

Basic Web

Node Modules, MVC, Express.js, Handlebars

http://

node
express

handlebars
mustache



SoftUni Team
Technical Trainers



Software University

<http://softuni.bg>

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



Have a Question?

sli.do

#tech-fund



**module
.exports**

Node Modules

Create a Basic Web Server

- A set of functions you want to include in your application.
- Include modules:

Use **require** to include a module

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

- Create a module:

Use **exports** to export a module

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

The HTTP Module

- Built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).
- The HTTP module 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.



- Here are some methods that are often used:
 - **writeHead()** - Sends a response header to the request. It requires status code (like 404), status message (optional) and the last argument are the response headers (object).
 - **write()** - sends a chunk of the response body. Chunk 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

- We can create a simple server using Node and HTTP:

We have to require
http in order to use it

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

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



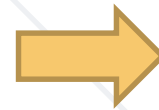
express

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

Define **URL** parameters

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

- 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

- Routing 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}`));
```

- 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

- To serve static files such as images, CSS files, and JavaScript files, use the **express.static** built-in **middleware** function in Express.
- Create a **public folder** and inside store static files after that write inside **index.js** the following:

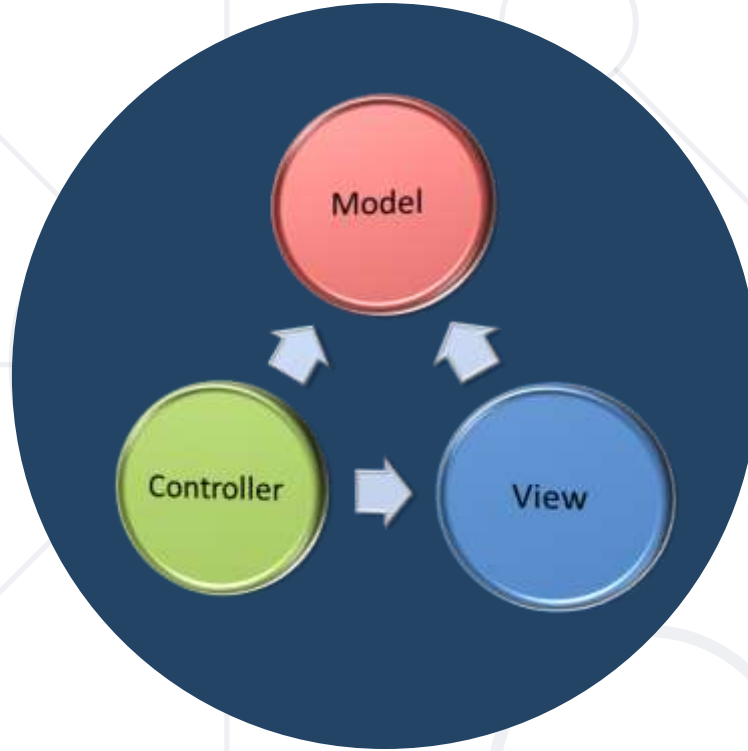
```
app.use(express.static('public'));
```


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  
}));
```

