

```
models > Js User.js > ...
1 // models/User.js
2
3 var mongoose = require('mongoose');
4
5 // schema // 1
6 var userSchema = mongoose.Schema({
7   username:{type:String, required:[true,'Username is required!'], unique:true},
8   password:{type:String, required:[true,'Password is required!'], select:false},
9   name:{type:String, required:[true,'Name is required!']},
10  email:{type:String}
11 },{
12   toObject:{virtuals:true}
13 });
14 // 첫번째는 true/false 값이고, 두번째는 에러메세지입니다.
15 // 그냥 true/false를 넣을 경우 기본 에러메세지가 나오고, 위와 같이 배열을 사용해서 에러메세지 내용을 원하는 대로 변경할 수 있습니다.
16
17 // virtuals // 2
18 userSchema.virtual('passwordConfirmation')
19   .get(function(){ return this._passwordConfirmation; })
20   .set(function(value){ this._passwordConfirmation=value; });
21
22 userSchema.virtual('originalPassword')
23   .get(function(){ return this._originalPassword; })
24   .set(function(value){ this._originalPassword=value; });
25
26 userSchema.virtual('currentPassword')
27   .get(function(){ return this._currentPassword; })
28   .set(function(value){ this._currentPassword=value; });
29
```



```

30 userSchema.virtual('newPassword')
31   .get(function(){ return this._newPassword; })
32   .set(function(value){ this._newPassword=value; });
33 // DB에 저장되는 값 이외의 항목이 필요할 땐 virtual 항목으로 만듭니다.
34 // 즉 passwordConfirmation, originalPassword, currentPassword, newPassword는 회원가입,
35 // 회원정보 수정을 위해 필요한 항목이지만, DB에 저장할 필요는 없는 값들입니다. 이처럼 DB에 저장될 필요는 없지만,
36 // model에서 사용하고 싶은 항목들은 virtual로 만듭니다.
37
38 // password validation // password를 DB에 생성, 수정하기 전에 값이 유효(valid)한지 확인(validate)을 하는 코드를 작성합니다
39 userSchema.path('password').validate(function(v) {
40   var user = this; // 3-1
41   // validation callback 함수 속에서 this는 user model입니다. 헷갈리지 않도록 user 변수에 넣었습니다.
42
43   // create user // 3-3
44   if(user.isNew){ // 3-2
45     //model.isNew 항목은 해당 모델이 생성되는 경우에는 true, 아니면 false의 값을 가집니다.
46     //이 항목을 이용해서 현재 password validation이 '회원가입' 단계인지, 아니면 '회원 정보 수정'단계인지를 알 수 있습니다.
47     if(!user.passwordConfirmation){
48       user.invalidate('passwordConfirmation', 'Password Confirmation is required.');

