动态服务器

从入门到工作：前后分离

版权声明

联系方式

如果你想要购买本课程
请微信联系 xiedaimala02 或 xiedaimala03

如果你发现有人盗用本课程
请微信联系 xiedaimala02 或 xiedaimala03

静态服务器 V.S. 动态服务器

也叫：静态网页 V.S. 动态网页

判断依据

- 是否请求了数据库
  - ✓ 没有请求数据库，就是静态服务器
  - ✓ 请求了数据库，就是动态服务器
- 今天要讲数据库吗
  - ✓ 数据库不属于前端范围，但程序员应该懂一点数据库
  - ✓ 今天直接用 json 文件当作数据库

/db/users.json

- 结构：一个数组

```
[
  {id:1, name: 'frank', password: '***', age: 18},
  {id:2, name: 'jack', password: '***', age: 20}
]
```

- ✓ 注意上面省略了双引号，是错误的写法

- 读取 users 数据
- ✓ 先 fs.readFileSync('./db/users.json').toString()
- ✓ 然后 JSON.parse 一下（反序列化），得到数组

- 存储 users 数据
- ✓ 先 JSON.stringify 一下（序列化），得到字符串
- ✓ 然后 fs.writeFileSync('./db/users/json', data)

目标1

- 实现用户注册功能
  - ✓ 用户提交用户名和密码
  - ✓ users.json 里就新增了一行数据
- 思路
  - ✓ 前端写一个 form，让用户填写 name 和 password
  - ✓ 前端监听 submit 事件
  - ✓ 前端发送 post 请求，数据位于请求体
  - ✓ 后端接收 post 请求
  - ✓ 后端获取请求体中的 name 和 password
  - ✓ 后端存储数据

目标2

- 实现用户登录功能
  - ✓ 首页 home.html，已登录用户可看到自己用户名
  - ✓ 登录页 sign_in.html，供提交用户名和密码
  - ✓ 输入的用户名密码如果是匹配的，就自动跳转首页
- sign_in.html 思路
  - ✓ 前端写一个 form，让用户填写 name 和 password
  - ✓ 前端监听 submit 事件
  - ✓ 前端发送 post 请求，数据位于请求体
  - ✓ 后端接收 post 请求
  - ✓ 后端获取请求体中的 name 和 password
  - ✓ 后端读取数据，看是否有匹配的 name 和 password
  - ✓ 如果匹配，后端应标记用户已登录，可是怎么标记?

目标2受阻，目标太大了

目标应该尽量小

目标3

标记用户已登录

Cookie

- 定义
  - Cookie 是服务器下发给浏览器的一段字符串
  - 浏览器必须保存这个 Cookie（除非用户删除）
  - 之后发起相同二级域名请求（任何请求）时，浏览器必须附上 Cookie
- 以公园门票作为对比（画图）
  - 假如你是公园检票员，你怎么知道谁能进谁不能进?
  - 有票能进，没票不能进
  - Cookie 就是门票
  - 有 Cookie 就是登录了，没 Cookie 就没登录
  - 那后端给浏览器下发一个 Cookie 不就完事了嘛

Set-Cookie 响应头

语法看 MDN 文档

home.html 怎么知道登录的是谁呢

把 logined 改成 user_id

目标4

- 显示用户名

- ✓ home.html 渲染前获取 user 信息
- ✓ 如果有 user，则将 {{user.name}} 替换成 user.name
- ✓ 如果无 user，则显示登录按钮

有一个大 bug

用户可以篡改 user_id 啊啊啊！

开发者工具或者 JS 都能改

目标5：防篡改 user_id

- 思路一：加密
  - 将 user_id 加密发送给前端，后端读取 user_id 时解密，此法可行，但是有安全漏洞
  - 漏洞：加密后的内容可无限期使用
  - 解决办法：JWT（这节课不讲）
- 思路二：把信息隐藏在服务器
  - 把用户信息放在服务器的 x 里，再给信息一个随机 id
  - 把随机 id 发给浏览器
  - 后端下次读取到 id 时，通过 x[id] 获取用户信息
  - 想想为什么用户无法篡改 id（因为 id 很长，而且随机）
  - x 是什么？是文件。不能用内存，因为断电内存就清空
  - 这个 x 又被叫做 session（会话）

