# 第十二章 Django ajax请求

本节所讲内容：

12.1 ajax介绍与编写

12.2 ajax提交文件对象

## 12.1 ajax介绍

Ajax（Asynchronous Javascript And XML：异步 JavaScript 和 XML），是指一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

使用Ajax的最大优点，就是能在不更新整个页面的前提下维护数据。这使得Web应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变的信息。

但是Ajax可能破坏浏览器的后退与加入收藏书签功能。在动态更新页面的情况下，用户无法回到前一个页面状态，这是因为浏览器仅能记下历史记录中的[静态页面](https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E9%A1%B5%E9%9D%A2)。

12.1.1 Ajax的定义

Ajax有最初是由js编写的，后来随着js框架的发展，jq，vue等大型框架都对ajax进行了封装，在这里我们主要学习jquery中的ajax请求。注意：我们就学习工作中100%会到的。

跟随上节课的ReqForm/models.py中的模型来进行操作，创建如下ajaxtest.html方便对ajax操作。如下所示：

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <script type="text/javascript" src="/static/js/jquery.js"></script>  
</head>  
<body>  
*{#{% csrf\_token %}#}*姓名:<input type="text" id = 'username'><br>  
密码:<input type="password" id = 'password'><br>  
邮箱:<input type="email" id = 'email'><br>  
电话:<input type="text" id = 'phone'><br>  
<button id = 'tj'>点击我</button>  
</body>  
</html>

注意：由于我们直接使用的jquery中的ajax 不要忘记在HTML中导入相关的js

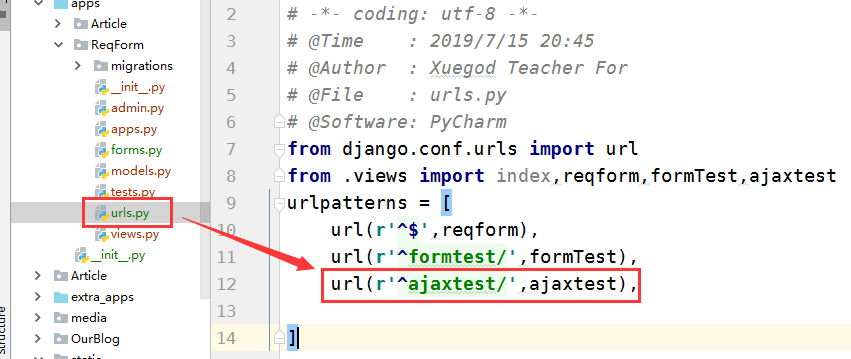
<script type="text/javascript" src="/static/js/jquery.js"></script>

在ReqForm/views.py中编写函数指定该模板

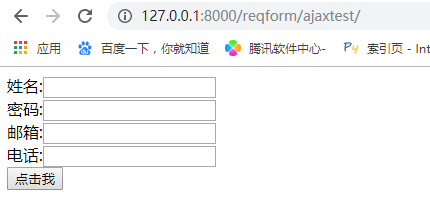
def ajaxtest(request):  
 return render(request,'ajaxtest.html')

在ReqForm/urls.py中配置路由

urlpatterns = [  
 url(r'^$',reqform),  
 url(r'^formtest/',formTest),  
 url(r'^ajaxtest/',ajaxtest),  
]



现在开启项目访问界面，结果如下所示：



12.1.2 Ajax基本语法

1. 语法格式：

jQuery.ajax( url [, settings ] )

url ：类型: [String](http://www.shouce.ren/api/view/a/13081#articleHeader0) ，一个用来包含发送请求的URL字符串。

settings ：类型: [PlainObject](http://www.shouce.ren/api/view/a/13081#articleHeader25) 一个以"{键:值}"组成的AJAX 请求设置。所有选项都是可选的。可以使用[$.ajaxSetup()](http://www.shouce.ren/api/view/a/12874)设置任何默认参数。看jQuery.ajax( settings )下所有设置的完整列表。

2. 常用settings配置介绍

****method**** :(default: 'GET') Type: [String](http://www.shouce.ren/api/view/a/13081#articleHeader0) HTTP 请求方法 (比如："POST", "GET ", "PUT")。(添加版本: 1.9.0)。（手册网注：如果你使用jQuery 1.9.0 之前的版本，你需要使用type选项。）

****success:**** Type: [Function](http://www.shouce.ren/api/view/a/13081#articleHeader28)( Anything data, String textStatus, jqXHR jqXHR )

请求成功后的回调函数。这个函数传递3个参数：从服务器返回的数据，并根据dataType参数进行处理后的数据或dataFilter回调函数，如果指定的话；一个描述状态的字符串;还有 jqXHR对象 。

