

Basic Web

Node Modules, MVC, Express.js, Handlebars

SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

Have a Questions?

sli.do

#fund-js

1. **Node** Modules
 - HTTP
 - Create a simple HTTP Server
2. **Express.js** Framework
3. **Model-View-Controller (MVC)**
4. MVC with Node, Express.js, Handlebars





**module
.exports**

Node Modules

Create a Basic Web Server

- A set of functions you want to include in your application

- Include modules:

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

Use **require** to include a module

- Create a module:

```
exports.myDateTime = function () {  
    return Date();  
};
```

Use **exports** to export a module

The HTTP Module

- Built-in module, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP)
- Can **create an HTTP server** that listens to server **ports** and gives a **response** back to the client
- Use the **createServer()** method to create an HTTP server



- **writeHead()** - sends a response header to the request.
Requires: **status code** (like 404), **status message** (optional) and **response headers** (object)
- **write()** - sends a chunk of the response body. Can be a string or a buffer

```
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.write('Hello Web!');  
  res.end();
```

ends the response

Creating a Simple Web Server

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

We have to require
http in order to use it

Here we start the server

```
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('Hello Web!');  
}).listen(8080);  
console.log('Listening on port 8080');
```

Here we choose a port

- Now type **node {filename}** and open **localhost:8080** in the browser

A background network diagram consisting of a central dark blue circle with the word 'express' in white. Surrounding this central circle are several smaller, light gray circles connected by thin gray lines, forming a web-like structure. The circles vary in size and are distributed across the frame, with some lines extending towards the edges.

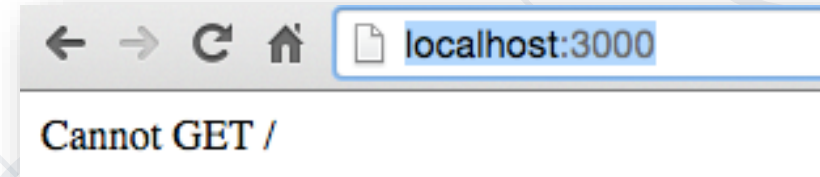
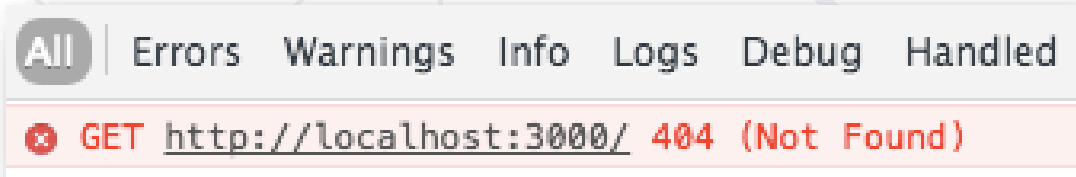
express

Express.js

Working with a Framework

What is ExpressJS?

- Web framework for Node.js
- Handles **GET / POST HTTP** requests
- Error handling (bad request, not found, unauthorized)



- Routing supported

```
app.post('/users/:id', function (req, res) {})
```

Define **URL** parameters

- Create a **directory** to hold your application

```
mkdir demoapp  
cd demoapp
```

- Create a **package.json** file that stores **dependency** information

```
npm init
```

- Now install **express** inside the directory

```
npm install express --save
```

Saves the **dependency** inside
package.json

- Refers to determining how an application responds to a **client request** to a particular **endpoint**
- Express executes different **functions**, based on **route**:

Specify **HTTP Request** method
(GET / POST)

Function to execute when the
route is matched

```
app.get('/api/todos', function(req, res) {})
```

Express **instance**

URL (path on server)

Request & Response

- Create an **index.js** file

node **index.js**

```
const express = require('express');  
const app = express();  
const port = 3000;
```

The function handles **HTTP GET**
requests at URL **'/'**

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

```
app.listen(port, () =>  
  console.log(`Example app listening on port: ${port}`));
```

