Урок N4

Фреймворк Express

Чтобы не писать избыточный код для удобной организации веб-приложения, мы будем использовать самый популярный на данный момент фреймворк **express**. Он компактен, быстр, легок и удобен в применении. В то же время он может расширяться с помощью дополнительных модулей, за счет чего обладает большой гибкостью в использовании и может служить надежным фундаментом практически для всех видов сетевых приложений.

Содержание урока

- Фреймворк для веб-приложений express
- Шаблонизатор handlebars
- Пример: программа-переводчик с HTML-интерфейсом

Устанавливается стандартно через npm под названием express.

Пример Hello, world:

```
var express = require('express');
var app = express();

app.get('/', function(req, res){
        res.send('hello world');
});
```

Для инициализации используется интерфейс Application (app).
Для работы со входящими запросами - Request (req).
Для работы с ответами на запросы - Response (res).
В некоторых случаях для маршрутизации используется Router.
Для расширения возможностей используются middleware-модули.

По сути, Application просто хранит настройки приложения и отвечает за маршрутизацию запросов. Request и Response похожи на стандартные объекты из node плюс для удобства содержат дополнительные поля и методы. Подробное описание всех методов и свойств этих объектов расположено на сайте express.

Middleware позволяет по желанию расширить возможности фреймворка, подключив дополнительные функции. Например, при подключении body-parser, в объект req в виде объекта body будут автоматически попадать данные из POST-запроса:

```
var express = require('express');
var app = express();

var bodyParser = require('body-parser');
app.use(bodyParser());
```

Основные расширения фреймворка перечислены здесь. Самые полезные из них: body-parser для получения данных из POST, serve-static для отдачи статичных файлов, cookie-parse, cookie-session и session для работы с сессиями и cookies.

В классической модели MVC шаблоны вредставлений необходимы для отделения визуального оформления в формате HTML от непосредственной исходной структуры данных и бизнес-логики приложения. Конечно же, express предоставляет удобные методы работы с шаблонами и их применение является хорошей практикой.

Пример шаблона в формате handlebars (с расширением html или hbs):

```
<div class="entry">
	<h1>{{title}}</h1>
	{{#if error}}
		<h2>Ошибка! {{error}}</p>
	{{/if}}
	<div class="body">
		<ul class="people_list">
			{{#each people}}
				<li>{{this}}</li>
			{{/each}}
		</ul>
	</div>
</div>
```

При вызове функции-шаблонизатора ей передается содержимое шаблона и объект с данными, которые будут подставлены в него. В примере выше значение поля title подставится в заголовок, заголовок второго уровня с сообщением error будет выведен только при наличии непустого поля error, а массив people будет преобразован в список.

Пример: hello, handlebars!

hello.js:

```
// подключаем express
var express = require('express');
// создаем приложение
var app = express();

// подключаем поддержку шаблонизаторов
var templating = require('consolidate');
// выбираем функцию шаблонизации для hbs
app.engine('hbs', templating.handlebars);
// по умолчанию используем .hbs шаблоны
app.set('view engine', 'hbs');
// указываем директорию для загрузки шаблонов
app.set('views', __dirname + '/views');

// обрабатываем запросы к главной странице
app.get('/', function(req, res){
        // выводим данные на основе шаблона
        res.render('hello', {
                title: 'Привет, handlebars!'
        });
});
```

hello.hbs:

```
<h1>{{title}}</h1>
```

Самоконтроль

✓ Преимущества использования express
✓ Роутинг в express
✓ Middleware в express
✓ Шаблонизаторы в express
✓ Шаблонизатор handlebars

Домашнее задание

1) Создать на основе express и handlebars веб-сервис с HTML-интерфейсом для динамической загрузки информации с одного из нескольких сайтов в выбранном формате. Зайдя на этот сервис, пользователь сможет с помощью формы настроить параметры информационной подборки (например, количество отображаемых новостей или их категорию) и получить ее в удобном виде. Форма должна отправляться на сервер методом POST.

2) Реализовать запоминание с помощью cookie текущих настроек формы и при заходе на сайт показывать последние использованные настройки. Если cookie не существует, можно при отображении формы дополнительно учитывать передаваемые GET-запросы (например, ?count=10&lang=ru и т.д.)