****type**** (默认: 'GET') :类型: [String](http://www.shouce.ren/api/view/a/13081#articleHeader0)

method选项的别名。如果你使用jQuery 1.9.0 之前的版本，你需要使用type选项。

****url**** (默认: 当前页面地址) 类型: [String](http://www.shouce.ren/api/view/a/13081#articleHeader0) 发送请求的地址。

****async**** (默认: true) 类型: [Boolean](http://www.shouce.ren/api/view/a/13081#articleHeader14) 默认设置下，所有请求均为异步请求（也就是说这是默认设置为 true ）。如果需要发送同步请求，请将此选项设置为 false 。跨域请求和 dataType: "jsonp" 请求不支持同步操作。注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行

****beforeSend**** 类型: [Function](http://www.shouce.ren/api/view/a/13081#articleHeader28)( jqXHR jqXHR, [PlainObject](http://www.shouce.ren/api/view/a/13081#articleHeader25) settings )

请求发送前的回调函数，用来修改请求发送前jqXHR（在jQuery 1.4.x的中，XMLHttpRequest）对象，此功能用来设置自定义 HTTP 头信息，等等。该jqXHR和设置对象作为参数传递。这是一个[Ajax事件](http://docs.jquery.com/Ajax_Events) 。在beforeSend函数中返回false将取消这个请求。

****contents****类型: [PlainObject](http://www.shouce.ren/api/view/a/13081#articleHeader25) 一个以"{字符串/正则表达式}"配对的对象，根据给定的内容类型，解析请求的返回结果。

****data**** 类型: PlainObject 或 [String](http://www.shouce.ren/api/view/a/13081#articleHeader0) 或 [Array](http://www.shouce.ren/api/view/a/13081#articleHeader21) 发送到服务器的数据。它被转换成一个查询字符串,如果已经是一个字符串的话就不会转换。查询字符串将被追加到GET请求的URL后面。对象必须为"{键:值}"格式。如果这个参数是一个数组，会自动转换成&name=for&user=if

****dataType**** (default: Intelligent Guess (xml, json, script, or html)) Type: [String](http://www.shouce.ren/api/view/a/13081#articleHeader0) 从服务器返回你期望的数据类型。 可用的类型（以及结果作为第一个参数传递给成功回调函数）有：

"xml": 返回 XML 文档，可以通过 jQuery 处理。

"html": 返回纯文本 HTML 文本；包含的script标签会在插入DOM时执行。

"script": 把响应的结果当作 JavaScript 执行，并将其当作纯文本返回。

"jsonp": 以 [JSONP](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/) 的方式载入 JSON 数据块。

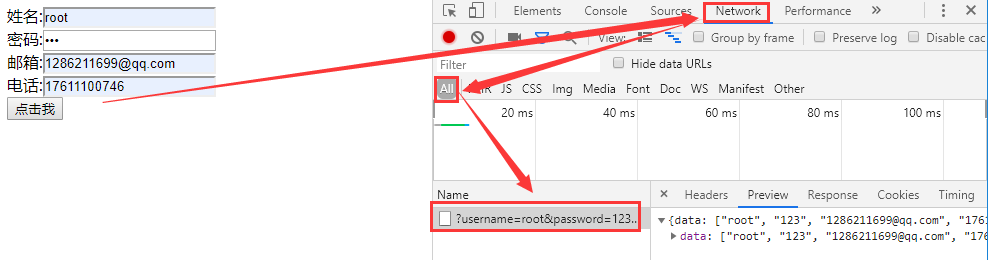
"text": 返回纯文本字符串。

首先我们先行使用GET请求搭建ajax，修改ajaxtest.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <script type="text/javascript" src="/static/js/jquery.js"></script>  
</head>  
<body>  
*{#{% csrf\_token %}#}*姓名:<input type="text" id = 'username'><br>  
密码:<input type="password" id = 'password'><br>  
邮箱:<input type="email" id = 'email'><br>  
电话:<input type="text" id = 'phone'><br>  
<button id = 'tj'>点击我</button>  
</body>  
<script>  
 $(function () {  
 $('#tj').click(function () {  
 var username = $('#username').val();  
 var password = $('#password').val();  
 var email = $('#email').val();  
 var phone = $('#phone').val();  
 $.ajax({  
 url:'/reqform/ajaxtest/',  
 type:'GET',  
 async:true,  
 data:{'username':username,'password':password,'email':email,'phone':phone},  
 dataType:'json',  
 success:function(data){  
 *console*.log(data)  
 },  
 error:function (data) {  
 *console*.log(data)  
 }  
 })  
  
