



Seoul  
Software  
ACademy

# MVC

- MVC 패턴 -

---

**SeSAC 도봉 1기**  
**웹 풀스택 과정**

---

# mvc/ 폴더에 프로젝트 생성

```
mkdir mvc # 폴더 생성
```

```
cd mvc # 폴더 이동
```

```
npm init -y # 프로젝트 시작 명령어 (-y 옵션: package.json 기본 값으로 생성)  
# package.json에서 "main" 값을 index.js 에서 app.js로 변경 (진입점 파일명)
```

```
# app.js 파일 생성
```

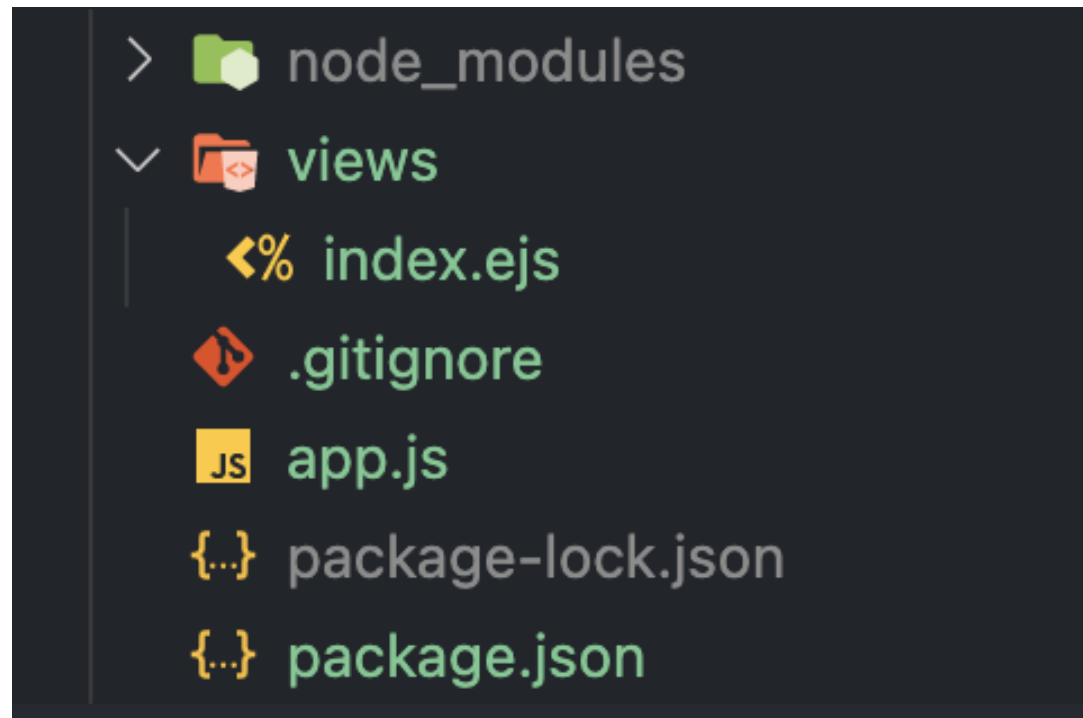
```
# views/index.ejs 파일 생성
```

```
# .gitignore 파일 생성
```

```
npm install express ejs # express와 ejs 패키지 설치
```

# mvc/ 기본 폴더 구조

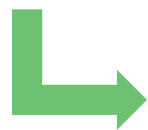
[미션] localhost:8000 접속시 메인 페이지 보이기



MVC 패턴 적용하기 전 폴더 구조

# MVC란?

- **Model View Controller**
- 소프트웨어 설계와 관련된 **디자인 패턴**
- MVC 이용 웹 프레임워크
  - PHP
  - Django
  - **Express**
  - Angular 등등



상황에 따라 자주 쓰이는 설계 방법을 정리한 코딩 방법론!!

