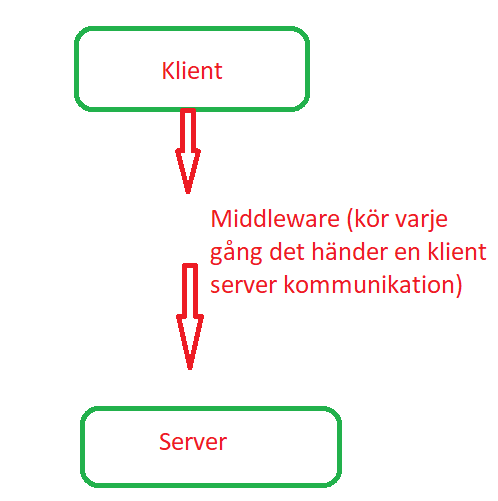
**Express middleware** :



I request response circle använder man middleware funktioner.

Ex middleware:

app.use(express.json());

Hur skriver man egna middleware:

app.use((req,res,next)=>{

// res.send("hello från middleware");

console.log(“hello from middleware”)

next();

})

Build in middleware i express : Middleware övning:

* Skapa en mapp med namnet public och skapa en index.html fil och skriv något.
* I main.js(eller app.js/index.js) filen skriv :

app.use(express.static("public"))

Gå till localhost:8000/index.html

**Resurser:** <https://expressjs.com/en/guide/using-middleware.html>

**Tredjeparts middleware:** <https://expressjs.com/en/guide/using-middleware.html#middleware.third-party>

<https://expressjs.com/en/resources/middleware.html>

Extra :

|  |  |
| --- | --- |
| **Arbeta med configuration:**  <https://www.npmjs.com/package/config>  **Extra:**  **för att läsa data från env filen : dotenv**  [**https://www.npmjs.com/package/dotenv**](https://www.npmjs.com/package/dotenv)   |  | | --- | | **.env fil:**  **// 1. Skapa en .env fil:**  **DB\_ADMIN\_USERNAME=Password**  **DB\_ADMIN\_PASSWORD=Username**  // 2. använd npm och sedan importera dotenv paketet  require('dotenv').config();  // 3.  console.log({  username: process.env.DB\_ADMIN\_USERNAME,  password: process.env.DB\_ADMIN\_PASSWORD,  });  ------  **Lägg till en info:**  **// .env fil**  **ENV=dev**  **OBS: Lägg aldrig komma mellan key:value pairs**  **// i din app**  **if (process.env.ENV === 'dev') console.log(“nu kör appen i dev environment”);** | |

Övningar Middleware:

* Använd Morgan som kan läsa och skriva ut alla request respons i konsolen.

package Länk:

<https://www.npmjs.com/package/morgan/v/1.1.1>

* Skapa en middleware som kan skicka användare meddelandet: “Tack för du använder vår sida!”
* valfria: Istället för console.log använd debug module i din app:

<https://www.npmjs.com/package/debug>

**Diskussion : Vad är skillnaden mellan app.use() och app.post/get osv?**

-----------------------------------lektion 4 slut--------------------------

MVC:

|  |
| --- |
| Står för  Model : Presentera data av applikationen till ex. fetch, save data från databas eller liknande  View : Vad ska visas till användare  Controller: Arbetar i mitten mellan model och view . Logic delen av appen .  Till ex. vilken data ska visas till vilken användare. |

Template engine:

Rendera data(dynamiska data) med html fil. Det finns många alternativa optioner på marknaden. Vi kommer att använda ejs

<https://expressjs.com/en/guide/using-template-engines.html>

Ejs

<https://www.youtube.com/watch?v=VM-2xSaDxJc>

npm i --save ejs

Ejs docs:

<https://ejs.co/#docs>

STEG 1: Skapa views mapp

Skapa en mapp i views och skapa en fil med namnet : admin.ejs

Steg 2:

const express = require("express");

const app = express();

//app.use middleware

//express.json() till vanligt js object

app.use(express.json())

//app.use(express.static("public"))

//Säger till vilken mapp express skulle kolla

// för views filerna

app.set("views", "views")

//registerar vilken typ template engine

// den skulle använda

app.set("view engine", "ejs")

Steg 3: Rendera ejs.

app.get("/admin", (req, res)=>{

res.render("admin", { title:"Min sida"} )

})

Gå till: [http://localhost:3004/admin](http://localhost:3004/admin/add-product)

Rendering dynamisk data med ejs:

Vi ska ändra title namnet för admin.ejs

Då kan vi göra så:

res.render("admin", {title:"Adminsite"})

och i ejs filen:

<title><%= title %></title>

Strukturen :

<%= dynamiska data %>

för att skriva js kod:

<% js kod %>

Loopa igenom en array:

Till ex. om vi har en product array :

<div>

<% product.forEach( (result)=> { %>

<li> <%= result.price %> </li>

<% }) %>

</div>

Övningar ejs:

