**INsoft Cloud Consulting Team Internship.**

2016.03.07

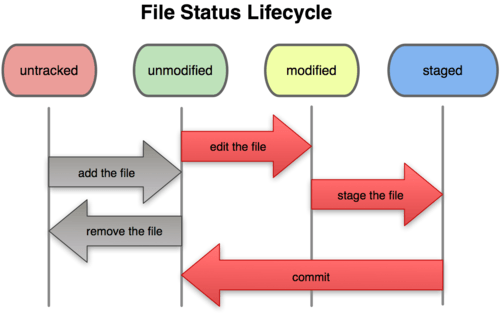
* Trello : 협업 툴.
* Gitlab : 버전 관리 툴, 사람 수에 관계없이 무료로 이용.
* github : private 저장소를 만들기 위해서는 비용을 지불.
* Git의 원격저장소(Gitlab.com)에서 clone 하고 수정한 뒤 다시 push 하는 법을 수행.
* Git push 가 되지 않는 문제 발생.
* JDK를 설치 후 환경변수를 설정하였지만 cmd 상에서 javac 명령어가 되지 않는다.

2016.03.08

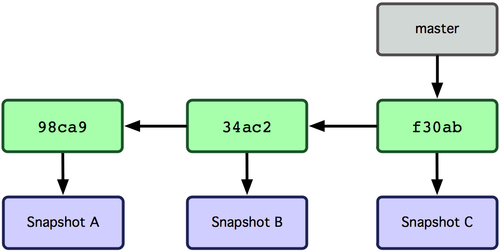
* 메이븐이란?
  + 라이브러리 관리 및 빌드 툴.
  + POM.xml 에 등록된 정보를 통해 자동으로 라이브러리를 다운로드 및 업데이트 해준다.
* JDK 설치 후 환경 설정 JAVA\_HOME 변수는 jre가 아니라 jdk로 해주어야 javac 명령어를 cmd에서 사용 가능하다(java complier는 jdk에 있기 때문).
* VCM(Version Control Management)
  + CVS
  + SVN
  + GIT
* GIT
  + Three states of GIT
    - Working directory -> staging area -> .git directory
    - GIT 명령어
    - git clone [url] : get a copy of an remote repository.
    - git status : see state of files that (un)tracked or modified.
    - git add : tracking a new file and staging.
* 빠른 영단어 해석을 위한 Google dictionary 크롬 익스텐션 설치.

2016.03.09

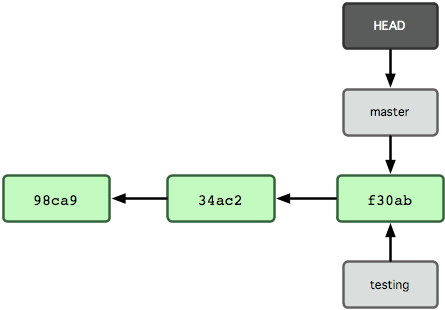
* GIT file
  + Tracked file : 이미 스냅샷에 포함돼 있던 파일.
    - Unmodified : 수정되지 않은 파일.
    - Modified : 수정된 파일.
    - Staged : index된 파일.
* Untracked file : 스냅샷에 포함돼 있지 않던 파일.
* 처음 clone을 하면 당연하게도 모든 파일이 tracked 이면서 unmodified 상태.