```

```
58
59 // update user // 3-4
60 if(!user.isNew){
61   if(!user.currentPassword){
62     user.invalidate('currentPassword', 'Current Password is required!');
63   }
64   else if(user.currentPassword != user.originalPassword){
65     user.invalidate('currentPassword', 'Current Password is invalid!');
66   }
67
68   if(user.newPassword !== user.passwordConfirmation) {
69     user.invalidate('passwordConfirmation', 'Password Confirmation does not matched!');
70   }
71 }
72 });
73 // 회원 정보 수정의 경우 current password값이 없는 경우와,
74 // current password값이 original password값과 다른 경우,
75 // new password값과 password confirmation값이 다른 경우 invalidate합시다.
76 // 회원정보 수정시에는 항상 비밀번호를 수정하는 것은 아니기 때문에 new password와 password confirmation값이 없어도 에러는 아닙니다.
77
78 // model & export
79 var User = mongoose.model('user',userSchema);
80 module.exports = User;
```



```
1 // routes/users.js
2
3 var express = require('express');
4 var router = express.Router();
5 var User = require('../models/User');
6
7 // Index // 1
8 router.get('/', function(req, res){
9   User.find({})
10     .sort({username:1})
11     .exec(function(err, users){
12       if(err) return res.json(err);
13       res.render('users/index', {users:users});
14     });
15 });
16 // 지금까지의 index route과는 다르게, 찾은 값을 정렬하는 기능이 추가되었습니다.
17 // sort함수가 추가되었는데요, 이 함수에는 {username:1} 이 들어가서 username을 기준으로 오름차순(asc)으로 정렬하고 있습니다.
18 // (-1을 넣으면 내림차순(desc)이 됩니다.)
19 // New
20 router.get('/new', function(req, res){
21   res.render('users/new');
22 });
23
24 // create
25 router.post('/', function(req, res){
26   User.create(req.body, function(err, user){
27     if(err) return res.json(err);
28     res.redirect('/users');
29   });
30 });
31
```

```
32 // show
33 router.get('/:username', function(req, res){
34     User.findOne({username:req.params.username}, function(err, user){
35         if(err) return res.json(err);
36         res.render('users/show', {user:user});
37     });
38 });
39
40 // edit
41 router.get('/:username/edit', function(req, res){
42     User.findOne({username:req.params.username}, function(err, user){
43         if(err) return res.json(err);
44         res.render('users/edit', {user:user});
45     });
46 });
47
48 // update // 2
49 router.put('/:username', function(req, res, next){
50     User.findOne({username:req.params.username}) // 2-1
51     // findOneAndUpdate함수대신에 findOne함수로 값을 찾은 후에 값을 수정하고 user.save함수로 값을 저장합니다.
52     // 단순히 값을 바꾸는 것이 아니라 user.password를 조건에 맞게 바꿔주어야 하기 때문이죠.
53
54     .select('password') // 2-2
55     // select함수를 이용하면 DB에서 어떤 항목을 선택할지, 안할지를 정할 수 있습니다.
56     //user schema에서 password의 select을 false로 설정했으니 DB에 password가 있더라도 기본적으로 password를
57     // 읽어오지 않게 되는데, select('password')를 통해서 password를 읽어오게 했습니다.
58     // 참고로 select함수로 기본적으로 읽어오게 되어 있는 항목을 안 읽어오게 할 수도 있는데 이때는 항목이름 앞에 -를 붙이면 됩니다.
59     // 또한 하나의 select함수로 여러 항목을 동시에 정할 수도 있는데, 예를 들어 password를 읽어오고, name을 안 읽어오게 하고 싶다면
60     // .select('password -name')를 입력하면 되겠습니다.
61
```



```
62     .exec(function(err, user){
63         if(err) return res.json(err);
64
65         // update user object
66         user.originalPassword = user.password;
67         user.password = req.body.newPassword? req.body.newPassword : user.password;
68         // user의 update(회원 정보 수정)은 password를 업데이트 하는 경우와, password를 업데이트 하지 않는 경우로 나눌 수 있는데,
69         // 이에 따라 user.password의 값이 바뀝니다
70         for(var p in req.body){ // 2-4
71             user[p] = req.body[p];
72         }
73         // user는 DB에서 읽어온 data이고, req.body가 실제 form으로 입력된 값이므로 각 항목을 덮어 쓰는 부분입니다.
74
75         // save updated user
76         user.save(function(err, user){
77             if(err) return res.json(err);
78             res.redirect('/users/'+user.username);
79         });
80     });
81 });
82
83 // destroy
84 router.delete('/:username', function(req, res){
85     User.deleteOne({username:req.params.username}, function(err){
86         if(err) return res.json(err);
87         res.redirect('/users');
88     });
89 });
90
91 module.exports = router;
```

