NodeJS 3일차 Page Router

index.js

const express = require("express");

const app = express(); //서버생성

const fs = require("fs");

const ejs = require("ejs");

const bodyparser = require("body-parser"); //form형식 넘길떄 이용

// 사용자화면에 보여줄 데이터

const mockData = [

{ subject : "Mock Subject", writer : "Mock Writer", content : "Mock Content"}

];

app.get("/board/:id",function(req,res){

let id = req.params.id; //params뒤에 붙는것이 :id이다

const item = mockData[id]; //위에서 지정한 데이터를 얻어와서 item에 넣어준다

if(item != undefined){

fs.readFile("view/detail.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data,item /\* 리터널만 적어줄수있다.\*/));

//item을 data에 보내주겟다 페이지를 보여줄떄 item도 같이 보여줘리

});

}

});

app.get("/board",function(req,res){

//list.html 보여줄것이다.

fs.readFile("view/list.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data, {

"list" : mockData

}));

});

});

app.use(function(req,res){

res.type("text/html");

res.sendStatus(404);

});

app.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

**view폴더에 list.html생성**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>List</title>

</head> <body>

<table> <tr>

<th>제목</th>

<th>작성자</th>

</tr>

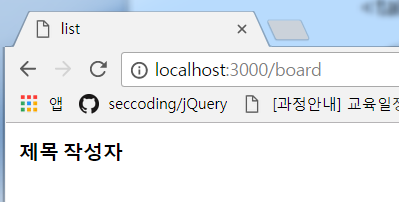
<% for (let i in list){ %> <!-- index.js에서 넘어온 list- -->

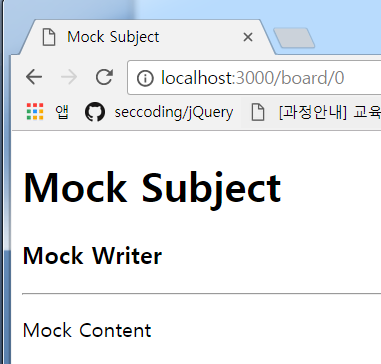
<tr>

<td><%=list[i].subject %></td>

<td><%=list[i].writer %></td>

</tr> <% } %> </table> </body></html>





**글쓰기작업**

/board/write 요청하면 app.get("/board/:id")의 미들웨어때문에 요청이 제대로 되지않을 것이다.

=> board/:id 보다 먼저 수행되도록 해주어야한다.

**index.js 수정**

const express = require("express");

const app = express(); //서버생성

const fs = require("fs");

const ejs = require("ejs");

const bodyparser = require("body-parser"); //form형식 넘길떄 이용

// 사용자화면에 보여줄 데이터

const mockData = [

{ subject : "Mock Subject", writer : "Mock Writer", content : "Mock Content"}

];

app.get("/board/write",function(req,res){

fs.readFile("view/write.html","utf-8",function(error,data){

res.type("text/html");

res.send(data);

//등록하는 페이지는 ejs일 필요는 없다.

});

});

app.get("/board/:id",function(req,res){

let id = req.params.id; //params뒤에 붙는것이 :id이다

const item = mockData[id]; //위에서 지정한 데이터를 얻어와서 item에 넣어준다

if(item != undefined){

fs.readFile("view/detail.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data,item /\* 리터널만 적어줄수있다.\*/));

//item을 data에 보내주겟다 페이지를 보여줄떄 item도 같이 보여줘리

});

}

});

app.get("/board",function(req,res){

//list.html 보여줄것이다.

fs.readFile("view/list.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data, {

"list" : mockData

}));

});

});

app.use(function(req,res){

res.type("text/html");

res.sendStatus(404);

