July31 AJAX

1 什么是AJAX

Asynchronous Javascript And Xml

异步的 js Xml

异步的向服务器发送请求并接受响应数据

响应回来的数据格式：早期是xml，现在使用JSON

1.1 同步访问

当客户端向服务器发送请求时，服务器在处理的过程中，浏览器只能等待

（效率偏低）

1.2 异步访问

当客户端向服务器发送请求时，服务器在处理操作的同时，客户端可以做其他的操作，不要一直等待（效率高）

1.3 使用场合

1.搜索建议

2.表单验证

3.前后端完全分离时

2 AJAX的核心对象

异步对象-XMLHttpRequest

主要称为异步对象，代替浏览器向服务器异步的发送请求并接受响应

该对象由JavaScript创建

2.1 创建异步对象

主流的异步对象是XMLHttpRequest类型的，并且主流浏览器（IE7+，Chrome，Firefox，Safari，Opera）全部支持XMLHTTPRequest。但是在IE的低版本中（IE6以及以下），不支持XMLHttpRequest，需要使用ActiveXObject来创建异步对象

判断浏览器是否支持XMLHttpRequest

if(window.XMLHttpRequest){

支持XMLHttpRequest

} else {

不支持XMLHttpRequest，需要使用ActiveXObject创建异步对象

var xhr = new ActiveXObject(“Microsoft.XMLHttp”);

}

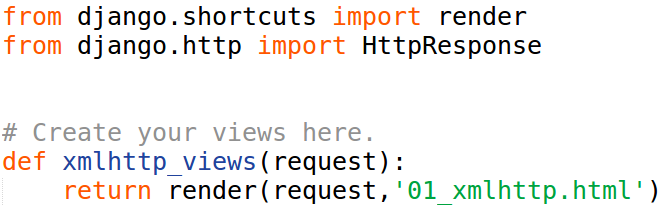
练习：

1.定义访问路径/01\_xmlhttp/,通过视图，显示01\_xmlhttp.html模板

2.在01\_xmlhttp.html模板中，创建一个按钮

3.点击按钮时，创建一个异步对象

建议：创建一个函数，根据浏览器对异步对象的支持情况，返回不同的异步对象



3 xhr的成员

3.1 方法open

创建请求

语法：

open(method,url,asyn)

method:异步请求的请求方式，取值是’get’或’post’

url:请求地址，String类型

asyn:指定使用同步还是异步的方式发送请求,true表示异步,false表示同步

3.2 属性-readyState

请求状态，通过不同的请求状态来表示xhr与服务器的交互情况

由0-4共5个值来表示5个不同的状态

|  |  |
| --- | --- |
| 0 | 请求尚未初始化 |
| 1 | 已经与服务器建立连接 |
| 2 | 服务器端的请求已接收 |
| 3 | 请求处理中 |
| 4 | 响应已完成 |

3.3 属性-status

服务器端的响应状态，用于表示服务器的状态

|  |  |
| --- | --- |
| 200 | 表示服务器正确处理所有的请求以及给出响应 |
| 404 | 请求资源未找到 |
| 403 | 无权限 |
| 500 | 服务器内部错误 |

3.4 属性-responseText

服务器端相应回来的数据（只能是string类型）

3.5 事件-onreadystatechange

每当readyState发生改变时要触发的操作——回调函数

只有当readyState的值为4的时候并且，status为200的时候，才可以正常的接收responseText

3.6 方法-send()

