

PHASE 2

WEEK 2

DAY 2



План

1. Cookies
2. Sessions
3. bcrypt (библиотека хеширования паролей)
4. IDOR (Insecure Direct Object Reference)
5. Skeleton (каркас приложения)*

Cookies

HTTP cookie

- Фрагмент данных в виде “ключ-значение”
- Хранится в браузере
- Привязан к хосту

HTTP cookie

Обычно используется для:

- аутентификации пользователя
- хранения сессии
- хранения корзины, игровой статистики, etc.
- хранения личных настроек
- сбора статистики о пользователях

Установка cookie

- Сервер — через заголовок ответа
- Клиент — с помощью JS

Установка сессионной cookie с сервера

```
res.cookie(  
    'test', // имя (ключ)  
    42, // значение  
); // метод .cookie() не завершает запрос  
  
// Response Headers:  
// Set-Cookie: test=42; Path=/  

```

Установка постоянной cookie с сервера

```
res.cookie('test', 42, {  
    maxAge: 90000,  
});
```

// значение в мс., кука удалится через 1.5 минуты

Установка постоянной cookie с сервера

```
res.cookie('test', 42, {  
    expires: new Date('Dec 31, 2022'),  
});  
// создаёт постоянную куку с датой истечения срока
```

Установка приватной cookie с сервера

```
res.cookie('test', 42, {  
  httpOnly: true,  
});
```

// создаёт куку недоступную через клиентский JS

Server > Set-Cookie > Client

```
res.cookie('test', 42, {  
    maxAge: 90000,  
    httpOnly: true,  
});
```

```
// Response Headers:
```

```
// Set-Cookie: test=42; Max-Age=90; Path=/;
```

```
Expires=Mon, 01 Oct 2021 13:10:21 GMT; HttpOnly
```

Чтение cookie на сервере

Куки отправляются с клиента на сервер в заголовке **Cookie**.

```
Cookie: test=42
```

Чтение cookie на сервере

Установка: `npm i cookie-parser`

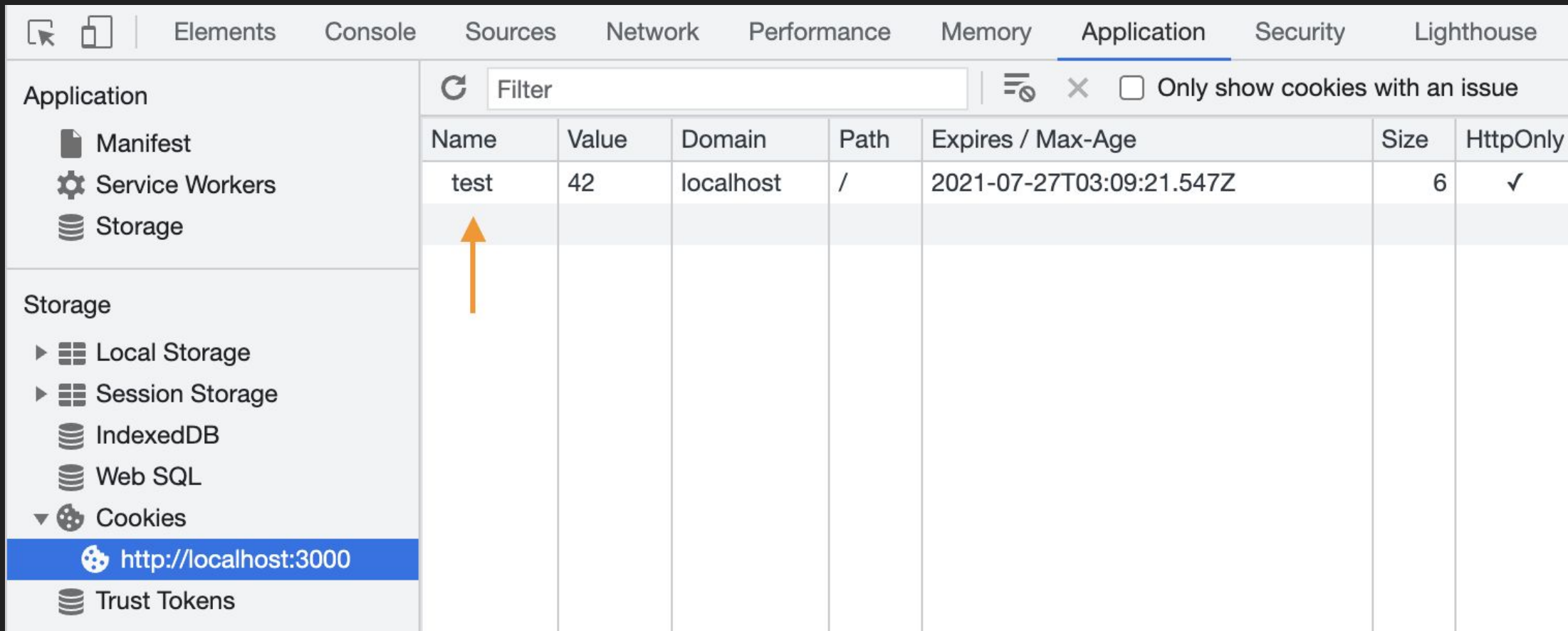
Middleware для чтения заголовка запроса `Cookie` и создания объекта `req.cookies` на основе его содержимого.