});

app.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

**list.html 수정**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="utf-8">**

**<title>list</title>**

**</head>**

**<body>**

**<table>**

**<tr>**

**<th>제목</th>**

**<th>작성자</th>**

**</tr>**

**<% for (let i in list){ %> <!-- index.js에서 넘어온 list- -->**

**<tr>**

**<td><a href="/board/<%=i %>"><%=list[i].subject %></a></td>**

**<td><%=list[i].writer %></td>**

**</tr>**

**<% } %>**

**</table>**

**<a href="/board/write">글쓰끼</a>**

**</body>**

**</html>**

**write.html 생성**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

</head>

<body>

<form action="/board/write" method="post">

<input type="text" name="subject"><br>

<input type="text" name="writer" ><br>

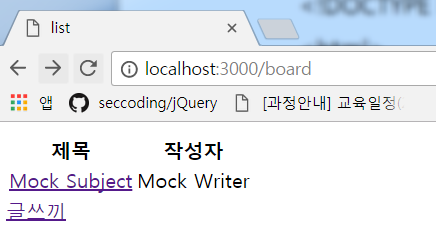
<textarea name="content" rows="8" cols="80"></textarea><br>

<input type="submit" value="등록">

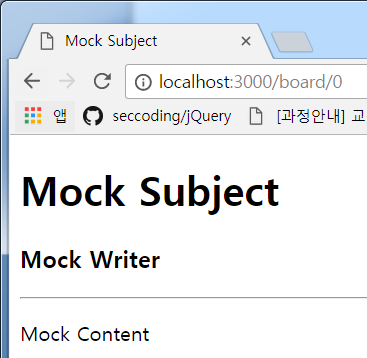
</form>

</body>

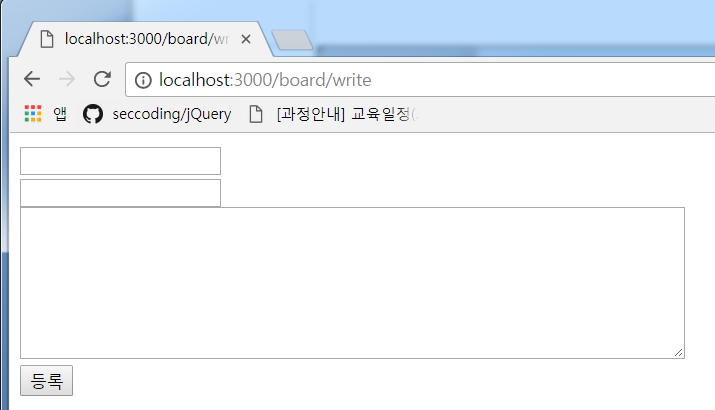
</html>



제목클릭시 =>



글쓰기 클릭시 =>



app.get("/board/:id",function(req,res,next){

else{

next();

}

next : 현재 미들웨어에서 요청할 것을 모두 했다. 다음 미들웨어로 넘어가라

}

write페이지에서 글을 썻을때 파라미터를 어떻게 보내겠느냐

Body Parser

var bodyParser = require("body-parser") 등록되어져 있어야 이용가능

**index.js**

const express = require("express");

const fs = require("fs");

const ejs = require("ejs");

const bodyparser = require("body-parser"); //form형식 파라미터 넘길떄 이용

const app = express(); //서버생성

//post로 전송된 데이터를 받아오는 미들웨어

//제일 먼저 실행되기떄문에 제일 위에 지정해주어야한다.

//req.body.name 과 같이 POST데이터를 얻어올 수 있다.

app.use(bodyparser.urlencoded({

"extended" : false

}));

// 사용자화면에 보여줄 데이터

const mockData = [

{ subject : "Mock Subject", writer : "Mock Writer", content : "Mock Content"}

];

app.get("/board/write",function(req,res){

fs.readFile("view/write.html","utf-8",function(error,data){

res.type("text/html");

res.send(data);

//등록하는 페이지는 ejs일 필요는 없다.

});

});

//post 데이터를 얻어온다.

app.post("/board/write",function(req,res){

const subject = req.body.subject;

const writer = req.body.writer;

const content = req.body.content;

//mockData에 POST데이터를 등록

mockData.push(

{

"subject" : subject,

"writer" : writer,

"content" : content

}

);

//페이지 redirect /board로 이동

res.redirect("/board");

});

app.get("/board/delete/:id",function(req,res){

const id = req.params.id; //delete/:id의 id값을 가져올수있다.