发送请求

语法：

send（body）

body：请求主体（请求体）

get请求的话，此处为null

post请求的话，此处为要提交的数据

4 AJAX的操作步骤

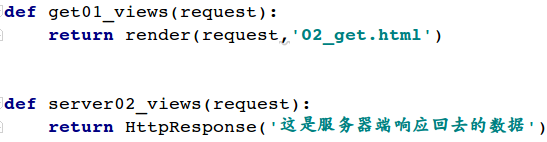
1.创建xhr对象

2.创建请求-open（）

3.设置回调函数-onreadystatechange

4.发送请求-send（）





July31 AJAX

1 AJAX发送get请求

1.1 发送异步请求的步骤

1.创建xhr对象

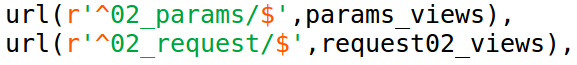
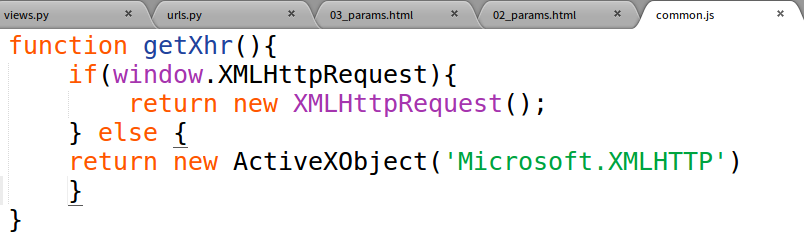
2.创建请求open()方法的调用

3.设置回调函数-onreadystatechange

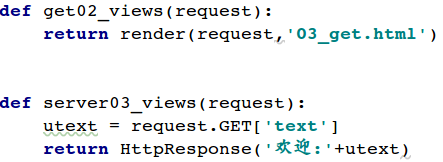
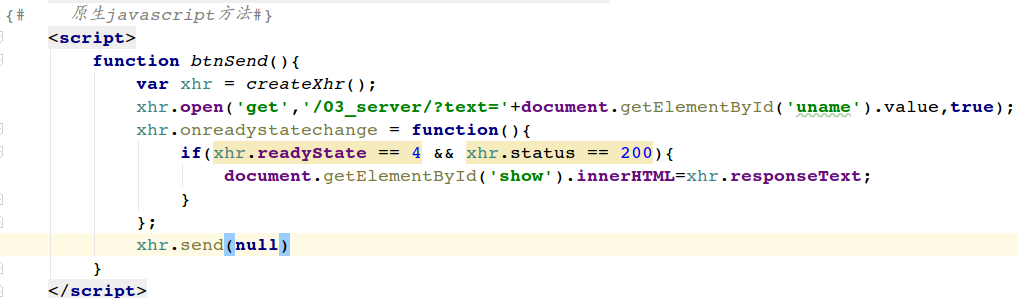
4.发送请求-send()

1.2 请求参数

推荐：地址栏传参http://localhost:8000/test/?name=value&name=value



==============================动态===========================



2 AJAX发送post请求

提交的参数要在send()中编写

var xhr = getXhr();

… …

… …

xhr.send(‘name1=value1&name2=value2’);

2.1 csrf验证

必须手动提交csrfmiddlewaretoken的值到服务器，否则无法通过csrf验证，服务器直接返回403

解决方案一：

1.在服务器端从COOKIES中获取csrftoken的值并发送到模板中

2.在模板中获取csrftoken的值，拼接成提交参数再发送给服务端

弊端：

由于要先走服务器，有可能在客户端中并没有csrftoken cookie，所以有可能报错

解决方案二：

1.在模板（前端页面）中，通过JS获取cookies中csrftoken的值

2.再构建成参数提交给服务器

弊端：

要求客户端cookies中必须包含csrftoken，否则获取不到对应的值就无法提交给服务器，结果依然是403。可以在网页中任意添加一个form，并且在form中只增加一个{% csrf\_token %}

