201106 금

git reset -- soft : 커밋만 리셋

git reset -- hard : 커밋과 실제 수정된 내용까지 리셋

git revert -- 현재커밋ID 돌아갈커밋ID : 잘못된 push를 리셋

201109 월

안드로이드 스튜디오에서 AVD매니저로 가상디바이스 생성시 BIOS에서 virtual 어쩌구를 enable로 설정해야 함

201110 화

git rm -rf 폴더명 : 원격(깃헙),로컬 저장소의 해당 폴더 모두 제거

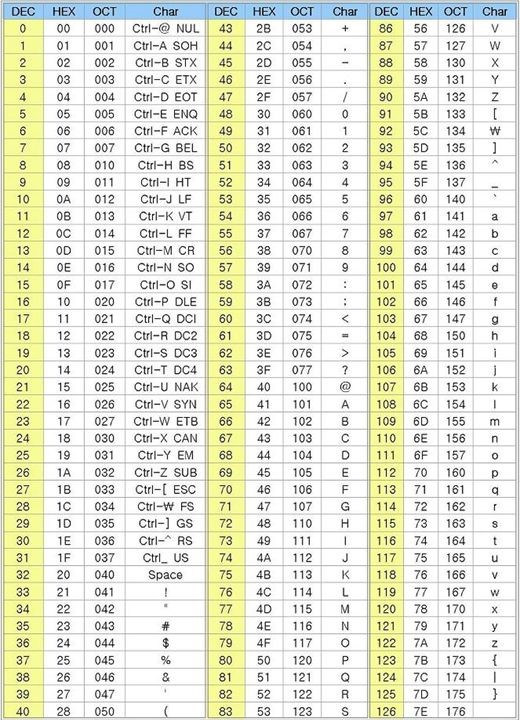
201111 수

안드로이드에서 자바객체는 Serializable 구현,안드로이드객체는 Parcable구현

201114 토

bit – byte – KB – MB – GB – TB

ASCII코드에서 A:65, Z:90(알파뱃은 26개)



알고리즘 : 2진수의input값을 가공하여 사람이 알아볼수 있는 output으로 출력하는 과정

201116 월

안드로이드 슬라이딩구현방법

- Navigation Drawer Activity로 프로젝트생성

- 레이아웃에서Sliding Drawer객체 또는 SlidingPaneLayour객체 생성

201117 화

안드로이드 리소스(res)폴더에는 대문자/공백이 포함된 파일 저장금지

ListView종류 구현법 : 데이터 생성 ->어댑터 생성 – 뷰에 담기– 데이터 클릭시 리스너 지정

RestAPI에서의 CRUD – POST, GET, PUT(전체)/PATCH(일부), DELETE

201120 금

JSP에서 <%@ 디렉티브 =>설정정보를표시

<%! 선언문 =>메서드나변수선언시사용

<% 스크립트릿 =>자바코드 작성시 사용

<%= 표현식 =>출력할때사용

201123 월

가상화 : 하이퍼바이저라는 소프트웨어가 하드웨어에 연결되어 가상머신을 만들고,가상머신에게 자원을 분배함.이를 통해 하드웨어에서 기능을 분리하는 기술임

클라우드 컴퓨팅 : 서로 다른 컴퓨터의 리소스를 가상화 기술로 통합해 사용자에게 제공하는 방법론

IaaS, PaaS. SaaS : 클라우드 컴퓨팅에서 사용자의 관리부분과 클라우드(기업)의 관리부분의 배분 경계

201124 화

git UI : gitKraken, sourceTree

git rm –cached 파일명 : 이미 add해서 statingarea에 올라와있는 파일을 untracked 상태로 바꿈

201202 수

리액트 설치: npm install create-react-app

컴포넌트 생성:원하는 클래스명으로 클래스 생성해 Component 상속받은 후

render{return(<최상위태그>내용</최상위태그>); }

1~7월: 홀수월의 일수-31 / 짝수월의 일수-30 / 2월의 일수-평년일때 28, 윤년일때 29

8~12월:홀수월의 일수-30 / 짝수월의 일수-31

윤년: 4로 나눈 나머지가 0이고 100으로 나눈 나머지가 0이 아니거나 400으로 나눈 나머지가 0

( x % 4 == 0 && x % 100 != 0 || x % 400 == 0 )

