express기반 프로젝트 생성

Node.js란

- Node.js는 자바스크립트를 브라우저 밖에서 실행할 수 있게 하는 런타임 환경입니다.
- 구글의 크롬 자바스크립트 엔진인 V8 엔진을 기반으로 구현된 일종의 Server-Side JavaScript 실행 환경입니다.
- Java언어가 각각의 OS운영체제의 JVM을 통해 실행되는 것처럼 Node.js는 웹브라우저에 종속적인 Javascript를
  각각의 OS운영체제에서 실행될 수 있도록 JavaScript Runtime 환경을 제공하고 있다.

Node.js설치하기

- Node.js 공식 홈페이지(https://nodejs.org/)에서 다운로드 받아 설치한다.

Node.js환경 구성하기

노드를 설치했으므로 이제 노드의 패키지 관리자인 npm을 사용하여 익스프레스 프로젝트를 자동으로 생성해 주는
express-generator 모듈을 설치한다.
참고로 익스프레스는 노드 기반에서 웹 애플리케이션 개발을 지원하는 프레임워크이다. npm의 기본 사용법은 아래와 같다

```
# 모듈을 전역으로 설치할 때 사용하는 방식이다. 일반적으로 특정 프로젝트가 아닌
# 전체 프로젝트에서 공통으로 사용할 수 있는 익스프레스 템플릿 생성 모듈인
# express-generator나 노드 프로젝트를 실행하는 nodemon 등을 전역으로 설치한다
npm install [모듈명] -g


# 현재 작업 중인 프로젝트에 모듈을 설치할 때 사용하는 방식이다. --save 옵션으로
# 현재 프로젝트의 package.json에모듈 이름과 버전을 추가한다. 이를 통해 현재
# 프로젝트가 사용하고 있는 모듈 이름과 버전을 알 수 있고 나중에 동일한
# 프로젝트를 다른 곳에 생성할 때 package.json을 복사해서 설치하면 되므로 매우 편리하다.
npm install [모듈명] --save

# 프로젝트 package.json에 작성된 모듈을 한 번에 설치하고 싶을 때 사용하는
# 명령어이다. package.json만 있다면 같은 버전의 모듈을 손쉽게 설치할 수 있다.<br>
npm install
npm i
```

1 프로젝트 디렉토리를 생성한 후 생성된 디렉토리로 이동한다.
2 익스프레스 프로젝트를 쉽게 생성해 주는 express-generator 모듈을 전역으로 설치한다. 전역으로 설치하면 다른 프로젝트에서도 사용할 수 있다.
사용법은 express -h로 살펴볼 수 있다.

```
express -h

npm install express-generator -g
```

3 express 명령어를 실행해서 프로젝트 디렉터리를 생성한다. 이때 템플릿 엔진은 ejs로 설치할 것이므로 -e 옵션을 추가한다.
이렇게 express-generator를 사용해서 프로젝트를 생성하면 기본적인 모듈들이 자동으로 package.json에 추가된다.

```
express -e cloud-app
```

4 package.json에 추가된 모듈을 설치한다. npm install 명령어를 사용하면 package.json에 선언된 모듈이 자동으로 설치된다.
이때 명령어는 web 디렉터리 아래에서 실행한다.

```
npm install
```

5 프로젝트를 실행한다. npm run start 명령어를 사용하면 프로젝트를 실행할 수 있다.
실행을 종료하고 싶다면 CTRL + C를 누르면 된다.

- 포트를 강제 종료하려면 아래와 같이 포트번호의 해당 pid번호 검색 후 포트번호를 kill한다. (npx kill-port 3000)

```
netstat -ano | findstr :3000 # 3000 포트를 사용하고 있는 pid번호 검색
taskkill /f /pid 185890 # pid번호를 사용하여 포트를 강제 종료
```

- 결과는 브라우저에 http://localhost:3000으로 접속해서 확인할 수 있다.
  노드는 기본적으로 3000번 포트에서 동작하도록 되어 있다.
  다른 포트 번호로 변경하려면 C:₩project₩cloud-app₩bin 디렉터리의 www 파일의 내용을 수정해야 한다.