//글삭제

mockData.splice(id,1);//요청한거 1개지워라

//board페이지로 이동

res.redirect("/board");

});

app.get("/board/:id",function(req,res){

let id = req.params.id; //params뒤에 붙는것이 :id이다

const item = mockData[id]; //위에서 지정한 데이터를 얻어와서 item에 넣어준다

item.id = id;

if(item != undefined){

fs.readFile("view/detail.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data,item /\* 리터널만 적어줄수있다.\*/));

//item을 data에 보내주겟다 페이지를 보여줄떄 item도 같이 보여줘리

});

}

});

app.get("/board",function(req,res){

//list.html 보여줄것이다.

fs.readFile("view/list.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data, {

"list" : mockData

}));

});

});

app.use(function(req,res){

res.type("text/html");

res.sendStatus(404);

});

app.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

**write.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

</head>

<body>

<form action="/board/write" method="post">

<input type="text" name="subject"><br>

<input type="text" name="writer" ><br>

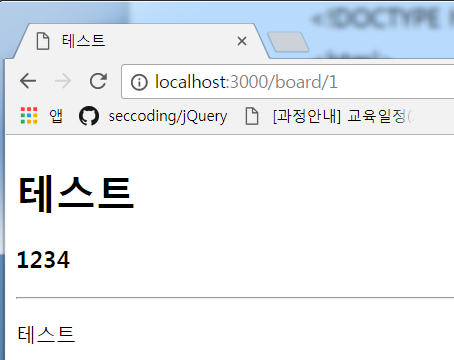
<textarea name="content" rows="8" cols="80"></textarea><br>

<input type="submit" value="등록">

</form>

</body>

</html>



**detail.html => 삭제 링크**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title><%= subject %></title>

<!-- 전달받은 데이터를 받아라-->

</head>

<body>

<h1><%= subject%> </h1>

<h3><%= writer%></h3>

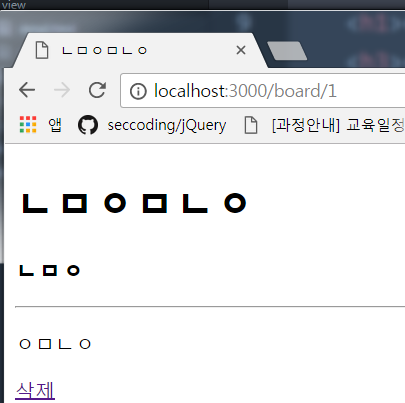
<hr/>

<p><%=content%></p>

<a href="/board/delete/<%=id%>">삭제</a>

</body>

</html>



Route 모듈화

보통 nodejs에 파일에 모듈화하여 분리시켜놓는 방법

수십개의 url을 모두 한파일에 관리하지않고 여러개의 파일로 관리하는방법

board폴더에 routes폴더생성 board.route.js 생성

**board.route.js**

const express = require("express");

const fs = require("fs");

const ejs = require("ejs");

const router = express.Router();

//router모듈만 가지고온다.

// 사용자화면에 보여줄 데이터

const mockData = [

{ subject : "Mock Subject", writer : "Mock Writer", content : "Mock Content"}

];

router.get("/write",function(req,res){

fs.readFile("view/write.html","utf-8",function(error,data){

res.type("text/html");

res.send(data);

//등록하는 페이지는 ejs일 필요는 없다.

});

});

//post 데이터를 얻어온다.

router.post("/write",function(req,res){

const subject = req.body.subject;

const writer = req.body.writer;

const content = req.body.content;

//mockData에 POST데이터를 등록

mockData.push(

{

"subject" : subject,

"writer" : writer,

"content" : content

}

);

//페이지 redirect /board로 이동

res.redirect("/board");

});

router.get("/delete/:id",function(req,res){

const id = req.params.id; //delete/:id의 id값을 가져올수있다.

