**리눅스(CentOS 7) 노드(node.js) 서버 운영기**

2016년 3월

1. 계기
2. 기본 환경 및 서버 구성
   1. 깃(github.com)
   2. 노드 서버 구성(with Express)
   3. 리눅스 환경 구성(CentOS 7)
3. 프로젝트 개발
   1. HR App(from ‘MEAN 스택 입문’)
4. 운영 및 관리

**1. 계기**

리눅스 기반 서버에 관심을 가져, 리눅스 기본 서적인 ‘이것이 리눅스다’를 모두 읽고 비교적 쉽게 운영할 수 있는 웹 서버를 개발하고 운영해 봐야겠다고 마음을 먹었다. 가장 흔히 사용되는 아파치와 php를 이용하여 서버를 구성하는 것도 좋은 방법이 되겠지만, 아파치와 php를 다뤄본 적이 없고, 이번 기회에 리눅스 환경에 더욱 적응하고 빠르게 웹 서버를 구성하기 위해서 node.js와 node.js의 express, MySQL, ejs를 이용하여 서버를 구성하기로 결정하였다.

서버 OS는 Centos 7-64bit 이고, cloudatcost.com에서 제공하는 호스팅을 이용하기로 했다.

**IP: 45.62.238.200  
PORT**(opened)**: 2222(ssh), 3000(node-service, web)**

-참고 도서  
‘이것이 리눅스다’, ‘모던 웹을 위한 Node.js 프로그래밍’, ‘풀스택 개발자를 위한 MEAN 스택 입문’, ‘git을 이용한 버전 관리’

\*’모던 웹을 위한 Node.js 프로그래밍’은 2016년 3월 기준으로 express의 용법이 크게 바뀌어 node.js 기본, mysql, ejs 부분만 의미가 있음.

\* 명령어, 요구사항, 개선가능사항

**2. 기본 환경 구성**

용산 미군 기지 내 사무실에서 프로젝트의 대부분을 진행했기 때문에 개발 환경이 그리 편하지 않았다. 개발은 IDE 웹 환경인 c9.io에서 진행하였는데, c9.io에서 지원하는 실행 환경이 우분투이었기 때문에 개발 환경에서 잘 실행되는 어플리케이션이 잘 실행되지 않는 경우가 종종 있었다. 우분투에서 cloudatcost(실제 실행 환경, CentOS)로 ssh로 접속하더라도 두 서버 모두 호스팅을 했기 때문에 너무 느려 개발에 집중할 수 있는 상황이 아니었다. 따라서 주중에는 우분투에서 실행되는 최소한의 구성만을 갖춰 개발에 초점을 맞추고 http://github.com을 이용해 저장소에 올린 뒤 주말을 이용해 CentOS 환경 관련 작업을 진행했다.

**A. 깃(github.com)**

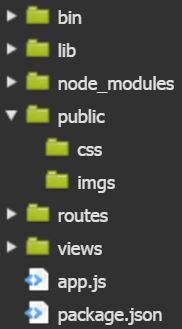
이번에 git을 사용하는 것은 단순 저장소의 성격이 강하나, 더욱 복잡한 프로젝트나 다수의 개발자가 참여하는 경우에 효과적일 것 같은 브랜치 기능을 사용해보고 싶다.

git address: git@github.com:llnntms/FirstWeb

* .gitignore

.gitinore 파일을 만들어 내부에 쓰여진 경로나 파일을 git이 처리하는 것을 막을 수 있음. 노드 서버 같은 경우에는 nodemodule/ 이 해당된다(npm install로 쉽게 설치 가능하고, 상대적으로 무거우므로, package.json은 그에 대한 목록을 나타냄).

**B. 노드 서버 구성(with Express)**

* Express

express를 설치(npm install express --save)한 뒤, express --ejs로 기본 프레임웤을 구성(사실 처음에는 혼자 디렉터리와 파일들을 구성하고 이리저리 옮겨 보았는데, 구성하다 보니 기본 구성에 점점 더 가까워지는 듯하다.여러 시행착오를 거쳐 디렉터리 구조는 다음과 같이 결정되었다.)

bin은 express의 www파일이 위치하고, 이는 express 앱의 시작점으로 사용된다(npm start의 시작점). 이 시작점은 package.json에서 설정이 가능하다.

lib는 개발자가 정의한 js 모듈이나 라이브러리를 가져오는 공간이다. 노드 앱은 npm install ~ 으로 모듈을 쉽게 node\_modules에 설치하므로, 일반적으로는 개발자 정의 모듈 자바스크립트 파일이 위치한다.