# MVC 장단점

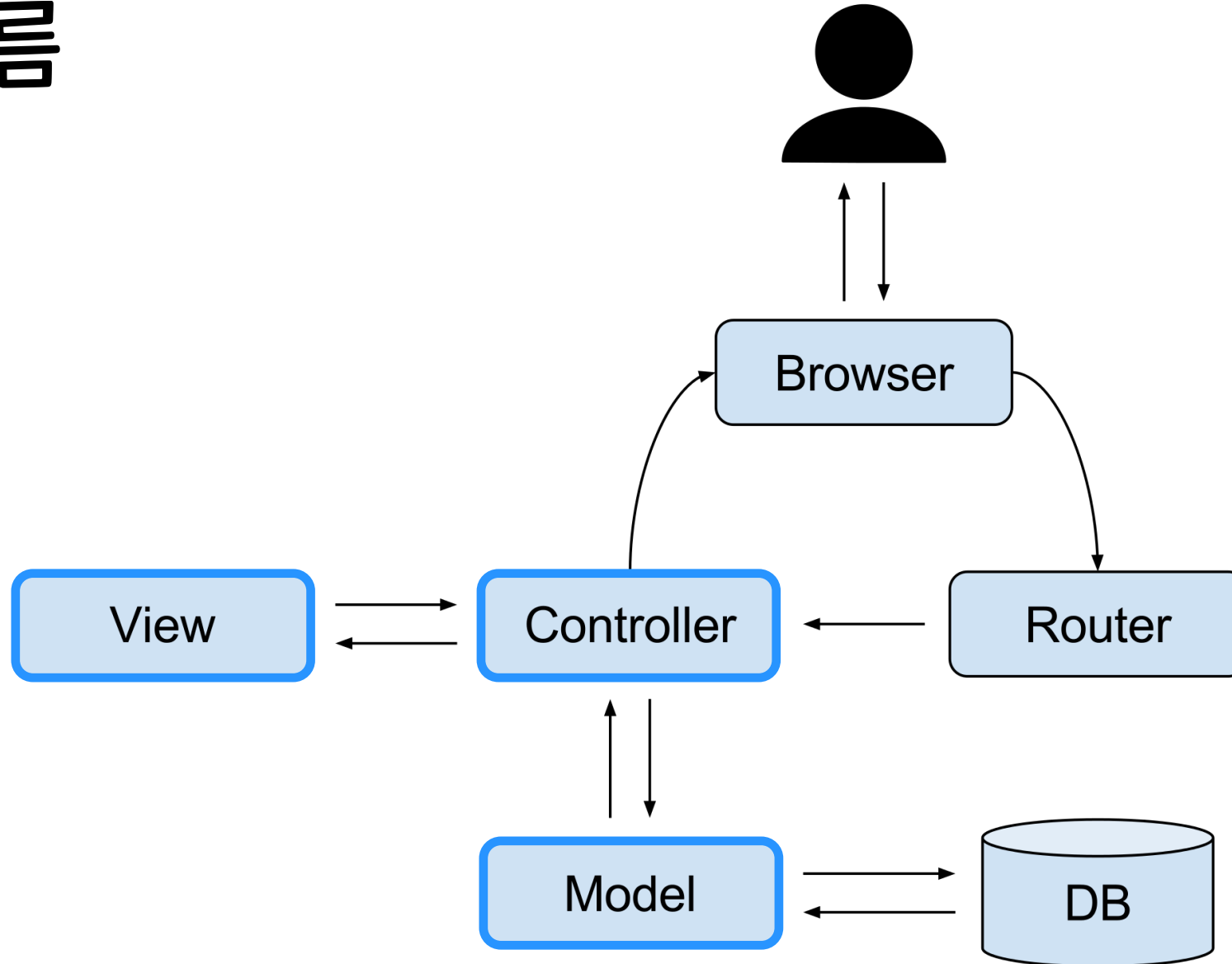
- 장점

- 패턴들을 구분해 개발한다.
- 유지보수가 용이하다.
- 유연성이 높다.
- 확장성이 높다.
- 협업에 용이하다.

- 단점

- 완벽한 의존성 분리가 어렵다.
- 설계 단계가 복잡하다.
- 설계 시간이 오래 걸린다.
- 클래스(단위)가 많아진다.

