Multimedia im Netz
(Online Multimedia)
Wintersemester 2014/15

Übung 07 (Hauptfach)
Today’s Agenda

• Announcements

• NodeJS Basics
  – Setting up NodeJS
  – Using Node at the CIP Pool
  – Basic middleware functionality
  – Side notes
  – Template Engines: Jade

• Solution: Currency converter.
OPEN LAB DAY
01.12.14, 18:00 bis 22:00
Amalienstraße 17

Die Arbeitsgruppen Medieninformatik und Mensch-Maschine-Interaktion präsentieren aktuelle Forschungs- und Studentenarbeiten.

www.medien.ifi.lmu.de/openlab
Looking back... The Challenge

- If more than **75** students (HF & NF) submit assignment 06, there will be a short **mock exam** prior to the final exam.

https://openclipart.org/detail/168636/challenge-accepted-by-tavin
Assignment Submissions

A01: 64 (HF), 9 (NF)
A02: 25 (HF), 3 (NF)
A03: 51 (HF), 4 (NF)
A04: 40 (HF), 2 (NF)
A05: 31 (HF), 4 (NF)
A06: 64 (HF), 5 (NF)

01.12.2014 15:50h
Announcements

• Assignment 06
  – Assignment 06 has been submitted 69 times (HF & NF)
  – Therefore there will be **no written mock exam (n < 75)**
  – However, we will discuss exemplary questions in the tutorials after the Christmas break.
Inofficial Evaluation Results

• Thank you for filling out the form and your valuable feedback!
• Changes:
  – Sample solutions: if you allow us, we’ll put a sample solution from your submissions on the website for all assignments
  – Corrections will be somewhat more detailed

• Find the results here:
  https://docs.google.com/forms/d/1q3lw3l59ixq2ZplYsE8afTavDS1AYNKIIdM6gdlqXgA/viewanalytics
Selected Results: Difficulty

The assignments are too difficult

- **strongly agree**: few votes
- **agree**: moderate votes
- **neither agree nor disagree**: many votes
- **disagree**: moderate votes
- **strongly disagree**: few votes

**Number of votes**
Selected Results: Discussion of Assignments

Discussing the assignments during the tutorials is important for me.

- strongly agree: [Full bar]
- agree: [Full bar]
- neither agree nor disagree: [Bar]
- disagree: [Bar]
- strongly disagree: [Bar]

Number of votes

0  2  4  6  8  10  12  14  16
Node JS – What is it?

• “Node.js® is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications” (official website, nodejs.org)

• Key features:
  – Asynchronous operations, event driven → callback pattern!
  – Server process included (no Apache is needed)
  – Modularized
  – Only language: JavaScript
Installing Node

- NodeJS is available for all major platforms
- Download at [www.nodejs.org](http://www.nodejs.org)
- Remarks:
  - on Ubuntu you might have to add the NodeJS repository to get a fairly up-to-date version. More info here: [https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-an-ubuntu-14-04-server](https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-an-ubuntu-14-04-server)
Hello World!

```javascript
var http = require('http');
var port = 1337;
var host = '127.0.0.1';

var server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World
');
});

server.listen(port, host);
console.log('Server running at http://' + host + ':' + port + '/');
```
Node Package Manager

• Easy handling of packages from the command line

• Most important operations:
  – Global package installation
    `npm install -g PACKAGE [PACKAGE2 PACKAGE3 ...]`
  – Local package installation
    `npm install PACKAGE [PACKAGE2 PACKAGE3 ...]`
  – Local package installation & saving to dependencies list:
    `npm install PACKAGE [PACKAGE2 ...] --save`

• Unfortunately, npm is not available at the CIP Pool.
Package script – package.json

```
{
  "name": "TestExpress",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "express": "~4.9.0",
    "body-parser": "~1.8.1",
    "cookie-parser": "~1.3.3",
    "morgan": "~1.3.0",
    "serve-favicon": "~2.1.3",
    "debug": "~2.0.0",
    "jade": "~1.6.0",
    "less-middleware": "1.0.x"
  }
}
```

$ npm install installs all dependencies that are listed in the package script into the node_modules directory
Excursus: NodeJS at the CIP Pool