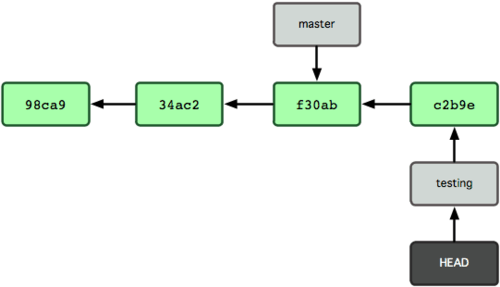
* .gitignore file : 어떤 파일들은 git이 자동으로 추가하거나 untracked 파일이라고 보여줄 필요가 없다. 그런 파일들은 .gitignore 파일 안에 무시할 패턴을 적는다. 표준 Glob 패턴을 사용해 작성한다.
* GIT 명령어
  + git commit -m “message” : “message”를 가지고 commit.
  + git commit –a -m “message” : -a 옵션을 사용하면 tracked 상태의 파일을 staging area를 거치지 않고 바로 commit.
  + rm [파일명] : 단순 파일 삭제, staging area 에는 남아있다.
  + git rm [파일명] : 파일을 삭제하면서도 staging area 에서도 삭제.
  + git rm –cached [파일명] : staging area 에서만 제거, 워킹 디렉토리에 있는 파일은 남겨둔다.
  + git fetch [remote-name] : 리모트 저장소에서 데이터를 가져온다. 로컬에는 없지만 리모트 저장소에는 있는 데이터를 모두 가져온다. 자동으로 merge 하지는 않는다.
  + git pull : 리모트 저장소에서 데이터를 가져와 현재 작업하는 코드와 merge 시킨다.
  + Git push : 서버로 push. 다른 사람이 push 한 후에 push 하려고 하면 push 할 수 없다. 먼저 다른 사람이 작업한 것을 가져와서 merge 한 후에 push 가능.
* GIT branch
  + 기본적으로 master branch.



* + git branch [branch name] : branch 생성.



* + HEAD 포인터가 현재 브랜치를 가리킨다.
  + git checkout [branch name] : 브랜치 이동. HEAD 포인터가 다른 브랜치를 가리킨다. 그 후 commit 하면 아래와 같다.



* myso-cloud 분석
  + spring framework
  + 자바 플랫폼을 위한 오픈소스 애플리케이션 프레임워크.
  + IoC 기술을 통해 애플리케이션의 느슨한 결합을 도모.
  + 의존성 주입 지원.
    - Xml에 작성된 명세서를 보고, IoC 컨테이너가 각 객체를 생성하고, 값을 주입해준다.
  + Spring Data JPA
    - Spring Data Java Persistence API.
    - 메소드 이름으로 Query 생성(규칙을 지켜 메소드 이름 설정).
    - Repository 인터페이스에 메소드 생성.
  + Spring data?
    - 오픈소스 프로젝트.
    - Spring 관련 어플리케이션에 “새로운 데이터 접근 기술(Non-Relational DB, 맵리듀스 프레임워크, 클라우드 기반 데이터 서비스)”을 쉽게 사용하도록 도와주는 역할.
    - 기존 RDBMS 역시 지원.
    - 데이터베이스 종류에 따라 수많은 세부 프로젝트로 나뉜다(하위 프로젝트에 JPA가 존재).
  + EJB?
    - Enterprise Java Beans.
    - SUN에서 개발한 자바 컴포넌트 개발을 위한 서버 측 컴포넌트 아키텍처 또는 모델.
    - JavaBeans?
    - EJB와 별개의 개념.
    - Java Component 제작 표준, 제작 표준에 따라 각각의 component를 bean이라고 한다.
  + JPA?
    - Java Persistence API.
    - Query 자동 생성, 실행.
  + CRUD
    - 대부분의 컴퓨터 소프트웨어가 가지는 기본적인 데이터 처리 기능인 Create, Read, Update, Delete를 지칭하는 말.
  + Guava
  + 구글에서 개발한 자바 오픈 소스 라이브러리.
  + 유틸리티 함수와 클래스 제공.