# MVC 흐름



# MVC 흐름

- **Model**

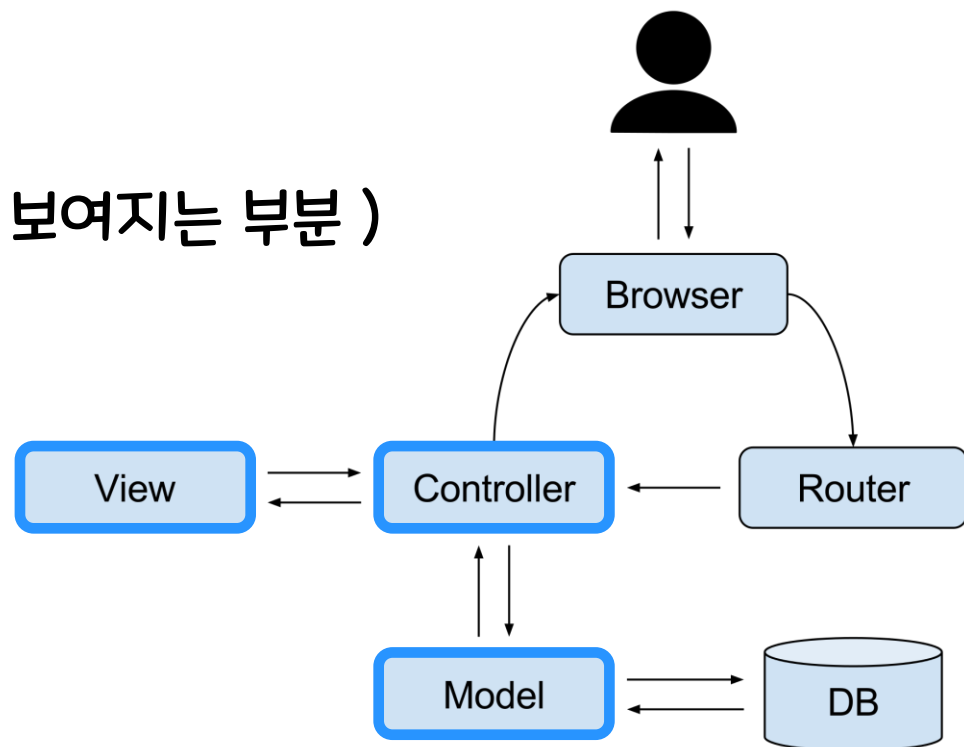
- 데이터 처리하는 부분

- **View**

- UI 관련된 것을 처리하는 부분 ( 사용자에게 보여지는 부분 )

- **Controller**

- View 와 Model을 연결해주는 부분

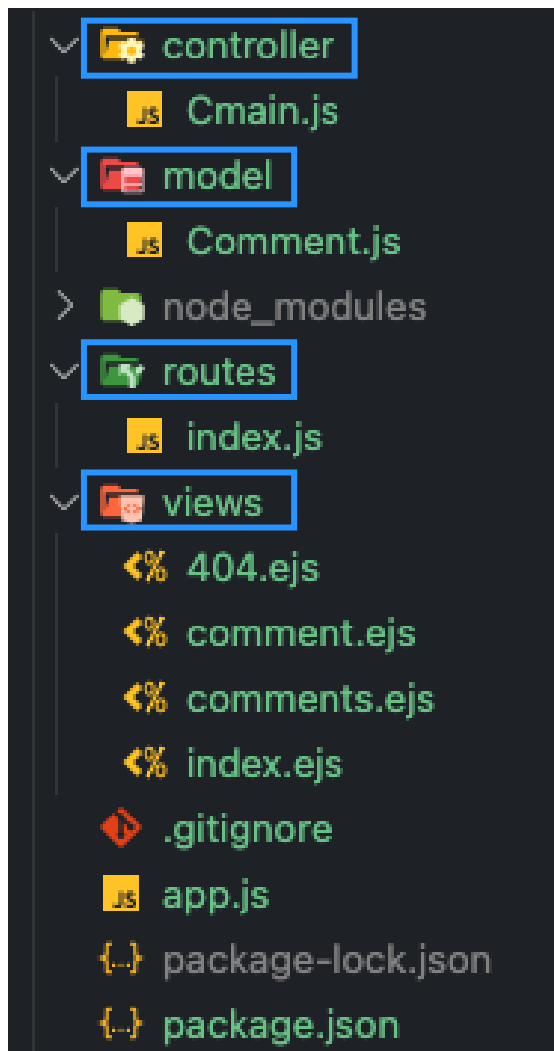


# Node.js MVC 구조



# 폴더구조

MVC 패턴을 적용한 폴더 구조



- view와 model 연결하는 부분
- 데이터 처리하는 부분
- 경로 설정하는 부분
- UI 관련 처리

# app.js

```
const express = require('express');
const app = express();
const PORT = 8000;

app.set('view engine', 'ejs');
app.use('/views', express.static(__dirname + '/views'));
app.use('/static', express.static(__dirname + '/static'));
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

const indexRouter = require('./routes'); // index는 생략 가능!
app.use('/', indexRouter); // localhost:PORT/ 경로를 기본으로 ./routes/index.js 파일에 선언한 대로 동작

// 404 error 처리
app.get('*', (req, res) => {
  // res.send('404 Error! 잘못된 주소 형식입니다.');
```

```
  res.render('404');
});

app.listen(PORT, () => {
  console.log(`http://localhost:${PORT}`);
});
```