• Unfortunately, npm is not available at the CIP pool at this time
• Solution: use a template project provided from us.
• Option 1 (preferred):
  – Sign up for a GitLab account here: https://tools.rz.ifi.lmu.de/cipconf/index.rb?op=gitlab
  – Generate an SSH Key (if you haven’t already)
    $ ssh-keygen
  – Copy the content of ~/.ssh/id_rsa.pub (SSH Key) to your GitLab profile: https://gitlab.cip.ifi.lmu.de/profile/keys
  – Clone the template to your computer:

    $ mkdir assignment07 && cd $_
    $ git clone git@gitlab.cip.ifi.lmu.de:stockinger/mmn-nodejs-template.git
Step-by-Step: WebStorm Project Setup (1)

1. Check out from Version Control ➔ Git
Step-by-Step: WebStorm Project Setup (2)

2. Enter repository URL
3. Choose a directory, into which the files are cloned
4. Choose a directory name (automatically created within parent)
Step-by-Step: WebStorm Project Setup (3)

5. Once the project window has opened, click this small arrow and choose „Edit Configurations...“
6. Click on the + Symbol and choose Node.js
Step-by-Step: WebStorm Project Setup (5)

7. Enter a run config name
   Check „Single instance only“

8. Configure path to node executable if empty.

9. Choose script to run (in the template it’s bin/www)
Step-by-Step: WebStorm Project Setup (6)

10. Run (or debug) the app

- Search Everywhere with Double ↩
- Open a file by name with ↑jectories
- Open Recent Files with 📚
- Open Navigation Bar with 🎨
- Drag and Drop file(s) here from Finder
Step-by-Step: WebStorm Project Setup (7)

11. Check log & stop app here.
Starting a node app from the Command Line

• On your own machine:
  $ node <path_to_app>

• On a CIP pool computer:
  $ nodejs <path_to_app>

• Generate an express app (alternatives)
  1. Create a new WebStorm project with NodeJS Express template
  2. install the Express generator:
     $ npm install –g express-generator
     $ express myApp
— a web application framework

• One of the most popular NodeJS frameworks.
• Characteristics:
  – minimalistic
  – easy to use API
  – many utility methods and middleware functionalities
  – thin layer on top of NodeJS
• Side notes:
  – responsible for the letter E in the MEAN stack
• Find the documentation here: [http://expressjs.com/](http://expressjs.com/)
Basic Express App

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

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

var server = app.listen(3000, function () {
    var host = server.address().address;
    var port = server.address().port;
    console.log('app listening at http://%s:%s', host, port)
});
```
Middleware & Routes

• Terminology:
  – Routes ≈ mount paths
  – At the end of routes: „end points“

• We can define more than one middleware for a route! (see example `03.middlewareChain.js`)

• Middleware delivers static or dynamic content (e.g. files in a directory or REST API call)

• Middleware is in the “middle” between the request and the response.

• Using a middleware function in express:
  ```javascript
  app.use(['path'], [function...], function)
  ```
Router (Express > 4.0)

- Lightweight express app only for routing
- Obtaining a router sub-app:
  ```javascript
  var router = express.Router();
  ```
- `.get` and `.post` routes posses different request objects
  - `req.params` inside a GET request
  - `req.body` inside a POST request
- Tutorials:
  - [http://scotch.io/tutorials/javascript/learn-to-use-the-new-router-in-expressjs-4](http://scotch.io/tutorials/javascript/learn-to-use-the-new-router-in-expressjs-4) (you can use the template and insert the code, if you are at the CIP Pool)
  - [https://www.packtpub.com/books/content/understanding-express-routes](https://www.packtpub.com/books/content/understanding-express-routes)
var express = require('express');
var app = express();
var router = express.Router();

router.get('/',function(req,res){
    res.send('Homepage');
});
router.get('/about',function(req,res){
    res.send('About Page');
});

app.use(router); // or: app.use('/',router);

var server = app.listen(3000, function () {
    var host = server.address().address;
    var port = server.address().port;
    console.log('Example app listening at http://%s:%s', host, port)
});
Extending the template: New route

• Task (5-10 Minutes):
  – add a new route that supports the POST method
  – send a response:
    • JSON object
    • includes all key/value pairs from the request.body object
    • to access the request.body object, make sure to require the body-parser module.
  – test your route with a REST client browser extension, e.g. Postman
Basic routes with JSON response