2016.3.10

* ORM 도입 이유
  + 관계형 데이터 -> 객체형 데이터로 사용.
  + 데이터를 쿼리가 아닌 객체지향 데이터로 사용
  + 쿼리를 짜는데 많은 시간이 소요되므로 생산성의 문제가 있었고, ORM은 개발자가 업무 프로세스 개발에 집중할 수 있도록 해준다.
* 하이버네이트
  + JPA를 구현한 ORM 프레임워크 중 하나.
  + Application (ORM API 사용) -> ORM Framework(JDBC API 사용) -> RDB
* Maven dependency scope
  + Compile : 기본영역, 컴파일 시점에 필요하고 배포할 때도 포함이 되어진다.
  + Provided : 컴파일 시점에는 필요하지만 배포시점에는 불필요한 라이브러리로 JDK, servlet API, Java EE API 등이 해당된다.
  + Runtime : 컴파일 시점에는 필요없지만 runtime 시점에 필요한 라이브러리.
  + System : repository에서 검색을 하지 않고 명시된 위치에서 jar를 이용한다. 그 외 provided와 유사함.
  + Test : 테스트 컴파일과 실행 시점에만 사용됨.
  + Import : maven 2.0.9 이후에서만 적용. 다른 pom 설정 파일에 정의되어 있는 의존 관계 설정을 현재 프로젝트로 가져온다.
* Slf4j
  + 스프링을 사용하면 기본적으로 JCL(Jakarta Commons Logging)를 사용하게 되는데 JCL이 실제 로거를 선택하는 시점이 런타임이라 클래스로더 문제라던가 런타임 시 오버헤드가 생길 수 있는데 이것을 개선한 구현체 SLF4J로 갈아타면 그런 문제걱정을 덜 수 있다.
* Application Layer
  + Presentation layer
    - 어플리케이션 중에서 화면을 담당하는 부분.
    - 보통 GUI 부분이라 칭함.
    - 화면을 통하여 사용자로부터 입력을 받거나 보여주는 역할.
    - 요구되는 기술
      * 화면 구성요소를 나타내는 방법.
      * 화면으로부터 받은 명령을 처리하는 방법.
      * 화면 컨트롤의 이벤트를 처리하는 방법.
      * 다른 Layer에게 입력받은 명령이나 데이터를 전달하는 방법.
  + Business logic layer
    - 사용자로부터 받은 입력을 기준으로 주어진 로직에 따라 각종 일 처리를 담당하는 부분.
    - 일을 처리하는 로직을 통상 비즈니스 로직이라 칭함.
    - 요구되는 기술
      * 하나의 로직 상태를 여러 화면과 공유하는 방법.
      * 데이터 layer에 데이터를 주고 받는 방법.
      * 여러 프로세스를 병렬로 처리하는 방법.
  + Persistence layer
    - 데이터를 영구적으로 저장하거나, 저장된 장소에서 데이터를 가져오는 역할을 담당.
    - 요구되는 기술
      * 데이터를 파일 및 데이터베이스에 저장, 수정, 삭제하는 방법.
      * 데이터베이스에 연결하고 연결을 유지하는 방법.
  + Integration layer
    - 어플리케이션이 다른 부분과 연동하거나, 통신을 할 필요가 있을 경우, 이에 대한 처리를 담당하는 부분.
    - 요구되는 기술
      * 통신 프로토콜, 분산 객체 호출 방법, 비동기식으로 외부 시스템에 내용 전달하는 방법.

2016.3.14

* Sourcetree
  + Git GUI 제공.
* Titan DB
  + 그래프 데이터베이스
* Myso-cloud 분석 시작.
  + RequestMapping 어노테이션에서 /console은 어디에?
  + 뷰를 Return 리턴하는 과정에서 overall.sub는 어디에?