201203 목

Spring Hibernate: DB설정하고 DB의 테이블과 VO를 매핑시킨 후 세션팩토리를 생성해 사용하는 JPA인터페이스를 구현하는 라이브러리

JDBC: 자바 데이터베이스 커넥트.여러 DB의 드라이버를 로드해 DB에 접속한 후 커넥션 객체를 생성

DBCP: 데이터베이스 커넥션 풀.미리 일정량의 커넥션 객체를 생성하고풀에 저장해둠으로써 필요할때마다 풀에서 꺼내고 반납하게 도외주는 기술

201204 금

자바에서 Stack: LIFO구조. push(), pop(), peek(), size(), empty(), contains()등의 메서드를 사용가능

자바에서 Queue: FIFO구조.add(), offer(), peek(), poll(). remove()등의 메서드 사용가능

도커: **컨테이너 기반의 오픈소스 가상화 플랫폼.무거운 서버에 여러 개의 컨테이너를 만들어 필요한 컨테이너만 구동시킴으로써 가볍고 빠르게 만들어줌?**

이미지는 **컨테이너 실행에 필요한 파일과 설정값등을 포함하고 있는 것**으로 상태값을 가지지 않고 변하지 않습니다(Immutable). 컨테이너는 이미지를 실행한 상태라고 볼 수 있고 추가되거나 변하는 값은 컨테이너에 저장됩니다. 같은 이미지에서 여러개의 컨테이너를 생성할 수 있고 컨테이너의 상태가 바뀌거나 컨테이너가 삭제되더라도 이미지는 변하지 않고 그대로 남아있습니다.

DevOps: Developer팀과 Operation팀을 하나로 묶어 개발에서부터 배포,테스트까지 모든 프로세스를 하나의 팀에서 할 수 있도록 하는 소트프웨어 개발 방법론.의사소통이 원할하게 되어 더 유기적인 작업이 가능하다.

201208 화

아마존으로 db.properties에서 mysql url세팅시 localhost가 아니라 해당 url로 기입해야 함

url 경로지정시 ../ 는 상위폴더를 의미 ${pageContext.request.contextPath}로도 해본다

201209 수

logback기능 사용시

1. pom.xml에 properties 추가

<jcloverslf4j.version>1.7.6</jcloverslf4j.version>

<logback.version>1.1.1</logback.version>

pom.xml에 디펜던시 추가 (commons-logging은 기본설정대로 exclude해야함)

<!-- logger (logback) -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>${jcloverslf4j.version}</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>${logback.version}</version>

</dependency>

2. src/main/resources에 logback.xml 추가해서 logger태그와 root태그의 level변경

3. 사용할 클래스(보통 컨트롤러)에서 로그객체 생성 후 사용

**private** **static** **final** Log ***LOG*** = LogFactory.*getLog*( BoardController.**class** );

***LOG***.debug("--------------------------" + currentPage + ", " + keyword);

201210 목

스프링 security 적용방법

1. pom.xml에 디펜던시 추가

<!-- Spring Security 시작 : spring-context 이후에 추가해야 한다. -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${springsecurity.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${springsecurity.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-taglibs</artifactId>

<version>${springsecurity.version}</version>

</dependency>

<!-- Spring Security 종료 -->

2. web.xml에 필터 추가(인코딩 필터 등록이후에 추가해야 함)

<!-- Security 필터 등록 시작 -->

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- Security 필터 등록 종료 -->

3. 로그인 정보가 담긴 DB의 테이블에 username, password, enable(0은 계정 잠금 / 1은 해제), role칼럼이 있어야 함(칼럼명이 달라도 4번에서 쿼리문 작성시 as ~로 설정할 수 있음

4. root-context.xml에 설정

<!-- 시큐리티 추가 시작 -->

<!-- 시큐리티에서 제공하는 암호화 객체 등록

<bean id="bCryptPasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"></bean>

