# 这一节我们来学习CSRF也就是跨站请求伪造

## 什么是CSRF

|  |
| --- |
|  |

# 1.假设我们有一个webA项目先看看下面的代码

|  |
| --- |
|  |

## 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" action="/transfer">          <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="submit" value="转账">          </p>      </form>  </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+'/');  }) |

### 这个小项目不是一个完整的项目，运行流程是：如果是get方法访问/,就会渲染登录页面，然后以post方式提交到/，然后在这里做模拟身份验证，然后保存session信息，post处理成功后跳转到/transfer，这个路由也会先执行get方法，验证用户是否登录，如果没有就重定向到/的get路由进行登录操作，如果用户已经登录了，就模拟转账，输入模拟转账成功操作

## 其实可以给transfer.htm的提交方法改为ajax需要用到jquery库，我们下载一个jquery到public/js文件夹下面，然后修改views/transfer.html文件内容如下，前提是app注册了静态资源服务

|  |
| --- |
| <!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> |

### 效果

|  |
| --- |
|  |

# 说明，这是一个普通网站，我们用来作为自己的信任网站，也是我们学习csrf攻击测试的网站，下一节我们来测试