public은 앞단에서 사용되는 css나 이미지들이 위치한다.

routes는 페이지 라우팅에 사용되는 자바스크립트 파일들이 위치한다. 여기에 위치한 자바스크립트 파일들은 라우팅을 진행한 후, views에 위치한 ejs파일에 데이터를 넘겨주며 ejs파일을 랜더링한다.

views에는 화면에 표시해주는 파일들(ejs를 사용함)이 위치한다.

app.js는 www가 불러오는 실제 express 서버 앱이며, 여러 설정, 에러 핸들러 등을 기술한 자바스크립트 파일이다.

package.json은 앱에서 사용하는 노드 모듈들의 버전정보를 json형태로 저장한 파일이다. express로 기본 앱을 설치했다면 이미 여러 모듈의 정보들이 들어가 있다.

**C. 리눅스 환경 구성(CentOS 7)**

* **ssh 포트 변경 (http://www.liberiangeek.net/2014/11/change-openssh-port-centos-7/)**

ssh 기본 포트(22)를 계속 사용하면 해킹의 위험이 높아진다 하여, 포트를 변경함

1. #vi /etc/ssh/sshd\_config  
2. #Port 22 -> Port 2222  
3. #semanage port –a –t ssh\_port\_t –p tcp 2222  
4. #firewall-cmd --permanent --zone=public --add-port=2222/tcp  
 (방화벽에 예외로 tcp 2222번 포트를 추가함 https://www.conory.com/note\_linux/42477)  
5. #systemctl restart sshd.service

설치되어 있지 않을 경우 semanage -> yum -y install policycoreutils-python  
 firewall -> yum -y install firewalld

* **웹 서버 포트를 예외에 등록**

방화벽을 설치하고 enable, start 시키니 연결이 안 되는 현상이 발생, 실험용으로 사용하고 있던 3000번 포트를 방화벽 예외로 등록 했다.

#firewall-cmd --permanent --zone=public --add-port=3000/tcp

참고로, --zone=public은 생략이 가능(기본 설정이 public).

* **MySQL 환경 설정**

1. root 계정 비밀번호 변경 - #mysql(root 계정으로 접속), use mysql, UPDATE user set password=password(‘비밀번호’) where user=’root’, **#systemctl restart mysql**

2. 비밀번호 변경 후 접속 - mysql -uroot -p -> 비밀번호 입력

3. mysql user 추가 - 현재는 root 계정에서 운영 중.. 보안 상의 문제가 있을 것이라고 판단될 때 계정을 새로 만들어 서비스할 예정

**3. 프로젝트 개발**

**A. HR App(from ‘MEAN 스택 입문’)**

첫 프로젝트는 ‘MEAN 스택 입문’이라는 책에 나와있는 예제와 동일한 기능을 가지면서 로그인 기능과 가입 기능을 추가한 노드 서버이다. 책 예제에는 MongoDB와 Angular.js를 사용하였다. MongoDB의 NoSQL과 Angular.js의 MVC 모델 기반 양방향 데이터 바인딩은 강력해 보였지만, 익숙한 MySQL과 ejs로 해당 부분을 대체하였다. 또, Angular.js에 관해서는 프론트엔드에 그렇게 관심이 가지 않고 ejs보다 복잡한 부분이 많아 추후 적절한 사용처가 있을 때 다시 한번 보기로 결정했다.

HR App은 직원들의 리스트를 관리할 수 있는 웹 앱이다. 이 어플리케이션에서는 로그인 기능이 있고, 로그인을 하게 되면 직원 리스트, 직원 상세, 팀 리스트, 팀 멤버 등을 볼 수 있게 된다. 이 때 로그인은 express-session을 이용하여 1시간 동안 세션으로 유지한다. 모든 정보(직원, 팀, 계정 등)들은 MySQL 서버에 저장되며, ejs파일로 이루어진 모든 페이지에서 저장된 정보를 불러와 동적으로 표현 가능하도록 하였다.

