高级node第十一单元（6.12）

2020年6月11日 15:21

从服务器下载文件
使用token进行登录验证
实现一个独立的用户系统（koa，mysql，token）
认识前端的一些新的API
- Blob
- URL.createObjectURL
- File
- FileReader
- ArrayBuffer

从服务器下载文件

1. 下载是浏览器的行为
2. 浏览器对于资源有两种策略
   a. 如果该资源类型可以识别，则会直接在浏览器中打开（解析）
   b. 如果该资源类型不可以识别，则会进行下载，一般不识别的资源类型设置成
   application/octet-stream

```
import fs from 'fs'
export default async (ctx, next) => {
  ctx.response.set('content-type', 'application/octet-stream')
  // ctx.response.set('content-type', 'application/json')
  console.log(ctx.response.header);
  ctx.body = fs.readFileSync('./package.json');
}
```

3. 具体的文件内容还是由响应内容提供
4. 下载文件的名字，取决于下载url地址的路径部分最后一段的设定

```
[
  {
    pathname: '/download/package.json',
    method: 'GET',
    application: download
  }
]
```

使用token进行登录验证

1. 使用过session-cookie的方式
   a. 登录成功之后，后端要生成一个sessionsid(用户身份标识)
   b. 在后端进行存储，这个行为叫session
   c. 把sessionid返回给前端（使用cookie存储）
   d. 之后的每次请求都要携带sessionsid
   e. 后端会对sessionsid进行比对
2. 使用token
   a. 登录成功之后，后端要根据用户的相关信息（用户名，登录时间等等）生成一个token（身份标识），token内存储了有用的信息
   b. 把token返回给前端，前端进行存储
   c. 之后的每次请求都要携带token
   d. 后端会对token进行解密（只有生成token的人才能解），进行验证

使用node-jsonwebtoken进行token的生成和验证

1. 生成token

```
import jwt from 'jsonwebtoken'

const token = jwt.sign(
  { username: 'zs' }, // 载体信息
  '1234', // 秘钥
  { expiresIn: 30000 } // 其他配置项，过期时间
);

console.log(token);
```

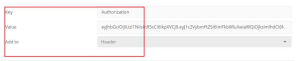
2. 解析token

```
// 解析token
const payload = jwt.verify(
  token, // 被解析的token字符串
  '1234' // 秘钥
);

console.log(payload)
```

前端请求的时候如何携带token

1. 通过cookie自动携带
2. 通过请求头携带，一般规定使用 Authorization字段保存token



在后端获取前端所携带的token

1. cookie中获取

```
// 获取前端传递过来的token
// 1. 通过cookie拿
let token = ctx.cookies.get('token');
console.log(token);
```

2. 通过Authorization请求头字段去拿token

```
// 2. 通过Authorization请求头字段去获取token
let AuthoToken = ctx.request.headers['authorization'];
if (AuthoToken) {
  token = AuthoToken;
}
```

更换用户的头像

1. 美化上传文件按钮
2. 让图片可以被预览
   a. 通过input file标签选中文件
   b. 通过 input.files拿到问到文件对象

   c. 如何把文件对象编程url地址
   Blob类型的数据可以通过window.URL.createObjectURL转换为url类型，被特殊的标签加载

```
> b = new Blob(['abc'], {type:'text/plain'})
< ▶Blob {size: 3, type: "text/plain"}
> window.URL.createObjectURL(b)
< "blob:http://localhost:3000/b1191a2a-1546-49ad-98b5-3aabefbd0b88"
> window.URL.createObjectURL(b)
```


```
console.log(inputFile.files);
showImg.src = window.URL.createObjectURL(inputFile.files[0]);
}
```

简单认识FileReader和ArrayBuffer

1. FileReader的作用就是去读取Blob实例

2. 类型化数组 ArrayBuffer
   直接可以存储资源的二进制形式，但是不能直接操作，需要有一个更加高级的接口（视图）

```
> buf = new ArrayBuffer(5)
< ▶ArrayBuffer(5) {}
> buf
< ▼ArrayBuffer(5) {}
    ▶[[Int8Array]]: Int8Array(5) [0, 0, 0, 0, 0]
    ▶[[Uint8Array]]: Uint8Array(5) [0, 0, 0, 0, 0]
    byteLength: 5
    ▶__proto__: ArrayBuffer
```