解决方案三：

1.在模板中（前端页面），通过JS获取

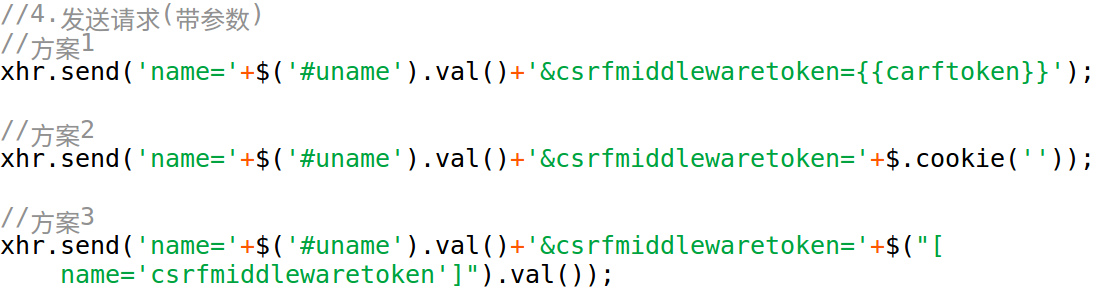
<input type=’hidden’ name=’csrfmiddlewaretoken’>的值

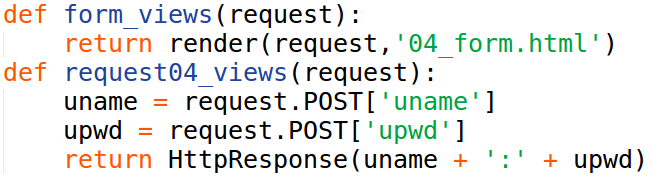
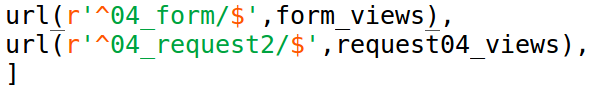
2.将取出来的值拼成参数发送给服务器

弊端：

要求在网页中必须存在<input type=’hidden’ name=’csrfmiddlewaretoken’>标记。可以在网页中任意添加一个form，并且在form中只增加一个{% csrf\_token %}

Content-Length:application/x-www-form-urlencoded







2.3 设置一个请求消息头

Content-Type

Content-Type:application/x-www-form-urlencoded

AJAX默认将此消息头设置为text/plain，导致服务器无法正常接收请求数据。

所以，必须将此消息头更新回原来的默认值

语法：

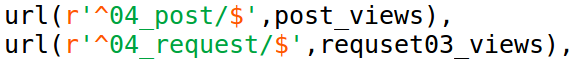
xhr.setRequestHeader(“Content-Type”,”application/x-www-form-urlencoded”);

注意：

该操作，要在xhr.send()之前执行

字符串的拼接





练习：

1.当光标从文本框中移除的时候，验证用户名在数据库中是否存在

2.用户名存在，在页面显示通过

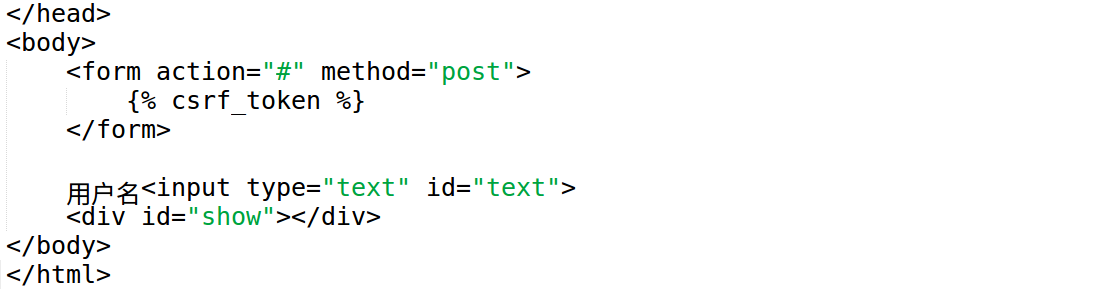
创建数据库ajax

创建models-Users(uname,upwd,uage,uemail)

通过后台管理增加数据到数据库





3 JSON介绍

JavaScript Object Notation

JS对象的表现形式

4 JSON表现

4.1 JSON表示单个对象

1.使用{}表示单个对象

2.在{}中使用key:value的形式来表示属性（数据）

3.key必须要使用双引号括起来

4.value如果也是字符串，必须使用双引号括起来

var obj = {

‘name’:’范冰冰’,

‘age’:26,

‘gender’:’女’

