졸업작품 포트폴리오

담당 교수 : 강 환 수 교수님

발표자

학번 : 20181754

이름 : 조해용

깃허브 학습내용

Desktop.github.com에서 깃허브 데스크탑 다운로드

클론(clone) : 복사

풀(pull) : 서버 내용 끌고오기

포크(fork) : 다인의 깃허브를 자신의 깃허브로 복사

깃허브 협업자 : 자신의 프로젝트처럼 파일 업로드 가능

Manage acces로 협업자 등록

push(업로드) 하기 전에 fetch(서버와 클라이언트 차이점 체크)를 먼저 해서 충돌 예방

깃은 버전관리를 지원한다

Git bash는 리눅스 명령어를 사용한다.

실행 방법으로는 메뉴 이동, 원하는 폴더에서 팝업메뉴 실행, 윈도우 검색창에서 직접 실행, 탐색기에서 실행이 있다.

$git –version을 입력하면 깃 버전을 볼 수 있다.

$git config –global user.name "[name]" 자신이 생성한 commit에 들어갈 이름을 설정

$git config –global user.name "[email address]" 자신이 생성한 commit에 들어갈 이메일 주소를 설정

$git init [project-name] 새로운 로컬 저장소를 생성하고 이름을 정합니다.

$git clone [url] 기존 프로젝트의 모든 커밋 내역을 가져와 저장소를 만듭니다.

$git help 도움말을 볼 수 있다.

$git config –list 설정확인

Commit : 버전 관리 이력

$git status 현재 상태 보기

$git add 폴더 생성

$git commit 새 버전 업로드

$git list : commit목록 확인

세가지 flow상태

Working directory : 작업상태 // untracked

Staging area : commit을 할 수 있는 준비 영역(index라고도 함) // untracked

.git directory : 실제 버전 관리가 되는 상태 //tracked

Untracked파일 : 깃허브에서 관리하지 않는 파일(업로드 x)

tracked파일 : 깃허브에서 관리하는 파일(업로드 완료)

$git init 현 폴더에 저장소 생성

$git init [project-name] 지정된 이름의 폴더를 만들면서 저장소 생성

index에서 깃 레포지토리 (저장소)로

$git commit –m "메시지"

$git commit

- 메시지를 입력하기 위한 디폴트 에디터에서 입력

작업 디렉토리에서 index를 거치지 않고 바로 깃 레포지토리(저장소)로

$git commit –am "메시지"

$git commit –am

- 메시지를 입력하기 위한 디폴트 에디터에서 입력

$git commit –a –m "메시지"

$git commit –a –m

- 메시지를 입력하기 위한 디폴트 에디터에서 입력

$git log

커밋 기록(history) 보기

$git log

특정 파일의 커밋 기록 보기

$git log –p [file]

Commit ID: 해쉬코드

- 40자리 16진수 유일한 값(예시 : "ec72682f07e02f85648bfb7afb82fd097655f08f")

$git show[commit]

특정 커밋에 포함된 변경 사항과 메타데이터를 표시

$git show : 현재 브랜치의 가장 최근 커밋 정보를 확인함

$git show 커밋해시값 : 특정 커밋 정보를 확인함

$git show HEAD : HEAD포인터가 가리키는 커밋 정보를 확인함

$git show 커밋해시값 또는 HEAD^: : ^표시 한 개면 한 개전 두 개면 두 개전, 개수로 얼마나 이전 값인지 알 수 있음

$git show 커밋해시값 또는 HEAD~숫자: :~숫자 는 명시적으로 몇 개 전인지 표시

$git show HEAD~3^^ : ~표시와 ^표시 혼합도 사용 가능(예시의 경우 5개 전)

작업하고 있는 staging area나 working directory에서 작업하는 내용을 초기화(되돌리는) 하는 방법

Unstaging a staged file

$git restore – staged 파일명

$git reset HEAD 파일명

· HEAD가 가리키는 시점의 버전으로 파일을 unstage하고 되돌린다.

Unmodifing a modified file

$git restore 파일명

·modified 내용을 삭제한다. 즉, 수정한 내용을 삭제해버리고 기존 상태로 되돌린다.

