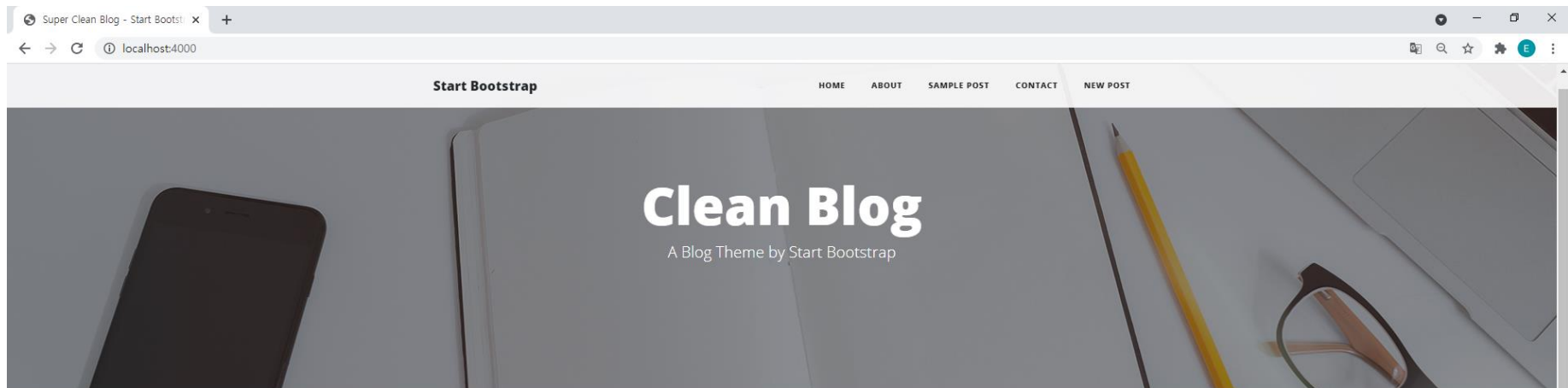


NodeJS 응용



Sweden's IVF programme

Fertility campaigners in Sweden say healthcare officials have broken a promise to help more single women get pregnant. In April 2016, Swedish women without partners were given the same rights as couples to access state-funded fertility treatments including IVF. But waiting times are so long in one part of the country that women have been told it's too late to join the list once they turn 37.

on Sun May 02 2021

North Korea accuses Joe Biden

North Korea has hit out at the Biden administration as it prepares to unveil its strategy for dealing with Pyongyang and its nuclear programme. The foreign ministry said recent comments out of Washington showed President Joe Biden was intent on maintaining a "hostile policy". Earlier this week, Mr Biden called North Korea's nuclear

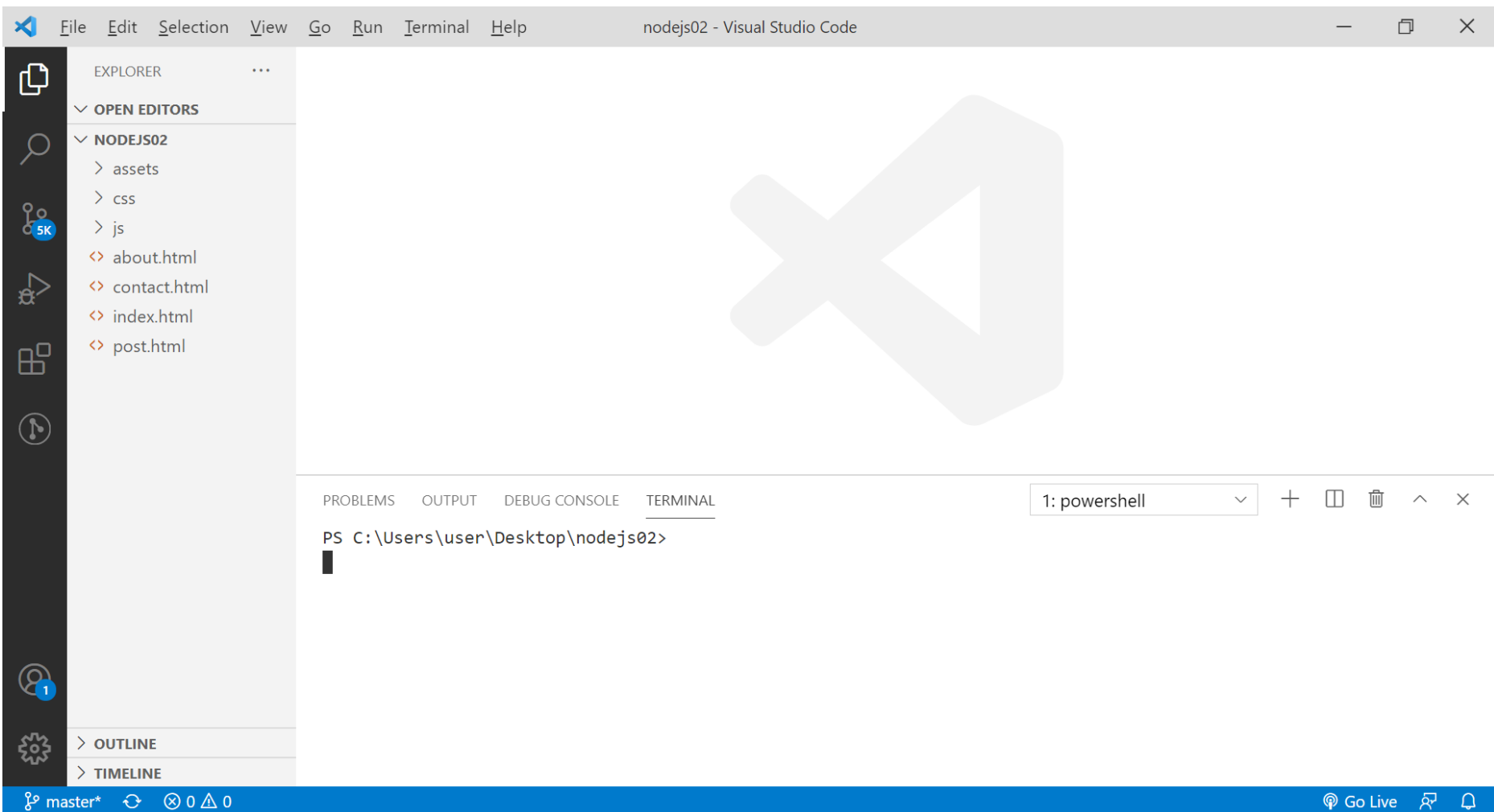
Downloading the Blog Template

- ◆ Extract it to a new folder for this project

The screenshot shows the Start Bootstrap Clean Blog theme page. The page features a header with the Start Bootstrap logo and navigation links. The main content area displays the theme's name, a description, and a featured article. A sidebar on the right contains a promotional banner for Creative Tim's Premium Bootstrap UI Kits and Dashboards, along with buttons for 'Free Download' and 'Upgrade to Pro'.

Below the webpage, a file explorer window shows the directory structure of the downloaded theme:

이름	수정된 날짜	유형	크기
assets	2021-04-30 오전 10:36	파일 폴더	
css	2021-04-30 오전 10:36	파일 폴더	
js	2021-04-30 오전 10:36	파일 폴더	
about	2021-04-01 오후 4:06	HTML 파일	6KB
contact	2021-04-01 오후 4:06	HTML 파일	9KB
index	2021-04-01 오후 4:06	HTML 파일	8KB
post	2021-04-01 오후 4:06	HTML 파일	10KB



◆ **> npm init**

◆ **> npm install express**

◆ **index.js :**

```
const express = require('express')
const app = new express()

app.listen(4000, ()=>{
  console.log('App listening on port 4000')
})
```

◆ **Start server:**

```
PS C:\Users\user\Desktop\nodejs02> node index.js
App listening on port 4000
```

Automatic Server Restart

- ◆ **Nodemon package**

- Automatically detects code changes and restart the server

- ◆ **> npm install nodemon --save-dev**

- ◆ **In package.json, go to "scripts" and make the following change:**

```
"scripts": {  
  | "start": "nodemon index.js"  
  },  
}
```

- ◆ **Starting our app :**

- > npm start**

Serving Static files

- ◆ Create a new folder :