}

4.2 JSON表示数组

1.使用中括号[]表示一个数组

2.数组中允许包含若干的JSON单个对象或字符串

ex：

1.使用JSON来表示多个人的姓名

var arr=[“范冰冰”,”李晨”,”洪金宝”];

2.使用JSON表示多个人的信息（name，age，gender）

var arr = [

{

‘name’:’范冰冰’,

‘age’:26,

‘gender’:’女’

},

{

‘name’:’李晨’,

‘age’:43,

‘gender’:’男’

},

];

jQuery的each函数

作用：快速且遍历的循环js数组

语法：

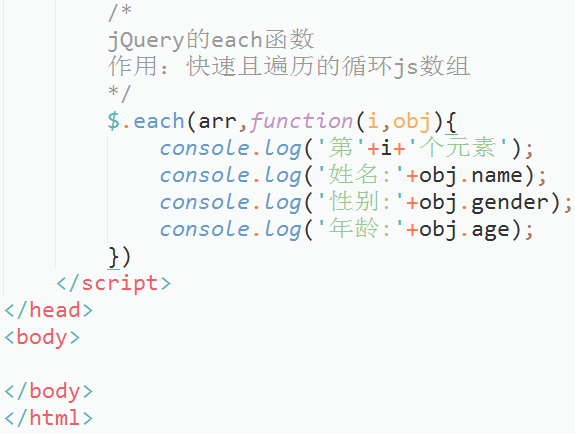
$.each(array,function(index,obj){

index：表示当前遍历的元素的下标

obj：表示当前遍历出来的元素

});





5 后台处理JSON

前后台JSON数据交互流程（后端->前端）

1.后台先获取数据

类型为：

1.元组

2.列表

3.字典

4.QuerySet

2.在后台将数据转换为符合JSON格式的字符串

3.将JSON格式的字符串相应给前端

return HttpResponse（JSON格式字符串）

4.在前端将响应回来的JSON串解析成JSON的对象

6 Python（Django）的JSON处理

6.1 元组、列表、字典

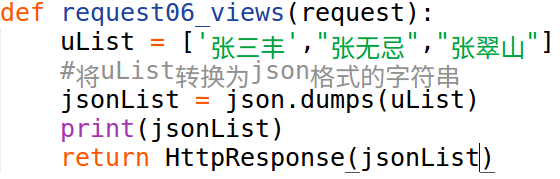
使用python中的JSON类就可以完成转换

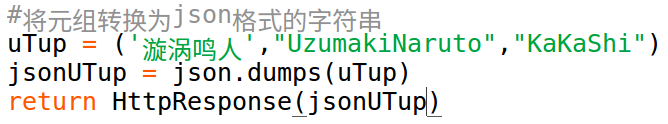
语法：

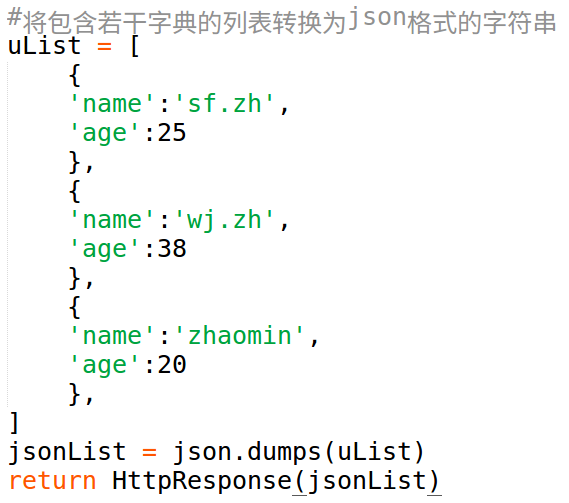
import json

jsonStr = json.dumps（元组|列表|字典）

return HttpResopnse(jsonStr)







