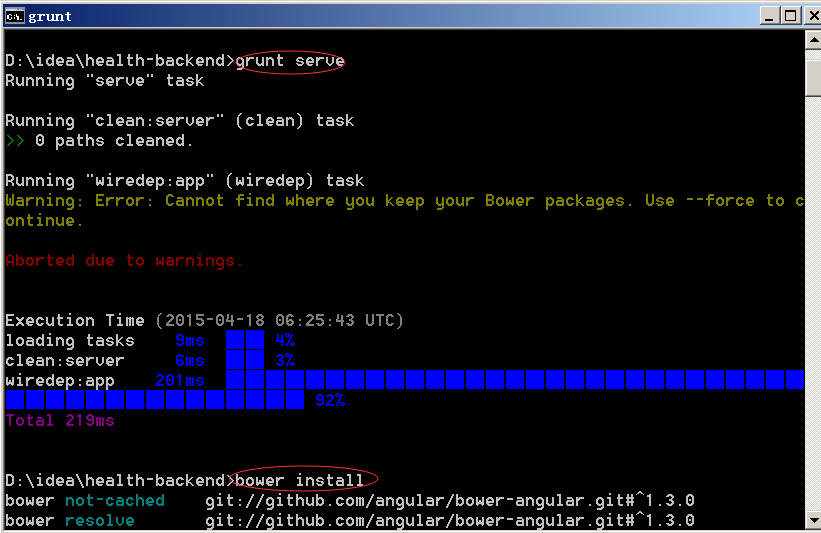
2、进入到需要安装模块的文件夹下

3、执行命令 “npm install 模块名称”进行安装

具体操作如下图：



## 安装bower



如果执行bower install 过程中，提示与git相关的错误，可能是系统环境变量 path 中没有添加git。在Path变量中加入：C:\Program Files (x86)\Git\bin;C:\Program Files (x86)\Git\cmd