- public

- ◆ In index.js :

```
const express = require('express')

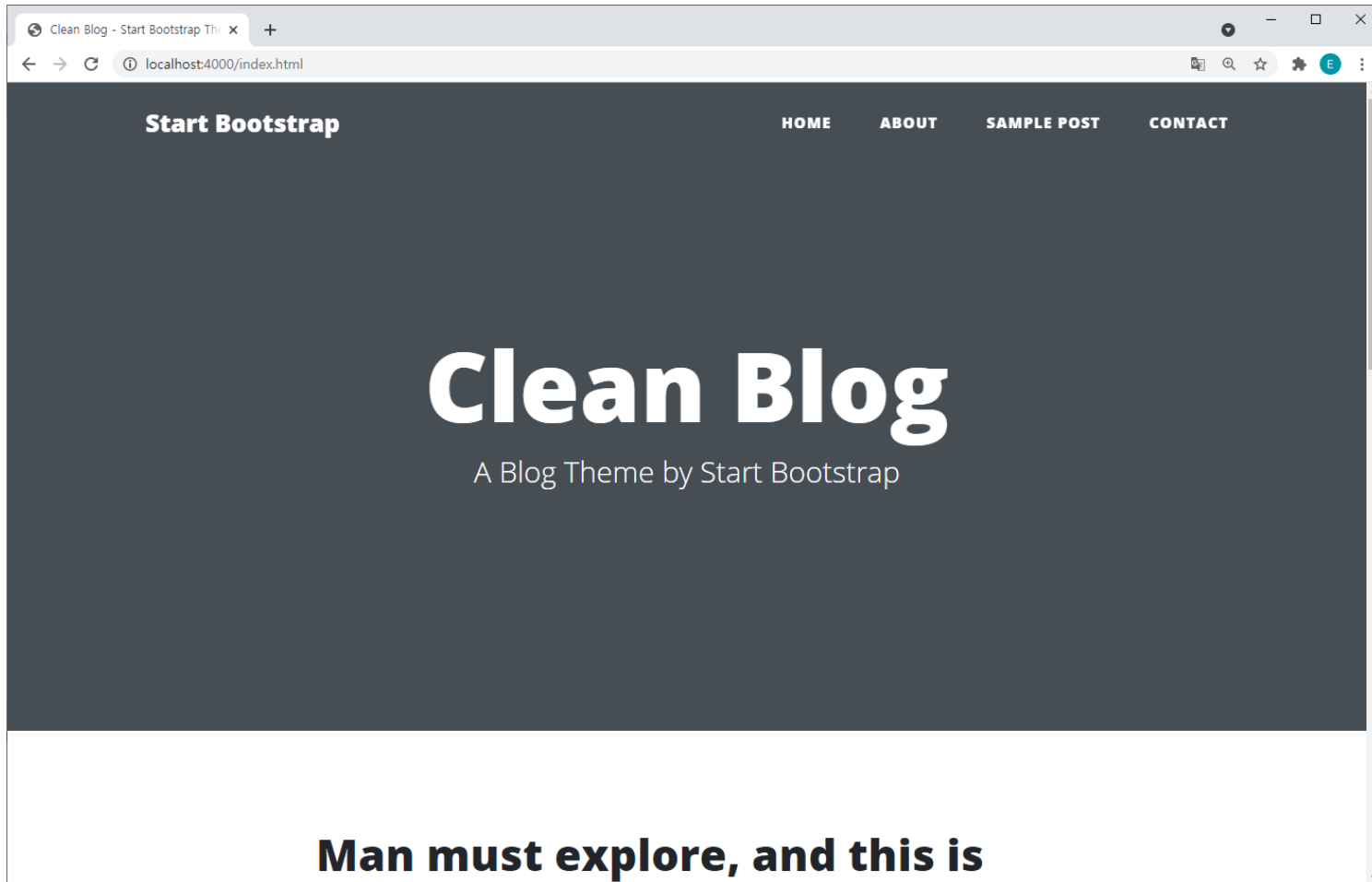
const app = new express()
app.use(express.static('public'))

app.listen(4000, ()=>{
  console.log('App listening on port 4000')
})
```

-
- ◆ Move the downloaded template files into public:
 - index.html
 - about.html
 - contact.html
 - post.html

 - ◆ Move the downloaded template folders into public.
 - css
 - img
 - js

◆ **http://localhost:4000/**



Creating Page Routes

← → ↻ ⓘ localhost:4000/about

Start Bootstrap

About

- ◆ Create a **pages** folder
 - Move all the HTML files from public
- ◆ **Setup the routes for the various pages:**

```
const express = require('express')
const path = require('path')

const app = new express()
...

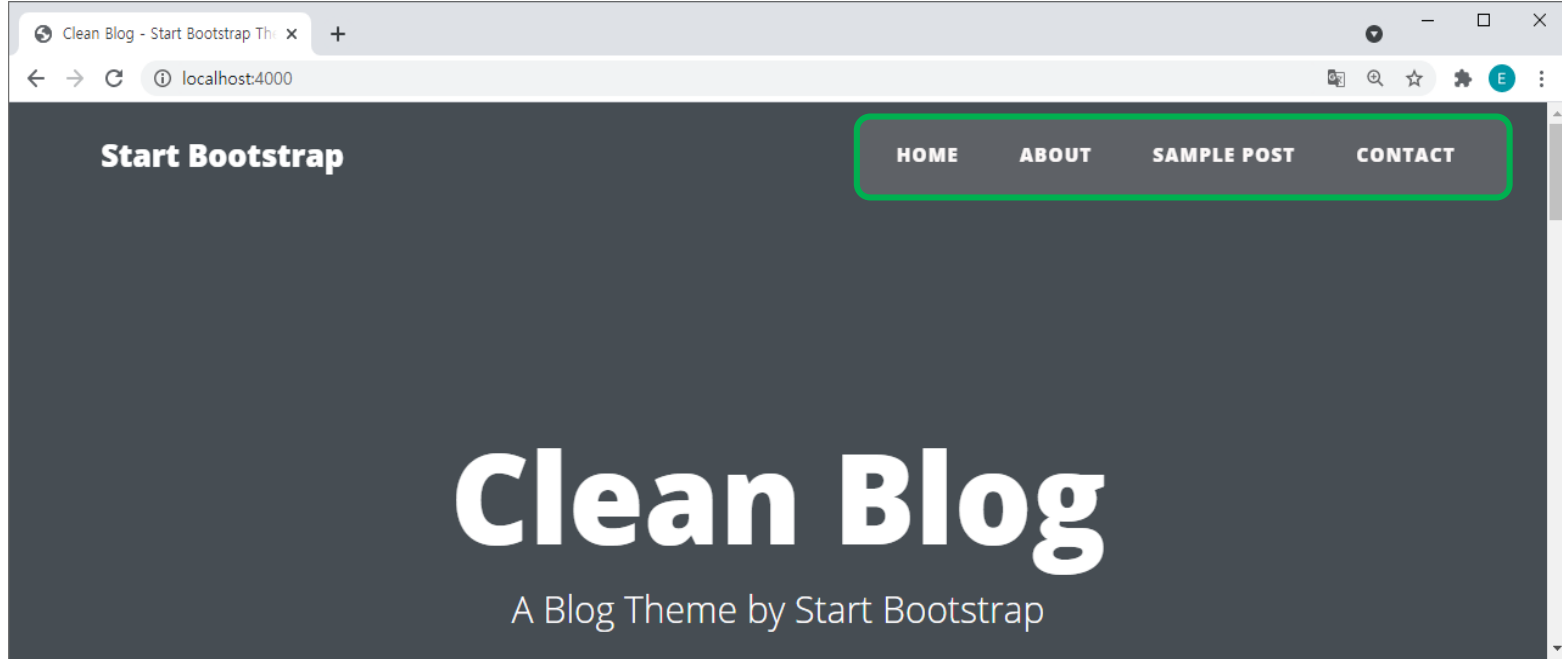
app.get('/', (req, res) => {
  res.sendFile(path.resolve(__dirname, 'pages/index.html'))
})
```

↳ 절대경로 필요

```
app.get('/about', (req, res) => {  
  res.sendFile(path.resolve(__dirname, 'pages/about.html'))  
})  
  
app.get('/contact', (req, res) => {  
  res.sendFile(path.resolve(__dirname, 'pages/contact.html'))  
})  
  
app.get('/post', (req, res) => {  
  res.sendFile(path.resolve(__dirname, 'pages/post.html'))  
})
```

