

Лабораторная работа «Быстрое создание механизма сессий в приложении Express»

Время выполнения: ~ 90 минут

Введение

Идентифицировать X – это означает распознать объект в качестве X , т.е. выяснить, что объект – это элемент X некоторого множества. Т.е. соотнести признаки объекта с признаками, которые служат для включения X в некоторое множество.

Например, чтобы доказать, что некто является клиентом банка, некто предъявляет паспорт. Сверив лицо предполагаемого клиента с фотографией в паспорте и сверив номер паспорта с записью в базе данных, сотрудник банка идентифицирует его или её как клиента. Паспорт бывает достаточен для снятия денег со счёта, но чтобы, к примеру, заблокировать карту по телефону, нужно назвать ещё и кодовое слово.

Всё это – вопросы точности идентификации. С одной стороны, паспорт однозначно представляет гражданина страны. С другой стороны, паспорт может быть похищен, а внешность – подделана. Для простоты, компьютерная система оперирует моделью пользователя, представленной на базовом уровне логином и паролем.

Если мы знаем, что два разных человека пользуются одним и тем же логином и паролем, то для системы в рамках такой модели это один и тот же пользователь.

Идентифицировать – это значит, условно говоря, найти пользователя в множестве пользователей – принять его или её заявку на вход в систему, а аутентифицировать – значит действительно подтвердить, что это тот самый пользователь, за которого кандидат себя выдаёт. Эти два акта тесно связаны и разделить их нелегко, да и нет необходимости. Что же касается авторизации, то это отнесение аутентифицированного пользователя к той или иной группе в аспекте *прав*, предоставление ему личного аккаунта / кабинета сообразно с этими правами.

Двухэтапная или двухфакторная аутентификация – такая, при которой используются два идентифицирующих фактора из разных категорий (их три: что **только этот** пользователь **знает**, что **имеет** и чем **является**). Например, пользователь **знает** пароль и **имеет** мобильный телефон. Значит, после успешного ввода пароля на телефон придёт сообщение с уникальным краткосрочным кодом. К третьей категории относятся ДНК, отпечатки пальцев и т.п.

Логин отсюда надо исключить, т.к. это общедоступная информация. Её знает не только пользователь. Грубая аналогия: лицо (логин) и паспорт (пароль). Это также можно сопоставить с шифрованием двумя ключами (публичный и секретный).

Цель умножения числа факторов – снизить вероятность идентификации, которая приведёт к неправильной аутентификации. Но вероятность того, что даже самые продуманные меры не помогут, всё равно ненулевая.

Выражаясь осторожно, закрытая часть сайта/приложения – это такая, доступ к которой можно получить только после успешной аутентификации. Т.е. приватный раздел увидит только тот, кто ввёл существующий логин и верный соответствующий этому логину пароль.

В рамках фреймворка Express это реализуется с помощью промежуточного программного обеспечения - `middleware`.

Идея в том, что обработчик маршрута к закрытому разделу не сразу выдаёт то, что там находится, а только если будет успешный ответ от соответствующего `middleware`.

Т.е., например, все маршруты под маршрутом `/users` или `/profile` или `/cabinet` должны выдаваться только через этот `middleware`.

Сессия – это состояние «включена аутентификация», которое может быть программно реализовано по-разному. Но начинается всё с того, чтобы в это состояние прийти, т.е. аутентифицироваться, т.е. «залогиниться».

Для этого логично выделить маршрут типа `/login`

1. Подготовьте рабочее место к выполнению задания в экосистеме Node.js (убедитесь, что **node.js** и **curl** установлены или воспользуйтесь docker-контейнером с работающим node последней стабильной версии, например <https://hub.docker.com/r/igossoudarev/nodesimple/>).

2. Выполнив команду `node -v` убедитесь, что версия node не ниже 7.4.x

3. Создайте папку для приложения и перейдите в неё в командной строке или терминале.

4. По адресу <https://kodaktor.ru/g/git> найдите и выполните команды для загрузки шаблона Express-приложения на локальный компьютер `curl -kSLO 'https://kodaktor.ru/et' && unzip et && rm et && npm i`

5. В файле `package.json` измените строку, содержащую описание сценария “start” на `"start": "port=7777 node index.js"`

6. Выполните для запуска приложения `npm start`

7. В другом окне командной строки или терминала выполните `curl localhost:7777` и убедитесь, что эта команда возвращает строку `<h1>Welcome to Express!</h1>` (или посетите этот адрес в браузере)

8. Аналогично убедитесь, что команда `curl localhost:7777/api` возвращает `{"gossApi":"started ok!"}` и не требует логина/пароля.