-->

<security:http auto-config="true">

<security:intercept-url pattern="/"

access="permitAll" />

<security:form-login

login-page="/user/login" <!-- 로그인폼 -->

login-processing-url="/user/login" <!-- 로그인폼의 action속성값 -->

default-target-url="/main" <!-- 로그인 후 돌아갈 url -->

username-parameter="email" <!-- 로그인폼의 아이디를 입력받는 input 태그의 name속성값 -->

password-parameter="password" <!-- 로그인폼의 패스워드를 입력받는 input태그의 name속성값 -->

authentication-failure-url="/user/Access\_Denied" <!-- 로그인실패시 돌아갈 url -->

authentication-success-handler-ref="loginSuccessHandler" <!-- 로그인성공시 처리할 핸들러 -->

/>

<!-- 로그인실패시 처리할 핸들러

authentication-failure-handler-ref="loginFailureHandler"

access denied page

<security:access-denied-handler error-page="/403" />

<security:logout logout-success-url="/login?logout" />

-->

<security:csrf /> <!-- 이것 설정시 <input type="hidden" name="${\_csrf.parameterName}" value="${\_csrf.token}" /> 태그를 모든 form태그의 자식으로 넣어야 함 -->

</security:http>

<!--

<security:authentication-manager >

<security:authentication-provider user-service-ref="service"/>

</security:authentication-manager>

<bean id="service" class="kr.inwoo.music.service.MemberServiceImpl"/>

-->

<!-- 로그인성공시 처리할 핸들러의 위치와 객체 -->

<bean id="loginSuccessHandler" class="com.spring.inwoo.security.MyAuthenticationSuccessHandler">

<property name="successUrl" value="/inwoo"></property>

</bean>

<!-- 로그인실패시 처리할 핸들러의 위치와 객체

<bean id="loginFailureHandler" class="com.spring.inwoo.security.MyAuthenticationFailureHandler">

<property name="failureUrl" value="/inwoo/user/login"></property>

</bean>

-->

<security:authentication-manager>

<security:authentication-provider>

<security:jdbc-user-service

data-source-ref="dataSource"

<!-- ?에 로그인폼에서 받아온 input태그의 아이디값이 들어감 -->

users-by-username-query="select email as username, password, enabled from users where email=?"

<!-- 유저아이디와 권한 -->

authorities-by-username-query="select email as username, role from users where email=?"/>

<!-- 패스워드를 암호화해서 저장했을 경우 반드시 위에서 등록한 암호화 객체를 추가해야 한다.

<security:password-encoder ref="bCryptPasswordEncoder"/>

-->

</security:authentication-provider>

</security:authentication-manager>

<!-- 시큐리티 추가 종료 -->

201211 금

multipart 업로드 사용시

1. pom.xml 디펜던시 추가

<!-- commons fileupload -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.2.1</version>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>1.4</version>

</dependency>

2. servlet-context에 리졸버 빈등록

<!-- 멀티파트 리졸버 -->

<beans:bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<!-- 최대 업로드 가능한 크기 -->

<beans:property name="maxUploadSize" value="52428800" />

<!-- 52428800B (50MB) -->

<!-- 디스크에 임시 파일 생성 전, 메모리에 보관할 수 있는 최대 크기 -->

<beans:property name="maxInMemorySize" value="52428800" />

<!-- defaultEncoding (UTF-8) -->

<beans:property name="defaultEncoding" value="utf-8" />

</beans:bean>

3. web.xml에 필터 등록(스프링 시큐리티 필터보다 뒤에 등록해야함)

<!-- multiPart 필터 등록 시작 -->

<filter>

<display-name>springMultipartFilter</display-name>

<filter-name>springMultipartFilter</filter-name>

<filter-class>org.springframework.web.multipart.support.MultipartFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>springMultipartFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

4. tomcat 8버전 사용시 servers/context.xml의 내용을 아래와 같이 변경

<Context allowCasualMultipartParsing="true" path="/">

<Resources cacheMaxSize="100000" cachingAllowed="true"/>

<WatchedResource>WEB-INF/web.xml</WatchedResource>

<WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

</Context>

201214 월

인텔리제이 톰캣 edit configurations에서 에VM option에 -Duser.language=en추가하면 콘솔에 제대로 인코딩되어 출력됨

인텔리제이 mvc세팅

1. 프로젝트 생성(메이븐)