- Router 불러오는 부분
- 위의 코드를 이용해 특정 시작 url의 역할 구분 가능

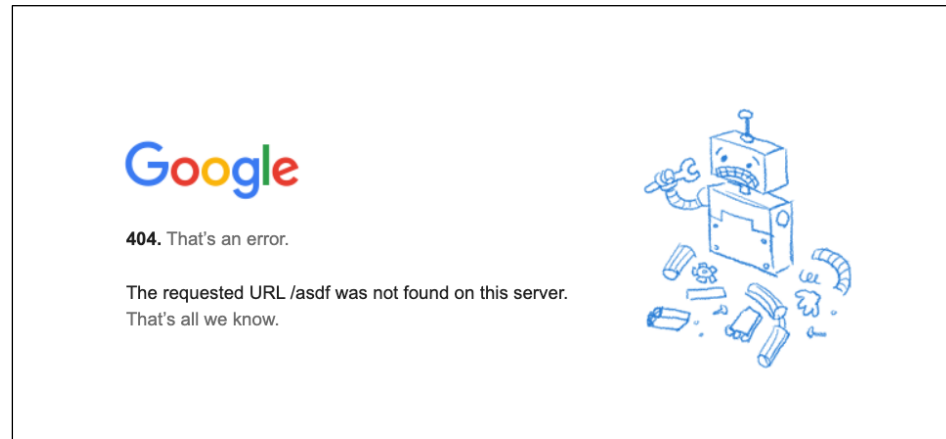
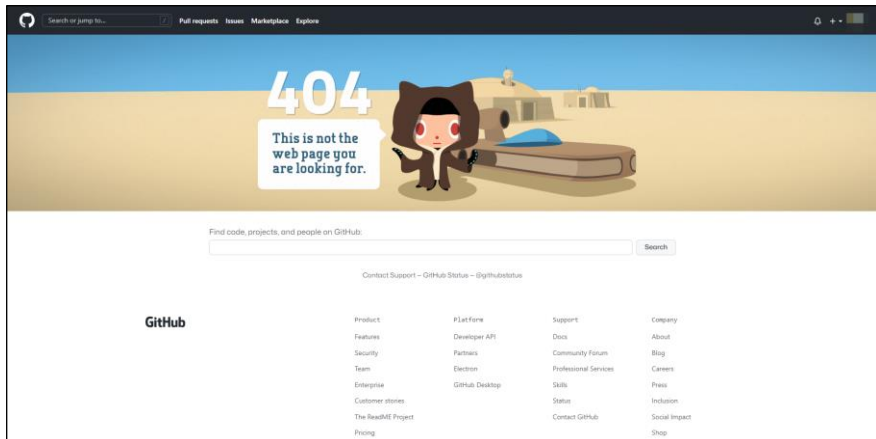
# routes/index.js

```
const express = require('express');  
const controller = require('../controller/Cmain.js');  
const router = express.Router();  
  
// localhost:PORT/  
router.get('/', controller.main); // GET /  
router.get('/commnets', controller.comments); // GET /comments  
  
module.exports = router;
```

- 경로를 controller와 연결해 설정 가능

# 참고) 404 Error 란?

- 404에러는 클라이언트가 잘못된 주소로 접속했을 때 발생하는 Error!



예시: GitHub/Google 404 에러 화면

# 참고) 404 Error 라우팅

```
app.get('*', (req, res) => {
  // res.send('404 Error! 잘못된 주소 형식입니다.');
```

```
  res.render('404');
```

```
});
```

- 맨 마지막 라우트로 선언
- \* : 그 외 나머지 주소는 모두(all) 잘못된 요청임을 사용자에게 알려야 함
- 클라이언트가 올바른지 않은 주소로 요청 시 Error 페이지 렌더링

# Controller/Cmain.js

```
exports.main = (req, res) => {
  res.render('index');
};

exports.comments = (req, res) => {
  res.render('comments');
};
```