* Förberedda projekt för express app
* Installera ejs template engine
* Skapa en views mapp i projektet
* Skapa en admin.ejs fil i views mappen
* Skapa en main.js fil för att hantera express server
* registrera med hjälp av app.set() metoden views mappen och view engine i din main.js filen
* Skapa en Array av produkter i main.js filen
* Skicka Arrayen till klienten när du renderer admin sidan i app.get(“/product”, ....) routen
* Loopa igenom arrayen i product.ejs filen och visa alla data till klient

till ex. localhost:8000/product urln

Hantera User input data:

Man behöver body parser middleware: npm install body-parser

Lägg till koden i din fil:

var bodyParser = require('body-parser')

// parse application/x-www-form-urlencoded

app.use(bodyParser.urlencoded({ extended: false }))

const product = [ {product\_id:1, price:100 },

{product\_id:2, price:200 },

{product\_id:3, price:300 }]

app.get("/admin", (req, res)=>{

res.render("admin", {product: product, title:"Min sida"} )

})

app.get("/products", (req, res)=> {

res.render("product", {product:product})

})

app.get("/add-product", (req, res)=> {

res.render("addProduct")

})

app.post("/add-product", (req, res)=> {

product.push( {product\_id: req.body.product\_id});

res.redirect("/products")

})

Två ejs filer som kommer att passera data mellan varandra:

till ex: product sidan kommer att ha produkt\_id som tillagts av add-product sidan:

add-product sida:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Add product sidan </title>

</head>

<body>

<form action="/add-product" method="post">

<input type="text" name="product\_id">

<button type="submit">Add product</button>

</form>

</body>

</html>

Product sida:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Produkt sida</title>

</head>

<body>

<div>

<% product.forEach( (result)=> { %>

<li> <%= result.product\_id %> </li>

<% }) %>

</div>

</body>

</html>

Diskussion/utmaning/Övning:

1. Hur man kan passera data mellan olika filer?

Till ex. om vi vill visa vilka produkter admin har tillagts ska visas i product sidan.

1. Gör övningen :

* Utgå från den ejs övningen:
* Skapa en till ejs fil med namnet admin.ejs
* Skapa ett formulär med två input (en för text och en för siffra) i admin.ejs filen
* formuläret kommer att ha post metoden och action i samma route som admin post routen i din main.js filen

till ex. action= ‘/admin’ om din app.post(“/admin”,...) har ‘/admin’.

* app.post() routen ska ta data från input och lagra i product listan
* visa product listan i product routen.

OBS: glöm ej bodyparser middleware och app.get() för admin sidan.

1. I vår appen vilka är model, view och controller?
2. Hur man kan loopa i ejs filen? <https://ejs.co/#docs>

---------------------------------------------------------

Extra:

Läs om ejs layout : <https://blog.soshace.com/express-js-ejs-templating/>

--------------------- slut lektion 5 ---------------------

Lektion : 6- Delete and update :-

Delete and update with client: (with ejs)

React, Vue js (eller liknande) brukar ha ajax, axios eller fetch för att kunna skicka CRUD request men HTML stödjer bara post and get dvs Create and read. Vi kommer att använda post and get för att kunna hantera både update and delete operationer.

i din main.js filen :

app.get("/product/:id", (req, res)=>{

console.log(req.params.id)

res.render("productEdit", {products, id:req.params.id})

})

app.post("/product/productEdit", (req, res)=>{

console.log(req.body.product\_name)

console.log("request id " , req.body)

//edit

// console.log(products)

products[Number(req.body.id)].product\_name

= req.body.product\_name;

products[Number(req.body.id)].id= req.body.id;

products[Number(req.body.id)].price = req.body.price

res.render("product", {products})

})

app.get("/productDelete/:id", (req, res)=>{

console.log("delete params" , req.params.id)

products.splice( req.params.id, 1)

res.send("product deleted")

})

och skapa en till ejs sida med namnet productEdit.ejs

<form action="/product/productEdit" method="post">

<input type="text" name="product\_name" id="" value="<%= products[id].product\_name %>">

<input type="text" name="id" id="" value="<%= products[id].id %>" >

<input type="text" name="price" id="" value="<%= products[id].price %>">

<button type="submit">Edit</button>

</form>

och i product.ejs filen:

<% for(let i= 0; i<products.length; i++){ %>

<div> <%= products[i].product\_name %>

<%= products[i].price %>

<%= products[i].id %>

<a href="/product/<%= products[i].id %>" >Edit</a>

<a href="/productDelete/<%= products[i].id %>" >Delete</a>

</div>

<% } %>

Övning:

Skapa en todo list med express ejs (en del av första inlämning)

…….

----------------------------------------

Omskrivna kod med router i en annan module:

Code ejs ex med router :

const express = require('express');

const app = express();

//kollar vilken mapp den ska referera för view.

app.set('views', 'views');

//sätter view engine till ejs

app.set('view engine', 'ejs')

const adminRoutes = require('./routes/admin');

const productRoutes = require('./routes/product');

app.use("/admin" , adminRoutes);

app.use(productRoutes);

const port = process.env.PORT || 3004;

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

const express = require('express');

const router = express.Router();

router.get('/add-product', (req, res, next) => {

res.render("admin")

});

router.post('/product', (req, res, next) => {

console.log(req.body);

res.redirect('/');

});

module.exports = router;