//글삭제

mockData.splice(id,1);//요청한거 1개지워라

//board페이지로 이동

res.redirect("/board");

});

router.get("/:id",function(req,res){

let id = req.params.id; //params뒤에 붙는것이 :id이다

const item = mockData[id]; //위에서 지정한 데이터를 얻어와서 item에 넣어준다

item.id = id;

if(item != undefined){

fs.readFile("view/detail.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data,item /\* 리터널만 적어줄수있다.\*/));

//item을 data에 보내주겟다 페이지를 보여줄떄 item도 같이 보여줘리

});

}

});

router.get("/",function(req,res){

//list.html 보여줄것이다.

fs.readFile("view/list.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data, {

"list" : mockData

}));

});

});

exports.router = router;

//router속성을 바깥으로 뺴주겠다.

**index.js**

const express = require("express");

const bodyparser = require("body-parser"); //form형식 파라미터 넘길떄 이용

const app = express(); //서버생성

//post로 전송된 데이터를 받아오는 미들웨어

//제일 먼저 실행되기떄문에 제일 위에 지정해주어야한다.

//req.body.name 과 같이 POST데이터를 얻어올 수 있다.

app.use(bodyparser.urlencoded({

"extended" : false

}));

app.use("/board",require("./routes/board.route").router);

// 사용자화면에 보여줄 데이터

const mockData = [

{ subject : "Mock Subject", writer : "Mock Writer", content : "Mock Content"}

];

app.get("/board/write",function(req,res){

fs.readFile("view/write.html","utf-8",function(error,data){

res.type("text/html");

res.send(data);

//등록하는 페이지는 ejs일 필요는 없다.

});

});

//post 데이터를 얻어온다.

app.post("/board/write",function(req,res){

const subject = req.body.subject;

const writer = req.body.writer;

const content = req.body.content;

//mockData에 POST데이터를 등록

mockData.push(

{

"subject" : subject,

"writer" : writer,

"content" : content

}

);

//페이지 redirect /board로 이동

res.redirect("/board");

});

app.get("/board/delete/:id",function(req,res){

const id = req.params.id; //delete/:id의 id값을 가져올수있다.

//글삭제

mockData.splice(id,1);//요청한거 1개지워라

//board페이지로 이동

res.redirect("/board");

});

app.get("/board/:id",function(req,res){

let id = req.params.id; //params뒤에 붙는것이 :id이다

const item = mockData[id]; //위에서 지정한 데이터를 얻어와서 item에 넣어준다

item.id = id;

if(item != undefined){

fs.readFile("view/detail.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data,item /\* 리터널만 적어줄수있다.\*/));

//item을 data에 보내주겟다 페이지를 보여줄떄 item도 같이 보여줘리

});

}

});

app.get("/board",function(req,res){

//list.html 보여줄것이다.

fs.readFile("view/list.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data, {

"list" : mockData

}));

});

});

app.use(function(req,res){

res.type("text/html");

res.sendStatus(404);

});

app.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

**=> router로 분리를 시켜도 웹페이지가 잘동작한다.**

static 미들웨어 =>정적 자원 라우팅

**index.js 추가**

app.use(express.static(\_\_dirname + "/resources"));//static을 등록시켜라

//현재경로 + 리소스폴더

앞에 /js붙이면 스크립트에서 붙여주어야한다.

app.use("/js",express.static(\_\_dirname + "/resources"));