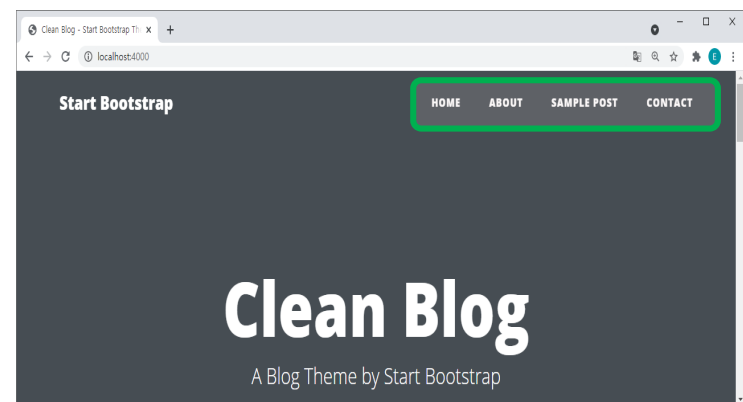
이 페이지를 response로 보냄

Links in index. html



◆ Index.html, about.html, contact.html and post.html

```
<li class="nav-item">
  <a class="nav-link" href="/">Home</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="/about">About</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="/post">Sample Post</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="/contact">Contact</a>
</li>
```



TEMPLATING ENGINES

Templating Engines

- ◆ **EJS'(Embedded JavaScript)**

- Generate HTML with plain JavaScript in script tags
- i.e. `<%=...%>`.

- ◆ **Install:**

- > npm install ejs**

♦ To use EJS in index.js

```
const express = require('express')  
const path = require('path')  
  
const app = new express()  
const ejs = require('ejs')  
app.set('view engine', 'ejs')
```

```
app.get('/', (req, res) => {  
  res.sendFile(path.resolve(__dirname, 'pages/index.html'))  
})
```



Now with EJS, we do the following,

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

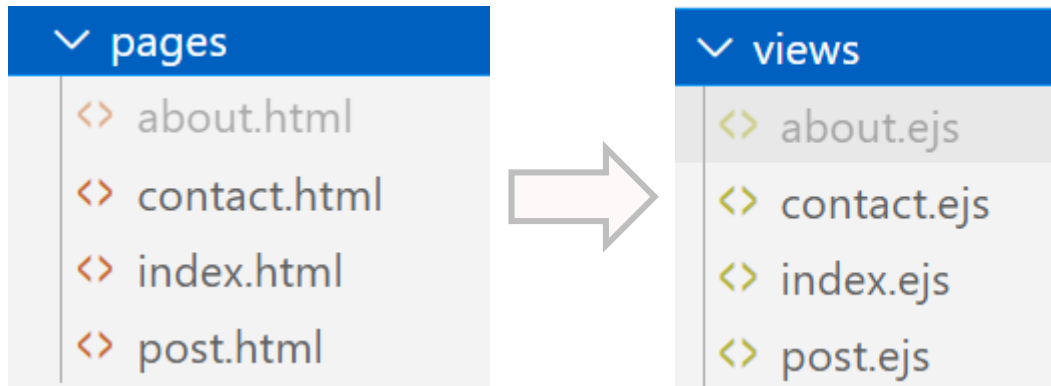
```
app.get('/', (req, res) => {
  res.render('index');
})

app.get('/about', (req, res) => {
  //res.sendFile(path.resolve(__dirname, 'pages/about.html'))
  res.render('about');
})

app.get('/contact', (req, res) => {
  //res.sendFile(path.resolve(__dirname, 'pages/contact.html'))
  res.render('contact');
})

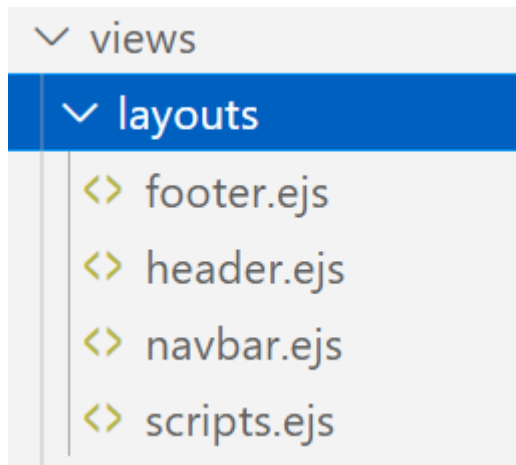
app.get('/post', (req, res) => {
  //res.sendFile(path.resolve(__dirname, 'pages/post.html'))
  res.render('post')
})
```

- ◆ **res.render('index')** will look in a **'views'** folder for the file index.ejs.



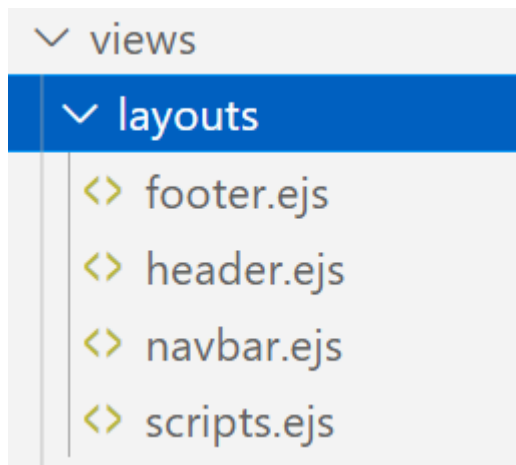
Layouts

- ◆ In **index.ejs**, `<head>`, `<nav>`, `<footer>` and `<script>` elements that also appear in the other views.
- ◆ **Layout file**
 - Extract these portion of common HTML code into separate files
 - **Include** the files that need them



◆ **Extract** (In index.js)

- `<head>` HTML into header.ejs
- `<nav>` HTML into navbar.ejs
- `<footer>` HTML into footer.ejs
- `<script>` elements into scripts.ejs



◆ Include the layout files : index.ejs, about.ejs, contact.ejs and post.ejs files

```
<!DOCTYPE html>
<html lang="en">

<%- include('layouts/header'); -%>

<body>
  <%- include('layouts/navbar'); -%>
  <!-- Page Header -->
  <header class="masthead" style="background-image: url('img/home-bg.jpg')">
    <div class="overlay"></div>
    <div class="container">
      <div class="row">
        <div class="col-lg-8 col-md-10 mx-auto">
          <div class="site-heading">
            <h1>Clean Blog</h1>
            <span class="subheading">A Blog Theme by Start Bootstrap</span>
          </div>
        </div>
      </div>
    </div>
  </header>

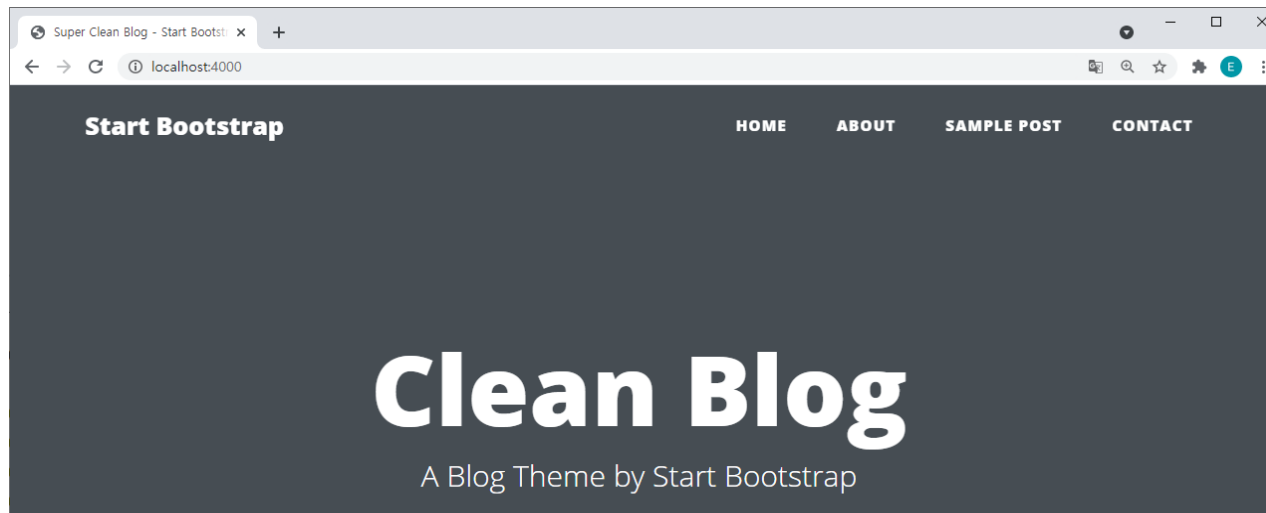
  <!-- Main Content -->
  ...
  <hr>

  <%- include('layouts/footer'); -%>
  <%- include('layouts/scripts'); -%>

</body>
</html>
```

- ♦ if you want to change the title:
change **header.ejs**

```
<head>  
  ...  
  
  <title>Super Clean Blog - Start Bootstrap Theme</title>  
  ...
```



INTRODUCTION TO MONGODB

Architecture of MongoDB

◆ MongoDB

- Works on concept of **collection** and **document**

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field

한개의
데이터
(행단위의 data)

Database

→ Products collection

→ Product document

```
{  
  price: 26,  
  title: "Learning Node",  
  description: "Top Notch Development book",  
  expiry date: 27-3-2020  
}
```

→ Product document

...