2016.3.15

* Myso-cloud 분석.
  + @Controller
    - 클래스 위에 선언하여 클래스를 콘트롤러로 사용함을 명시.
* @RequestMapping
  + DispatcherServlet이 RequestMapping되어 있는 url을 보고 해당하는 controller 함수를 호출해 준다. Return되는 string 값은 다시 DispatcherServlet으로 전달되고 DispatcherServlet은 해당하는 string 값의 이름을 가진 jsp 파일을 호출한다.
  + 즉, Spring은 요청 서블릿을 DispatcherServlet 한 개를 가지고 수행한다(물론 여러개를 만들수 있다).
* EL
  + Expression Language.
* Apache titles
* DashBoardMain 등의 ~~~Main 클래스 들은 MainLayoutHelper 클래스를 상속하여 부모 클래스의 함수를 재사용하고 있다.
* MainLayoutHelper 클래스 에서 UserHelper클래스의 getCurrentUserId()를 사용해서 유저의 아이디를 반환받아 Model에 저장한다..
* Jsp에서 HTML class, JSTL의 반복조건을 EL로 사용하기 위해 Model에 attribute로 저장시키는 것이다.
* 클라우드 서비스 웹 화면의 서비스 관리 -> 서비스 로그 보기 에서 검색 기능에 대한 코드를 찾지 못하였다. Search 함수를 호출하지만 실질적으로 검색을 하는 코드나 쿼리는 찾지 못함.
* Overall.sub는 layout-tiles.xml에 definition으로 정의되어 있었다. Layout-tiles.xml은 layout-servlet.xml에 등록되어 있다.
* j\_spring\_security\_check
  + Spring Security.
  + Spring Security에서 j\_spring\_security\_check 에서 반환된 사용자 정보를 이용. Password가 일치하는지 확인.
  + Password가 일치하는 경우, authentication-success-handler-ref에 정의된 handler를 이용해서 page의 이동이나 json 응답을 반환.
  + Password가 일치하지 않는 경우, authentication-failure-handler-ref에 정의된 handler를 이용해서 page의 이동이나 json 응답을 반환.
  + Web.xml에 반드시 securityFilterChain과 ContextLoaderListener가 선언되 있어야 한다.
* Spring-security.xml
  + Spring security 로그인을 구현하기 위해 필요한 xml 명세.
* <welcome-file-list>
  + Web.xml에 명세.
  + Url에 file명을 명시하지 않을 시 자동으로 디폴트 파일을 보여주는 역할(index.html).

2016.3.16

* 클라우드 인스턴스 생성에 대해서 학습.
* 인스턴스 연동 API 및 그 외의 정보에 대해서 알기 위해 인스턴스 region 정보를 어떻게 가져오는지 추적.
  + Hashtable.java 의 get 메소드를 통해서 region 정보를 가져오는 듯 하다.
* 디자인 패턴의 Factory 기법.
* Mysocloud 구번전과 신버전의 병합
  + IntelliJ IDEA 에서 신버전 프로젝트를 import 하질 못하겠다.

2016.3.17

* IntelliJ 폰트 색깔별 상태
  + Red : workspace file
  + Green : staged file
  + Blue : modified file
  + White : commited file
  + Gray : deleted file
* pom.xml 에서 source가 error가 난다.
* Autowired 어노테이션이 되지 않는 error.
* Class duplicate 문제가 발생하여 찾아보니 src와 target에 중복 클래스가 존재했다.
* Pom.xml 의 tomcat7-maven-plugin error는 지원하지 않는 <source>,<target>,<encoding> tag를 사용하였기 때문에 삭제하였다.
* Class duplicate 문제는 target 폴더의 commit 때문에 발생하였는데 target 폴더는 컴파일러가 자동으로 생성하는 파일이었기 때문에 제거해도 무방하다.

2016.3.18

* Revert : staged file을 workspace file로 강등시키는 작업.
* Error는 없지만 File name이 red인 문제.
* File name이 red인 문제는 src를 복사 붙여넣기 하는 과정에서 기존에 없었던 파일들이 추가되었고 이 파일들은 staged 되지 않았기 때문에 workspace에 잔재해 있었다. 따라서 이 파일들을 add 시켜주고 commit 하여 모든 파일을 repository에 저장하였다.
* dev 브랜치를 taejintest 브랜치와 merge 성공.
* Myso-cloud 인스턴스 추적했지만 javascript showList메소드에서 loadService 메소드를 호출하는 코드를 찾지 못하였다.