<script src="js/jquery-3.2.1.min.js" charset="utf-8"></script>

**write.html 추가**

<script src="/jquery-3.2.1.min.js" charset="utf-8"></script>

<script type="text/javascript">

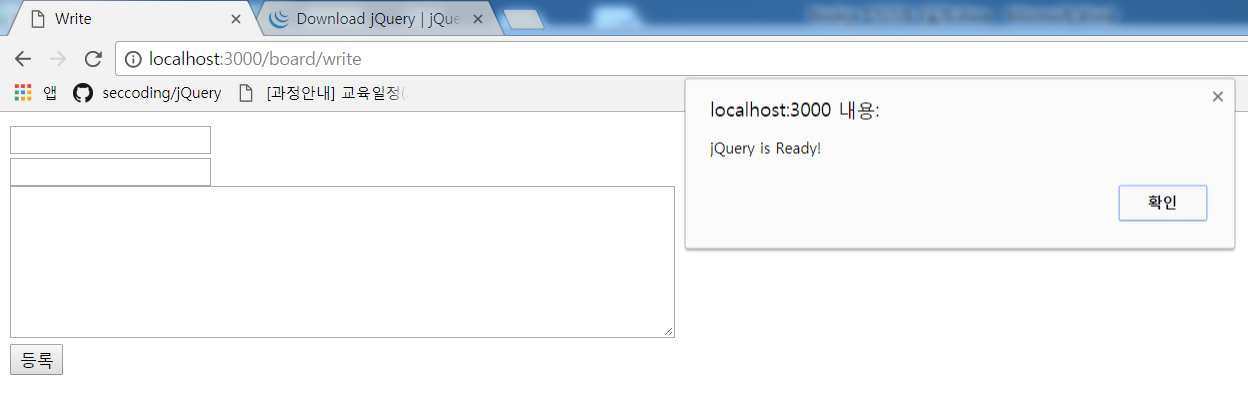
$().ready(function(){

alert("jQuery is Ready!");

});

</script>

글쓰기 클릭시 jquery function이 동작하여 alert창이 나타난다.



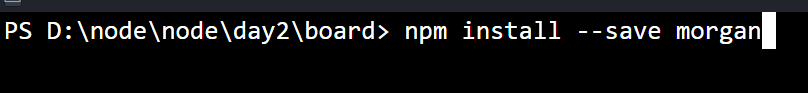
/resources 2개 지정된 url을 가질 수 없기 떄문에 각각 위치를 지정해주어야한다.

app.use("/js",express.static(\_\_dirname + "/resources/js"));//static을 등록시켜라

//현재경로 + 리소스폴더

app.use("/img",express.static(\_\_dirname+ "/resources/ing")); //폴더를 분리시켜주어야한다.

Morgan[Log] => 요청정보가 들어오면 로그를 작성시켜준다.



설치해주어야한다.

**index.js**

const express = require("express");

const morgan = require("morgan");

const bodyparser = require("body-parser"); //form형식 파라미터 넘길떄 이용

const app = express(); //서버생성

app.use(morgan("combined"));//모든요청에 로그를 다 찍는다.

app.use("/js",express.static(\_\_dirname + "/resources/js"));//static을 등록시켜라

//현재경로 + 리소스폴더

app.use("/img",express.static(\_\_dirname+ "/resources/ing")); //폴더를 분리시켜주어야한다.

app.use("/css",express.static(\_\_dirname + "/resources/css"));

//post로 전송된 데이터를 받아오는 미들웨어

//제일 먼저 실행되기떄문에 제일 위에 지정해주어야한다.

//req.body.name 과 같이 POST데이터를 얻어올 수 있다.

app.use(bodyparser.urlencoded({

"extended" : false

}));

app.use("/board",require("./routes/board.route").router);

// 사용자화면에 보여줄 데이터

const mockData = [

{ subject : "Mock Subject", writer : "Mock Writer", content : "Mock Content"}

];

app.get("/board/write",function(req,res){

fs.readFile("view/write.html","utf-8",function(error,data){

res.type("text/html");

res.send(data);

//등록하는 페이지는 ejs일 필요는 없다.

});

});

//post 데이터를 얻어온다.

app.post("/board/write",function(req,res){

const subject = req.body.subject;

const writer = req.body.writer;

const content = req.body.content;

//mockData에 POST데이터를 등록

mockData.push(

{

"subject" : subject,

"writer" : writer,

"content" : content

}

);

//페이지 redirect /board로 이동

res.redirect("/board");

});

app.get("/board/delete/:id",function(req,res){

const id = req.params.id; //delete/:id의 id값을 가져올수있다.

