

# NodeJS 응용 II

---

# UPLOADING AN IMAGE WITH EXPRESS

# Uploading an Image

Title  
China rocket debris 'disintegrates over Indian Ocean

Description  
Debris from a Chinese rocket that had been hurtling back towards Earth has disintegrated over the Indian Ocean, China says.

The bulk of the rocket was destroyed during the re-entry, but parts landed at a location 72.47° East and 2.65° North, Chinese state-run

Image  
파일 선택 Rocket.jpg

SEND

ADD DATA VIEW {}

```
_id: ObjectId("60948eb3d4559069504a1ef9")
datePosted: 2021-05-07T00:46:37.647+00:00
title: "Covid: Germany rejects US-backed proposal to waive vaccine patents"
body: "Germany has voiced opposition to a US-backed proposal to waive patents..."
image: "/img/img4.jpg"
__v: 0
```

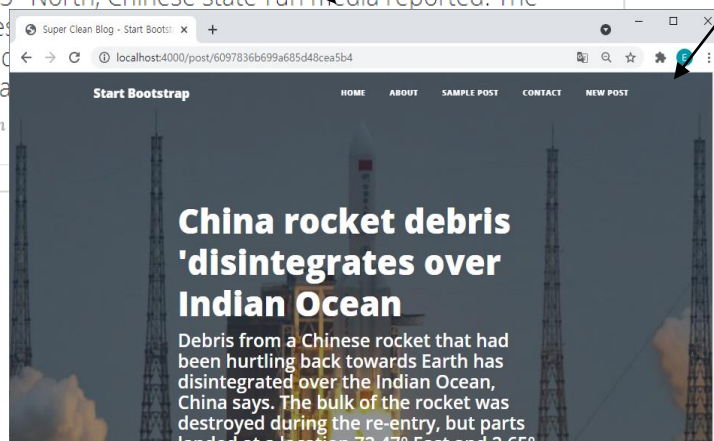
```
_id: ObjectId("6094a5693740be6240e3783f")
datePosted: 2021-05-07T02:24:21.186+00:00
title: "IBM 2nm chip breakthrough claims more power with less energy"
body: "IBM says it has made a significant breakthrough in computer processors..."
image: "/img/ibm.jpg"
__v: 0
```

```
_id: ObjectId("6097836b699a685d48cea5b4")
datePosted: 2021-05-09T06:38:35.344+00:00
title: "China rocket debris 'disintegrates over Indian Ocean"
body: "Debris from a Chinese rocket that had been hurtling back towards Earth..."
image: "/img/Rocket.jpg"
__v: 0
```

**China rocket debris 'disintegrates over Indian Ocean**

Debris from a Chinese rocket that had been hurtling back towards Earth has disintegrated over the Indian Ocean, China says. The bulk of the rocket was destroyed during the re-entry, but parts landed at a location 72.47° East and 2.65° North, Chinese state-run media reported. The point lies sites had Long Ma

Posted by on



◆ > npm install express-fileupload

◆ In create.ejs : add the code

```
<form action="/posts/store" method="POST"
enctype="multipart/form-data">
  ...
  <div class="control-group">
    <div class="form-group floating-label-form-group controls">
      <label>Image</label>
      <input type="file" class="form-control" id="image"
        name="image">
    </div>
  </div>
  <br>
  <div class="form-group">
    <button type="submit" class="btn btn-primary"
      id="sendMessageButton">Send</button>
  </div>
</form>
```

◆ In index.js :

- In the request handler for '/posts/store', add:

```
...
const fileUpload = require('express-fileupload')
app.use(fileUpload())

...
app.post('/posts/store', (req,res)=>{
  let image = req.files.image;
  image.mv(path.resolve(__dirname, 'public/img', image.name), async
    (error)=>{
      await BlogPost.create(req.body)
      res.redirect('/')
    })
  })
})
```

# Saving Uploaded Images to DB

## ♦ In BlogPost.js :

```
...  
const BlogPostSchema = new Schema({  
  title: String,  
  body: String,  
  username: String,  
  datePosted:{  
    type: Date,  
    default: new Date()  
  },  
  image: String  
});  
...
```