6 npm start로 노드를 실행하면 소스 코드를 수정한 것이 반영되지 않는다. 그래서 이런 경우를 위한 여러 모듈이 있는데
그중에서 간단히 사용할 수 있는 노드몬을 설치한다.

```
npm install nodemon -g
```

- 프로젝트 디렉터리에서 nodemon이라고 실행하면 소스 코드가 수정될 경우 수정을 감지하고 다시 시작하는 것을 확인할 수 있다.
  따라서 소스 코드를 변경할 때마다 서버를 재 시작해야 하는 번거로움을 줄일 수 있다.

```
C:₩project₩cloud-app>nodemon
```

- nodemon 실행 오류 시에는 권리자 권한으로 PowerShell을 실행하여 실행한다.

```
C:₩windows₩system32>Set-ExecutionPolicy Unrestricted  #(Resticted를 Unrestricted
로 변경한다.)
```

프로젝트 기본 디렉토리

1 bin 디렉터리 : bin 디렉터리에는 www 이름의 파일이 한 개 존재한다. 이 파일은 확장자가 없지만, 내부는 노드가 서버로서 동작하기 위한 기본적인 코드가 자바스크립트로 작성되어 있다.
또한 서버를 시작할 포트가 지정되어 있다.
2 node_modules 디렉터리 : 이 디렉터리는 npm install을 실행하면서 생긴 디렉터리이다.
그래서 package.json에 선언되어 있는 모듈 과 이 모듈을 실행가기 위해서 필요한 의존관계의 모듈이 설치되어있다.
3 public 디렉터리 : 이 디렉터리에는 이미지, 자바스크립트, 스타일시트와 관련된 디렉터리로 구성되어있다.
4 routes 디렉터리 : 이 디렉터리에는 index.js와 users.js 파일이 존재한다. 기본적으로 생성되는 파일이며, 라우트를 처리하기 위한 코드 가 작성되어 있다.
5 views 디렉터리 : 뷰를 처리하는 파일이 위치한 곳이며, 프로젝트를 생성할 때 템플릿 엔진으로 ejs를 지정했으므로 이 디렉터리에는 확장자가 ejs인 파일이 위치한다.

프로젝트 기본 파일

1 bin/www에 위치한 www 파일
이 파일에서는 기본적으로 노드를 실행할 포트를 설정하고 웹서버를 생성하는 코드가 작성되어 있는 자바스크립트 파일이다

```
 //사용할 모듈지정
var app = require('../app');
var debug = require('debug')('cloud-app:server');
var http = require('http');
//환경 설정에 지정된 포트가 있다면 이를 사용하고 없다면 지정한 포트(3000)를 사용하도록
설정
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);
//HTTP 서버 생성
var server = http.createServer(app);
//지정된 포트 상에서 응답 대기하도록 설정
server.listen(port);
```

2 app.js
이 파일은 웹 애플리케이션을 위한 기본적인 설정을 가지고 있는 파일이다. 모듈을 로딩하고 템플릿 엔진을 설정하며,
라우트를 설정한다. 파일 상단부분에는 사용할 모듈을 로딩 하는 코드가 작성되어있다.
외부 모듈을 해당 파일에서 사용하고 싶다면 require() 함수를 호출해야 한다.

```
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
// 라우트 코드를 로딩 하는 코드이다 require() 함수를 사용하며, 로딩 한 라우트 함수들을
//  지정된 변수로 사용할 수 있게 해 준다
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
// 익스프레스 객체를 app 변수로 선언한다. 앞으로 app을 통해 익스프레스 함수를 호출할 수
// 있게 된다
var app = express();
// 라우트를 설정하는 코드이다. 여기서 말하는 라우트는 url 경로의 뒷부분을 의미한다.
// 그래서 '/'와 관련된 라우트는 routes 폴더의 index 파일에 작성된 라우트 함수를
// 통해 처리되며,'/users'와 관련된 라우트는 users 파일에 작성된 라우트 함수를
// 통해 처리된다
app.use('/', indexRouter);
app.use('/users', usersRouter);
// 익스프레스에서 사용할 템플릿 엔진을 설정하는 코드이다. __dirname은 현재 디렉터리를
//  의미하며 path.join() 함수는 경로를 연결하는 기능을 한다. 그래서 현재 디렉터리에
// 있는 views 디렉터리를 의미한다. 그리고 app.set() 함수는 익스프레스의 환경을
// 설정하는 함수이다.
// 첫 번째 라인은 경로를 두 번째 라인은 템플릿 엔진의 종류를 설정한다. <br>
// 참고로 app.get() 함수도 정의되어 있으며, 이 함수는 값을 반환한다.<br>
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
```

3 routes/index.js

'app.js'에서 라우트를 처리할 때는 app.get()나 app.post() 함수를 사용한다. 그리고 별도 파일에서 라우트 함수를 작성할 경우에는 express.Router() 함수를 통해 호출하며 별도 파일에서 함수를 사용할 수 있도록 module.export=router를 추가해야 한다.

index.js에 작성된 router.get() 함수는 URL 경로가 '/'일 때 호출되는 함수이며 실제로 호출되어 실행될 코드는 function(req, res, next){ } 함수에 작성하면 된다.

req는 HTTP 요청 객체에 대한 정보이며, res는 HTTP 응답 객체에 대한 정보를 가지고 있다.

```
var express = require('express');
var router = express.Router();

router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});
module.exports = router;
```