//글삭제

mockData.splice(id,1);//요청한거 1개지워라

//board페이지로 이동

res.redirect("/board");

});

app.get("/board/:id",function(req,res){

let id = req.params.id; //params뒤에 붙는것이 :id이다

const item = mockData[id]; //위에서 지정한 데이터를 얻어와서 item에 넣어준다

item.id = id;

if(item != undefined){

fs.readFile("view/detail.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data,item /\* 리터널만 적어줄수있다.\*/));

//item을 data에 보내주겟다 페이지를 보여줄떄 item도 같이 보여줘리

});

}

});

app.get("/board",function(req,res){

//list.html 보여줄것이다.

fs.readFile("view/list.html","utf-8",function(error,data){

res.type("text/html");

res.send(ejs.render(data, {

"list" : mockData

}));

});

});

app.use(function(req,res){

res.type("text/html");

res.sendStatus(404);

});

app.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

Cookie Parser : 쿠키를 생성 추출 위한 미들웨어



다운받아서 사용해주어야한다.

const cookieparser = require("cookie-parser");

app.use(cookieparser()); // 미들웨어 위치는 제약받지않는다.

**view폴더에 use 폴더생성 login.html생성**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

</head>

<body>

<form action="/user/login" method="post">

<input type="text" name="id" placeholder="ID"><br>

<input type="password" name="pwd"placeholder="Password"><br>

<!--아아디 저장하기 -->

<input type="checkbox" id="saveId" name="saveId" value="Y">

<label "saveId">아이디저장</label>

<input type="submit" value="Login">

</form>

</body>

</html>

**routes폴더에 user.route.js 생성**

const express = require("express");

const router = express.Router();

const fs = require("fs");

const ejs = require("ejs");

router.get("/login",(req,res) => { //fat arow function

fs.readFile("view/user/login.html","utf-8",(error,data) => {

res.type("text/html");

res.send(data);

});

});

router.post("/login",(req,res) => {

const id = req.body.id;

const pwd = req.body.pwd;

const saveId = req.body.saveId;

//html에 input데이터를 가지고올수 있다.

//cont data = req.body; //id,pwd,saveId를 한번에 가져올수 있다/

if(saveId == "Y"){ //체크를 했다면 쿠키를 만들어서 돌려줘라

//쿠키생성

res.cookie("save\_id", saveId);

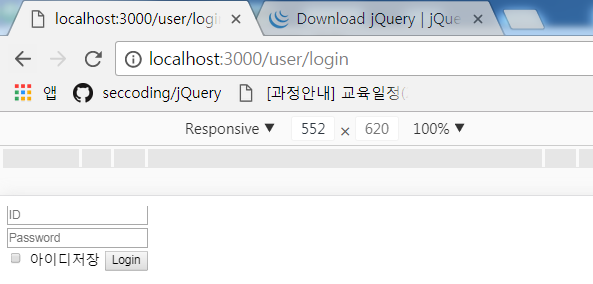
res.cookie("use\_Id", id);

}

res.redirect("/board");

});

exports.router = router;



**login.html 아이디 저장**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

</head>

<body>

<form action="/user/login" method="post">

<%

var name ="";

if(save\_id == "Y"){

name = user\_id;

}

%>

<input type="text" name="id" placeholder="ID" value="<%=name%>"/><br>

<input type="password" name="pwd"placeholder="Password"><br>

<!--아아디 저장하기 -->

<input type="checkbox" id="saveId" name="saveId" value="Y">

<label "saveId">아이디저장</label>

<input type="submit" value="Login">

</form>

</body>

</html>

routes폴더에 middlewares.js 생성

세션 사용하기 위한 미들웨어

router.use(session({

secret: "node-express-board",

resave: false,

saveUninitialized: true

}));

user.route.js

if(id == "admin" && pwd == "admin"){

const user = {};

user.id = id;

user.paswword = id;

req.session.USER = user;

}

로그아웃시 세션초기화

router.get("/logout",(req,res) => ){

req.session.destory();

res.redirect("/user/login");

}