```javascript
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res) {
  res.json({
    hello: 'world'
  })
});

module.exports = router;
```

```javascript
var index = require('./routes/index');
var stuff = require('./routes/stuff');
var app = express();
app.use('/', index);
app.use('/stuff', stuff);
```
Common Problems

• Error: listen EADDRINUSE
  – usually means that you tried to listen on a port that’s already used
  – solution: quit any running instances of nodejs and restart

• Express dependencies are not resolved
  – npm install express -g
## PHP & NodeJS – advantage comparison

<table>
<thead>
<tr>
<th>PHP</th>
<th>NodeJS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Widely distributed, large community</td>
<td>Fast, performant execution</td>
</tr>
<tr>
<td>Finding a web hoster is easy</td>
<td>Variable access (same variables for multiple requests possible)</td>
</tr>
<tr>
<td>PHP code can be embedded into HTML</td>
<td>Less memory intensive (arguably)</td>
</tr>
<tr>
<td>No restart after code modifications</td>
<td>Shared JS code on Client and Server</td>
</tr>
<tr>
<td>Easier for less dynamic websites</td>
<td>Suitable for websites with many database queries</td>
</tr>
<tr>
<td>Easy to learn for beginners</td>
<td>Traffic management for large websites</td>
</tr>
<tr>
<td></td>
<td>WebSockets allow the subscriber pattern</td>
</tr>
</tbody>
</table>

http://nodecode.de/php-oder-nodejs
Template Engines

• Render templates and values to finished web pages.
• Examples:
  – Jade
  – Handlebars
  – Hogan
  – ... many more
• Usage with Express:
  – in the app module:
    ```javascript
    app.set('views', path.join(__dirname, 'views'));
    app.set('view engine', 'jade');
    ```
  – in the route handlers:
    ```javascript
    router.get('/', function(req, res) {
      res.render('index', { title: 'Express' });
    });
    ```
Jade – Basic Layout

index.jade

extends layout

block content
    h1= title
    p Welcome to #{title}

layout.jade

doctype html
html
    head
        title= title
        link(rel='stylesheet', href='/stylesheets/style.css')
body
    block content
Rendering the template

• Providing values:
  – pass a JS object / JSON
  – access the values from the jade template

• Note: we can easily switch the template engine and this still works!

```javascript
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```
Extending the template: New template

• Task (5 minutes):
  – add a new template file
  – elements:
    • dynamic heading (h1) content
    • div with class container
    • dynamic content inside the container
  – new route /templateTest
    • render the template
    • pass random parameters for the heading and content
Jade: Further details

- Javascript code can be escaped with a leading dash (-), e.g.

```javascript
for(var i=0; i<someCollection.length; i++){
  var element = someCollection[i];
  console.log(element);
  div(class='element')
    div(class='text') = element.text
    div(class='number') = element.someNumber
}
```

- Side notes:
  - there are file watchers in WebStorm that allow generating the HTML files for preview on the fly
  - You can convert your existing HTML files to Jade using this web app: http://html2jade.aaron-powell.com/
CSS Preprocessors

• **Problem:**
  – CSS files become messy and unreadable
  – CSS files don’t match the HTML file’s structure

• **Solution:**
  – CSS Preprocessors. Most famous examples are SASS & LESS
  – Key features:
    • Variables and Operators
    • Nesting
    • Mixins

• Make sure to add file watchers to the SASS or LESS files.
Assignment 7

• Topic: Feedback voting app
• Due in: 2 Weeks
• Due date: 15.12.2014
• Please note: There will be another assignment next week!

Assignment 07

Mehrere kleinere Hausaufgaben auf den Übungsblättern, nicht eine große

Den schwerpunkt zukünftig weiter weg vom LAMP hin zum MEAN stack.

Mehr Codebeispiele auf den Übungsfolien

Ich fände es gut, wenn man in die Übung mehr Theorie einbauen könnte.

Das zweite Übungsblatt hat meiner Meinung nach zu viele Aspekte in einer Aufgabe behandelt.

Es ist nicht gerade motivierend für die nächste Übung, wenn die eigene Lösung besser war als die "Musterlösung" (dann besser - nach meiner Meinung - gar keine Musterlösung vorführen also so)

Die Lösung der MySQL Aufgabe aus Übung 2 wäre cool zu besitzen.
JS Code Retreat Munich

• Code all day long on your project with other JS enthusiasts

• http://www.meetup.com/JavaScript-CodeRetreat/events/209200652/

• Thanks to V. Böhner for pointing out this event
Thanks!

What are your questions?