 });  
 })  
</script>  
</html>

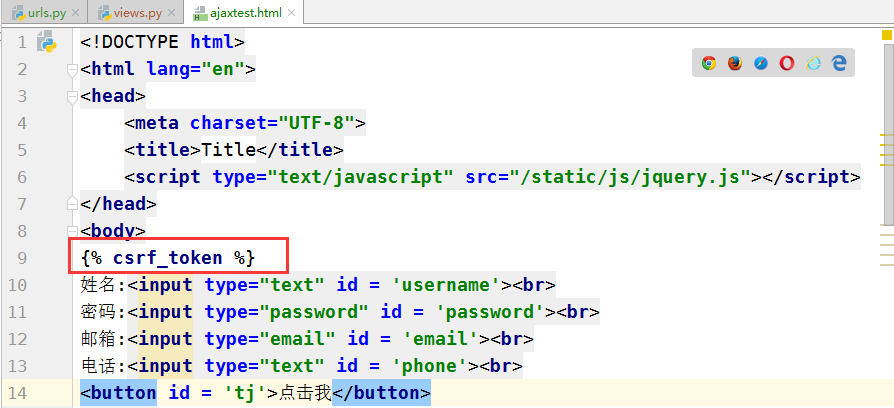
**修改ReqForm/views.py/ajaxtest函数  
def ajaxtest(request):  
 if request.method == 'GET' and request.GET:  
 username = request.GET.get('username')  
 password = request.GET.get('password')  
 email = request.GET.get('email')  
 phone = request.GET.get('phone')  
 print(password)  
 return JsonResponse({'data':[username,password,email,phone]})  
 return render(request,'ajaxtest.html')**

运行结果如下：



接下来让我们尝试下post请求

修改ajaxtest.html中的ajax为post，并在body标签中添加{%csrf\_token%}跨域伪造，如下图所示：





上节课我们已经学到csrf的一些问题，所以在这里我们要伪造请求头部信息或者是直接把csrf\_token中的数据伴随着data进行发送：

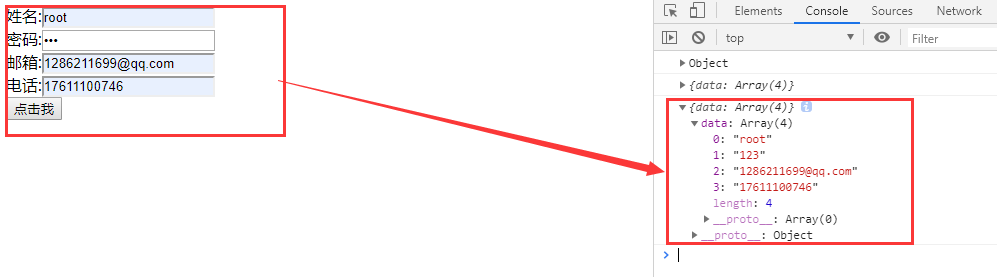
<script>  
 $(function () {  
 $('#tj').click(function () {  
 var username = $('#username').val();  
 var password = $('#password').val();  
 var email = $('#email').val();  
 var phone = $('#phone').val();  
 var csrftoken = $('input[name="csrfmiddlewaretoken"]').val();  
 $.ajax({  
 url:'/reqform/ajaxtest/',  
 type:'POST',  
 *{#headers:{"X-CSRFToken":csrftoken},#}* async:true,  
 data:{'username':username,'password':password,'email':email,'phone':phone,'csrfmiddlewaretoken':csrftoken},  
 dataType:'json',  
  
 success:function(data){  
 *console*.log(data)  
 },  
 error:function (data) {  
 *console*.log(data)  
 }  
 })  
  
 });  
 })

第二种方法：

在发送请求之前给xhr设置请求头部信息如下所示：

**<script>  
 $(function () {  
 $('#tj').click(function () {  
 var username = $('#username').val();  
 var password = $('#password').val();  
 var email = $('#email').val();  
 var phone = $('#phone').val();  
 var csrftoken = $('input[name="csrfmiddlewaretoken"]').val();  
 $.ajax({  
 url:'/reqform/ajaxtest/',  
 type:'POST',  
 *{#headers:{"X-CSRFToken":csrftoken},#}* async:true,  
 data:{'username':username,'password':password,'email':email,'phone':phone},  
 dataType:'json',  
  
 success:function(data){  
 *console*.log(data)  
 },  
 beforeSend:function (xhr,settings) {  
 xhr.setRequestHeader("X-CSRFToken",csrftoken)  
 },  
 error:function (data) {  
 *console*.log(data)  
 }  
 })  
  