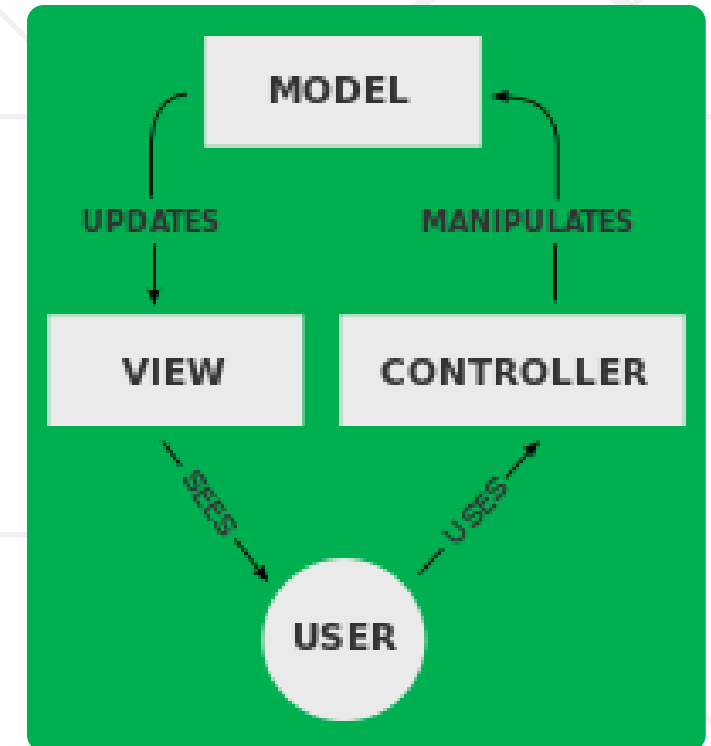


Model-View-Controller

The MVC Pattern

The MVC Pattern

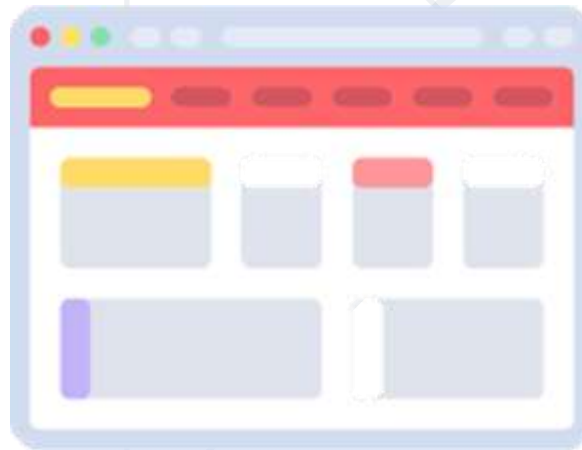
- Design pattern with three independent components
 - **Model (data)**
 - Manages data and database logic
 - **View (UI)**
 - Presentation layer (renders the UI)
 - **Controller (logic)**
 - Implements the application logic
 - Processes user request, performs an action, updates the data model and invokes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

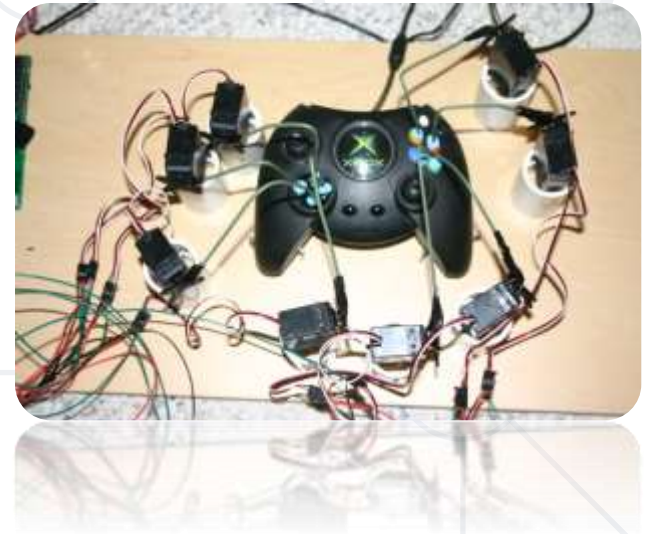
```
const exerciseDataTypes = {  
  deadlift: Number,  
  benchpress: Number,  
  militaryPress: Number,  
  squat: Number  
};  
  
const programSchema = mongoose.Schema({  
  user: {  
    ref: "User",  
    type: String  
  },  
  startingOneRepMaxes: exerciseDataTypes,  
  cycles: [cycleSchema]  
});
```

