Backendprogrammering 1

MER EJS

Utbildare: Mikael Olsson

mikael.olsson@emmio.se
076-174 90 43

NACKADEMIN

Express application generator

- Hjälper oss att generera ett ”skelett” för ett projekt.
- `npm install -g express-generator`

```
→  03 git:(master) ✗ express --help

  Usage: express [options] [dir]

  Options:

        --version        output the version number
    -e, --ejs            add ejs engine support
        --pug            add pug engine support
        --hbs            add handlebars engine support
    -H, --hogan          add hogan.js engine support
    -v, --view <engine>  add view <engine> support (dust|ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
        --no-view        use static html instead of view engine
    -c, --css <engine>   add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
        --git            add .gitignore
    -f, --force          force on non-empty directory
    -h, --help           output usage information
```

Express application generator

- Skapa projekt med de options du vill, t ex:

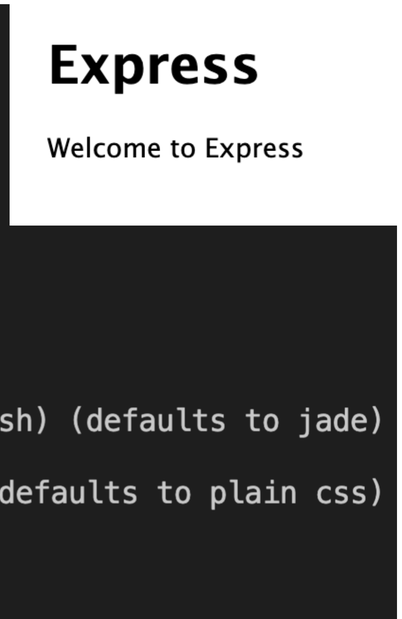
- `express --view=ejs --css=sass --git my_proj`
- `cd my_proj`
- `npm install`
- `DEBUG=myapp:* npm start`

```
→  03 git:(master) ✗ express --help

  Usage: express [options] [dir]

  Options:

        --version        output the version number
    -e, --ejs            add ejs engine support
        --pug            add pug engine support
        --hbs            add handlebars engine support
    -H, --hogan          add hogan.js engine support
    -v, --view <engine>  add view <engine> support (dust|ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
        --no-view        use static html instead of view engine
    -c, --css <engine>   add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
        --git            add .gitignore
    -f, --force          force on non-empty directory
    -h, --help           output usage information
```



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

EJS - Includes

- Includes are relative to the template with the `include` call. (This requires the 'filename' option.) For example if you have "./views/users.ejs" and "./views/user/show.ejs" you would use `<%- include('user/show'); %>`.

- You'll likely want to use the raw output tag (`<%-`) with your include to avoid double-escaping the HTML output.

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

EJS - Layouts

- EJS does not specifically support blocks, but layouts can be implemented by including blocks like headers and footers, like so:

```
<%- include('header'); -%>

<h1>Title</h1>
<p>My page</p>

<%- include('footer'); -%>
```

Uppgift

- Skapa en enkel sida med en header, en footer och en kontakta oss-”modul”. Styling är inte det viktiga i denna kurs, men visa var era block finns med en bakgrundsfärg, en border eller liknande.

Session

- En session är lite som en cookie. Kan användas för att spara temporära data på servern, t ex om användaren är inloggad eller inte.

- https://www.npmjs.com/package/cookie-session

- Låt oss kolla in ett exempel.

cookieSession

- We can generate a session using the following command:

```
app.use(session({
  secret: 'veryimportantsecret',
}))
```

- The secret is used to sign the cookie using the cookie-signature library. Cookies are signed using Hmac-sha256 and converted to a base64 string. We can have multiple secrets as an array. The first secret will be used to sign the cookie. The rest will be used in verification.

```
app.use(session({
  secret: ['veryimportantsecret', 'notsoimportantsecret',
'highlyprobablysecret'],
}))
```

https://blog.jscrambler.com/best-practices-for-secure-session-management-in-node/

Redirect

- Ett sätt att hantera olika typer av requests är att göra en redirect när man är klar.

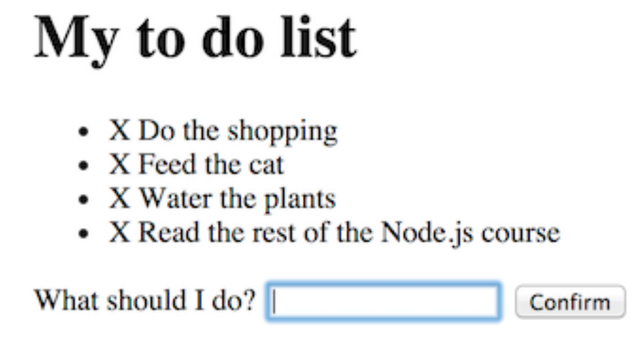
- Säg att vi ska ta emot ett post-request som ska skapa en ny användare och sedan återgå till listan av användare.

- ```
  app.post('/user', (request, response) => {
    // Process the data received in request.body
    response.redirect('/');
  });
  ```
  ```

Todo-list

- *Uppgift*: Skapa en todo-lista. Skapa applikationen mha express-generate.
- We can add elements to the to do list via the form.
- We can delete items by clicking on the crosses in the list.
- The list is stored in the visitor's session. If someone else connects to the site, they will have their own list.
- In theory, we can associate a route to each of these features:

- `/todo`: list of tasks.
- `/todo/add`: add a task.
- `/todo/delete/:id`: delete task n°id.



Todo-list

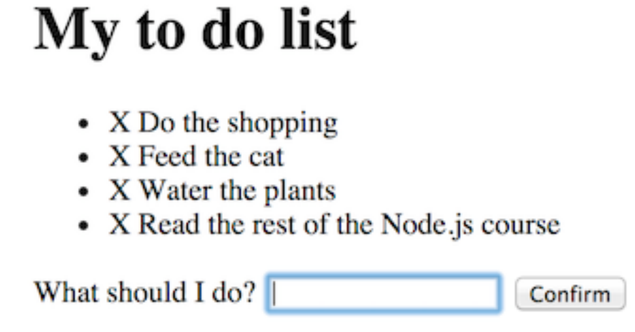
- *Uppgift*: Skapa en todo-lista. Skapa applikationen mha express-generate.

- Hur ska vi låta användaren redigera uppgifter?

  - Vi kan sätta upp en route som ska rendera alternativet i textrutan.

  - Vi kan också låta ett javascript på klientsidan fylla i värdet man klickar på i textrutan.

  - I båda fallen behöver vi ta hand om svaret och uppdatera våra uppgifter.



Fler små projekt

- Skapa en gästbok. Längst upp ska det finnas ett formulär där användaren får fylla i namn och meddelande.
- Låt användaren skicka in meddelandet.
- Under formuläret ska alla inskickade meddelanden visas med namn, meddelande och tid som meddelandet skickades.

Fler små projekt

- Skapa en blog. Vi har inga databaser än, men vi kan skapa en blogpost-klass och skapa några objekt av den. Låt användaren CRUD:a bloggposter.

    - Bonus-uppgift: Lägg till möjligheten att CRUD:a kommentarer till varje inlägg.

Fler små projekt

- Skapa en frågesport. Om du vill kan du hämta frågor från https://quizapi.io/, eller så kan du skapa ett eget system för att hantera frågor, t ex en lokal json-fil eller skapa en frågeklass och skapa ett par objekt med frågor.
    - Skapa en template som skriver ut frågorna och svaren (t ex med radioknappar) i ett formulär.
    - Skriv en route som tar emot formuläret och skriver ut antalet rätta frågor.