2016.3.21

* Myso-cloud 코드 분석.
* Myso-cloud 인스턴스 추적 중 자바스크립트의 showList에서 loadService(null, “User”)의 형식으로 호출함을 확인.
* loadService 메소드를 추적하다보면 컨트롤러로 json 요청을 보낼 때가 있다. 이 때 getAll, getOne 메소드를 사용하여 데이터를 가져오는데 이 메소드들의 구현체를 찾지 못하였다.
* 검색 기능(ctrl + H)에서 file mask를 xml로 해서 검색하면 getOne, getAll 메소드들의 sql문이 정의되 있는 것을 확인할 수 있었고, 이는 MyBatis의 학습이 요구된다.
* MyBatis를 사용하면 xml에 정의된 sql 구문을 통해 DB에 쿼리를 날릴 수 있다.
* MyBatis의 #{변수명}으로 parameter에 정의된 속성으로 접근.
* 인스턴스 종료 버튼을 추적해서 MySQL에서는 Data를 지우는 코드를 발견했지만 실질적으로 클라우드 벤더에 존재하는 인스턴스는 어떻게 지우는가?
* 클라우드 벤더에 존재하는 인스턴스를 지우는 코드는 구현되어 있지 않았다.
* MySQL 스키마 덤프 추출 및 복원.
  + DUMP 추출
    - mysqldump -u[사용자아이디] -p 데이터베이스명 [테이블명] > 저장될 파일명
  + 복구
    - mysql -u[사용자아이디] -p [디비명] < 덤프파일명
  + DB 스키마만 추출.
    - -d 옵션을 사용.
    - mysqldump -u mysql\_admin -p -d edu > edu\_db.sql
* MySQL 비밀번호 변경
  + UPDATE user set password=password(“\*\*\*\*\*\*”) where user = ‘root’;
  + flush privileges;
* MySQL dump 복원 작업 시
  + ERROR 1067 (42000) at line 200: Invalid default value for ‘VERSION\_DATE’
  + Error 발생.
  + VERSION\_DATE의 CURRENT\_TIMESTAMP는 5.6.5 이상의 버전부터 지원이 되기 때문에 문제가 발생한 것.

2016.3.22

* MySQL 5.7.11 버전 설치.
* Utf8 변환 작업
  + my.ini를 아래와 같이 수정.
    - [client]
    - default-character-set=utf8
    - [mysqld]
    - init\_connect=SET collation\_connection = utf8\_general\_ci
    - init\_connect=SET NAMES utf8
    - #default-character-set=utf8 # deprecated
    - character-set-server=utf8
    - collation-server=utf8\_general\_ci
* MySQL mysocloud dump 복원 성공.
* Tomcat 구동 시 웹 화면이 나오지 않는 문제 발생. 원인 불명.

2016.3.23

* 사용자 계정 이름이 한글이면 문제 발생의 소지가 있어서 영어로 변환하는 작업을 하였다. 이를 위해 pc초기화를 진행.
* 초기화된 pc에 기본 세팅을 다시 시행.
* IntelliJ를 사용해 웹 프로젝트 생성 시도.
* 첫 화면이 띄어지지 않는 문제 발생.

2016.3.24

* IntelliJ 자동 import 및 정리 단축키 : Alt + enter
* 기본 maven, spring mvc, web project 성공 및 Tomcat 실행 성공
* helloWorld로 RequestMapping이 되지 않는 문제 발생.
* IntelliJ 자동 정렬 : Ctrl + Alt + L
* helloWorld로 ReqestMapping 되지 않는 문제는 xsi:Location을 xsi:Loacation으로 잘못 타이핑 했기 때문에 component-scan을 할 수 없었기 때문에 발생한 것이었다.
* SignUp page front end 구현.