→ Users collection

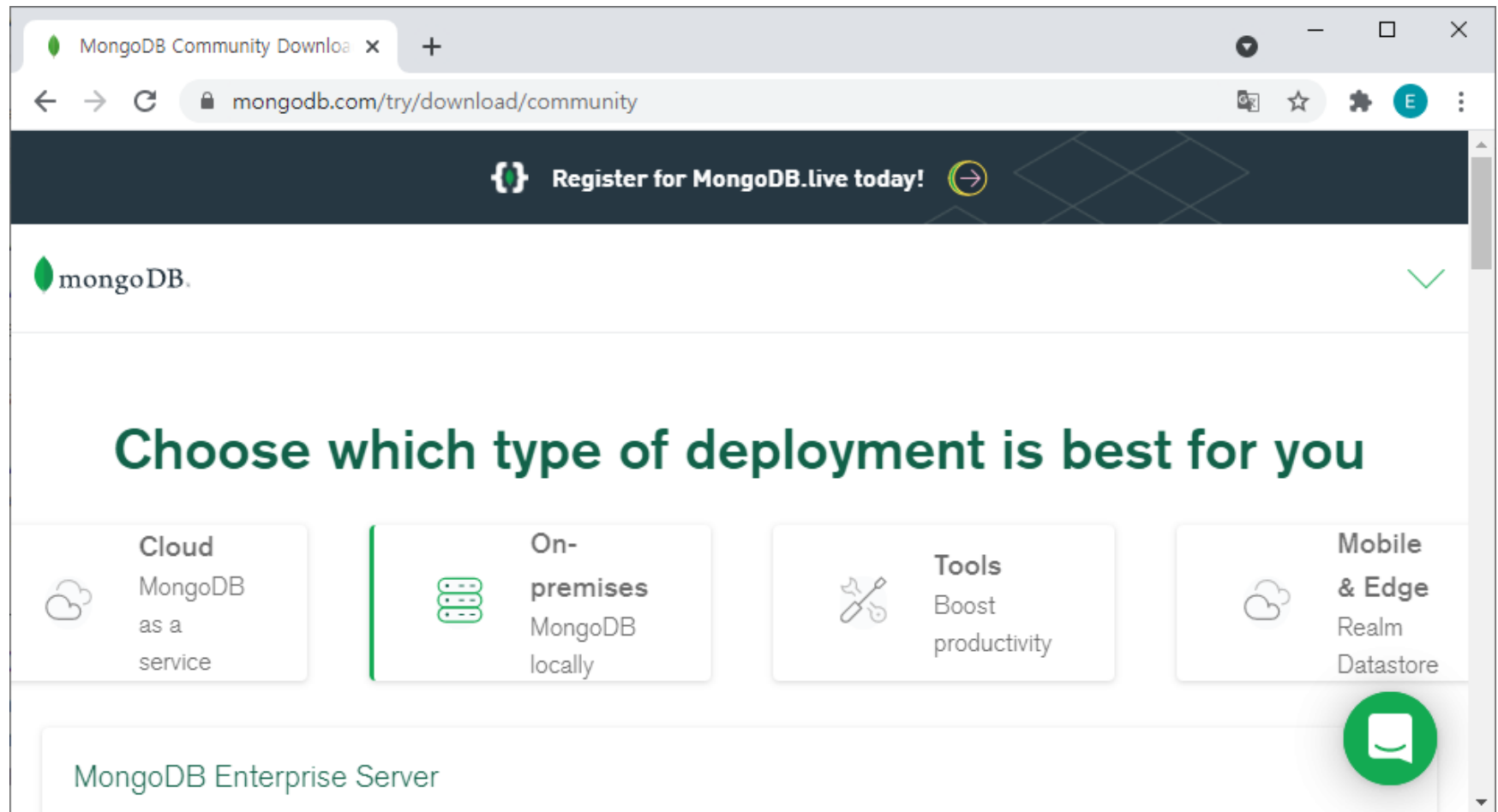
→ User document

```
{  
  username: "123xyz",  
  contact:  
    {  
      phone: "123-456-7890",  
      email: "xyz@example.com"  
    }  
}
```

→ User document

...

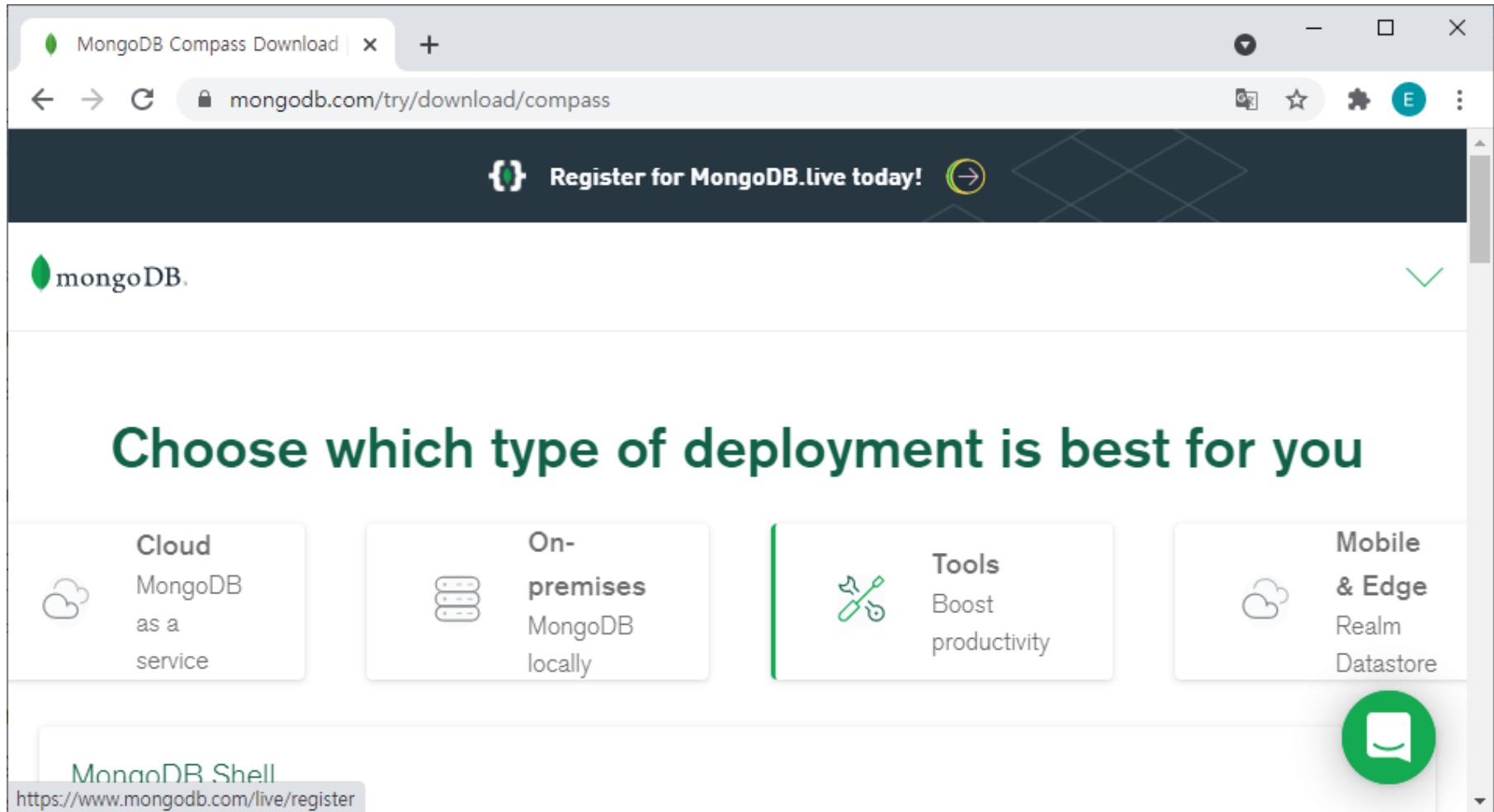
Installing MongoDB



Running MongoDB

- ♦ **C:\Program Files\MongoDB\Server\4.4\bin> mongod**

Installing MongoDB ComPass



Mongoose

- ♦ To talk to MongoDB from Node, we need a library
- ♦ Mongoose is an officially supported Node.js package (<https://www.npmjs.com/package/mongoose>)
- ♦ `> npm install mongoose`