Handle Different HTTP Methods

- Routing in express gives you the ability to handle **different** HTTP requests

```
app.post('/login', function(req, res) {})
```

```
app.put('/books/:id', function(req, res) {})
```

```
app.delete('/books/:id', function(req, res) {})
```

- You can get a URL parameter from **req.params**

```
app.get('/books/:id', function(req, res) {  
  let bookId = req.params.id;  
  console.log(bookId);  
});
```

- Chaining **multiple** parameters

```
app.get('/user/:first/:second', function(req, res) {  
  console.log([req.params.first, req.params.second]);  
});
```

Serving Static Files in Express

- Use the **express.static** built-in **middleware** function in Express to serve static files
- Create a **public folder** and inside store static files after that write inside **index.js** the following:

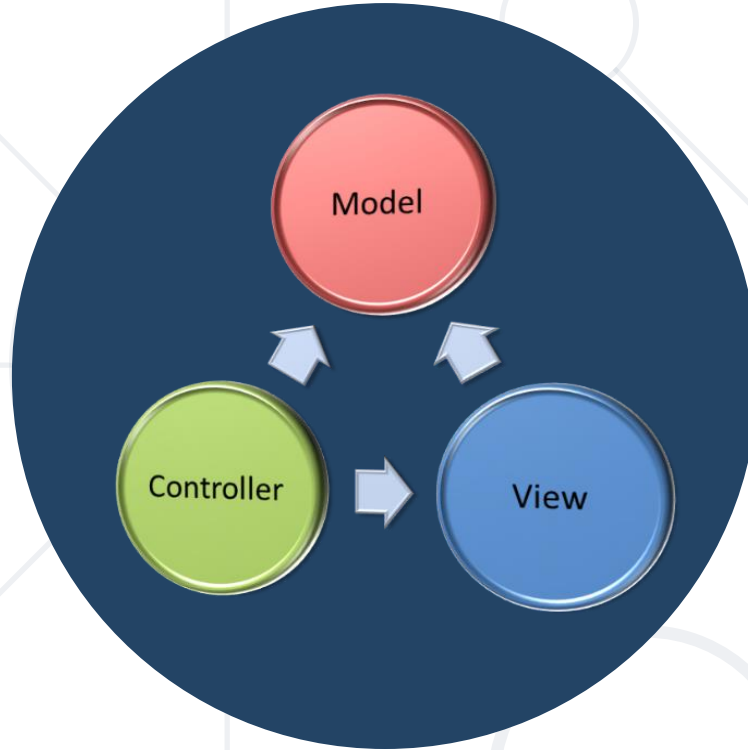
```
app.use(express.static('public'));
```


Parsing Incoming Requests

- Use **body parser** to parse incoming request bodies available under the **req.body** property

```
npm install body-parser --save
```

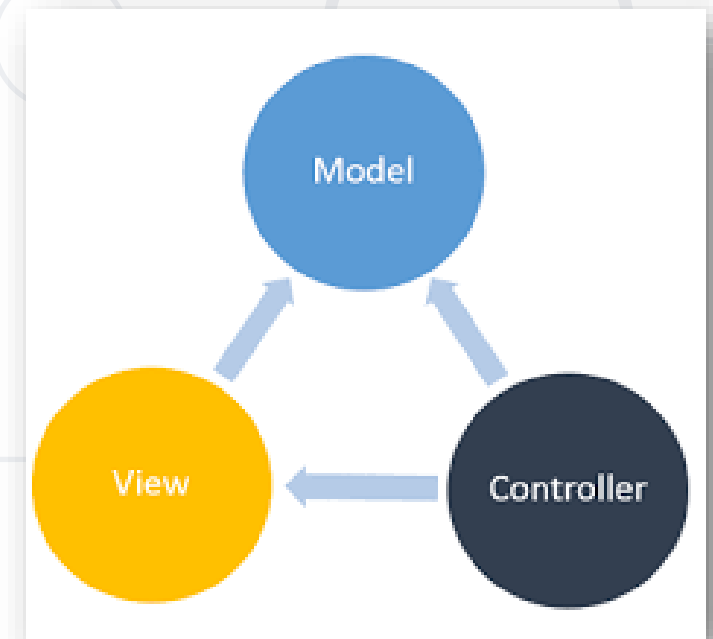
```
const bodyparser = require('body-parser');  
  
app.use(bodyparser.urlencoded({  
  extended: true  
}));
```