2016.3.25

* HttpServletRequest 클래스는 javax.servlet.http 패키지에 속함.
* myBatis를 사용한 db insert가 되지 않는 문제 발생.
* DB insert 문제는 @Autowired를 선언하지 않아 UserService 객체가 주입이 되지 않았던 것이다.
* 경량형 회원가입 페이지 구현.
* @Autowired : springframework에서 자동 setter의 역할.

2016.3.28

* 이메일 보내기 및 MyBatis 학습.
* SMTP Error가 발생.
* 이메일을 보내기 위해선 SMTP server가 필요하다. 이는 외부 [www.makebuildngo.com의](http://www.makebuildngo.com의) SMTP server를 사용해 해결하였다.
* 5.7.1 relaying denied error 발생.
* [www.makebuildngo.com은](http://www.makebuildngo.com은) 외부 접속을 차단하고 있었다. 이를 [smtp@gmail.com](mailto:smtp@gmail.com)으로 사용해서 문제 해결.
* 알고리즘 퀴즈 문제 수행.

2016.3.29

* Quiz 1 수행
* Quiz 2 메모리 낭비 문제.

2016.3.30

* Quiz 2 에 대한 문제 접근 방법이 잘못 되었다. 문제 속의 규칙을 찾지 않고 모든 것을 다 조사해야 한다는 생각에 빠져 있었다.
* Quiz 2 수행.

2016.3.31

* Spring 기본적인 학습.
* 회고
  + 장점
    - Git을 학습.
    - 프레임워크를 접해본 것.
    - 자유로운 분위기.
    - 여유로운 출근 시간.
    - 질문에 잘 대답해주신다.
    - 아침 회의, 저녁 회고가 유익하다.
    - Quiz 가 신선했다.
* 단점
  + 사무실 공기가 너무 탁하다.
  + 많은 프레임워크를 알게 되었는데 결국 용어만 알게 되고 실습이 부족했다.
  + 건물이 공사중이라 너무 시끄럽고 냄새가 진동.
  + 조명이 너무 어두움, 특히 노란 불빛.
  + 개인적인 작업만 한 것 같다.
* 앞으로의 방향
  + 공동 프로젝트(협업)이 필요하다.
  + 서로 개발한 코드를 리뷰하는 코드리뷰 시간을 가져서 개발방법에 대해 의견을 교환하고 조언해주는 시간을 가지면 좋을 듯.

2016.4.1

* Spring 학습.
* Hibernate 학습.
* Hibernate 의 작동 원리 및 사용 방법에 대한 이해 부족.

2016.4.4

* GCS 혁신과제 회의.
* uCloud 가상 인스턴스 생성.
* Myso Cloud 를 통한 uCloud 가상 인스턴스 생성.
* 인텔리j, 하이버네이트 연동 실패.

2016.4.5

* AWS EC2 인스턴스 생성.
* MYSQL Version Issue 가 생겨서 우분투 버전으로 EC2를 다시 생성.
* MYSQL table doesn’t exist 문제 발생.
* MySQL table doesn’t exist error는 MySQL의 대소문자 구분 때문에 발생했음. My.cnf 파일의 [mysqld] 하단에 lower\_case\_table\_names = 1 로 처리하여 대소문자 구분을 없앴다.

2016.4.6

* 하이버네이트 연동이 실패되어서 일단 보류.
* Spring Boot 학습
* H2에 대한 학습이 더 필요할 듯 하다.

2016.4.7

* H2 연동.
* CRUDRepository 인터페이스를 활용한 기본적인 Spring Boot Data JPA 수행.
* CRUDRepository 의 구현체를 찾을 수 가 없다. 구현체 없이 실행되는 것이 의문이 든다.

2016.4.8

* Myso Cloud 사용법 숙지.
* Repositiroy 는 구현체가 없이 작동된다. 메서드명에 기반에 쿼리를 날려주기 때문에 구현체가 필요하지 않다.
* DataSourse에 대한 이해 필요.