6.2 Django中的查询结果集

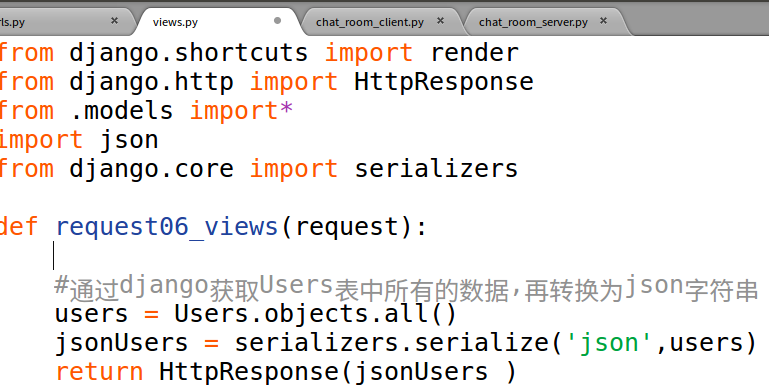
使用Django提供的序列换类，来完成QuerySet到JSON字符串的转换

from django.core import serializers

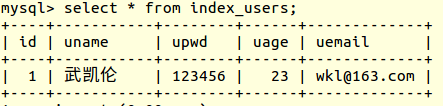
str = serializers.serialize(‘json’,QuertSet)

return HttpResponse(str)

数据库中查询并且转换为json字符串







id就是pk主键primary key

6.3 Django中的查询结果集

使用Entry.objects.get()得到的单个对象是不可序列化的，也不能用json.dumps()去转换

方案1

使用Entry.objects.filter()替换.Entry.objects.get(),替换后可以正常使用序列化

方案2

将Entry.objects.get()得到的对象转换为字典然后再使用json.dumps()

class Users(models.Model):

… …

def to\_dict(self):