9. Выполните `npm i -S express-session body-parser lodash`

10. Выполните `curl -kSL 'https://kodaktor.ru/g/session_post' -o './public/login.html'` и `curl -kSL 'https://kodaktor.ru/j/users' -o './users.json'`

11. После этого в подпапке `public` вашей папки приложения окажется файл `login.html`

12. В файле `server.js`

(а) в раздел зависимостей добавьте

```
session = require('express-session'),
bodyParser = require('body-parser'),
u = require('./users'),
_ = require('lodash'),
```

(б) и далее объявите функцию `middleware`:

```
const checkAuth = (req, res, next)=>{
  console.log( req.session.auth);
  if (req.session.auth==='ok') {
    next();
  } else {
    res.redirect('/login');
  }
};
```

(в) в раздел маршрутов добавьте маршрут

```
.get('/login', (req,res)=>{
  res.sendFile( path.join(__dirname, 'public/login.html') );
})
.post('/login/check/', (req, res)=>{
  const login = req.body.login,
  user = _.find(u.users, {login});
  if (user) {
    if ( user.password === req.body.pass ) {
      req.session.auth='ok';
      res.send('Good!');
    } else {
      res.send('Wrong pass!');
    }
  }
  else {
    res.send('No user!') ;
  }
})
```

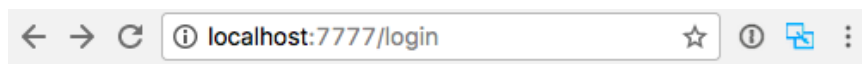
(г) в раздел `use` добавьте

```
.use ( bodyParser.json() )
.use ( bodyParser.urlencoded({ extended: true}) )
.use(session({ secret: 'mysecret', resave: true, saveUninitialized: true })))
```

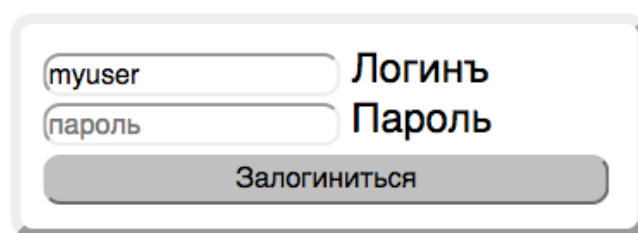
(д) Измените начало обработчика маршрута `/api`, добавив к нему вызов промежуточного программного обеспечения `checkAuth`:

```
.get('/api', checkAuth, (req, res) => {
```

13. Перезапустите приложение (`Ctrl C`, `npm start`) и перейдите на его адрес в браузере:



Введите логин и пароль!

A login form with a light gray border and rounded corners. It contains two input fields: the first is labeled 'Логинъ' and contains the text 'myuser'; the second is labeled 'Пароль' and contains the text 'пароль'. Below these fields is a wide button labeled 'Залогиниться'.

14. Убедитесь, что теперь попытка посетить маршрут `/api` перенаправляет клиента на форму с логином и паролем.

15. Убедитесь, что после успешного заполнения формы (`myuser`, `myras`) попытка перейти на маршрут `/api` выводит `{"gossApi":"started ok!"}` как ранее в п.8

16. Реализуйте маршруты:

(а) `/logout` для выхода из сессии

(б) `/name`, возвращающий ваше имя и фамилию на русском языке с правильным заголовком с кодировкой `utf-8`

17. Реализуйте запоминание посещённого маршрута (до авторизации) так, чтобы после переадресации на форму и успешного логина приложение возвращалось к закрытому маршруту

18. Составьте отчёт о проделанной работе по обычной схеме, разместите файлы приложения в портфолио (`git push`) и поместите в форум с заданием ссылку на получившийся результат.

Критерии оценивания

Критерии и типичные ошибки	Результат
Работа выполнена полностью и в срок, без ошибок.	Задание принято
Всё сделано в целом верно, однако допущены ошибки из числа следующих: <ul style="list-style-type: none">• Веб-приложение не полностью корректно проходит процедуру проверки• В коде веб-приложения присутствуют предупреждения, не влияющие в целом на его работу• ...	Задание принято с учётом замечаний
Работа не выполнена, допущены ошибки из числа следующих: <ul style="list-style-type: none">• Неправильно указано имя автора• Адрес задания недоступен (ошибка в написании адреса)• В коде приложения обнаруживаются фатальные ошибки, неустраняемые уязвимости для атак• В приложении обнаружен вредоносный код, осуществлена попытка атаки на веб-сервер	Задание не принято или возвращено на доработку

Внимание! В таблице указаны примерные критерии, и по сумме допущенных ошибок преподаватель может снизить итоговый балл за всю работу.

Размещение отчёта

https://kodaktor.ru/report_template.pptx

В качестве отчётов по заданиям дисциплины студентам необходимо подготовить слайды/презентацию, если не оговорено отдельного способа сдачи работ.

Пример-шаблон отчётной презентации: [j.mp/report_template](https://kodaktor.ru/report_template.pptx). Основные требования:

1. На первом/титальном слайде укажите ФИО, группу и цель работы
2. На слайдах разместите отчёт (текст, изображения, ссылки, видео)
3. [На предпоследнем слайде оформите список использованной литературы или источников]
4. На последнем слайде сформулируйте выводы по результатам выполнения задания. Какая цель достигнута?
5. Опубликуйте презентацию. Если вы используете Google Docs, то дайте ссылку на расшаренный ресурс. Иначе ссылку на папку или файл Яндекс.Диска. Ссылки опубликуйте в ответе на тему в форуме в Moodle
6. Разместите ответ в форуме и дождитесь, чтобы запись была оценена преподавателем.