Чтение cookie на сервере

```
const cookieParser = require('cookie-parser');  
app.use(cookieParser());  
  
app.get('/my-cookies', (req, res) => {  
  console.log(req.cookies); // { test: '42' }  
  res.json(req.cookies);  
});
```

Просмотр cookie на клиенте

Chrome DevTools / Application / Storage / Cookies / <ХОСТ>



The screenshot shows the Chrome DevTools Application panel with the 'Cookies' sub-panel selected. The left sidebar shows the 'Application' tree with 'Storage' expanded and 'Cookies' selected for the host 'http://localhost:3000'. The main area displays a table of cookies. An orange arrow points to the first row of the table.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly
test	42	localhost	/	2021-07-27T03:09:21.547Z	6	✓

Чтение cookie на клиенте

`document.cookie`

// геттер, возвращает строку со всеми доступными
куками

Важно!

Куки с `httpOnly` недоступны на клиенте.

Установка сессионной cookie с клиента

```
document.cookie = 'myCookie=the best cookie';  
  
// сеттер для добавления или перезаписи куки с  
таким же именем
```

Важно!

Не меняет другие куки.

Установка постоянной cookie с клиента

Можно указывать дополнительные опции через точку с запятой.

```
document.cookie = "cookievalue;  
expires=Sat, 03 Sep 2022 20:00:00 UTC; path="/";
```

Удаление cookie на клиенте

Делается через установку куки с истекшим сроком годности:

```
document.cookie = "cookieName=; expires=Thu, 01  
Jan 1970 00:00:00 GMT";
```

Удаление cookie на сервере

// удалить куку с сервера по ключу

```
res.clearCookie('user_id');
```

// метод clearCookie не завершает запрос

Sessions

Сессия

Данные, связанные с определённым посетителем. Хранятся на сервере.

На клиент обычно передаётся id сессии – случайная строка, зашифрованная по секретному слову, которая хранится в cookie.

Подключение сессий в Express

Установка: `npm i express-session`

Middleware для генерации сессий и отправки куки с id сессии на клиент.

Создаёт объект `req.session`, который сохраняется между запросами.

Конфигурация сессий в Express

```
const session = require('express-session');

const sessionConfig = {
  name: 'user_sid',           // Имя куки для хранения id сессии. По умолчанию - connect.sid
  secret: process.env.SESSION_SECRET ?? 'test', // Секретное слово для шифрования, может быть любым
  resave: false,             // Пересохранять ли куку при каждом запросе
  saveUninitialized: false,   // Создавать ли сессию без инициализации ключей в req.session
  cookie: {
    maxAge: 1000 * 60 * 60 * 12, // Срок истечения годности куки в миллисекундах
    httpOnly: true,              // Серверная установка и удаление куки, по умолчанию true
  },
};

app.use(session(sessionConfig));
```


Пример использования req.session

```
app.get('/', (req, res) => {  
  if (req.session.count) {  
    req.session.count += 1;  
  } else {  
    req.session.count = 1;  
  }  
  res.json(req.session);  
});
```

Хранение сессии

По умолчанию сессия хранится в памяти серверного приложения. Это приводит к утечке памяти и не предназначено для production-приложений.