Model-View-Controller

The MVC Pattern

- Design pattern with **three** independent components
 - **Model (data)**
 - Manages **data** and **database logic**
 - **View (UI)**
 - Presentation layer (renders the UI)
 - **Controller (logic)**
 - Implements the application logic
 - Processes user request, performs an **action**, updates the data model and invokes a view to render some UI



- Set of classes that describes the **data** we are working with
- Rules for how the data can be **changed** and **manipulated**
- May contain **data validation** rules
- Often **encapsulates** data stored in a database
 - As well as code used to manipulate the data



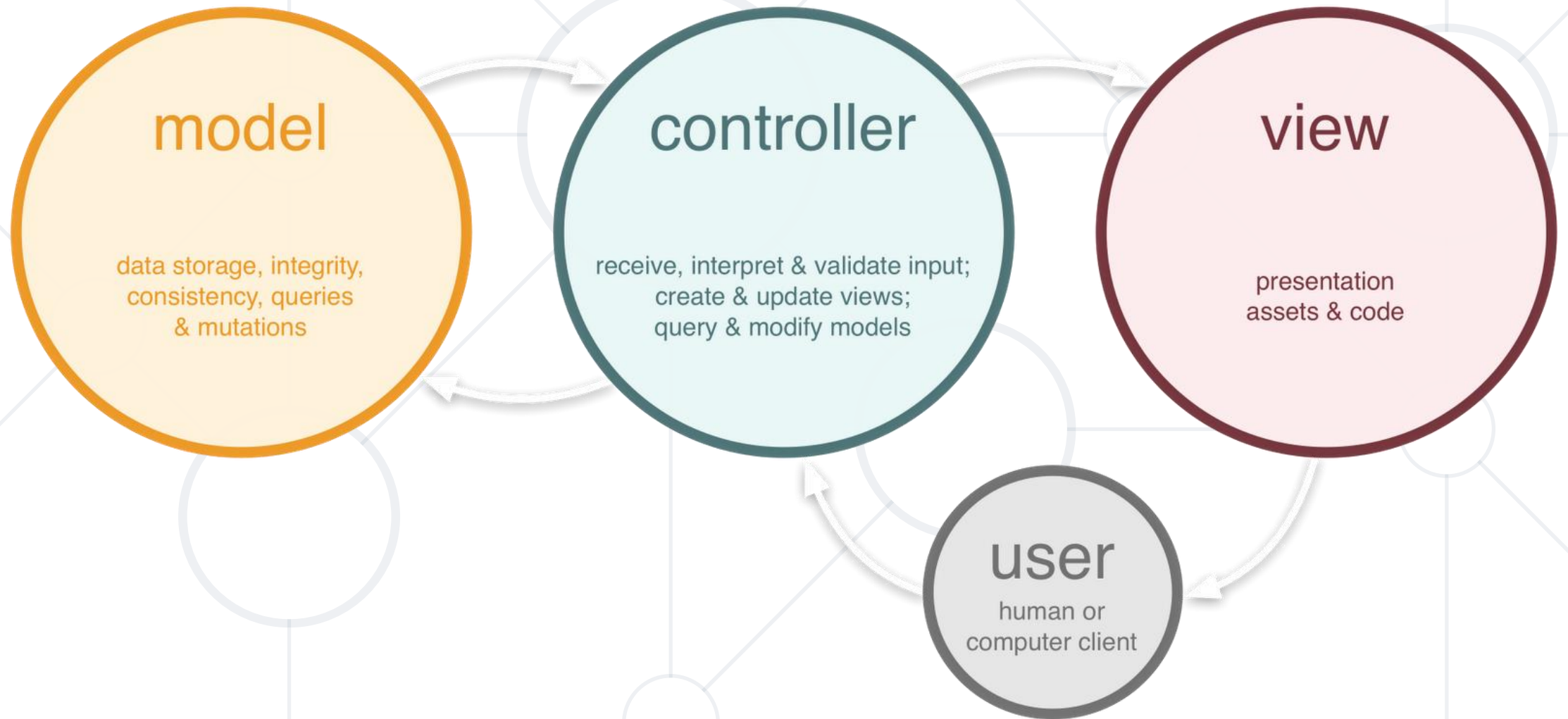
- Defines how the application's **user interface** (UI) will be displayed
- May support **master views** (layouts)
- May support **sub-views** (partial views or controls)
- May use **templates** to dynamically generate HTML