- ◆ Add the following code: **index.js**,

```
const mongoose = require('mongoose');  
  
mongoose.connect('mongodb://localhost/my_database', {useNewUrlParser:  
true})
```


Defining a Model

- ♦ Create a directory : **models**
- ♦ Create a model file in it: **BlogPost.js**

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema;

const BlogPostSchema = new Schema({
  title: String,
  body: String
});

...
const BlogPost = mongoose.model('BlogPost', BlogPostSchema);

module.exports = BlogPost
```

CRUD Operations with Mongoose Models

♦ Create a file: **test.js**

```
const mongoose = require('mongoose')

const BlogPost = require('./models/BlogPost')

mongoose.connect('mongodb://localhost/my_database', {useNewUrlParser:
true});

BlogPost.create({
  title: 'The Mythbuster's Guide to Saving Money on Energy Bills',
  body: 'If you have been here a long time, you might remember when I
went on ITV Tonight to dispense a masterclass in saving money on energy
bills. Energy-saving is one of my favourite money topics, because once
you get past the boring bullet-point lists, a whole new world of thrifty
nerdery opens up. You know those bullet-point lists. You start spotting
them everything at this time of year. They go like this:'
}, (error, blogpost) =>{
  console.log(error, blogpost)
})
```

◆ > node test.js

- Notice that there is an **additional field** **_id**. a unique id provided by MongoDB for every document.

```
null { _id: 5cb436980b33147489eadfbb,  
  title: 'The Mythbuster's Guide to Saving Money on Energy Bills',  
  body:  
    'If you have been here a long time, you might remember when I went on  
    ITV Tonight to dispense a masterclass in saving money on energy bills.  
    Energy-saving is one of my favourite money topics, because once you get  
    past the boring bullet-point lists, a whole new world of thrifty nerdery  
    opens up. You know those bullet-point lists. You start spotting them  
    everything at this time of year. They go like this:',  
    __v: 0 }
```

- ◆ **To see the data visually in MongoDB:**

[illegible]

Reading Data from MongoDB using Mongoose

- ♦ To select **all documents** in BlogPosts collection:

```
BlogPost.find({}, (error, blogspot) =>{  
    console.log(error, blogspot)  
})
```

- ♦ To find all documents in BlogPosts collection **with a particular title**:

```
BlogPost.find({  
    title: 'The Mythbuster's Guide to Saving Money on Energy Bills'  
}, (error, blogspot) =>{  
    console.log(error, blogspot)  
})
```

- ♦ To find all documents in BlogPosts collection **with 'The' in the title**

```
BlogPost.find({  
  title:/The/}, (error, blogspot) =>{  
  console.log(error, blogspot)  
})
```

- ♦ To get single database documents, i.e. to retrieve single documents **with unique id,**

```
var id = "5cb436980b33147489eadfbb";  
  
BlogPost.findById(id, (error, blogspot) =>{  
  console.log(error, blogspot)  
})
```

Updating Records

- ♦ To update a record :

```
var id = "5cb436980b33147489eadfbb";  
BlogPost.findByIdAndUpdate(id, {  
  title: 'Updated title'  
}, (error, blogspot) => {  
  console.log(error, blogspot)  
})
```

Deleting Single Record

- ♦ To delete a record:

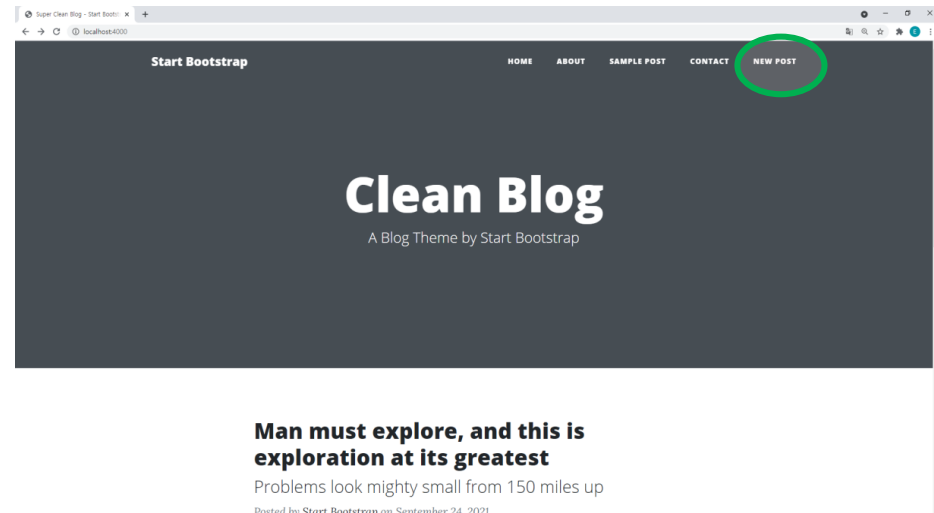
```
var id = "5cb436980b33147489eadfbb";

BlogPost.findByIdAndDelete(id, (error, blogspot) =>{
    console.log(error, blogspot)
})
```

APPLYING MONGODB TO OUR PROJECT

- ◆ In the views folder, create a new file : **create.ejs**.
 - **Copy** the code from **contact.ejs** into **create.ejs**
 - Change the <h1> text to **'Create New Post'**:
- ◆ Register the route for 'Create New Post' : **index.js**:

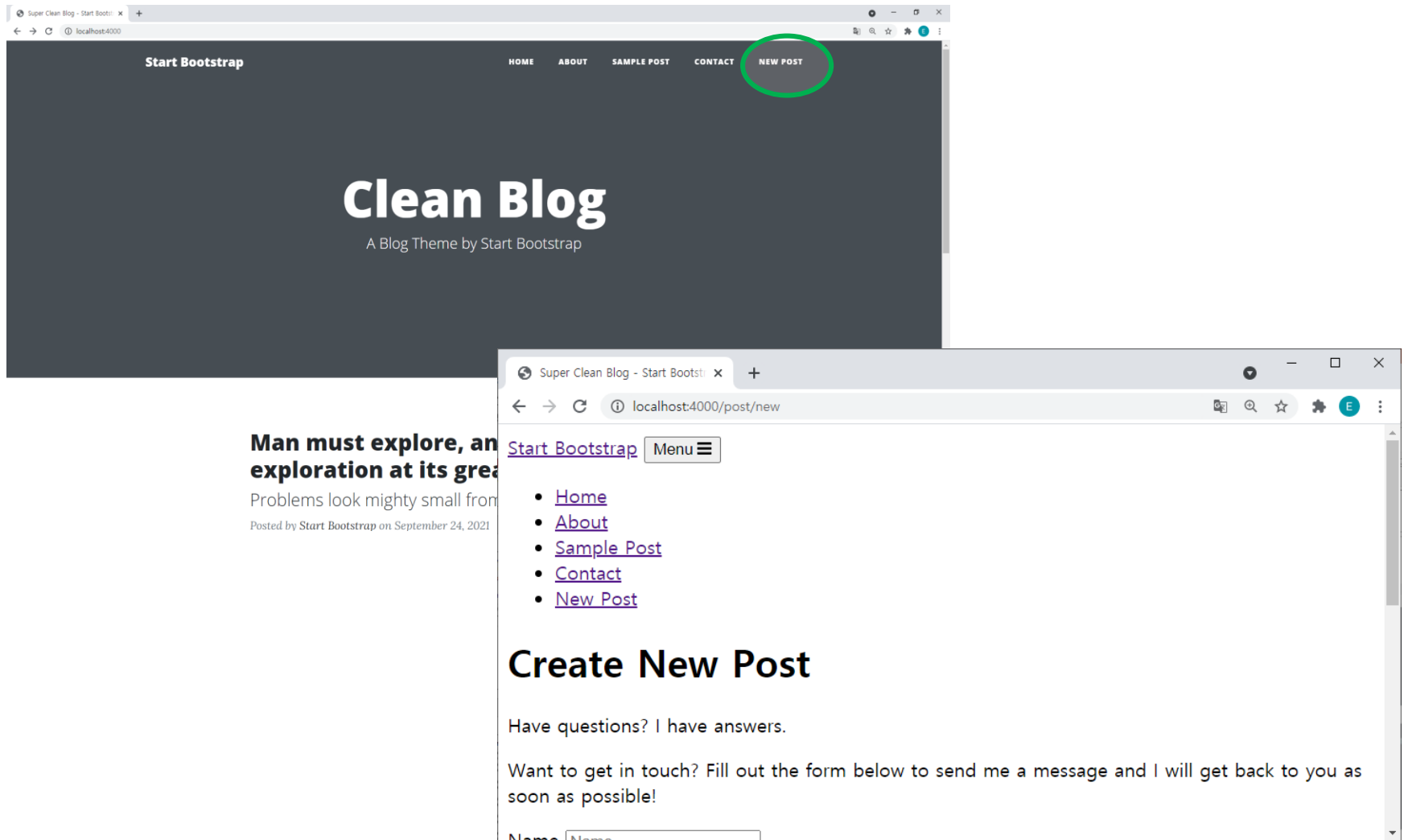
```
app.get( '/posts/new' , (req,res)=>{  
    res.render( 'create' )  
})
```



- ◆ Add the '**New Post**' in the nav bar : views/layouts/navbar.ejs