Cookie / Session 总结

- 服务器可以给浏览器下发 Cookie
  - ✓ 通过 Response Header
  - ✓ 具体语法见 MDN
- 浏览器上的 Cookie 可以被篡改
  - ✓ 用开发者工具就能改
  - ✓ 弱智后端下发的 Cookie 用 JS 也能篡改
- 服务器下发不可篡改的 Cookie
  - ✓ Cookie 可包含加密后的信息（还得解密，麻烦）
  - ✓ Cookie 也可只包含一个 id（随机数）
  - ✓ 用 session[id] 可以在后端拿到对应的信息
- 这个 id 无法被篡改
  - ✓ 但可以被复制，问题不大

目标6：注销

- 思路
  - ✓ 将 session id 从 session 里删掉
  - ✓ 将 cookie 从浏览器删掉（可选）
  - ✓ 即可
- 实现
  - ✓ 前端制作注销按钮
  - ✓ 前端监听注销按钮点击事件
  - ✓ 前端发送 delete session 请求
  - ✓ 后端接受 delete session 请求
  - ✓ 后端获取当前 session id，并将其删除
  - ✓ 后端下发一个过期的同名 Cookie（因为没有删除）

还有一个大 bug

用户密码被泄露了怎么办

目标7：防止密码泄露

- 不要存明文
  - ✓ 拿到明文之后，bcrypt.js 加密，得到密文
  - ✓ 将密文保存到 users.json
  - ✓ 用户登录时，加密后对比密文是否一致
  - ✓ 一致则说明用户知道密码，可以登录
- 不要用 MD5
  - ✓ 傻 X 才用 MD5 来处理密码
  - ✓ MD5 甚至都不是一种加密算法，是 hash 算法
  - ✓ 不要被辣鸡教程误导
  - ✓ 深入阅读，请看这篇博客

我们终于做出了一个最简单的动态网站

涉及的知识非常多

大总结

- 如何获取 post 请求体（后端知识）
  - ✓ 其实我在把大家培养成全栈
- 如何使用 Cookie（后端知识）
  - ✓ 永远不要用 JS 操作 Cookie，要 http-only
- 什么是 Session（后端知识）
  - ✓ 就是后端的一个文件，保存会话数据，一般存用户信息
- 注册、登录、注销的实现（后端）
  - ✓ 每个 Web 程序员都应该搞清楚这些知识
  - ✓ 然而 80% 的前端并不会这些
  - ✓ 我们的课程，让你完爆大部分竞争者

应试技巧

- Cookie 和 Session 的区别
  - ✓ Cookie 存在浏览器，Session 存在服务器
  - ✓ Cookie 长度有限制，Session 一般没有
  - ✓ Session 是借助 Cookie 实现的，id 存放在 cookie
- Cookie 和 LocalStorage 的区别
  - ✓ Cookie 会被放在每一次的请求里，LS 不会
  - ✓ Cookie 长度有限制，LS 的长度一般为 5M~10M
- 题外话
  - ✓ 其实我还蛮鄙视一些面试官的，为了问问题而问问题
  - ✓ 这些都是完全不同的概念，硬要放在一起比较
  - ✓ 你需要面试之前记忆这些答题关键点，否则吃亏

Cookie 其他细节

- 以下内容为选学内容，有兴趣的话自己看

相同二级域名

- 不是同源
- 举例：a.qq.com 和 qq.com 的二级域名都是 qq.com，所以 a.qq.com 发起请求时，必须附上 qq.com 里的 cookie

功能

- 具体请看 MDN 文档
- 后端可以设置 Cookie 的过期时间
- 后端可以设置 Cookie 的域名（不能乱设置）
- 后端可以设置 Cookie 是否能被 JS 读取（禁止 JS）
- 后端可以设置 Cookie 的路径（用得较少）

再见

下节课学习 MVC 设计模式