dic = {

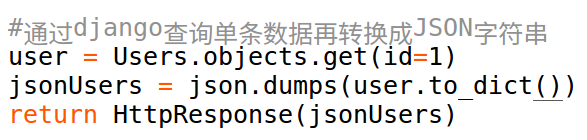
‘name’:self.name,

‘age’:self.age,

}

return dic





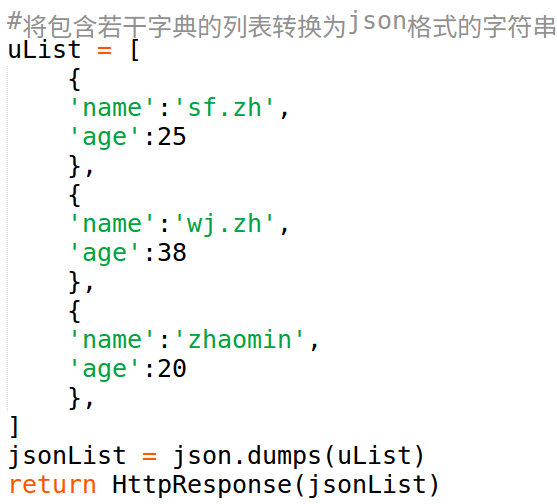
7 前端处理JSON

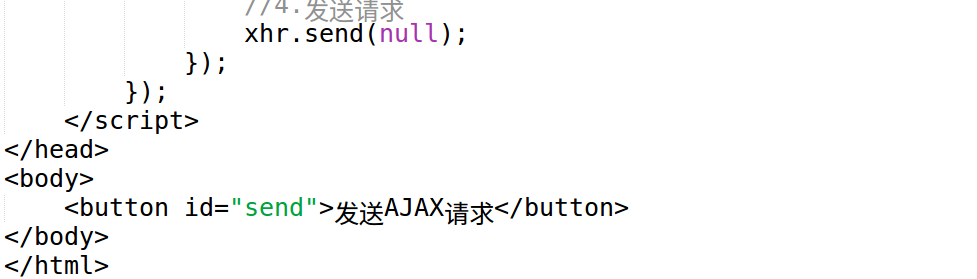
将后端响应回来的JSON字符串转换为JSON对象即可

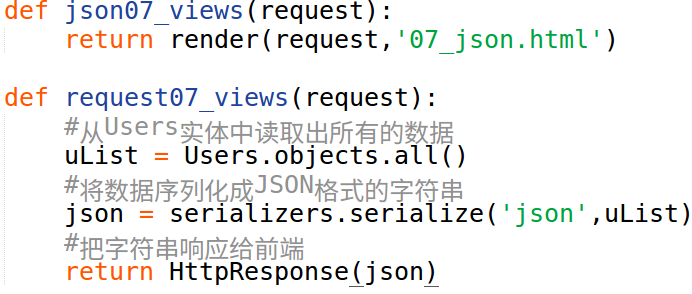
语法：

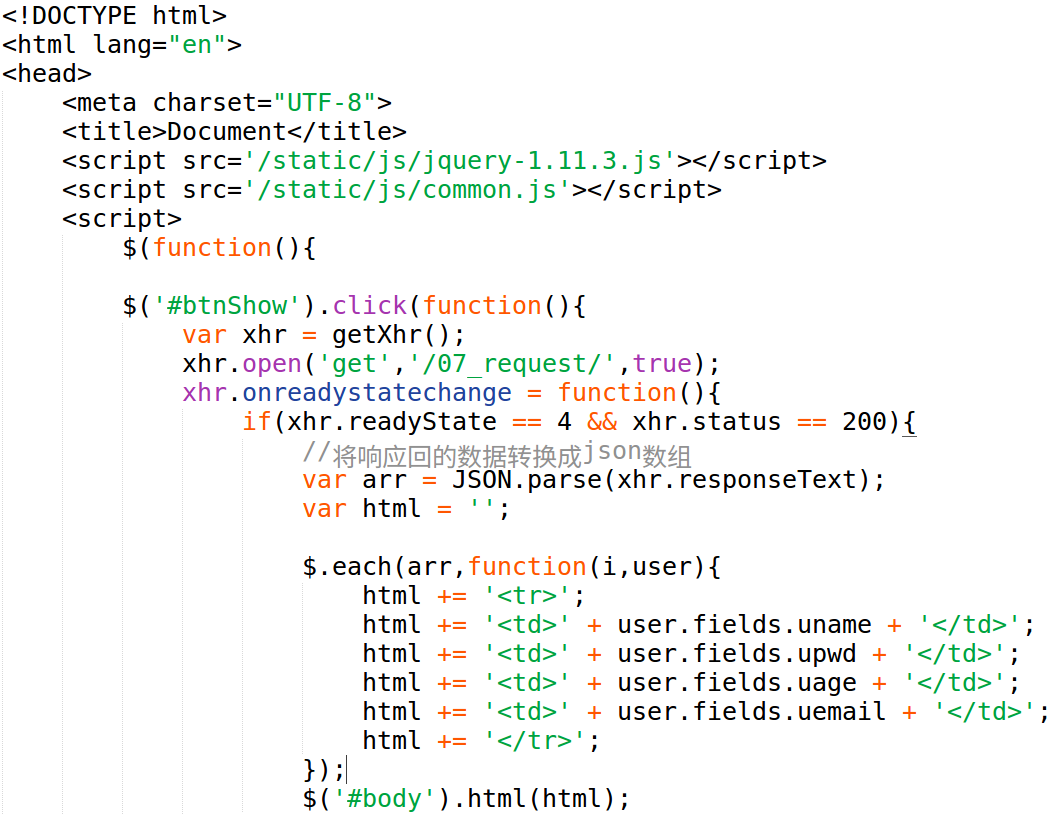
在JS中

JSON对象 = JSON.parse(JSON字符串);

reuqest06\_views







8 JQuery中的AJAX

在jQuery中提供了对原生AJAX的封装

8.2 $obj.load()