## ◆ In index.js

- Specify the full image file path to the BlogPost image attribute

```
app.post('/posts/store', (req,res)=>{
  let image = req.files.image;
  image(move).mv(path.resolve(__dirname,'public/img',image.name),async (error)=>{
    await BlogPost.create({
      ...req.body,
      image: '/img/' + image.name
    })
    res.redirect('/')
  })
})
```

```
_id: ObjectId("60948eb3d4559069504a1ef9")
datePosted: 2021-05-07T00:46:37.647+00:00
title: "Covid: Germany rejects US-backed proposal to waive vaccine patents"
body: "Germany has voiced opposition to a US-backed proposal to waive patents..."
image: "/img/img4.jpg"
__v: 0
```

```
_id: ObjectId("6094a5693740be6240e3783f")
datePosted: 2021-05-07T02:24:21.186+00:00
title: "IBM 2nm chip breakthrough claims more power with less energy"
body: "IBM says it has made a significant breakthrough in computer processors..."
image: "/img/ibm.jpg"
__v: 0
```

## ◆ In Post.ejs :

- To display the image in the post view, change the hardcoded filepath from

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

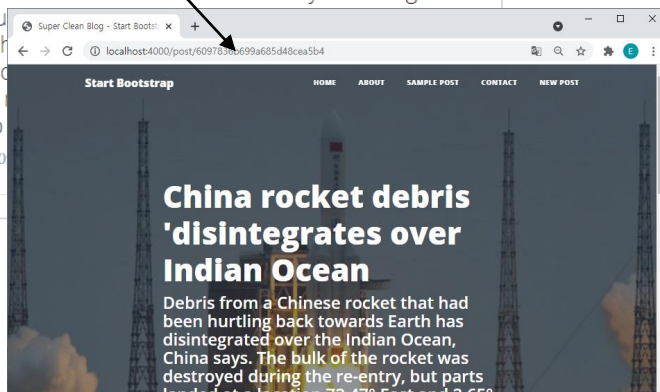
to:

```
<header class="masthead" style="background-image: url('<%= blogpost.image %>')">
```

### China rocket debris 'disintegrates over Indian Ocean

Debris from a Chinese rocket that had been hurtling back towards Earth has disintegrated over the Indian Ocean, China says. The bulk of the rocket was destroyed during the re-entry, but parts landed at a location 73.47° East and 2.65° North. The point lies west of the Indian Ocean. The sites had been predicted by the Indian Space Research Organisation (ISRO) on Long March-5B.

Posted by on Sun May 0





# INTRODUCTION TO EXPRESS MIDDLEWARE

# Middleware

- ◆ The *use* registers a middleware with our Express app

요청의 본문을 해석해줌 (

```
app.use(express.static('public'))
app.use(express.urlencoded({extended:true}))
app.use(express.json())
app.use(fileUpload())
```

파일업로드

↳ 따로 설치가 필요한 3rd 모듈을 사용하여 쿼리 스트링 해석

- ◆ When a browser makes a request to a page, Express will execute all the **'use' statements sequentially** before handling the request  
순차적
- ◆ It might make changes to the request and response objects.
  - ex) app.use(fileUpload())
    - It modifies the **request** object and adds the **request.files** property to it.

# Custom Middleware

## ◆ ex)

```
...                               인자
const customMiddleWare = (req,res,next)=>{
  console.log('Custom middle ware called')
  next() 다음 미들웨어 실행
}
app.use(customMiddleWare)
...
```