- The **core** MVC component – holds the logic
- Process the requests with the help of views and models
- A set of classes that handles
 - Communication from the user
 - Overall application flow
 - Application-specific logic (business logic)
- Every controller has one or more "actions"



The MVC Pattern (in Web Apps)





MVC with Express.js

Using Node.js, Express.js, Handlebars

Handlebars Templates

- Handlebars provides the power necessary to let you build **semantic templates** effectively
- It is based on the **Mustache** template language, but improves it in several important ways
- To install it inside an Express.js project type in cmd:

```
npm install express-handlebars --save
```



- HTML views with Handlebars templating syntax:

HTML Code

```
<div class="container body-content">
  <div class="row">
    <h2>{{cat.name}}</h2>
    <p>Age: {{cat.age}}</p>
    {{#if cat.isAlive}}
    <p>Status: Alive</p>
    {{else}}
    <p>Status: Deceased</p>
    {{/if}}
  </div>
</div>
```

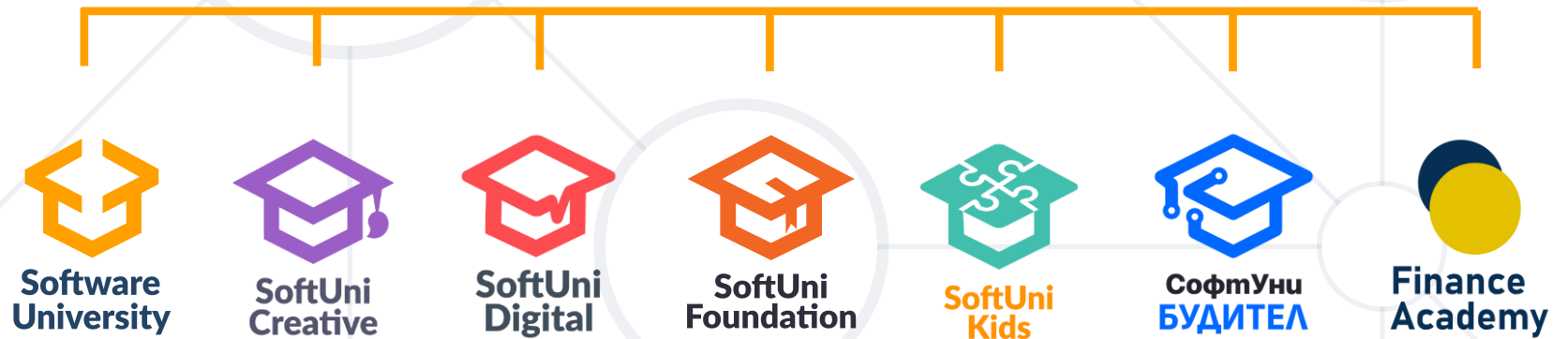
Handlebars
syntax

Handlebars
If/else

- Node.js – JavaScript **runtime environment**
- We use **Node** and **HTTP** to create servers
- MVC is a **design pattern** with individual components
- Express.js – **Web Framework** for building **server-side** JavaScript apps



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