作用：

异步加载数据到$obj元素中

语法：

$obj.load(url,data,callback);

url：异步请求的地址

data：传递给请求地址的参数（可选）

可以传递普通的字符串’name=sf.zhang&age=85’

可以传递JSON

{

‘name’:’sf.zhang’,

‘age’:85

}

callback：异步请求完成后要执行的操作（可选）

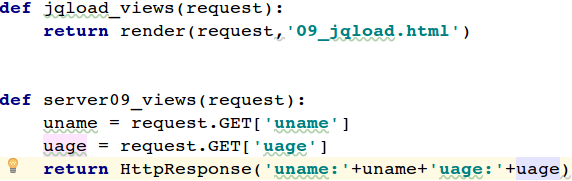
function(resText,statusText){

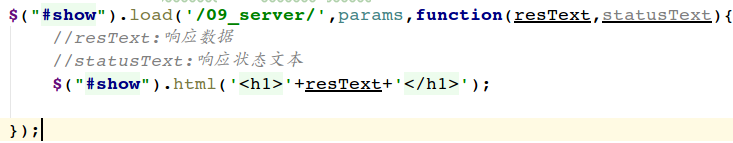
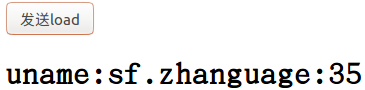
resText:响应数据

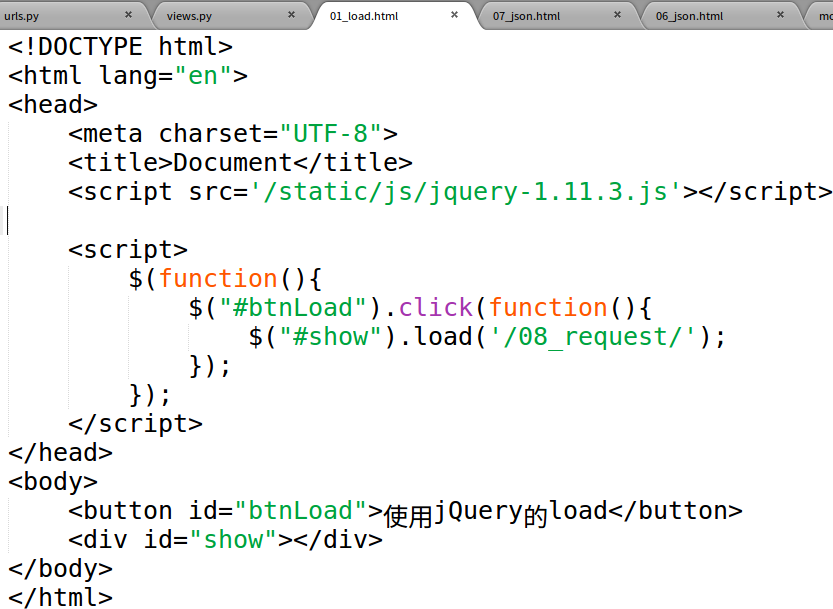
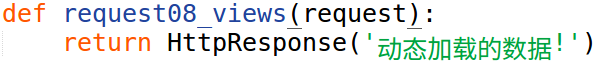
statusText:响应状态文本(原因短句)

}

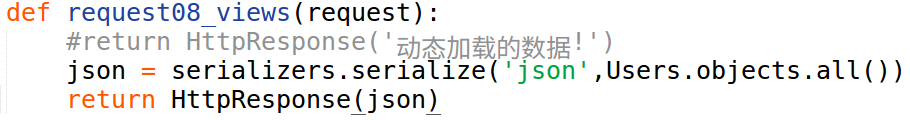








==================================================================





8.2 $.get()