```
<div class="collapse navbar-collapse" id="navbarResponsive">
<ul class="navbar-nav ml-auto">
  <li class="nav-item">
    <a class="nav-link" href="/">Home</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/about">About</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/post">Sample Post</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/contact">Contact</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/posts/new">New Post</a>
  </li>
</ul>
```

◆ Run the app:



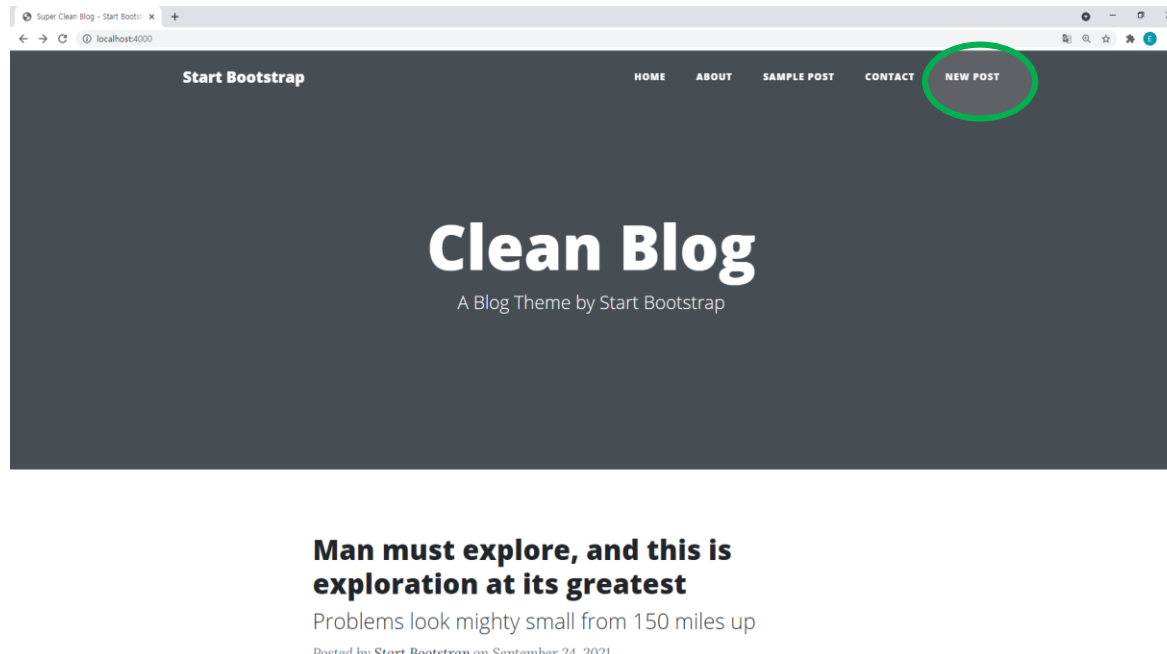
- ♦ To rectify this, we need href to point to an **absolute reference** by adding a '/' before it: **header.ejs, scripts.ejs, create.ejs** :

```
<!-- Custom styles for this template -->  
<link href="/css/clean-blog.min.css" rel="stylesheet">
```

```
<!-- Custom scripts for this template -->  
<script src="/js/clean-blog.min.js"></script>
```

```
<!-- Page Header -->  
<header class="masthead" style="background-image: url('/img/contact-bg.jpg')">  
  ...  
</header>
```

◆ Run the app:



New Post

Title

SEND

♦ In create.ejs :

```
<!-- Main Content -->
<div class="container"> // bootstrap classes
  <div class="row">
    <div class="col-lg-8 col-md-10 mx-auto">
      <form action="/posts/store" method="POST">
        <div class="control-group">
          <div class="form-group floating-label-form-group controls">
            <label>Title</label>
            <input type="text" class="form-control" placeholder="Title"
              id="title" name="title" >
          </div>
        </div>
        <div class="control-group">
          <div class="form-group floating-label-form-group controls">
            <label>Description</label>
            <textarea rows="5" class="form-control"
              id="body" name="body" ></textarea>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
```

◆ In index.js :

```
app.use(express.urlencoded({extended:true}))  
app.use(express.json())
```


```
app.post('/posts/store', (req, res) => {  
    console.log(req.body)  
    res.redirect('/')  
})
```

Super Clean Blog - Start Bootstrap x

localhost:4000/post/new

Start Bootstrap

HOMEABOUTSAMPLE POSTCONTACTNEW POST



Create New Post

Have questions? I have answers.

Title

Message