2016.4.11

* Spring Boot 와 MySQL 을 연동하는데 오류 발생.
* Spring 으로 개발했던 Demo 사이트를 Spring Boot 를 사용해 개발.
* Spring Boot 와 MySQL 연동 성공. @Entity 가 붙은 클래스의 @Id 어노테이션을 붙여주지 않아 생겼던 오류.

2016.4.12

* Spring Boot 를 이용해서 데이터베이스의 정보를 가져와 출력.
* Ajax를 사용해 json 형태로 데이터 값을 받아와 화면에 출력.
* 쉘 스크립트 기본 학습.
* 정규 표현식

|  |  |  |
| --- | --- | --- |
| 문자 | 설명 | ex |
| **^** | 라인 또는 문자열의 시작과 부합 | ^1234 : 1234로 시작 |
| **$** | 라인 또는 문자열의 끝과 부합 | 1234$ : 1234로 끝남 |
| **\*** | 바로 앞의 문자를 나타내며 그 문자가 없거나 하나 이상을 의미 | s\*e : e, se, sse, ssse….. |
| **?** | 바로 앞의 문자가 없거나 있음을 의미 | fath\*r : fatr, fathr….. |
| **.** | 임의의 한 글자만 | a.c : aac, abc, acc, adc….. |
| **+** | q로 앞의 문자가 하나 이상 | a+c : aac, aaac, aaaac ….. |
| **[ ]** | 각괄호[ ]안에 있는 문자열 중 하나의 문자만 | [a-z] : 소문자 한 글자를 포함하는 문자열과 일치 |
| **[^]** | 원하지 않는 문자를 제외한 나머지 문자 | [^ab] : a와 b를 제외한 문자열 |
| **{ }** | 중괄호{ } 앞에 있는 문자나 문자열 개수 | a{2}c : aac |
| **|** | OR를 의미 | a|b : a OR b |
| **\** | 정규식에 사용되는 문자를 그대로 적용 | \^ : ^ |
| **\b** | 단어의 시작 또는 끝과 부합 |  |
| **\B** | 단어의 시작 이나 끝이 아닌 것에 부합 |  |

2016.4.14

* 쉘 스크립트 학습.
* Head, tail, find, grep 명령어 학습.
* 하이버네이트 session.query 학습.

2016.4.15

* Spring data JPA persistence context 학습.
* 쉘 스트립트 학습.
* 쉘 스크립트 세미나 준비.

2016.4.18

* 쉘 스크립트 세미나 및 피드백 받은 부분 수정.
* Spring boot documentation reading, 특별한 것은 없었다.
* Hibernate 혹은 data JPA를 이용해 복잡한 데이터베이스 쿼리를 다루는 법이 필요하다.

2016.4.19

* Mysocloud spring boot 화 작업.
* 회원가입 된 user 목록 출력.
* 회원가입에서 문제 발생.
* Mysocloud mc\_t\_user table에 필요없는 데이터 스키마가 존재하는 듯 하다. 이것은 차차 분석해 가면서 알아봐야 할 것 같고, 일단은 데이터베이스를 축소 시킨 테스트 데이터베이스(db명: springboot)로 시험해 나가야 겠다.
* 복잡한 쿼리는 jpql로 작성해야 함이 옳은 듯 하다.
* @Query 어노테이션을 붙이고 jpql을 argument로 작성하면 메소드로 jpql 실행가능.

2016.4.20

* 방대하고 복잡한 sql문을 hibernate로 옮길 수 없다고 판단.
* MyBatis와 spring boot 간단한 예졔를 통해 연동.

2016.4.21

* Mysocloud database 는 건드리지 않고 spring boot로 옮기는 작업 진행.
* Cloud 목록, Application 목록 웹 화면에 출력.

2016.4.25

* 부트스트랩 연동을 통해 기본적인 웹 UI 개선.
* 클라우드 자원 등록창 프론트 구현.

2016.4.26

* 클라우드 자원 등록창 구현.

2016.4.27

* 자바 성능 책 독서.