JS index.js > ...


```
1  var express = require('express');
2  var mongoose = require('mongoose');
3  var bodyParser = require('body-parser');
4  // body-parser module를 bodyPaser 변수에 담습니다
5  // 역할 조사
6  var methodOverride = require('method-override');
7  // method-override module을 methodOverride변수에 담습
8  // 역할 조사
9  var app = express();
10
11  // DB setting
12  mongoose.set('useNewUrlParser', true);
13  mongoose.set('useFindAndModify', false);
14  mongoose.set('useCreateIndex', true);
15  mongoose.set('useUnifiedTopology', true);
16  // mongoose의 몇몇 글로벌 설정을 해 주는 부분입니다. 저 부분이 바뀔 일은 웬만하면 없기 때문에 그냥 항상 저렇게 설정하고 쓰시면 됩니다.
17  // mongoose.connect(process.env.MONGO_DB); -- 윈도우에서 사용시 이용하는 코드
18  mongoose.connect('mongodb+srv://test_username:0000@cluster0-amulf.mongodb.net/test?retryWrites=true&w=majority');
19  // mongo altes의 주소 - 맥이나 우분투에서 사용시 주소를 통채로 넣어야 한다
20
21  var db = mongoose.connection;
22  db.once('open', function(){
23    console.log('DB connected');
24  });
25  // 'open' db연결이 성공 했을때 나오는 메시지
26  db.on('error', function(err){
27    console.log('DB ERROR : ', err);
28  });
29  // 'error' db연결이 실패 했을때 나오는 메시지
30
31
```



```
31
32 // Other settings
33 app.set('view engine', 'ejs');
34 //조사 -
35 app.use(express.static(__dirname+'/public'));
36 //조사 -
37 app.use(bodyParser.json());
38 // json형식의 데이터를 받는다는 설정
39 app.use(bodyParser.urlencoded({extended:true}));
40 // urlencoded data를 extended 알고리즘을 사용해서 분석한다는 설정입니다
41 app.use(methodOverride('_method'));
42 // _method의 query로 들어오는 값을 HTTP method를 바꾼다.
43 // ex) http://example.com/category/id?_method=delete를 받으면 _method의 값인 delete을 읽어
44 // 해당 request의 HTTP method를 delete으로 바꾼다.
45
46 // Routes
47 // 라우팅 리소스 별로 모듈을 만들어 라우팅 로직을 구현. 클라이언트에서 요청 별로 어떤 로직을 수행할지 정해놓은 파일
48 // (java에서 Controller 역할)
49 app.use('/', require('./routes/home'));
50 app.use('/posts', require(['./routes/posts']));
51 app.use('/users', require('./routes/users'));
52
53 // Port setting
54 var port = 3000;
55 app.listen(port, function(){
56 |   console.log('server on! http://localhost: '+port);
57 | });
```


partials > % nav.ejs > nav.navbar.navbar-expand-sm.navbar-light.bg-light.mb-3 > div.container > div#navbarSupportedContent.collapse.

```
1  <!-- views/partials/nav.ejs -->
2  <!-- 네비게이션 메뉴는 주소록 만들기 예제와 구조 -->
3
4  <nav class="navbar navbar-expand-sm navbar-light bg-light mb-3">
5      <div class="container">
6          <div class="navbar-brand">My Website</div>
7          <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent">
8              <span class="navbar-toggler-icon"></span>
9          </button>
10         <div class="collapse navbar-collapse" id="navbarSupportedContent">
11             <ul class="navbar-nav">
12                 <li class="nav-item"><a href="/" class="nav-link">Home</a></li>
13                 <li class="nav-item"><a href="/about" class="nav-link">About</a></li>
14                 <li class="nav-item"><a href="/posts" class="nav-link">Board</a></li>
15             </ul>
16             <ul class="navbar-nav ml-auto">
17                 <li class="nav-item"><a href="/users" class="nav-link">Users</a></li>
18                 <li class="nav-item"><a href="/users/new" class="nav-link">Sign Up</a></li>
19             </ul>
20         </div>
21     </div>
22 </nav>
```

