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

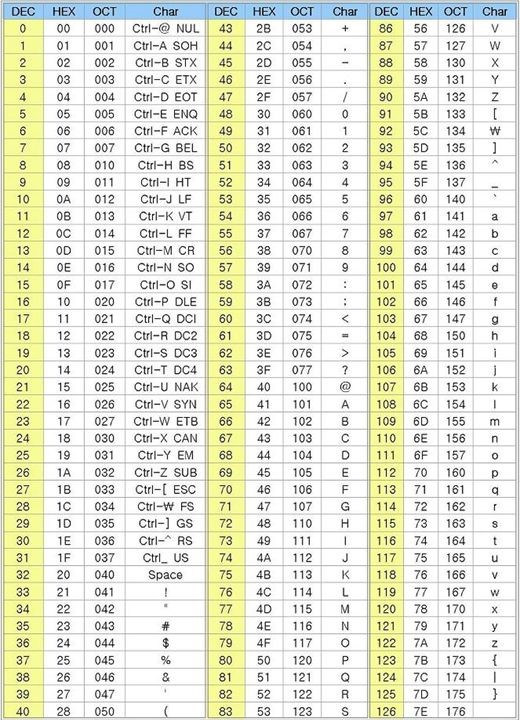
201111 수

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>