Альтернативные способы хранить сессию:

<https://www.npmjs.com/package/express-session#compatible-session-stores>

Хранение сессии в файле

Установка: `npm i session-file-store`

Хранилище для сессий в виде файлов. Создаёт папку `sessions` на сервере.

1. Эту папку нужно добавить в файл `.gitignore`
2. Если используете `nodemon`, то игнорируйте эту папку в скрипте запуска в файле `package.json`:

```
"dev": "nodemon server.js --ignore ./sessions"
```

Хранение сессии в файле

```
const session = require('express-session');  
const FileStore = require('session-file-store')(session);  
  
const sessionConfig = {  
  store: new FileStore(),  
  /* ... */  
};  
  
app.use(session(sessionConfig));
```

Серверное удаление сессии и куки

```
app.get('/logout', (req, res) => {  
  // удаление сессии на сервере  
  req.session.destroy((error) => {  
    if (error) {  
      return res.status(500).json({ message: 'Ошибка при удалении сессии' });  
    }  
  
    res  
      .clearCookie('user_sid') // серверное удаление куки по имени  
      .json({ message: 'Успешный выход' });  
  });  
});
```

bcrypt

bcrypt

Установка: `npm i bcrypt`

Библиотека для хэширования (шифрования) строк – обычно паролей.

Подключается к тому файлу, в котором мы хотим что-то шифровать.

bcrypt hashing, example code

```
const bcrypt = require('bcrypt');

app.post('/register', async (req, res) => {
  // Пропустить пароль и случайно сгенерированную соль
  // через алгоритм хэширования 2^10 раз
  const hash = await bcrypt.hash(req.body.password, 10);

  /* ... */
});
```


bcrypt hashing, salt

Соль (salt) – случайно сгенерированная строка.

Соединяется со строкой, которую нужно зашифровать, и пропускается через алгоритм.

saltRounds определяет, сколько раз исходная строка + соль будут пропущены через алгоритм хэширования.

При `saltRounds = 10` данные будут обработаны 2^{10} раз.

bcrypt compare, example code

```
app.post('/login', async (req, res) => {  
  const user = await User.findOne({ where: { email: req.body.email } });  
  
  if (user) {  
    // Сравнить пароль из формы логина с хэшированным паролем из БД  
    const isSame = await bcrypt.compare(req.body.password, user.password);  
    /* ... */  
  } else {  
    /* ... */  
  }  
});
```

bcrypt compare

Расшифровать захэшированную строку нельзя, но можно получить из хэша **соль** и **количество раундов**, использованные для хэширования исходной строки.

Используя соль и количество раундов из существующего хэша, можно захэшировать другую строку и сравнить, идентичны ли эти хэши.

Если они одинаковы — значит, обе исходные строки совпадают.

IDOR

Insecure Direct Object Reference, небезопасная прямая ссылка на объект

IDOR

IDOR, Insecure Direct Object Reference — это небезопасная прямая ссылка на объект / ресурс на сервере.

С помощью этой атаки злоумышленник может получить приватную информацию пользователей, например их профиль, файлы или другие данные.

IDOR: применения и последствия

Характерные действия при IDOR:

- Чтение чужих данных
- Изменение чужих данных
- Повышение привилегий (частный случай изменения данных)
- Захват аккаунта (частный случай повышения привилегий)

IDOR: базовые принципы безопасности

Предотвращение IDOR:

- Пользователь может редактировать / удалять только свои ресурсы
- Нет возможности редактировать / удалять ресурсы через HTTP клиенты
- Нет возможности повышения привилегий (частный случай изменения данных)

Skeleton*

Skeleton

- “Скелет” - готовый каркас приложения
- Изучайте и разбирайте “скелеты” пошагово
- Составляйте собственные каркасы для приложений
- Не используйте их бездумно