* 디렉터리 구성

기본적인 구성은 2-B. 노드 서버 구성에서 기술한 구조와 일치한다.

lib/: 노드 mysql 모듈을 이용해 여러가지 쿼리를 함수로 실행해주는 모듈을 모아놓은 디렉터리이다. 다른 파일에서 require하여 사용할 수 있도록 exports를 이용한다.

public/: public용 파일들이 위치한다. 뷰페이지를 위한 css와 css에서 사용하는 이미지 파일들이 위치해 있다.

routes/: 라우팅용 자바스크립트 파일들이다. 이 파일들은 권한 별로 분류되었다. public.js는 홈, 로그인, 가입 페이지에 대한 라우팅(누구나 접근 가능), hr.js는 앱의 실제 기능과 관련된 라우팅(로그인이 되어있어야 접근 가능), manager.js는 계정 관리와 가입 신청 승인과 관련된 라우팅(level 0인 계정만 접근 가능)관련 라우팅 함수들이 정의되어있다. 라우팅 후에는 필요한 데이터와 함께, 또는 데이터 없이 적절한 ejs파일을 랜더링하여 요청에 응답한다.

views/: 각각의 페이지 뷰를 위한 ejs파일들을 모아놓은 디렉터리이다. routes에서 처리한 요청에 대해 랜더링되어 동적으로 제공된다.

* 기능 구현

라우팅 - 라우팅은 express에서 제공하는 라우터 기능을 사용하였다. 라우팅 관련 파일들은 routes디렉터리에 위치해 있고, 권한별로 3개의 파일로 분리했다(public.js, hr.js, manager.js). public.js는 로그인하지 않은 상태에서도 접근할 수 있는 라우트(홈, 로그인, 로그아웃, 가입), hr.js는 로그인을 해야 접속할 수 있는 라우트(직원 정보, 추가, 수정, 삭제 팀 정보 - 기본적인 CRUD 기능들), manager.js는 level 0으로 등록된 관리자만 접속할 수 있는 라우트(계정 관리, 가입 승인)을 라우팅한다. hr.js와 manager.js는 각각의 CheckAuth함수를 통해 권한을 확인하고 권한이 충분치 않을 경우 적당한(로그인이 필요하다면 로그인 페이지로, 권한이 없는 경우에는 홈으로) 페이지 우회시킨다. 이를 통해 권한을 가지지 않은 사용자가 특정 페이지에 접근하는 것을 막는다. 라우팅 관련 에러가 발생하면(존재하지 않는 페이지 접근 등) next로 error를 넘겨 에러페이지로 응답한다.

로그인, 세션 기능 – 로그인 기능은 앱을 초기화 하면서부터 시작된다. 앱을 초기화하면서 express-session을 사용하여 세션을 활성화하고, 로그인 세션을 초기화한다. 로그인 세션을 초기화할때는 이미 로그인 세션이 존재하면 그 값을 res.locals에 담아 ejs뷰 파일에서 사용할 수 있도록하고, 세션이 존재하지 않으면 res.locals에 undefined을 담는다. 이것이 세션이 유지되고 있는지 ejs뷰 파일에서 구분하는 기준이 된다. 로그인이 되어있지 않으면 홈의 네비게이션바에서 employee와 team 메뉴를 보여주지 않고(home.ejs) url을 통해 /employees나 /teams에 접근하더라도 로그인 페이지인 /login으로 리다이렉트 시켜준다(home.js). 또 홈에서 로그인이 되었다는 의미로 로그인 된 계정명을 보여준다. 실제 로그인 기능은 /login에서 제공하는데, 폼을 입려하고 log-in 버튼을 누르면 lib/accountDB.js 모듈에서 로그인이 성공했는지 실패했는지 콜백함수로 인자를 전달한다. 로그인이 성공했으면 정보를 세션으로 저장하고, /로 리다이렉트 시킨다. 로그인이 실패했다면 로그인이 실패했다는 것을 알리는 인자와 함께 login.ejs를 랜더링하여 보내준다. 이때 login.ejs에서 login failed라는 라벨을 함께 출력한다. 로그인이 성공한 뒤에는 모든 hr관련 메뉴가 보이게 되며, 접근할 수 있게 된다. 만약 로그인한 계정이 관리자 계정(level 0)이라면 Accounts, Signup Request 메뉴를 같이 보여준다. 홈에 있는 로그아웃 버튼을 누르면 세션을 모두 삭제한 뒤 홈으로 리다이렉트 시킨다. 로그아웃을 하지 않으면 세션이 1시간 동안 유지되어 다음에 접속했을 때 로그인된 상태의 홈페이지를 볼 수 있다.

