1. HTTP协议：

**基于TCP/IP协议**

**短连接**

**被动响应**

**无状态**

**HTTP是基于socket**

**引用cookie 和 session（解决http的无状态）**

2. 用Socket实现一个Web服务器

3. Django跨站请求资源

**1. 通过Jsonp**

只能发get请求，尽量不使用

**2.通过cors跨域资源共享(添加请求头)**

1. 简单请求基本跨越：B中的Ajax不带请求头

比如说：A通过Django写了一个视图来处理B用户的请求。

B用户在自己的页面上通过按钮触发Ajax事件来访问A的地址，A服务器收到了来自B的请求，并且返回数据给B，但是返回数据时存在跨域问题。假如B的地址为localhost:6342

**A下的views大致如下：**

def service(request):

v1 = request.GET.get('v1')

v2 = request.GET.get('v2')

resullt = v1 + v2

obj = HttpResponse(result)

**# 添加请求头即cors跨域，全部都访问则**obj['Acess-Control-Allow-Origin']=**"\*"**

obj['Acess-Control-Allow-Origin'] = 'http://localhost:6342'

return obj

**B中的Ajax请求如下：**

<input type='button' value='按钮' onclick=getData()>

<script>

function getData(){

$.ajax({

url:'http://127.0.0.1:8000/service/?v1=1&v2=1',

type:'GET',

success:function(arg){

alter(arg);

}

})

}

</script>

点击按钮执行结果：弹出2

2. 复杂请求自定义请求头：B中的Ajax自带请求头

浏览器在跨域的时候会自动先发起一个options请求作预检，即请求对方允不允许我

自己自带请求头访问。先发options再发get或post

因此服务端（A下的views）需要更改以让对方请求通过

API:

def service(request):

if request.method == 'OPTIONS'

print('进行预检')

obj = HttpResponse()

obj['Acess-Control-Allow-**Headers**'] = 'k1'

obj['Acess-Control-Allow-Origin'] ="\*"

return obj

else:

v1 = request.GET.get('v1')

v2 = request.GET.get('v2')

resullt = v1 + v2

obj = HttpResponse(result)

obj['Acess-Control-Allow-**Origin**'] = '\*'

return obj

ajax:

function getData(){

$.ajax({

url:'http://127.0.0.1:8000/service/?v1=1&v2=1',

type:'GET',

headers:{

'k1':'v1'

},

success:function(arg){

alter(arg);

}

})

3. 复杂请求PUT请求

API:

def service(request):

if request.method == 'OPTIONS'

print('进行预检')

obj = HttpResponse()

obj['Acess-Control-Allow-Origin'] ="\*"

obj['Acess-Control-Request-**Methods**'] = 'PUT'

return obj

else:

v1 = request.GET.get('v1')

v2 = request.GET.get('v2')

resullt = v1 + v2

obj = HttpResponse(result)

obj['Acess-Control-Allow-Origin'] = '\*'

return obj

ajax:

function getData(){

$.ajax({

url:'http://127.0.0.1:8000/service/?v1=1&v2=1',

type:'PUT',

success:function(arg){

alter(arg);

}

})

obj['Acess-Control-Allow-Headers'] = 'k1'

------------------------------------------------------------------------------

**简单请求与非简单请求**

条件：

    1、请求方式：HEAD、GET、POST

    2、请求头信息：

        Accept

        Accept-Language

        Content-Language

        Last-Event-ID

        Content-Type 对应的值是以下三个中的任意一个

                                application/x-www-form-urlencoded

                                multipart/form-data

                                text/plain

注意：同时满足以上两个条件时，则是简单请求，否则为复杂请求

**简单请求与非简单请求区别**

简单请求：一次请求

简单请求：两次请求，在发送数据之前会先发一次请求用于做“预检”，只有“预检”通过后才再发送一次请求用于数据传输

**关于“预检**

请求方式：OPTIONS

“预检”其实做检查，检查如果通过则允许传输数据，检查不通过则不再发送真正想要发送的消息

如何“预检”

     => 如果复杂请求是PUT等请求，则服务端需要设置允许某请求，否则“预检”不通过

        Access-Control-Request-Method

     => 如果复杂请求设置了请求头，则服务端需要设置允许某请求头，否则“预检”不通过

        Access-Control-Request-Headers

**应用:**

这样一个一个写比较复杂，通常我们在**django中间件实现cors跨域**