- 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

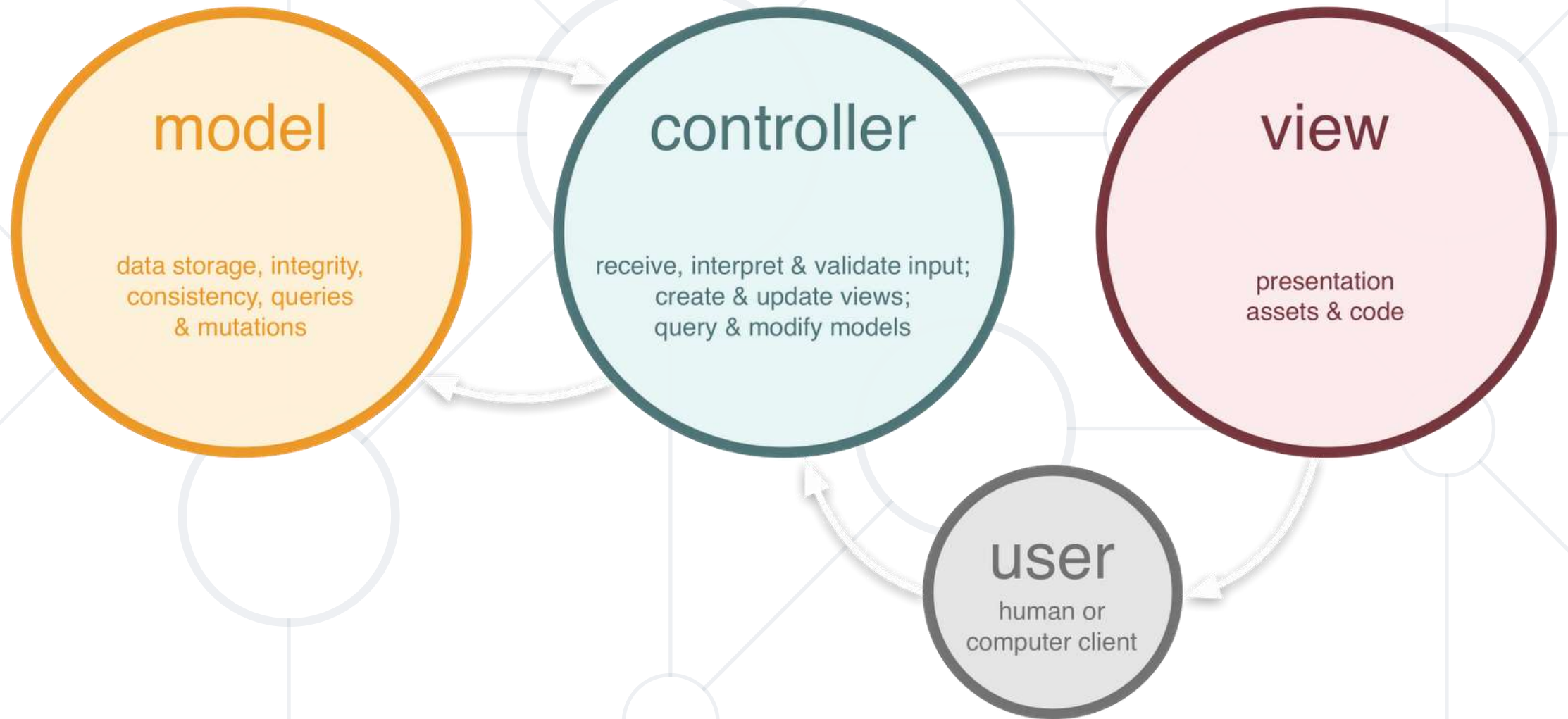


Controller (Logic)