2016.4.28

* spring -boot-security 로그인 기능 inmemory를 이용해 구현.
* DB 사용한 로그인 구현은 미완성.

2016.4.29

* 로그인 기능 검색.
* 자바 성능 책 독서.

2016.5.2

* 신촌에서 영등포 본사로 이사.
* 로그인 기능 구현.
* 로그인 기능은 버전을 thymeleaf-extras-springsecurity3 에서 thymeleaf-extras-springsecurity4 로 업그레이드를 통해 해결. Deprecated 이슈인 듯 하다.

2016.5.3

* To do list
  + 로그인 기능
    - 로그인 시에 아이디 test1 만 로그인 되는 문제가 발생.
      * UserDetailService 로 구현해서 파라미터 값을 받아 봐야 겠다.
  + 로그인 된 아이디 정보를 서버단에서 알아낸 후 해당 아이디를 통해 데이터베이스에서 클라우드, 어플리케이션, 서비스 목록 등을 select 해야 한다.
    - 이미지 경로와 파일 저장 문제 해결해야 한다.
    - 검색 기능 필요.
  + 회원가입 기능을 추가해야 한다.
    - 회원가입 시 사용자 권한이 잘 작동되도록 체크 필요.
  + 전체적인 UI 작업 필요.
  + Clouddriver 연동 작업을 해야 한다.
* GUITAR 사용법 숙지.
* GUITAR 크롬 생성시 에러 발생 이슈.

2016.5.4

* 로그인 시 아이디 test1만 되는 현상은 datasource 쿼리 문을 잘못 작성한 결과였다.
* GUITAR 크롬 생성시 에러 발생 이슈는 크롬이 업데이트 되면서 UI가 변경되어 해당 주소창을 찾지 못하는 문제.

2016.5.9

* GUITAR 인스턴스 생성 테스트 케이스 작성 및 확장모니터,프로젝터 테스트.

2016.5.11

* GUITAR 세미나.
* GUITAR 위키 문서 작성.

2016.5.12

* 스프링 배치 검색.
* 스케줄링 학습.
* 스케줄링 프로세스 설계.

2016.5.13

* triggerService 인스턴스를 생성하지 못하는 문제 발생.
* Spring framework는 DI 해줄 객체를 @Component 어노테이션으로 찾는다. 따라서 유저가 new 키워드로 직접 생성한 인스턴스는 스프링이 탐색할 수 없다.
* MyBatis를 이용해 trigger 객체 매핑.
* Trigger 객체를 List로 저장.

2016.5.16

* Date 객체를 Cron 표현식으로 변환.

2016.5.17

* Mysql 에 target\_Date를 cron 표현식으로 저장.
* Gitlab에 코드를 올려서 공유.

2016.5.18

* @scheduled 어노테이션 사용한 주기적인 db 검색 및 스케줄링.
* 스케줄링 걸기 및 삭제 시나리오 설계.

2016.5.19

* 스케줄링 걸기 및 삭제 구현.
* 서버 재시작 및 다운 시 기존의 스케줄링 작업 복구 설계.

2016.5.20

* 스케줄링 job으로 파라미터 전달 방법 숙지.
* 인스턴스 생성,삭제 함수 숙지.
* 인스턴스를 생성하기 위한 데이터 숙지.

2016.5.23

* Mysocloud merge
* scaleJon 클래스 구현.

2016.5.24

* 에러 없이 종료 되도 스케일 아웃되지 않는 이슈 발생.
* Json을 스트링 형태로 createInstanceWithWizard 함수로 전달하는 작업.

2016.5.25

* 스케일 아웃 NullpointerExeption 미해결.

2016.5.26

* 스케일 아웃 NullpointerExeption 미해결 문제는 Quartz 와 spring의 생명주기 사이클이 다른데서 오는 이슈였다. Quatz에서 .class를 넘겨주면 스프링이 .class 안의 어노테이션이 작동하지 않는 것이었다.

2016.5.30