가입 기능 – 가입은 자동으로 이루어지지 않고 HRAPP을 사용하려는 사용자가 가입 신청을 제출하고 관리자가 요청을 승인하거나 거부하는 형식으로 이루어진다. 가입신청을 제출하면 Account 테이블과 SignupRequest 테이블에서 이메일의 존재 여부를 확인한 후, 존재하지 않는다면 제출된 이메일과 비밀번호가 DB의 SignupRequest 테이블로 입력된다. 이때, 비밀번호는 암호화(sha256)되어 테이블에 저장된다.

계정 확인 및 가입 승인 – 관리자(level 0)로 등록된 계정으로 로그인할 경우 나타나는 Accounts와 Signup Request에서 각 메뉴를 통해 관리자는 모든 계정의 이메일 주소와 그 계정의 레벨을 볼 수 있고, 들어온 가입 요청을 승인하거나 거부할 수 있다. 이때, 승인이면 SignupRequest 테이블에서 Account 테이블로 정보를 옮겨주고, SignupRequest 테이블에서 해당 정보를 삭제한다. 이는 트랜잭션으로 묶어 무결성을 보장하도록 하였다. 거부이면 SignupRequest 테이블에 대해 삭제 쿼리만을 실행한다. 이는 accountDB.js 에서 일어난다.

DB 관련 기능 - DB내의 테이블은 Account, Employee, Team, SignupRequest로 총 4개이다. Account와 SignupRequest는 위에 기술한 것과 같고, Employee와 Team은 HR 앱의 실제 데이터를 저장하는 공간이다. DB에 접근하여 정보를 읽거나 수정하는 작업(CRUD 작업)은 lib디렉터리에 hrDB.js와 accountDB.js에서 이루어 지는데, 여기에서 미리 작성해둔 SQL문과 입력 받은 값을 적절히 결합하여 MySQL에 넘겨주면, 라우팅 파일에서 실행 결과를 인자로 전달하며 ejs페이지를 랜더링한다. 따라서 사용자는 최신의 정보를 볼 수 있게 된다.

보안 – 사용자에게 받은 입력, 즉 쿼리에 대입되어 mysql에서 실행되는 모든 입력은 sql injection 공격의 가능성을 줄이기 위해 node-mysql 모듈이 지원하는 escape 기능을 사용한다(node-mysql이 지원하는 구문 ‘?’을 사용하여 적용함). 또 모든 비밀번호는 sha256 해시 함수를 이용해 64자리 해시값의 형태로 표현하여 저장한다.

* 부족하거나 생각해 볼 점

가장 부족하다고 생각되는 점은 역시 **디자인**이다. 아무리 기능적인 면에 치중했다고 해도, 사용환경이 뒷받침되지 않으면 그 기능을 사용하기 어려운 것이 현실이다. 다음 프로젝트 때는 웹 디자이너가 직접 개발한 css나 디자인 등을 구입해 사용하는 것도 고려해 볼 수 있을 듯 하다.

**여러 명의 사용자가 동시에 사용**하고 있을 경우 일어날 일에 대해 대비가 되어있지 않다. 이번 프로젝트를 진행하면서 가장 현실과 동떨어져있다고 생각한 부분이다. 실제 웹을 쓰는 사용자는 한 명일 수 없기 때문에 이런 부분에 대해 대비가 필요한데 어떤 방식이 있는지 어떤 방식을 쓰는 것이 좋은지 더 공부해봐야 할 것 같다.

실제 **자바스크립트**를 제대로 공부해 본 적이 없어서 매번 이런 식으로 쓰는 것이 맞는지 헷갈릴 때가 많이 있었다. 자바스크립트를 조금 더 심도 있게 공부한다면 코드의 질적인 면이 많이 향상될 것이라고 생각한다. 나중에 기회가 되어 이 프로젝트를 더욱 보기 좋고, 빠른 구조로 만들 수 있다면 좋을 것 같다.

**리눅스 보안** 기능도 아쉬운 부분이 많으므로 추후 리눅스 서버에 관한 보안 책을 살펴본 뒤 보안 기능을 추가하고, 리눅스로 마스터 네임서버를 구축하여 도메인 명으로 접속할 수 있게 업데이트하겠다.

그리 어렵지 않은 프로젝트였지만, 처음 해보았던 만큼 **시간이 생각보다 오래 걸렸다**. 아무래도 여러 책이나 블로그를 보면서 프로젝트를 진행했기 때문에 각기 다른 구조를 따와 가장 적절한 구조로 변환하는데 시간이 오래 걸린 것 같다. 다음부터는 이번과 같거나 비슷한 구조로 프로젝트를 진행해야겠다.

**B. 채팅 웹 앱**

추가중..