- 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

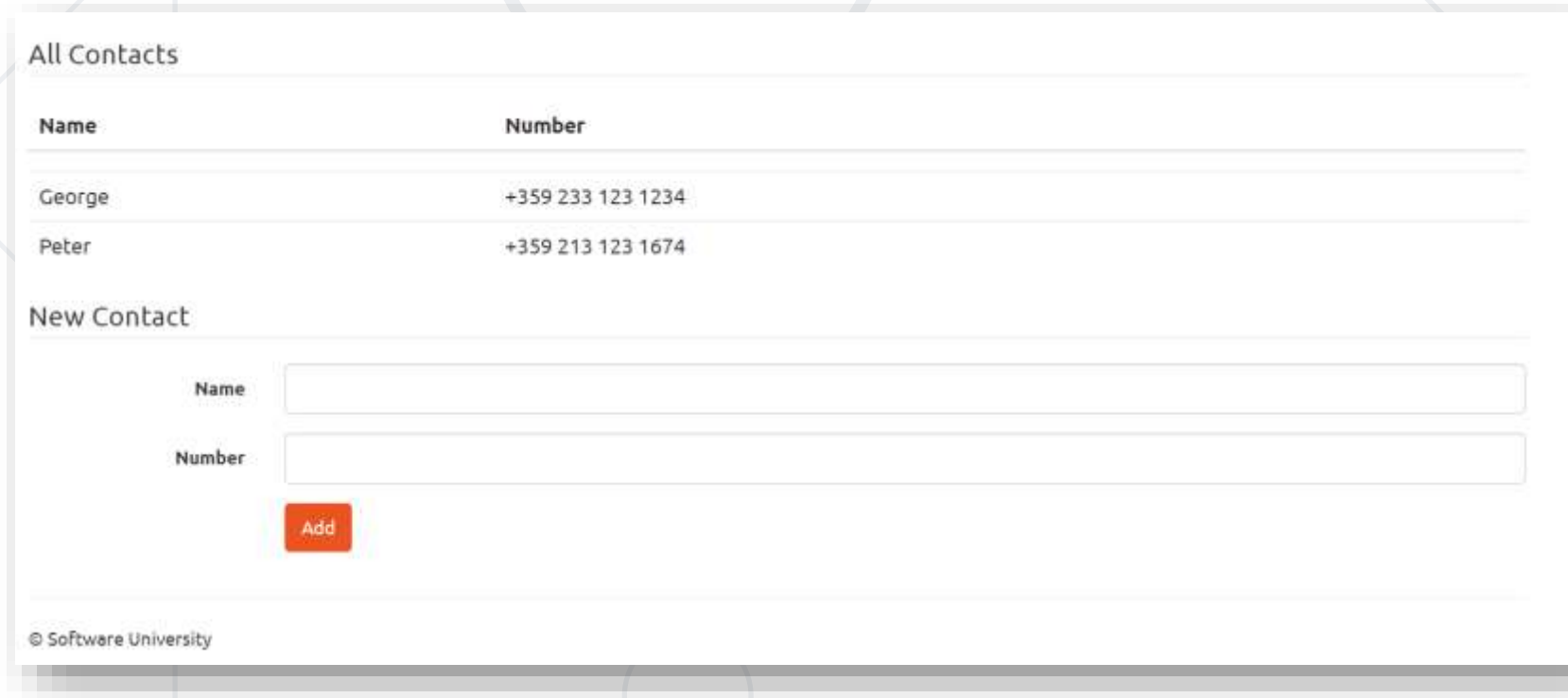
```
<div class="container body-content">
  <div class="row">
    <h2>{{cat.name}}</h2>
    <p>Age: {{cat.age}}</p>
    {{#if cat.isAlive}}
    <p>Status: Alive</p>
    {{else}}
    <p>Status: Deceased</p>
    {{/if}}
  </div>
</div>
```

Handlebars
syntax

Handlebars
If/else

Problem: Simple Phonebook Application

- Write a web application, which displays and creates **contacts** in a **phonebook**
- Implement **listing and adding** contacts



The mockup shows a web application interface for a phonebook. It features a header 'All Contacts' above a table with two columns: 'Name' and 'Number'. The table contains two entries: 'George' with number '+359 233 123 1234' and 'Peter' with number '+359 213 123 1674'. Below the table is a section titled 'New Contact' with two input fields labeled 'Name' and 'Number', and a red 'Add' button. The footer of the mockup reads '© Software University'.