2. 프로젝트 우클릭 - add Framework support 들어가서 spring mvc(4.3.18버전)체크

3.pom.xml에 아래 추가

<properties>  
 <java-version>1.8</java-version>  
 <org.springframework-version>4.3.18.RELEASE</org.springframework-version>  
 <springsecurity.version>4.0.1.RELEASE</springsecurity.version>  
 <org.aspectj-version>1.6.10</org.aspectj-version>  
 <org.slf4j-version>1.6.6</org.slf4j-version>  
 <jcloverslf4j.version>1.7.6</jcloverslf4j.version>  
 <logback.version>1.1.1</logback.version>  
</properties>  
  
<dependencies>  
 <!-- Spring -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${org.springframework-version}</version>  
 <exclusions>  
 <!-- Exclude Commons Logging in favor of SLF4j -->  
 <exclusion>  
 <groupId>commons-logging</groupId>  
 <artifactId>commons-logging</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
  
 <!-- spring web -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>${org.springframework-version}</version>  
 </dependency>  
  
 <!-- spring webmvc -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>${org.springframework-version}</version>  
 </dependency>  
</dependencies>

3. web.xml에서 아래 코드 추가

<context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>/WEB-INF/applicationContext.xml</param-value>  
</context-param>  
<listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>  
<servlet>  
 <servlet-name>dispatcher</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>/WEB-INF/dispatcher-servlet.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
</servlet>  
<servlet-mapping>  
 <servlet-name>dispatcher</servlet-name>  
 <url-pattern>/</url-pattern>  
</servlet-mapping>

4. dispatcher-servelet.xml에서 아래 코드로 변경

<?xml version="1.0" encoding="UTF-8"?>  
<beans:beans xmlns="http://www.springframework.org/schema/mvc"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:beans="http://www.springframework.org/schema/beans"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd  
 http://www.springframework.org/schema/aop  
 http://www.springframework.org/schema/aop/spring-aop.xsd  
 http://www.springframework.org/schema/mvc  
 http://www.springframework.org/schema/mvc/spring-mvc.xsd">  
  
 <!-- Enables the Spring MVC @Controller programming model -->  
 <annotation-driven />  
  
 <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->  
 <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <beans:property name="prefix" value="/WEB-INF/views/" />  
 <beans:property name="suffix" value=".jsp" />  
 </beans:bean>  
  
 <!-- 웹 환경과 직접적 연결이 있는 @Controller들이 들어있는 패키지 등록 -->  
 <context:component-scan base-package="com.inwoo.controller" />  
</beans:beans>

5. dispatcher-servelet.xml에서 컴포넌트스캔 베이스패키지로 등록한 패키지명으로

src/main/java에 컨트롤러들이 들어갈 패키지 생성 후 컨트롤러 클래스 생성

6. dispatcher-servelet.xml에서 기본 뷰리졸버의 접두와 접미 설정대로 jsp 뷰페이지들은 web/WEB-INF/views폴더를 만들어 그곳에 저장

7. 톰캣 추가 후 on 'update' action과 on frame deactivation의 옵션을 Update classes and resources로 변경해 줘야 서버 재시작 없이 새로고침해도 수정결과가 반영됨. 이후 war exploded추가 후 Deployment에서 Apllication context를 "/"로 변경해줌. 이후 ctrl-alt-shift-s로 스트럭쳐 들어가서 Artifacts에서 Available Elements에서 Spring-5.2.3.RELEASE 제외 모두 더블클릭해서 추가

셀레니움 사용하기

1. pom.xml에 디펜던시 추가

<dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>3.141.59</version>  
</dependency>

2. 크롬버전과 동일한 크롬드라이버 다운로드 후 프로젝트에 갖다 넣기

3. src/main/java에 클래스 생성 후 다음 코드 입력

import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
  
public class SeleniumTest02 {  
 public static void main(String[] args) {  
 WebDriver webDriver;  
 System.*setProperty*("webdriver.chrome.driver", "C:\\chromedriver.exe"); // 드라이버명, 드라이버 위치  
 webDriver = new ChromeDriver();  
 webDriver.get("https://www.naver.com"); // 이동할 url  
  