```
{ title: 'title1', body: 'body1' }
```

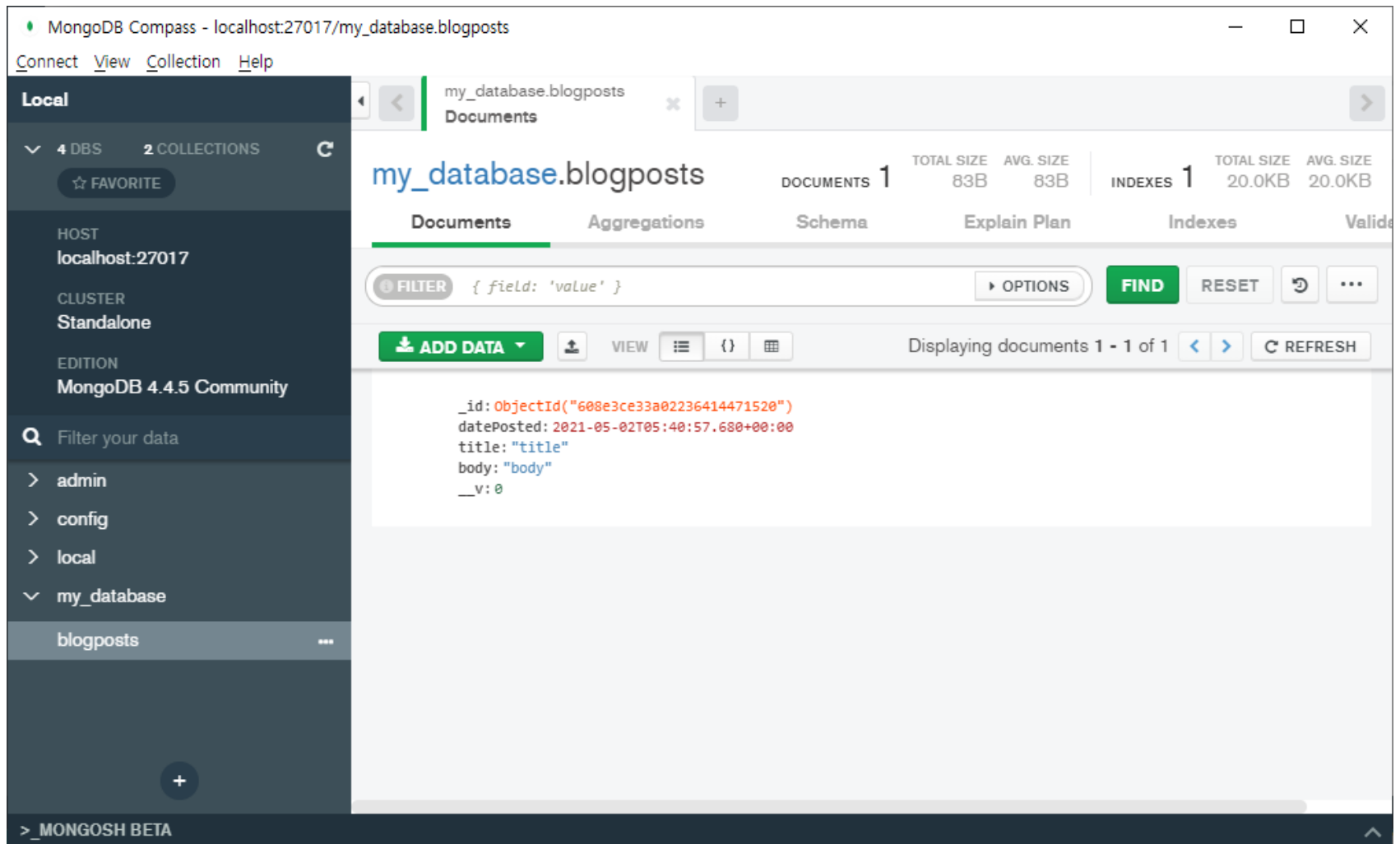
Saving Posts to the Database

◆ In index.js :

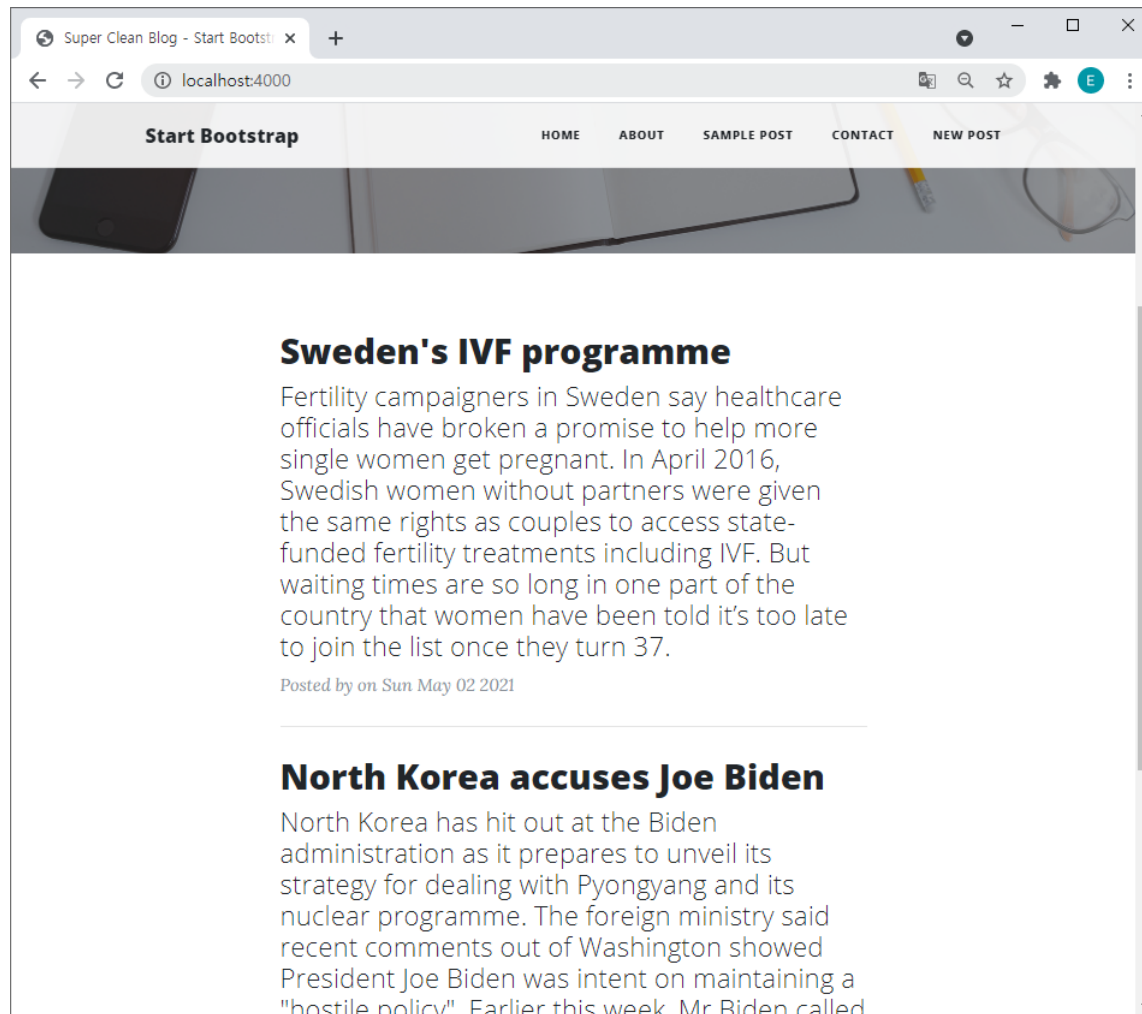
```
...  
const BlogPost = require('./models/BlogPost.js')  
...  
  
app.post('/posts/store', async (req,res)=>{  
  await BlogPost.create(req.body)  
  res.redirect('/')  
})  
...
```

↑

```
app.post('/posts/store', (req,res)=>{  
  // model creates a new doc with browser data  
  BlogPost.create(req.body,(error,blogpost) =>{  
    res.redirect('/')  
  })  
})
```



Displaying a List of Blog Posts



◆ In index.js :

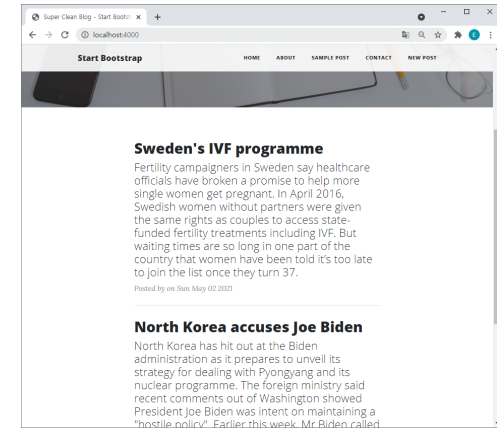
```
app.get('/', async (req, res) => {  
  const blogposts = await BlogPost.find({})  
  res.render('index', {  
    blogposts  
  });  
})
```

- By `console.log(blogposts)` : ...
- With this, index.ejs now has access to the ***blogposts*** variable.

Dynamic Data with Templating Engines

◆ In index.ejs

```
<!-- Main Content -->
<div class="container">
  <div class="row">
    <div class="col-lg-8 col-md-10 mx-auto">
      <% for (var i = 0; i < blogposts.length; i++) { %>
        <div class="post-preview">
          <a href="post.html">
            <h2 class="post-title">
              <%= blogposts[i].title %>
            </h2>
            <h3 class="post-subtitle">
              <%= blogposts[i].body %>
            </h3>
          </a>
          <p class="post-meta">Posted by
            <a href="#">Start Bootstrap</a>
            on September 24, 2019</p>
        </div>
        <hr>
      <% } %>
    <!-- Pager -->
    ...
  </div>
</div>
</div>
```



◆ For the blog-post route :

