PHASE 2

WEEK 1

DAY 2

ELBRUS/BOOTCAMP

План

1. Routing, Router (маршрутизация)
2. Request parameters (параметризованные запросы)
3. Middleware (промежуточные функции)
4. Static (статические файлы)
5. res.locals, app.locals

Routing, Router

Routing, Router

Express позволяет задавать обработчики маршрутов по одному, либо отдельным модулем — роутером.

Express Router позволяет группировать обработку запросов для маршрутов с единым префиксом в отдельный модуль.

Одиночные обработчики

```
app.get('/students', (req, res) => {
 res.send('GET-запрос к маршруту /students');
});

app.post('/students', (req, res) => {
 res.send('POST-запрос к маршруту /students');
});
```

Отдельный модуль — Router

```
// routes/students.route.js
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
 res.send('GET-запрос к маршруту /students');
});

router.post('/', (req, res) => {
 res.send('POST-запрос к маршруту /students');
});

module.exports = router;
```

Express Router: подключение

```
// app.js
const express = require('express');
const studentsRouter = require('./routes/students.route.js');

const app = express();

app.use('/students', studentsRouter);
// ...
```

Важно:

Префикс роутера добавляется ко всем адресам в роутере.

Parameterized Queries

```
app.get('/students/:id', async (req, res) => {
 const student = await Student.findByPk(req.params.id);
 res.json({ student });
});
```

Сработает на маршруты:

- /students/123
- /students/asdfsdf

НЕ сработает на маршруты:

- /students/
- /students/123/comments
- /student/123

Middleware

(промежуточные функции)

Middleware: подключение

Промежуточные функции подключаются с помощью метода **use()**

```
app.use(function myMiddleware(req, res, next) {
 // сделать что-нибудь

 // вызвать следующую промежуточную функцию или обработчик маршрута
 next();
});
```

Express Static

```
app.use(express.static(path.join(__dirname, 'public')));
```

Если после указания статики положить в папку public какие-нибудь файлы, то они будут доступны браузеру (Express создаст маршрутизацию для каждого файла самостоятельно).

Нужно для клиентских скриптов, картинок, стилей, etc.

Клиентский и серверный JS

Надо всегда понимать, где и в какой момент будет выполняться код, который вы сейчас пишите. На клиенте или на сервере?

- Если js лежит в папке статики – скорее всего на клиенте
- Если вне её – скорее всего на сервере.

res.locals

Объект для передачи свойств между middleware, обработчиками и шаблонами.

```
app.use(async (req, res, next) => {
 const user = await User.findByPk(req.params.id);

 // Теперь переменная user придёт в React-компонент при рендеринге
 res.locals.user = user;
 next();
});
```

app.locals

app.locals — объект имеющий свойства, которые являются локальными переменными в пределах всего приложения (в React-компонент эта переменная не придёт автоматически), пример установки:

```
app.locals.title = 'Hello!'
```

После установки значения app.locals свойства сохраняются на протяжении всего жизненного цикла приложения, в отличие от свойств res.locals, которые действительны только в течение времени существования запроса.

Пример middleware

Отправляет HTML на основе React-компонента

Пример кастомного middleware

```
// middleware/ssr.js
// Middleware для добавления метода renderComponent в объект res
function ssr(req, res, next) {
 res.renderComponent = renderComponent;
 next();
}

module.exports = ssr;
```

Пример кастомного метода для res

```
// Вспомогательная функция для отправки HTML на основе React-компонента
function renderComponent(reactComponent, props = {}, options = { doctype: true }) {
 const reactElement = React.createElement(reactComponent, {
   ...this.app.locals, // передать app.locals
   ...this.locals, // передать res.locals
   ...props, // передать пропсы
 });

 const html = ReactDOMServer.renderToStaticMarkup(reactElement);
 if (!options.doctype) return this.send(html);

 const document = '<!DOCTYPE html>' + html;
 this.send(document);
}
```

Подключение кастомного middleware

```
// app.js
// ...
const ssr = require('./middleware/ssr');

// ...
app.use(ssr);

// ...
```

Использование кастомного метода

```
// Если нужна страница целиком
res.renderComponent(ProductListPage, { products });

// ИЛИ если нужен фрагмент страницы (без doctype)
res.renderComponent(ProductItem, { product }, { doctype: false });
```