views > users > <% index.ejs >  html

```
1  <!-- views/users/index.ejs -->
2
3  <!DOCTYPE html>
4  <html>
5    <head>
6      <%- include('../partials/head') %>
7    </head>
8    <body>
9      <%- include('../partials/nav') %>
10
11     <div class="container mb-3">
12
13       <h3 class="mb-3">Users</h3>
14
15       <ul class="list-group">
16         <% if(users == null || users.length == 0){ %>
17           <li class="list-group-item"> There is no user yet.</div>
18         <% } %>
19         <% users.forEach(function(user) { %>
20           <li class="list-group-item">
21             <a href="/users/<%= user.username %>"><%= user.username %></a>
22           </li>
23         <% }) %>
24       </ul>
25
26     </div>
27   </body>
28 </html>
```


views > users > <% new.ejs >  html

```
1  <!-- views/users/new.ejs -->
2
3  <!DOCTYPE html>
4  <html>
5      <head>
6          <%- include('../partials/head') %>
7      </head>
8      <body>
9          <%- include('../partials/nav') %>
10
11      <div class="container mb-3">
12
13          <h3 class="contentBoxTop mb-3">New User</h3>
14
15          <form action="/users" method="post">
16
17              <div class="form-group row">
18                  <label for="username" class="col-sm-3 col-form-label">Username*</label>
19                  <div class="col-sm-9">
20                      <input type="text" id="username" name="username" value="" class="form-control">
21                  </div>
22              </div>
23              <div class="form-group row">
24                  <label for="name" class="col-sm-3 col-form-label">Name*</label>
25                  <div class="col-sm-9">
26                      <input type="text" id="name" name="name" value="" class="form-control">
27                  </div>
28              </div>
29              <div class="form-group row">
30                  <label for="email" class="col-sm-3 col-form-label">Email</label>
31                  <div class="col-sm-9">
32                      <input type="text" id="email" name="email" value="" class="form-control">
33                  </div>
```

```
34 </div>
35 <div class="form-group row">
36   <label for="password" class="col-sm-3 col-form-label">Password*</label>
37   <div class="col-sm-9">
38     <input type="password" id="password" name="password" value="" class="form-control">
39   </div>
40 </div>
41 <div class="form-group row">
42   <label for="passwordConfirmation" class="col-sm-3 col-form-label">Password Confirmation*</label>
43   <div class="col-sm-9 col-sm-offset-3">
44     <input type="password" id="passwordConfirmation" name="passwordConfirmation" value="" class="form-control">
45   </div>
46 </div>
47 <p>
48   <small>*Required</small>
49 </p>
50
51 <div class="form-group">
52   <button type="submit" class="btn btn-primary">Submit</button>
53 </div>
54 </form>
55
56 </div>
57 </body>
58 </html>
```