```
<div class="post-preview">
  <a href="/post/<%= blogposts[i]._id %>">
...
  </a>
...
</div>
```



Sweden's IVF programme

Fertility campaigners in Sweden say healthcare officials have broken a promise to help more single women get pregnant. In April 2016, Swedish women without partners were given the same rights as couples to access state-funded fertility treatments including IVF. But waiting times are so long in one part of the country that women have been told it's too late to join the list once they turn 37.

Posted by on Sun May 02 2021

North Korea accuses Joe Biden

North Korea has hit out at the Biden administration as it prepares to unveil its strategy for dealing with Pyongyang.

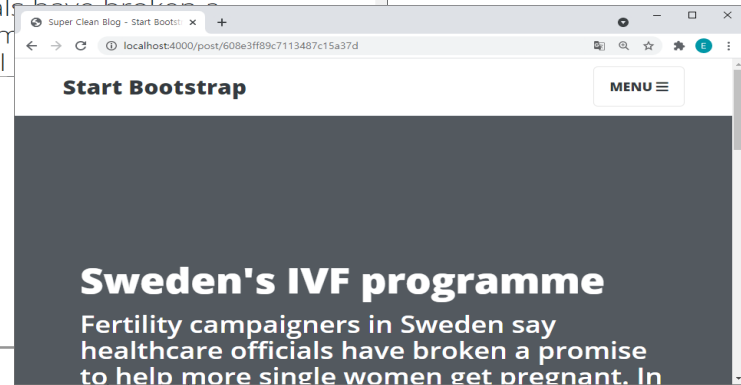
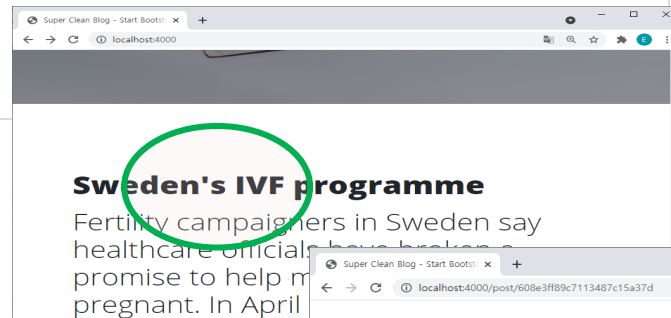
Single Blog Post Page

- ◆ To display each blog post's detail in their own specific url

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

Change it to:

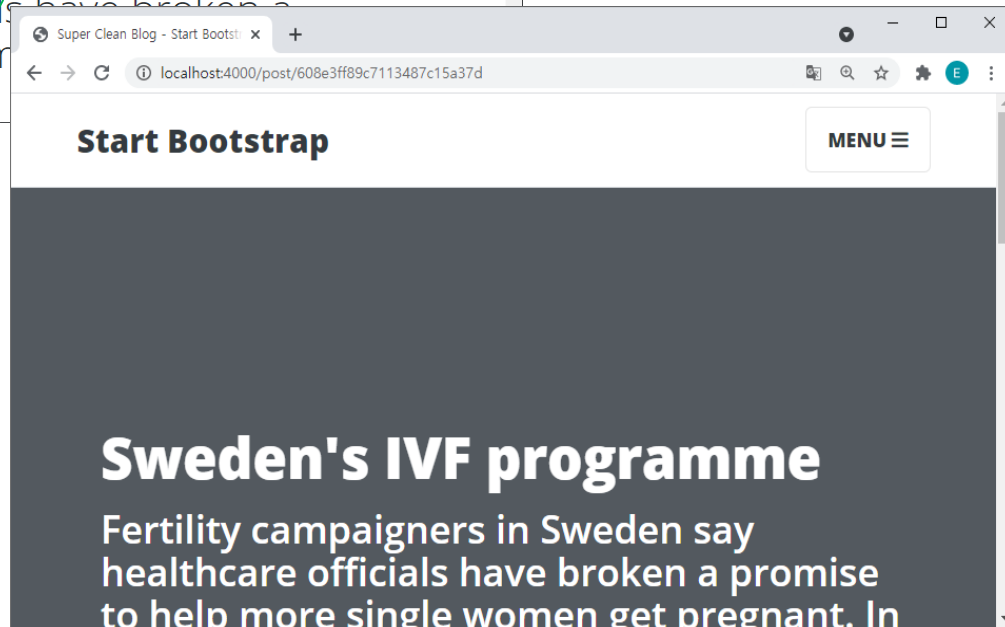
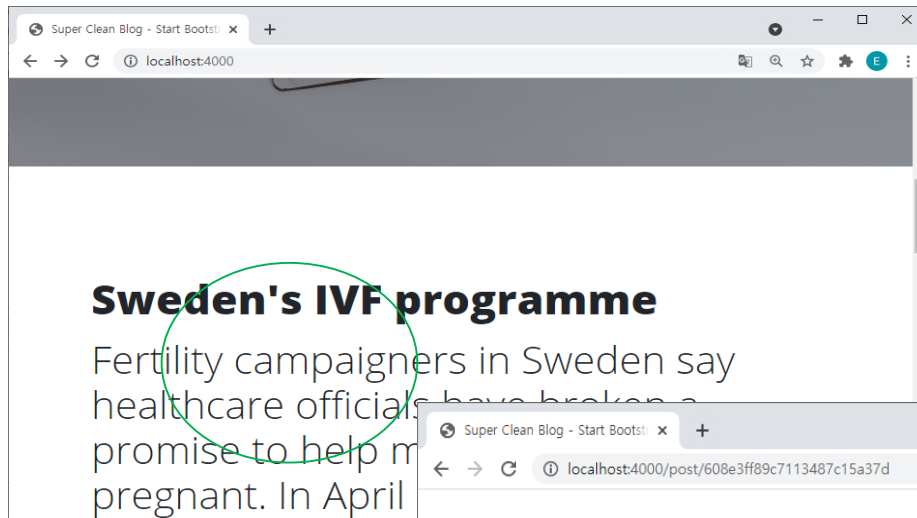
```
app.get('/post/:id', async (req, res) => {  
  const blogpost = await BlogPost.findById(req.params.id)  
  res.render('post', {  
    blogpost  
  })  
})
```



- ♦ To dynamically display each post's unique data : **post.ejs**

```
<!-- Page Header -->
<header class="masthead" style="background-image: url('img/post-
bg.jpg')">
  <div class="overlay"></div>
  <div class="container">
    <div class="row">
      <div class="col-lg-8 col-md-10 mx-auto">
        <div class="post-heading">
          <h1><%= blogpost.title %></h1>
          <h2 class="subheading"><%= blogpost.body %></h2>
          <span class="meta">Posted by
            <a href="#">Start Bootstrap</a>
            on August 24, 2019</span>
        </div>
      </div>
    </div>
  </div>
</header>

<!-- Post Content -->
<article>
  <div class="container">
    <div class="row">
      <div class="col-lg-8 col-md-10 mx-auto">
        <%= blogpost.body %>
      </div>
    </div>
  </article>
```



Adding Fields to the Schema

◆ In BlogPost.js

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema;

const BlogPostSchema = new Schema({
  title: String,
  body: String,
  username: String,
  datePosted: { /* can declare property type with an object like this
because we need 'default' */
    type: Date,
    default: new Date()
  }
});

const BlogPost = mongoose.model('BlogPost', BlogPostSchema);
module.exports = BlogPost
```

♦ index.ejs, post.ejs

index.ejs

```
...
    <p class="post-meta">Posted by
      <a href="#"><%= blogposts[i].username %></a>
      on <%= blogposts[i].datePosted.toDateString() %></p>
  </div>
  <hr>
  <% } %>
  <!-- Pager -->
  <div class="clearfix">
    <a class="btn btn-primary float-right" href="#">Older Posts
&rrarr;</a>
  </div>
</div>
</div>
```

post.ejs

...

```
<!-- Page Header -->
<header class="masthead" style="background-image: url('img/post-
bg.jpg')">
  <div class="overlay"></div>
  <div class="container">
    <div class="row">
      <div class="col-lg-8 col-md-10 mx-auto">
        <div class="post-heading">
          <h1><%= blogpost.title %></h1>
          <h2 class="subheading"><%= blogpost.body %></h2>
          <span class="meta">Posted by
            <a href="#"><%= blogpost.username %></a>
            on <%= blogpost.datePosted.toDateString() %></span>
        </div>
      </div>
    </div>
  </div>
</div>
</header>
```

Clean Blog

A Blog Theme by Start Bootstrap

Sweden's IVF programme

Fertility campaigners in Sweden say healthcare officials have broken a promise to help more single women get pregnant. In April 2016, Swedish women without partners were given the same rights as couples to access state-funded fertility treatments including IVF. But waiting times are so long in one part of the country that women have been told it's too late to join the list once they turn 37.

on Sun May 02 2021

North Korea accuses Joe Biden

North Korea has hit out at the Biden administration as it prepares to unveil its strategy for dealing with Pyongyang and its nuclear programme. The foreign ministry said recent comments out of Washington showed President Joe Biden was intent on maintaining a "hostile policy". Earlier this week, Mr Biden called North Korea's nuclear