 });  
 })  
</script>**

运行效果如下：



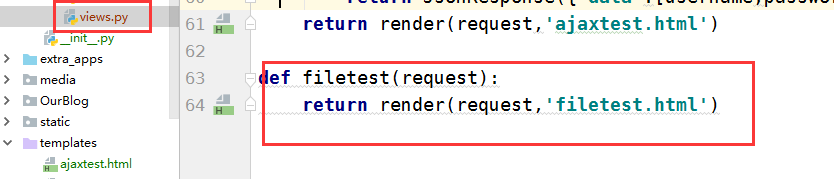
## 12.2 ajax提交文件对象

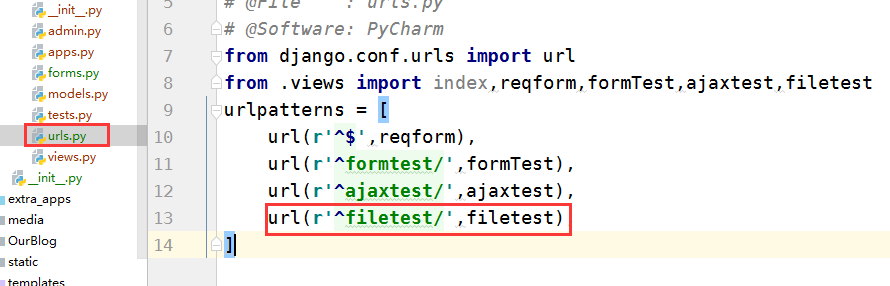
12.2.1 form表单提交文件对象

在学习ajax提交文件之前我们先行学习form表单提交文件。让我们另行创建filetest.html文件

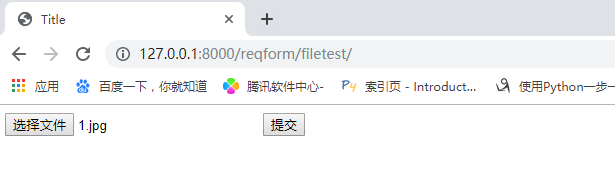
**<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<from>  
 <input type="file" name="file">  
 <input type="submit" value="提交">  
</from>  
  
</body>  
</html>**

**编写ReqForm/views.py  
def filetest(request):  
 return render(request,'filetest.html')**

****

编写ReqForm/urls.py  


运行结果如下所示：



接下来编写逻辑进行后端保存修改filetest.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<form action="/reqform/filetest/" method="post" enctype="multipart/form-data">  
 {% csrf\_token %}  
 <input type="file" name="file">  
 <input type="submit" value="提交">  
</form>  
  
</body>  
</html>

enctype="multipart/form-data" //不对字符编码。在使用包含文件上传控件的表单时，必须使用该值。  
views.py中编写

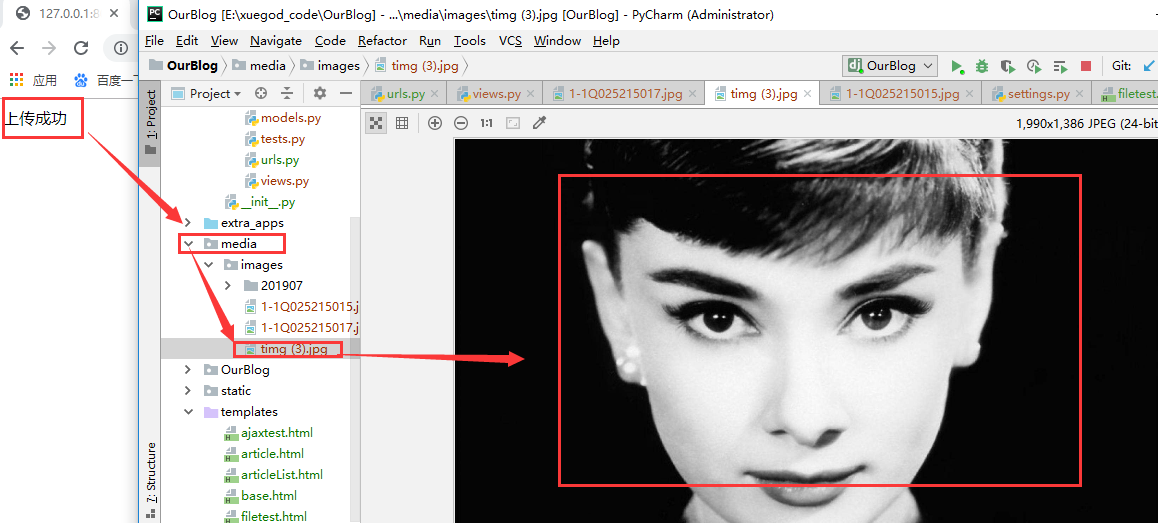
import os  
def filetest(request):  
 if request.method=='POST' and request.POST:  
 *#获取文件名字* filename = request.FILES.get('file').name  
 *#拼接文件路径，课存储到model中方便项目寻找到* filepath = os.path.join('media/images',filename)  
 *#保存图片* with open(filepath,'wb') as f:  
 *# UploadedFile.chunks()：如果上传的文件足够大需要分块就返回真。默认的这个值是2.5M，当然这个值是可以调节的。* for chunk in request.FILES.get('file').chunks():  
 if chunk:  
 f.write(chunk)  
 return HttpResponse('上传成功')  
 return render(request,'filetest.html')