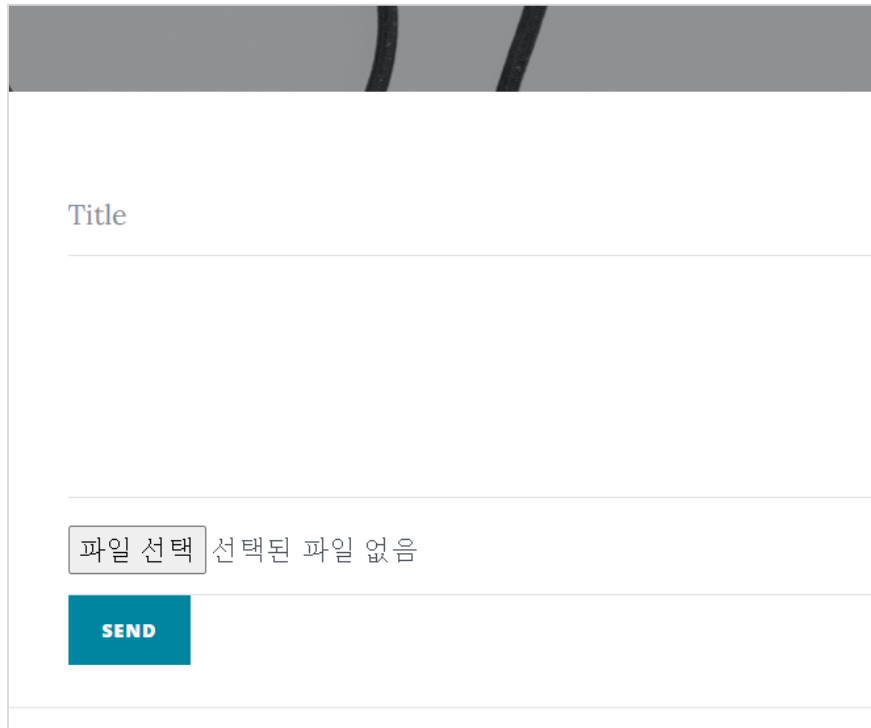
## ◆ next()

- Tells Express that the middleware is done and Express should call the next middleware function
- **All middlewares** called by **app.use** calls next0.

# Registering Validation Middleware

## ◆ Form validation

- When you run your app and to submit a create post form with one of the **fields missing**, you will be re-directed to the form page.



A screenshot of a web form. At the top is a grey header bar. Below it, the form has a label "Title" followed by a text input field that is empty. Further down, there is a file upload section with a button labeled "파일 선택" (File Select) and the text "선택된 파일 없음" (No file selected). At the bottom of the form is a blue button labeled "SEND".

## ◆ Checking for the validity of form field values (field not blank etc.) :

```
...
const validateMiddleWare = (req,res,next)=>{
  if(req.files == null || req.body.title == null || req.body.title == null){
    return res.redirect('/posts/new')
  }
  next()
}
...

...
app.use('/posts/store',validateMiddleWare)
...
```

Note: make sure the above statement is **after** `app.use(fileUpload())` since we depend on the req object having the files property.

# REFACTORING TO MVC

# MVC Pattern

## ◆ MVC (Model-View-Controller)

- A pattern in software design commonly used to implement **user interfaces**, **data**, and **controlling logic**.

## ◆ Model

- Represents the structure of the data, the format and the constraints with which it is stored.
- In essence, it is the database part of the application

## ◆ View

- What is presented to the user  
보여주다

## ◆ Controller

- Controls the requests of the user and then generates appropriate response rendered back to the user.

- ◆ Create a new folder **controllers**
  - In it, create a new file **newPost.js**

In *newPost.js*, fill in the code with:

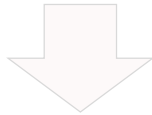
```
module.exports = (req, res) =>{  
  res.render('create')  
}
```



- ◆ In index.js, replace the request handler

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

with:



```
const newPostController = require('./controllers/newPost')  
  
...  
  
app.get('/posts/new', newPostController)
```

- ◆ **Remove the About, Contact and Sample Post pages:**

So, in *index.js*, remove:

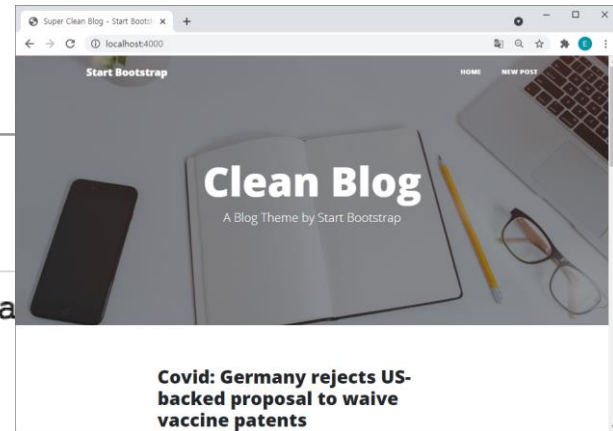
```
app.get('/about', (req, res) => {  
  res.render('about');  
})
```