views > users > ⏪ edit.ejs > 📦 html

```
1  <!-- views/users/edit.ejs -->
2
3  <!DOCTYPE html>
4  <html>
5    <head>
6      <%- include('../partials/head') %>
7    </head>
8    <body>
9      <%- include('../partials/nav') %>
10
11     <div class="container mb-3">
12
13       <h3 class="mb-3">Edit User</h3>
14
15       <form action="/users/<%= user.username %>?_method=put" method="post">
16
17         <div class="form-group row">
18           <label for="currentPassword" class="col-sm-3 col-form-label">Current Password*</label>
19           <div class="col-sm-9 col-sm-offset-3">
20             <input type="password" id="currentPassword" name="currentPassword" value="" class="form-control">
21           </div>
22         </div>
23
24       <hr></hr>
```

```
2
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <%- include('../partials/head') %>
7   </head>
8   <body>
9     <%- include('../partials/nav') %>
10
11     <div class="container mb-3">
12
13       <h3 class="contentBoxTop"><%= user.username %></h3>
14
15       <form class="user-form" action="/users" method="post">
16         <fieldset disabled>
17           <div class="form-group row">
18             <label for="name" class="col-sm-3 col-form-label">Name</label>
19             <div class="col-sm-9">
20               <input class="form-control" type="text" id="name" name="name" value="<%= user.name %>">
21             </div>
22           </div>
23           <div class="form-group row">
24             <label for="email" class="col-sm-3 col-form-label">Email</label>
25             <div class="col-sm-9">
26               <input class="form-control" type="text" id="email" name="email" value="<%= user.email %>">
27             </div>
28           </div>
29         </fieldset>
30       </form>
```



```
31
32     <div>
33         <a class="btn btn-primary" href="/users">Back</a>
34         <a class="btn btn-primary" href="/users/<%= user.username %>/edit">Edit</a>
35         <form action="/users/<%= user.username %>?_method=delete" method="post" class="d-inline">
36             <a class="btn btn-primary" href="javascript:void(0)" onclick="confirm('Do you want to delete this?')?this.parentElement.submit():false">Delete</a>
37         </form>
38     </div>
39
40 </div>
41 </body>
42 </html>
```

```
public > css > master.css > .board-table .date
1  body {
2    font-family: 'Open Sans', sans-serif;
3  }
4  .breadcrumb-item {
5    font-size: 0.8em !important;
6  }
7  .ellipsis{
8    display: block;
9    width: 100%;
10   white-space: nowrap;
11   overflow: hidden;
12   text-overflow: ellipsis;
13 }
14
15 .board-table {
16   table-layout: fixed;
17 }
18 .board-table .date {
19   width: 100px;
20 }
21
22 .post-body{
23   white-space: pre-line;
24 }
25 .post-info{
26   font-size: 0.8em;
27 }
28 .user-form {
29   width: 400px;
30 }
```




My Website



← → ↻ ⓘ localhost:3000/users/new



Guest



My Website [Home](#) [About](#) [Board](#)

[Users](#) [Sign Up](#)

New User

Username*

Name*

Email

Password*

Password
Confirmation*

*Required

Submit



My Website



localhost:3000/users



Guest



My Website

Home

About

Board

Users

Sign Up

Users

test1



My Website



localhost:3000/users/test1



Guest



My Website

Home

About

Board

Users

Sign Up

test1

Name

test1name

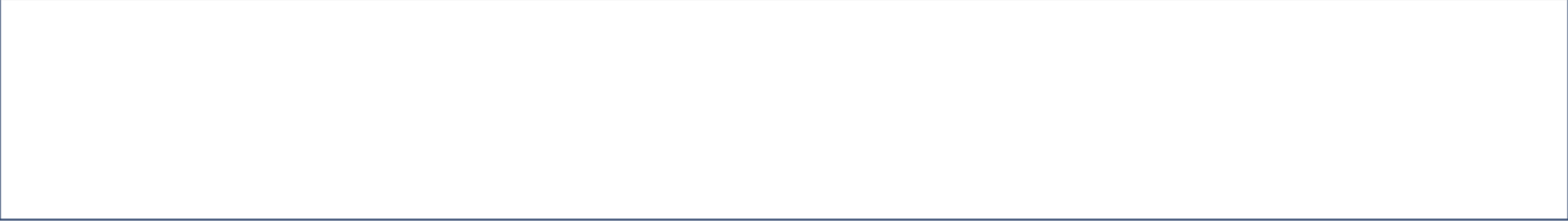
Email

test1email

Back

Edit

Delete



edit입니다. 회원가입할때는 password가 필수지만, 정보 수정시의 new password는 필수가 아닙니다. 다만 current password가 DB에 저장된 값과 일치해야합니다.

My Website

localhost:3000/users/test1/edit

Guest

My Website

HomeAboutBoard

UsersSign Up

Edit User

Current Password*

Username*

test1

Name*

test1name

Email

test1email

New Password

Password Confirmation

*Required

Back

Submit

마치며..

긴 강의 읽으시느라 수고하셨습니다. 이번 포스팅은 코드가 좀 많긴한데, 비슷한 내용을 자꾸 반복하면서 계속 글을 나눌 수 없어서 한번에 해봤습니다.





Edit User

Current Password*

Username*

test1

Name*

test1name

Email

test1email

New Password

Password
Confirmation

*Required

Back

Submit