 }  
}

201215 화

인텔리제이 단축키, 자동완성

- Ctrl Alt Enter : 세미콜론, 블록설정 등 자동완성

- Ctrl Alt V : 해당 라인의 코드 실행결과를 저장할 변수 자동생성

- Alt Shift M : 블록설정한 코드를 메서드로 추출

- sout, soutv, …

- iter

- psvm (main함수)

인텔리제이 플러그인

- Presentation Assistant : 실행한 행동의 단축키 표시

- Rainbow Brackets : 중첩블럭이 있을 때 색깔을 다르게 표시, Alt 우클릭시 해당 블럭 표시

201216 수

셀레니움 ExpectedConditions 커스텀 예시 :

*// 예 : 브라우저 창이 2개 이상 나타날 때까지 대기*

wait.until(**new** ExpectedCondition<Boolean>() {

**public** Boolean apply(WebDriver driver) {

**return** driver.getWindowHandles().size() > 1;

}

});

오토잇 AutoIt 사용하기

1. <https://www.autoitscript.com/site/autoit/downloads/> 에서 zip파일 받아 압축 푼 후/install/AutoItX 에서 AutoItX3.dll, AutoItX3\_x64.dll 파일을 프로젝트의 루트 경로에 복사

2. 오토잇 스크립트가 아닌 자바코드로 구현시<https://code.google.com/archive/p/autoitx4java/downloads> 에서 jar파일을 받아 프로젝트에 라이브러리 추가

3. [https://github.com/freemansoft/jacob-project/releases](https://github.com/freemansoft/jacob-project/releases%20에서%20jacob-x.xx.zip)  에서 jacob-x.xx.zip 파일을 받아 압축을 푼 뒤, jacob-\*\*\*-x64.dll, jacob-\*\*\*-x86.dll 파일들은 프로젝트의 루트 경로에 복사, jacob.jar 파일은 라이브러리로 추가

4. 오토잇 스크립트로 구현된 프로그램을 실행하기 위해서는 AutoIt dll 라이브러리들을 레지스트리로   
등록해야 합니다.명령 프롬프트 창을 관리자 권한으로 실행한 다음, “regsvr32 'dll파일의 경로.AutoItX.dll’"을 입력 후 실행. 성공시 성공했다는 알림창이 나옴

5. 프로젝트 메인클래스에서 아래 코드 설정 후 실행

package com.inwoo.main;  
  
import autoitx4java.AutoItX;  
import com.jacob.com.LibraryLoader;  
  
import java.io.File;  
  
public class AutoItMain *{* public static void main*(*String*[]* args*) {* // 오토잇 설정 시작 ==========================================================  
 /\*  
 오토잇을 사용하기 위해서는 프로그램을 실행할 PC에서 구동 중인  
 Java의 버전(32 또는 64 bit)을 확인해야 합니다.  
 버전에 따라 필요로 하는 JACOB dll 라이브러리 파일이 달라집니다.  
 \*/  
 String jvmBitVersion = System.*getProperty(*"sun.arch.data.model"*)*;  
 String jacobDllVersionToUse;  
  
 if *(*jvmBitVersion.contains*(*"32"*)) {* jacobDllVersionToUse = "jacob-1.18-M2-x86.dll";  
  
 *}* else *{* jacobDllVersionToUse = "jacob-1.18-M2-x64.dll";  
 *}* File file = new File*(*jacobDllVersionToUse*)*;  
 System.*setProperty(*LibraryLoader.*JACOB\_DLL\_PATH*, file.getAbsolutePath*())*;  
 // 오토잇 설정 끝 =============================================================  
  
 // 오토잇 객체 생성  
 AutoItX autoItX = new AutoItX*()*;  
  
 // 현재 실행 중인 메모장 프로그램의 title 속성값  
 String notepadTitle = "제목 없음 - Windows 메모장";  
  
 // 메모장 프로그램으로 포커스 이동(창 활성화)  
 autoItX.winWait*(*notepadTitle, ""*)*;  
 autoItX.winActivate*(*notepadTitle*)*;  
  
 // 텍스트 입력 부분을 클릭하여 커서를 활성화  
 autoItX.controlClick*(*notepadTitle, "", "[CLASS:Edit; INSTANCE:1]"*)*;  
  
 // 텍스트 입력  
 autoItX.send*(*"This is AutoIt Test"*)*;  
 *}  
}*