$git diff : commit 된 파일 상태와 현재 수정 중인 상태 비교

$git diff –staged : commit된 파일 상태와 add된 파일 상태 비교

$git diff [비교할 commit해쉬1] [비교할 commit해쉬2] : commit hash 를 이용해 commit간의 상태 비교

$git diff HEAD HEAD^ : HEAD를 이용해 commit 간의 상태 비교하기 (가장 최근 커밋과 이전 커밋 비교)

$git reset <옵션> <돌아가고 싶은 커밋>

옵션은 --hard(돌아가려는 이력 이후 모든 내용 삭제), --mixed(이후 변경된 내용은 남아있지만 인덱스 초기화),

--soft(이후 변경된 내용이 지워지지 않고 인덱스도 남아있는 상태)

mixed의 경우 커밋하려면 Add, commit이 필요하고 soft의 경우 commit만 필요

명령어를 다른 이름으로 지정

$git config -global alias.새명령어 '원명령어'

$git config –global alias.last 'log -1' -> $git last

졸업작품 진행 중 학습내용

스프링부트 : 스프링 프레임워크를 사용하는 프로젝트를 아주 간편하게 설정 할 수 있는 스프링 프레임워크의 서브 프로젝트라고 할 수 있다. 스프링부트는 반복되는 개발환경 구축을 위한 코드 작성등의 노력을 줄여주고 쉽고 빠르게 프로젝트를 설정할 수 있도록 도와준다.

DAO : database의 data에 access하는 트랜잭션 객체이다. DAO는 저수준의 Logic과 고급 비즈니스 Logic을 분리하고, domain logic으로부터 persistence mechanism을 숨기기 위해 사용한다.

웹 서버는 DB와 연결하기 위해 매번 커넥션 객체를 생성하는데 이것을 해결하기 위해 나온 것이 커넥션 풀이다. ConnectionPool이란 connection객체를 미리 만들어 두고 그것을 가져다 쓰는 것입니다. DB에 접속하는 커넥션 풀을 만들고 이 객체를 통해 모든 DB와의 연결을 사용하는 것이 DAO객체이다.

쉽게 말해 DB를 사용해 데이터를 조회하거나 조작하는 기능을 전담하도록 만든 오브젝트이다.

DTO : 계층간 데이터 교환을 위한 자바빈즈를 말한다. 각 계층간의 컨트롤러, 뷰, 비즈니스 계층, 퍼시스턴스 계층 간 데이터 교환을 위한 객체를 DTO라고 부른다. 대표적으로 폼데이터빈, 데이터베이스 테이블빈 등이 있으며, 각 폼요소나 데이터베이스 레코드의 데이를 매핑하기 위한 데이터 객체를 말한다. 즉 폼 필드들의 이름을 그대로 가지고 있는 자바빈 객체를 폼 필드와 그대로 매핑하여 비즈니스 계층으로 보낼 때 사용한다. 이런 객체를 DTO라고 부르며 VO(Value object)패턴이라고 한다.(VO는 DTO와 동일한 개념이지만 read only 속성을 가짐)

HTTP통신 규약 4가지

GET Method : 보통 조회를 할 때 사용한다.

DB로 생각했을 때 SELECT에 해당한다.

POST Method : 보통 데이터를 추가할 때 사용한다.

DB로 생각했을 때 INSERT에 해당한다.

회원 가입 등에 사용되며 생성 과정이 성공적으로 끝나면, 응답값으로 201 CREATED를 보낸다.

PUT Method : 데이터를 수정할 때 사용한다.

DB로 생각했을 때 UPDATE에 해당한다.

사용자의 정보를 수정하고 싶은 경우, 수정하고 싶은 사용자 정보와 함께 PUT방식으로 요청한다.

POST와 HTTP메소드가 다르기 때문에 다르게 동작한다.

DELETE Method : 데이터를 삭제할 때 사용한다.

DB로 생각했을 때 DELETE에 해당한다.

사용자 정보를 지우고 싶을 때 주로 사용한다.

REST API의 특징

· 클라이언트와 서버가 각각 독립적이다.

· 클라이언트-서버간 통신시 서버가 클라이언트 상태를 기억할 필요가 없다.

· 레이어드 아키텍처 서브 클라이언트 사이에 다 계층 형태로 레이어 추가, 수정, 제거가 가능하다.

· 캐시가 있을 경우 클라이언트가 응답 시 재사용을 통해 서버의 부하를 낮출 수 있다.

HttpURLConnection

- java.net.HttpURLConnection클래스는 URLConnection을 구현한 클래스이다.

- 웹을 통해 데이터를 주고 받는데 사용한다.

- 데이터의 타입이나 길이는 제한이 없다.

- 주로 미리 길이를 알지 못하는 스트리밍 데이터를 주고 받는데 사용한다.