Name	Number
George	+359 233 123 1234
Peter	+359 213 123 1674

New Contact

Name

Number

© Software University

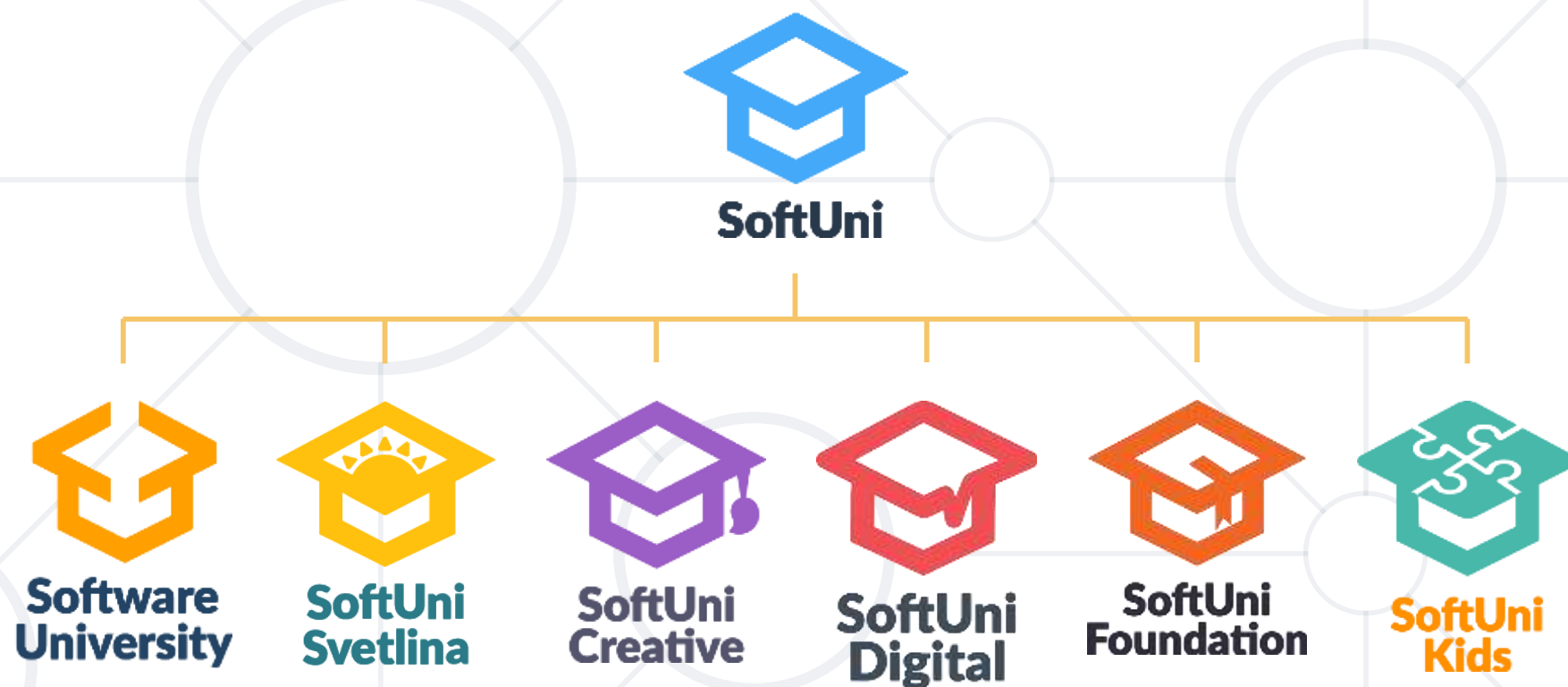


Live Exercises

- 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



XSsoftware



SBTech
we know sports



telenor



SoftwareGroup
doing it right

NETPEAK



SmartIT



Postbank

Решения за твоето утре

**SUPER
HOSTING**
.BG

INDEAVR

Serving the high achievers



INFRAGISTICS®

LIEBHERR



aeternity



codexio

SoftUni Organizational Partners



Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
 - softuni.bg
- Software University Foundation
 - <http://softuni.foundation/>
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