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

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

## ◆ In navbar.ejs:

```
<div class="collapse navbar-collapse" id="navba
  <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>
</div>
```



- ◆ In controllers folder, create **home.js**, **storePost.js** and **getPost.js**

*home.js*

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

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

*getPost.js*

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

module.exports = async (req,res)=>{
  const blogpost = await BlogPost.findById(req.params.id)
  console.log(blogpost)
  res.render('post',{
    blogpost
  });
}
```

*storePost.js*

```
const BlogPost = require('../models/BlogPost.js')
const path = require('path')

module.exports = (req,res)=>{
  let image = req.files.image;
  image.mv(path.resolve(__dirname, '..', 'public/img', image.name), async
(error)=>{
    await BlogPost.create({
      ...req.body,
      image: '/img/' + image.name
    })
    res.redirect('/')
  })
}
```

◆ In index.js:

```
const path = require('path')  
const BlogPost = require('../models/BlogPost.js')  
  
...  
const homeController = require('./controllers/home')  
const storePostController = require('./controllers/storePost')  
const getPostController = require('./controllers/getPost')  
...  
  
app.get('/', homeController)  
app.get('/post/:id', getPostController)  
app.post('/posts/store', storePostController)
```

# Refactoring Validation Layer

- ♦ Create a **middleware** directory and in it, create a new file **validationMiddleware.js**
  - Cut and paste the validationMiddleware function from **index.js** into validationMiddleware.js :

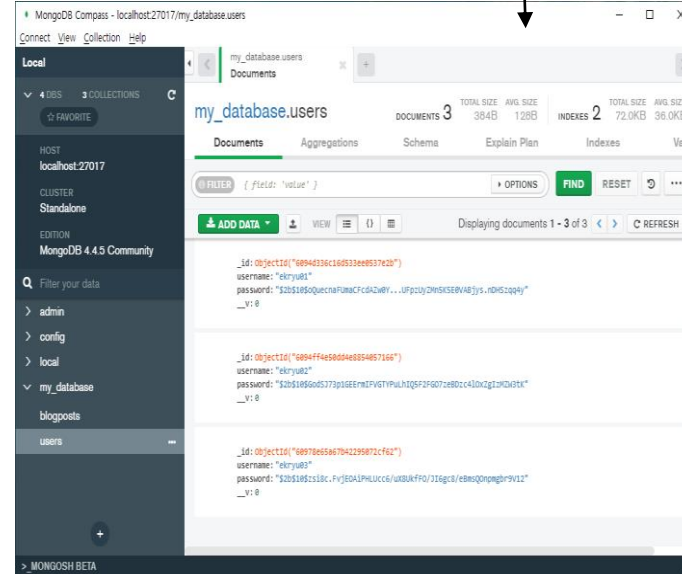
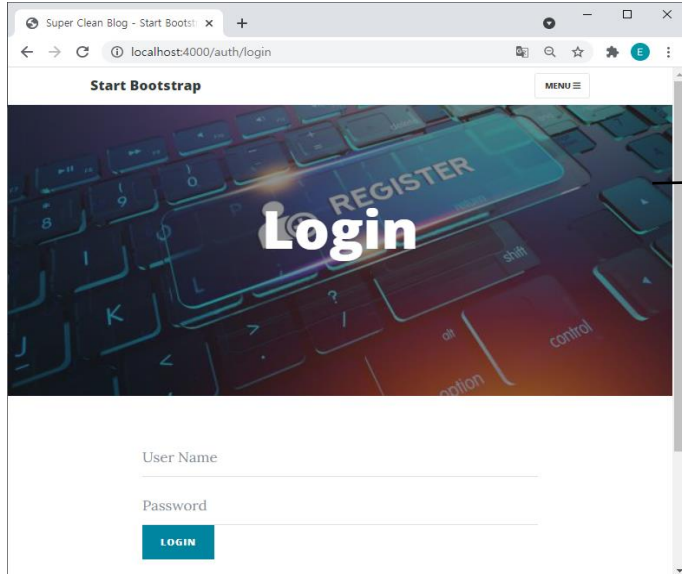
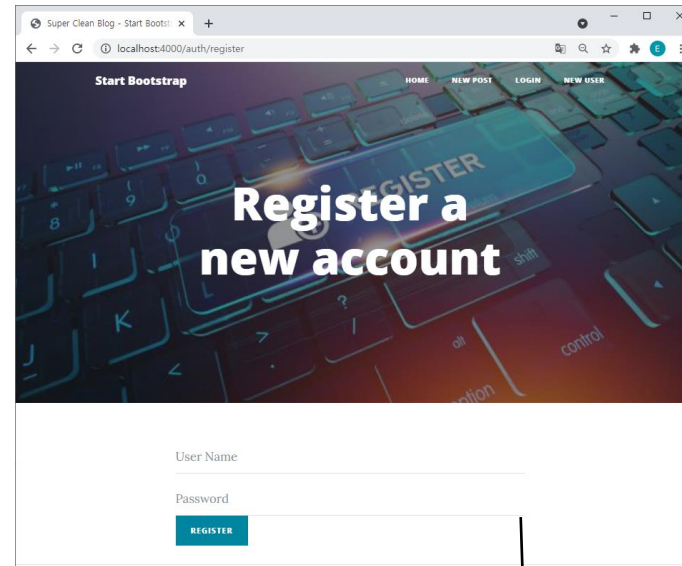
```
module.exports = (req,res,next)=>{
  if(req.files == null || req.body.title == null || req.body.title == null){
    return res.redirect('/posts/new')
  }
  next()
}
```

- ♦ In **index.js**, import the middleware with:

```
const validateMiddleware = require("../middleware/validateMiddleware");
```



# **USER REGISTRATION**



- ◆ Create a new file **register.ejs** in 'views' folder (using create.ejs)

In *register.ejs*, change the header to "Register a new account"

...

```
<div class="page-heading">  
  <h1>Register a new account</h1>  
</div>
```

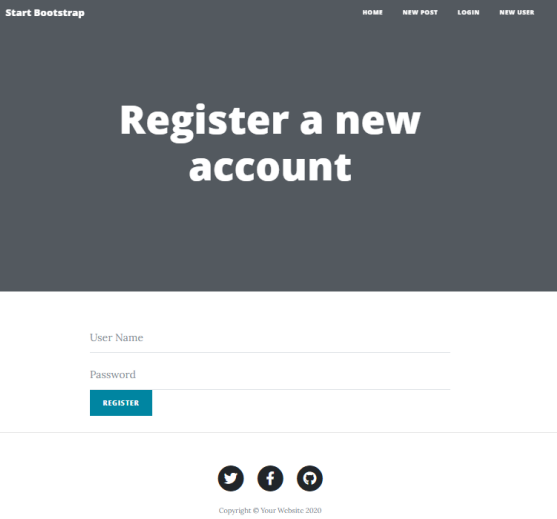
...

Next, change *form action* to:

...

```
<form action="/users/register" ...>
```

...



The screenshot shows a web application interface for user registration. At the top, there is a dark navigation bar with the text "Start Bootstrap" on the left and links for "HOME", "NEW POST", "LOGIN", and "NEW USER" on the right. Below the navigation bar is a large dark section with the heading "Register a new account" in white. Underneath this heading is a registration form with two input fields: "User Name" and "Password". A blue "REGISTER" button is positioned below the "Password" field. At the bottom of the page, there is a footer containing three social media icons (Twitter, Facebook, and GitHub) and the text "Copyright © Your Website 2020".

- ♦ In the title field, change it to **username field** :

```
<div class="control-group">
  <div class="form-group floating-label-form-group controls">
    <label>User Name</label>
    <input type="text" class="form-control" placeholder="User
Name" id="username" name="username">
  </div>
</div>
```

- ♦ **Password field** : Duplicate the above code and rename it

```
<div class="control-group">
  <div class="form-group floating-label-form-group controls">
    <label>Password</label>
    <input type="password" class="form-control" placeholder="Password"
id="password" name="password">
  </div>
</div>
```

- ◆ Rename the submit button to **Register**

