NODE

SERVER SIDE RENDERING CRUD USING EXPRESS

Express Hello World

```
const express = require('express')
const app = express()
const port = 3000
app.get('/', (req, res) => {
  res.send('Hello World!')
})
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Basic Routing

app.METHOD(PATH, HANDLER)

- app is an instance of express.
- METHOD is an HTTP request method, in lowercase
- PATH is a path on the server.
- HANDLER is the function executed when the route is matched.

Example Code for routes

```
app.get('/', (req, res) => {
  res.send('Hello World!')
})
app.post('/', (req, res) => {
  res.send('Got a POST request')
})
```

Serving Static Files

app.use(express.static('public’))

- Now, you can load the files that are in the public directory:
  - http://localhost:3000/images/kitten.jpg
  - http://localhost:3000/css/style.css
  - http://localhost:3000/js/app.js
  - http://localhost:3000/images/bg.png
  - http://localhost:3000/hello.html

EJS

EJS is a simple templating language that lets you generate HTML markup with plain JavaScript

- Fast compilation and rendering
- Simple template tags: <% %>
- Custom delimiters (e.g., use [? ?] instead of <% %>)
- Sub-template includes
- Ships with CLI
- Both server JS and browser support
- Static caching of intermediate JavaScript
- Static caching of templates
- Complies with the Express view system

EJS (Embeded JS) user variable must be passed

```
<% if (user) { %>
 <h2><%= user.name %></h2>
<% } %>
```

<%= EJS %>

Embedded JavaScript templating.

EJS Tags

<% 'Scriptlet' tag, for control-flow, no output

<%_ ‘Whitespace Slurping’ Scriptlet tag, strips all whitespace before it

<%= Outputs the value into the template (HTML escaped)

<%- Outputs the unescaped value into the template

<%# Comment tag, no execution, no output

<%% Outputs a literal '<%'

%> Plain ending tag

-%> Trim-mode ('newline slurp') tag, trims following newline

_%> ‘Whitespace Slurping’ ending tag, removes all whitespace after it

Include EJS into another

```
<ul>
    <%- include('user/show', {user: user}); %>
</ul>
```

Layouts in EJS

```
<%- include('header'); -%>
<h1>
  Title
</h1>
<p>
  My page
</p>
<%- include('footer'); -%>
```

express-ejs-layouts
npm install express-ejs-layouts

```
var expressLayouts = require('express-ejs-layouts');

var app = express();

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

app.use(expressLayouts);
```

Set custom default layout

By default 'layout.ejs' is used.

If you want to specify your custom layout (e.g. 'layouts/layout.ejs'), just set layout property in express app settings.

```
app.set('layout', 'layouts/layout’);

app.get('/', function(req, res) { // no layout
  res.render('the-view', { layout: false });
);
```

Where Child View Will be rendered

Place below line in layout.ejs file

```
<%-body%>
```

Optional Sections

LAYOUT.EJS

```
1

<%-defineContent('a')%>

2

<%-defineContent('b')%>

3
```

VIEW.EJS

```
<%- contentFor('a') %>

1.5
```

Will render

```
1

1.5

2

3
```

Passing Variables to View

```
app.get('/', function(req, res) {
  var locals = {
    title: 'Page Title',
    description: 'Page Description',
    header: 'Page Header'
  };
  res.render('the-view', locals);
});
```

Using EJS with Express

```
app.set('view engine', 'ejs');
app.get('/', (req, res) => {
  res.render('index', {foo: 'FOO'});
});
```

(This assumes a views directory containing an index.ejs page.)

// middleware to set variables to all views

```
app.use((req, res, next) => {
  // Set variables to be available in all views
  res.locals.siteTitle = 'Todo App';
  next();
});
```

Express Server Side CRUD

npm install express mongoose ejs

File Structure

```
- /models
  - todo.js
- /views
  - index.ejs
- app.js
```

models/todos.js

```
const mongoose = require('mongoose');
const todoSchema = new mongoose.Schema({
  title: String,
  completed: {
    type: Boolean,
    default: false
  }
});
module.exports = mongoose.model('Todo', todoSchema);
```

app.js

```
const express = require('express');
const mongoose = require('mongoose');
const Todo = require('./models/todo');

const app = express();
```

Set up view engine and mongoose

```
app.set('view engine', 'ejs');
app.use(express.urlencoded({ extended: true }));
mongoose.connect('mongodb://localhost:27017/todo-app', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => {
  console.log('Connected to MongoDB');
}).catch(err => {
  console.error('Error connecting to MongoDB:', err.message);
});
```

// Index route - list all todos

```
app.get('/', async (req, res) => {
  try {
    const todos = await Todo.find();
    res.render('index', { todos });
  } catch (err) {
    console.error('Error fetching todos:', err.message);
    res.status(500).send('Server Error');
  }
});
```

views/index.ejs

```
----
<ul>
    <% todos.forEach(todo => { %>
      <li><%= todo.title %></li>
      <a href="/todos/<%= todo._id %>/delete">Delete</a>
    <% }); %>
  </ul>
----
```

// Start the server

```
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Render Form

```
app.get('/todos/new', (req, res) => {
  res.render('new');
});
```

new.ejs

```
<h1>Add New Todo</h1>
  <form action="/todos" method="POST">
    <input type="text" name="title" placeholder="Todo Title"
required>
    <button type="submit">Add Todo</button>
  </form>
```

Form will be posted to this route

```
app.post('/todos', async (req, res) => {
  const { title } = req.body;
  try {
    await Todo.create({ title });
    res.redirect('/');
  } catch (err) {
    console.error('Error creating todo:', err.message);
    res.status(500).send('Server Error');
  }
});
```

href="/todos/<%= todo._id %>/delete"

```
app.get('/todos/:id/delete', async (req, res) => {
  const { id } = req.params;
  try {
    await Todo.findByIdAndDelete(id);
    res.redirect('/');
  } catch (err) {
    console.error('Error deleting todo:', err.message);
    res.status(500).send('Server Error');
  }
});
```

href="/todos/<%= todo._id %>/edit"

```
app.get('/todos/:id/edit', async (req, res) => {
  const { id } = req.params;
  try {
    const todo = await Todo.findById(id);
    res.render('edit', { todo });
  } catch (err) {
    console.error('Error fetching todo:', err.message);
    res.status(500).send('Server Error');
  }
});
```

edit.ejs

```
<h1>Edit Todo</h1>
  <form action="/todos/<%= todo._id %>/edit" method="POST">
    <input type="hidden" name="_method" value="PUT">
    <input type="text" name="title" value="<%= todo.title %>"
required>
    <button type="submit">Update</button>
  </form>
```

Handle edit form submission

```
app.put('/todos/:id/edit', async (req, res) => {
  const { id } = req.params;
  const { title } = req.body;
  try {
    await Todo.findByIdAndUpdate(id, { title });
    res.redirect('/');
  } catch (err) {
    console.error('Error updating todo:', err.message);
    res.status(500).send('Server Error');
  }
});
```