- 경로와 연결될 함수 내용을 정의
- 경로와 연결되는 함수이기에 req 객체와 res 객체를 사용 가능

# model/Comment.js

```
exports.commentInfos = () => {
  return [
    {
      id: 1,
      userid: 'helloworld',
      date: '2022-10-31',
      comment: '안녕하세요^^',
    },
    {
      id: 2,
      userid: 'happy',
      date: '2022-11-01',
      comment: '반가워유',
    },
    {
      id: 3,
      userid: 'lucky',
      date: '2022-11-02',
      comment: '오 신기하군',
    },
    {
      id: 4,
      userid: 'bestpart',
      date: '2022-11-02',
      comment: '첫 댓글입니당ㅎㅎ',
    },
  ];
};
```

- ( 임시 ) DB에서 댓글 목록 데이터를 가져왔음을 가정
  - 댓글 목록은 배열로 가져옴
  - 각 댓글은 객체로 저장됨

# Controller - model

```
const Comment = require('../model/Comment');

exports.main = (req, res) => {
  res.render('index');
};

exports.comments = (req, res) => {
  console.log(Comment.commentInfos()); // 댓글 목록이 [ {}, {}, {}, {} ] 형태로 출력
  res.render('comments', { commentInfos: Comment.commentInfos() });
};
```

- 컨트롤러와 모델을 연결한다.

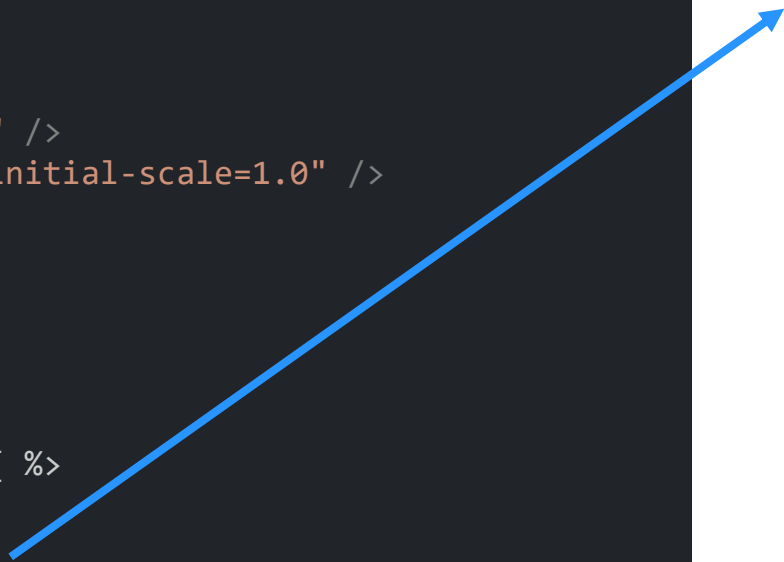


# views/comments.ejs

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>댓글 목록 보기</title>
  </head>
  <body>
    <h1>댓글 모음</h1>
    <a href="/">홈으로 이동하기</a>
    <ul>
      <% for (let i = 0; i < commentInfos.length; i++) { %>
        <li>
          <b><%= commentInfos[i].userId %></b> -
          <a href="/comment/<%= i + 1 %>"><%= commentInfos[i].comment %></a>
        </li>
      <% } %>
    </ul>
  </body>
</html>
```

```
router.get("/comment/:id", controller.comment);
```

routes/index.js



# controller/Cmain.js

```
exports.comment = (req, res) => {  
  console.log(req.params); // 라우트 매개변수에 대한 정보 담겨 있음  
  console.log(req.params.id); // id 고유 값  
  
  const commentId = req.params.id; // 댓글 id: url로 들어온 매개변수  
  const comments = Comment.commentInfos();  
  console.log(comments[commentId - 1]);  
  
  // :id 변수에 숫자가 아닌 값이 온다면 404 페이지  
  if (isNaN(commentId)) {  
    return res.render("404");  
  }  
  
  // 존재하지 않는 댓글 id 접근시 404 페이지  
  if (commentId < 1 || commentId > comments.length) {  
    return res.render("404");  
  }  
  
  res.render("comment", { commentInfo: comments[commentId - 1] });  
};
```

# views/comment.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>댓글 자세히 보기</title>
  </head>
  <body>
    <h1><%= commentInfo.userId %>님의 댓글입니다.</h1>
    <a href="/comments">댓글 목록 보기</a>

    <p>작성일: <%= commentInfo.date %></p>
    <p>댓글 내용: <%= commentInfo.comment %></p>
  </body>
</html>
```