```
<div class="form-group">  
  <button type="submit" class="btn btn-primary">Register</button>  
</div>
```

- ◆ Delete the code for description

- ♦ In controllers folder, create the **newUser.js** :

```
module.exports = (req, res) =>{  
  res.render('register') // render register.ejs  
}
```

## ◆ In index.js

```
...  
const newUserController = require('./controllers/newUser')
```

and apply it to the route:

```
app.get('/auth/register', newUserController)
```

♦ In views/layouts/navbar.ejs, add:

```
<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="/posts/new">New Post</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/auth/register">New User</a>
    </li>
  </ul>
</div>
```



# User Model

- ◆ In **models folder**, create a new file **User.js**
  - copy/edit contents from **BlogPost.js** to create the User schema:

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

const UserSchema = new Schema({
  username: String,
  password: String
});

// export model
const User = mongoose.model('User', UserSchema);
module.exports = User
```

- ◆ **In index.js** : Register a new route

```
...  
const storeUserController = require('./controllers/storeUser')  
...  
  
app.post('/users/register', storeUserController)  
...
```

- ◆ **In register.ejs** : Specify this same route in the form action of the register user form

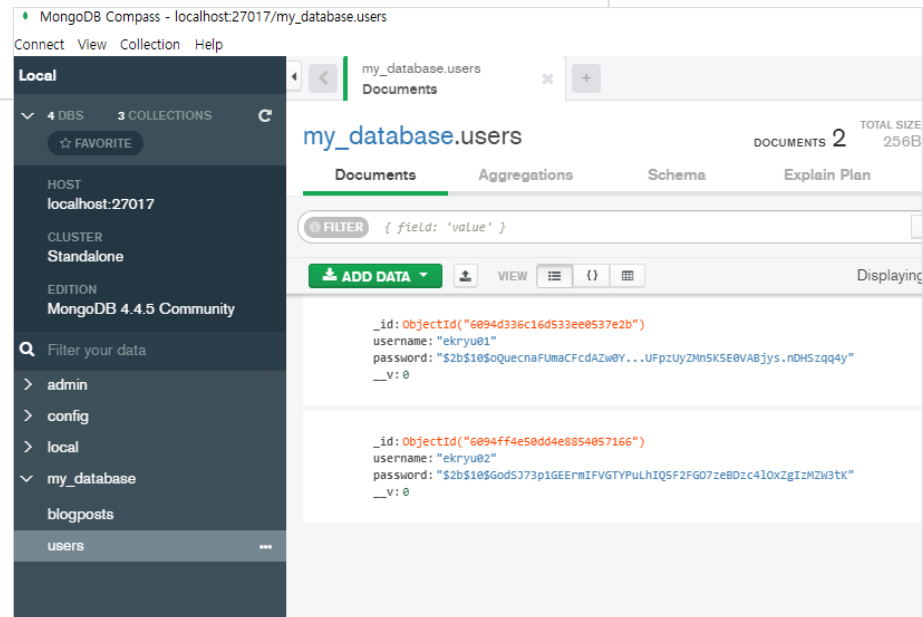
```
<form action="/users/register" method="POST" enctype="multipart/form-data">
```

# Handle User Registration

- ◆ In controllers folder : Create a new file **storeUser.js**

```
const User = require('../models/User.js')
const path = require('path')

module.exports = (req,res)=>{
  User.create(req.body, (error, user) => {
    res.redirect('/')
  })
}
```



# Password Encryption

---

- ◆ To encrypt a password before storing it
  - > npm install bcrypt

◆ In /models/ User.js : add the codes:

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

const UserSchema = new Schema({
  username: String,
  password: String
});

UserSchema.pre('save', function(next){
  const user = this

  bcrypt.hash(user.password, 10, (error, hash) => {
    user.password = hash
    next()
  })
})

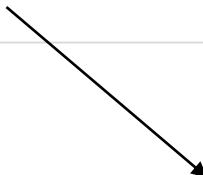
const User = mongoose.model('User', UserSchema);
module.exports = User
```

# Mongoose Validation

## ◆ In User.js :

- To ensure that the **username** is **required** and **unique** in DB