作用：使用get方式异步向服务器发送请求

语法：

$.get(url,data,callback,type)

url:待载入页面的URL地址

data:待发送 Key/value 参数。

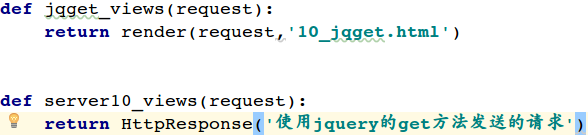
callback:载入成功时回调函数。

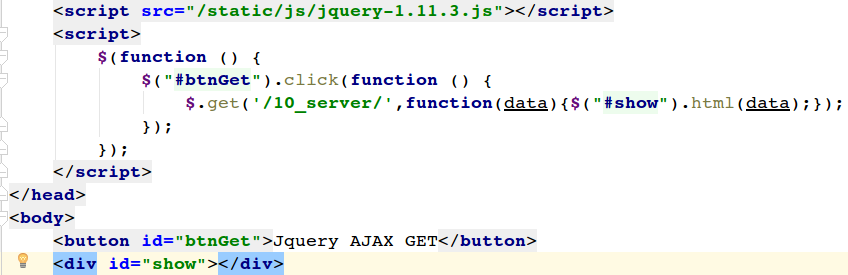
function(data){

data:表示的是响应回来的数据

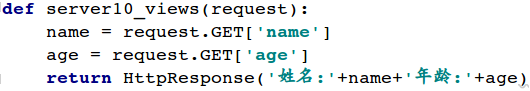
}

type:返回内容格式，xml, html, script, json, text, \_default。

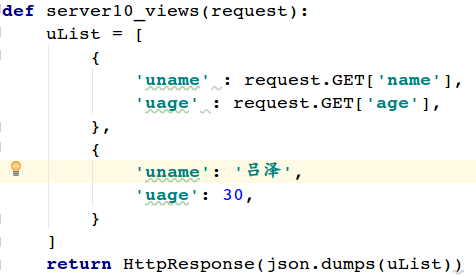




=======================使用第二个参数data==================



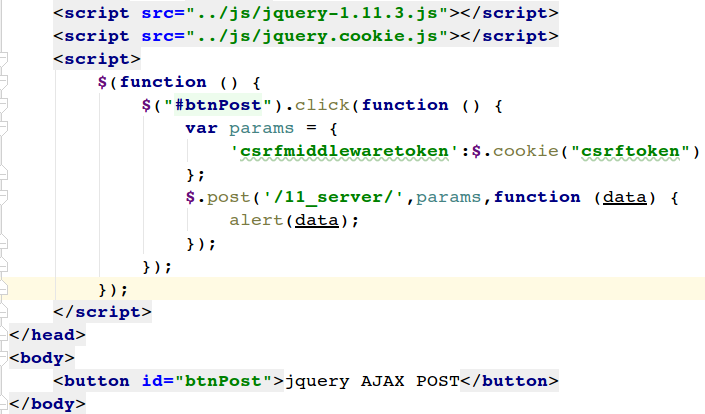
=======================使用第四个参数type==================



8.3 $.post()

作用：使用post方式异步的向服务器发送请求





8.4 $.ajax()

作用：所有的操作均可自己定制

语法：$.ajax({传递参数的JSON对象});

JSON对象：

1.url：字符串，要异步请求的地址

2.type：字符串，提交方式，get或post

3.data：JSON对象或字符串，传递给后台服务器的参数，拼字符串

4.dataType：字符串，响应回来数据的类型：

“html”:响应回来的数据是HTML文本

“text”:响应回来的数据是普通文本

“script”:响应回来的数据是js脚本

“json”:响应回来的数据是JSON对象

“xml”:响应回来的数据是xml文档

“jsonp”:JSONP格式，跨域时使用

5.success：回调函数，请求和响应成功时才回来执行的操作

function(data,statusText,jqXHR){

data:响应回来的数据

textStatus：响应状态文本

statusText:状态文本（可选）

jqXHR:jq中的XMLHttpRequest对象

}

6.error：回调函数，请求或响应产生错误是的回调函数

error:function(xhr,textStatus){

}

7.async：指定是否为异步方式

true：使用异步方式

false：使用同步方式

默认为true

========================模糊查询===========================