注意：UploadedFile.chunks()：如果上传的文件足够大需要分块就返回真。默认的这个值是2.5M，当然这个值是可以调节的。

有没有同学有疑惑，为什么在django中用chunks来替代read（）？

如果当UploadedFile.read（）：从文件中读取整个上传的数据。小心整个方法：如果这个文件很大，你把它读到内存中会弄慢你的系统.

运行效果如下所示：

上面我们已经学到了如何使用form表单进行发送数据了，接下来让我们使用下ajax方法发送数据

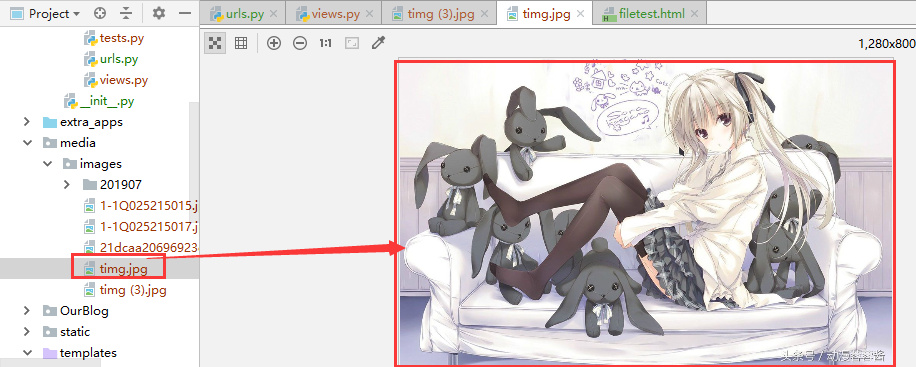
修改filetest.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>代码统计</title>  
 <script src="/static/js/jquery-3.4.1.min.js"></script>  
</head>  
<body>  
<form>  
 {% csrf\_token %}  
 <input type="file" name="file" value="上传文件">  
 <button type="button" id="btn">提交</button>  
</form>  
  
  
<script>  
 *$*(function () {  
 *$*("#btn").click(function () {  
 *//创建一个FormData对象用来存储数据* var file\_obj = new *FormData*;  
 *//通过jquery的的属性操作找到上传的文件,  
 // 注意files方法是js对象的特有的方法，所以需要将jquery对象索引转化成js对象调用此方法* file\_obj.append('file', *$*("input[type='file']")[0].files[0]);  
 *//jquery的属性操作获取csrftoken值来防御CSRF攻击* file\_obj.append('csrfmiddlewaretoken', *$*('[name=csrfmiddlewaretoken]').val());  
 *$*.ajax({  
 url: '/reqform/filetest/',  
 type: 'post',  
 processData: false,*//不让jQuery处理我的file\_obj* contentType: false,*//不让jQuery设置请求的内容类型* data: file\_obj,  
 *//成功回调函数* success: function (res) {  
 *console*.log(res)  
 }  
 })  
 })  
 })  
  
</script>  
</body>  
</html>

略微修改ReqForm/views.py中的filetest 不用改

import os  
def filetest(request):  
 if request.method=='POST' and request.POST:  
 *#获取文件名字* filename = request.FILES.get('file').name  
 *#拼接文件路径，课存储到model中方便项目寻找到* print(filename)  
 filepath = os.path.join('media/images',filename)  
 *#保存图片* with open(filepath,'wb') as f:  
 *# UploadedFile.chunks()：如果上传的文件足够大需要分块就返回真。默认的这个值是2.5M，当然这个值是可以调节的。* for chunk in request.FILES.get('file').chunks():  
 if chunk:  
 f.write(chunk)  
  
 return JsonResponse({'status':'success'})  
 return render(request,'filetest.html')

运行结果如下：

总结：

12.1 ajax介绍与编写

12.2 ajax提交文件对象