```
...  
const UserSchema = new Schema({  
  username: String,  
  password: String  
});  
...
```



```
const UserSchema = new Schema({  
  username: {  
    type: String,  
    required: true,  
    unique: true  
  },  
  password: {  
    type: String,  
    required: true  
  }  
});
```

- ◆ **In storeUser.js** : when there is an error, we will redirect back to the user register form

```
...
module.exports = (req,res)=>{
  User.create(req.body, (error, user) => {
    if(error){
      return res.redirect('/auth/register')
    }
    res.redirect('/')
  })
}
```

↳ 시작페이지

# User Login Process

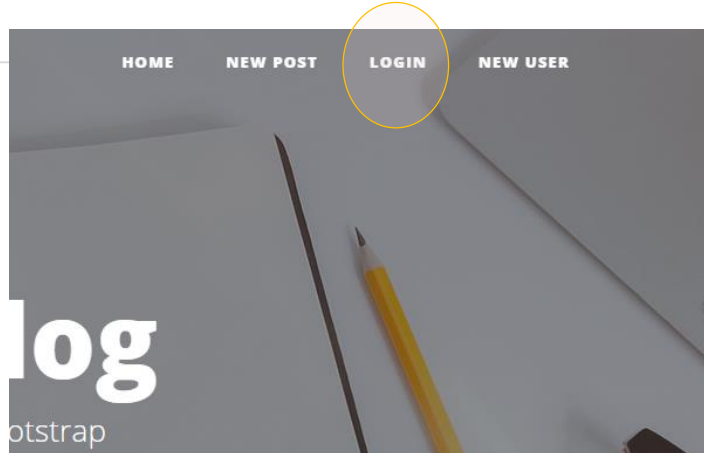




- ◆ In views folder, create a new file login.ejs :  
(copying register.ejs)

In *login.js*, change the page heading and button label to 'Login':

```
<div class="page-heading">
  <h1>Login</h1>
</div>
...
<div class="form-group">
  <button type="submit" class="btn btn-primary">Login</button>
</div>
```

A screenshot of the login form. It has a dark grey header with the word 'Login' in white. Below the header, there are two input fields: 'User Name' and 'Password'. At the bottom, there is a blue 'LOGIN' button.

Next in form action, change from `<form action="/users/register"...` to:

`<form action="/users/login"...`

and in *controllers* folder, create a new file *login.js* with the following code:

```
module.exports = (req, res) =>{  
  res.render('login')  
}
```

*views 폴더안 login.ejs*

◆ In index.js :

```
const loginController = require('./controllers/login')
```

```
...
```

and register the route with :

```
...
```

```
app.get('/auth/login', loginController);
```

```
...
```

- ◆ **In view/ layouts/navbar.ejs:** Add login to the navbar

```
<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="/posts/new">New Post</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/auth/login">Login</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/auth/register">New User</a>
  </li>
</ul>
```

# Login Process

- ◆ In controllers folder, create a new file loginUser.js

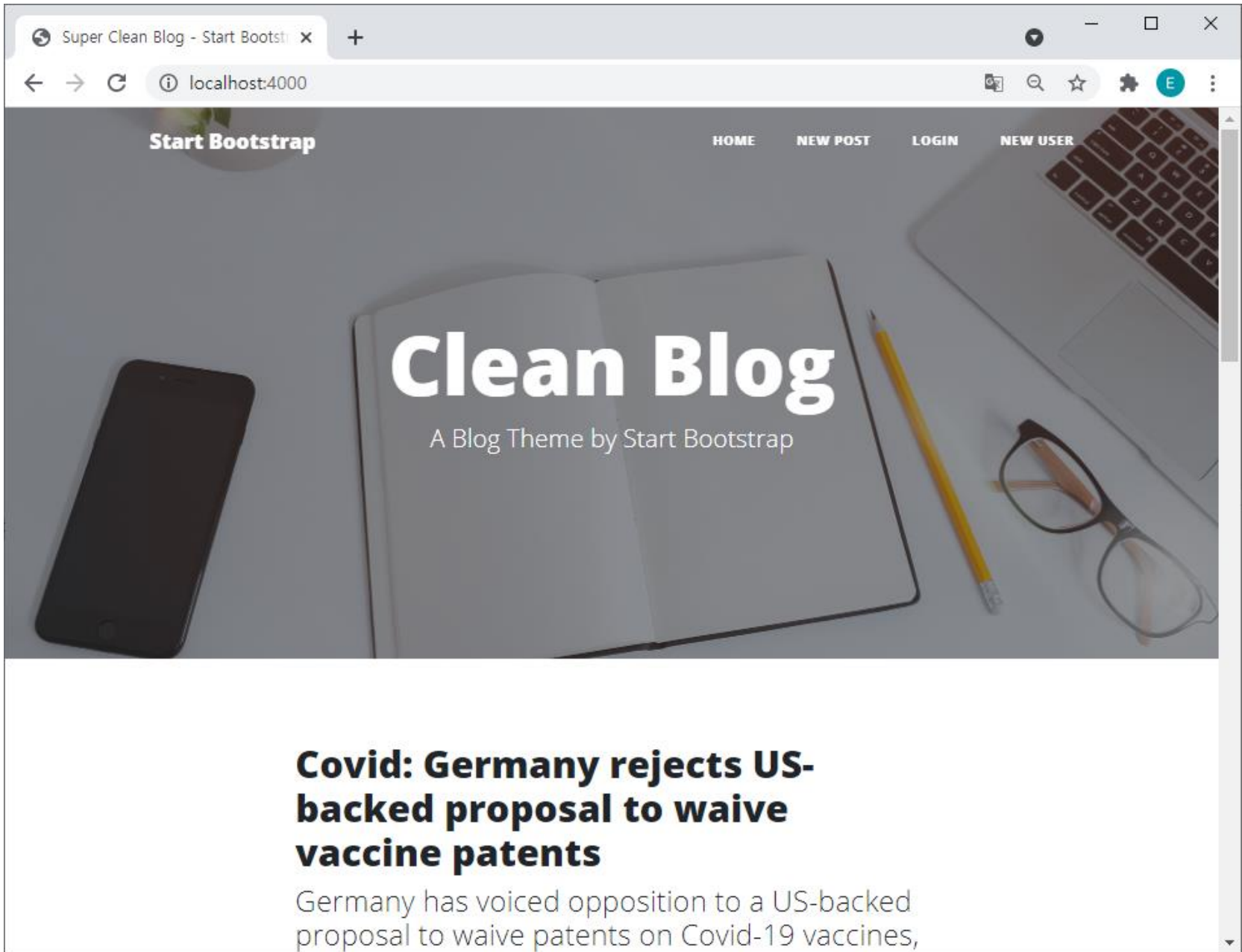
```
const bcrypt = require('bcrypt')
const User = require('../models/User')

module.exports = (req, res) => {
  const { username, password } = req.body;

  DB User.findOne({username:username}, (error, user) => {
    if (user){
      bcrypt.compare(password, user.password, (error, same) => {
        if(same){ // if passwords match
          // store user session, will talk about it later
          res.redirect('/') 시작
        }
        else{
          res.redirect('/auth/login') 다시 로그인페이지
        }
      })
    }
    else{
      res.redirect('/auth/login')
    }
  })
}
```

- ◆ **In index.js** : To apply the loginUserController

```
const loginUserController = require('./controllers/loginUser')  
  
app.post( '/users/login',loginUserController)
```



Start Bootstrap

HOME

NEW POST

LOGIN

NEW USER

# Clean Blog

A Blog Theme by Start Bootstrap

## Covid: Germany rejects US-backed proposal to waive vaccine patents

Germany has voiced opposition to a US-backed proposal to waive patents on Covid-19 vaccines,

# Summary

---

- ◆ APPLYING **MONGODB** TO Blog PROJECT
- ◆ **UPLOADING IMAGE** WITH EXPRESS
- ◆ INTRODUCTION TO EXPRESS **MIDDLEWARE**
- ◆ REFACTORING TO **MVC**
- ◆ **USER REGISTRATION**

index.js 의 get → 경로에 따라 메뉴클릭시  
post → 버튼(로그인, 회원가입)  
서버에서 받아서 처리해야하는 부분.