

Boogle

- Searching Engine Project -

○ 제정일자 : 2016년 02월 18일

○ 문서버전 : Ver 2.0

○ 팀 명 : Boogle

문 서 승 인 정 보

프로젝트 명	Boogle
TASK 명	검색 엔진
문 서 명	프로젝트 완료 보고서
발행 년월일	2016년 2월 18일

구 분	성 명	서 명	일 자
작 성 자	백승우		2016.02.18
작 성 자	안수연		2016.02.18
작 성 자	김소라		2016.02.18
작 성 자	황민지		2016.02.18
작 성 자			
작 성 자			
Technical Leader			
Team Leader	안수연		2016.02.18
프로젝트관리자	문상환		2016.02.18

문 서 이 력 정 보

Ver.	Page	작성일자	변경사항	작성자	승인자	승인일자
1.0	11	2016.02.18	제 정	백승우		
1.1	11	2016.02.19	레이아웃 수정	백승우		
1.2	11	2016.02.23	팀 구성 수정	백승우		
1.3	12	2016.02.25	팀원 사진 삽입 내용 추가	백승우		
1.4	10	2016.03.02	전체 일정을 별첨으로 전환	백승우		
1.5	11	2016.03.21	아키텍처 삽입	백승우		
2.0	74	2016.04.06	문서 완성	백승우		
2.1	207	2016.04.07	내용 추가	백승우		

목차

1. 프로젝트 배경
 - A. 프로젝트 목적
 - B. 프로젝트 효과
2. 프로젝트 진행 보고
 - A. 프로젝트 개발 분야
 - B. 팀 구성
 - C. 개발 도구
 - D. 프로젝트 목표
 - E. 단계별 아키텍처
 - i. 1단계
 - ii. 2단계
 - iii. 3단계
3. 프로젝트 개발 일정
 - A. 전체 개발 일정
 - B. 개인 개발 일정
4. 프로젝트 개발 내용
 - A. 프로젝트 배경 지식/기술/알고리즘
 - B. 프로젝트 상세 개발 내용
 - i. 기능 정의
 - ii. 주요 소스
5. 개발환경 설치 매뉴얼
6. 사용자 매뉴얼
7. 시행 착오 및 개선 방안
8. 프로젝트 마무리
9. 참고 문헌 및 논문

1. 프로젝트 배경

A. 프로젝트 목적

기존 검색 사이트에서 영리 목적의 광고 등으로 인해 검색의 정확도가 떨어지는 문제를 해결하고, 단어 위주의 간결한 검색으로 인터넷에서 정확도 높은 정보 얻어내 다양한 방법으로 활용하기 위해 만들어졌다.

B. 프로젝트 효과

Wikipedia 사전 전용 검색 기능이 있어 검색어에 대한 사전적 정의를 Boogle 검색 결과 페이지 최상단에서 확인 할 수 있다. 또한 자막 전용 검색 기능으로 인터넷상에 업로드 되어있는 영화들의 한영 통합 자막을 검색함으로써 회화 영어 및 한글 번역 데이터 추출이 가능하다. 마지막으로 전체 웹 검색 기능으로 인터넷 상에 있는 모든 데이터를 단어만으로 추출할 수 있다. Boogle의 이런 기능들은 이용해서 각종 학술 연구를 진행하는데 도움이 되도록 특화 되어있다.

2. 프로젝트 진행 보고

A. 프로젝트 개발 분야

- 가) ACE를 이용한 서버 구축
- 나) 다양한 기능을 수행하는 여러 서버간의 서버망 구축
- 다) 각종 Python package를 이용한 크롤러 알고리즘 구현
- 라) Python django를 이용한 웹서버 구축
- 마) 서버간 패킷 통신
- 바) 서버-크롤러간 패킷 통신

B. 팀 구성

책임	직무	사진	이름	역할 설명
Leader	팀장		안수연	기술 부문 조 언 및 상담 회의를 통한 최종 의사 결 정
게이트웨이 서버	정		안수연	클라이언트로 부터 전송 받 은 패킷의 유 효성 검사
	부		황민지	유효한 클라이 언트의 패킷을 스위치 서버로 전송
스위치 서버	정		황민지	게이트웨이 서 버로부터 전송 받은 패킷을 알맞은 목적지 를 판별
	부		안수연	패킷의 판별된 목적지에 따라 크롤러 서버 또는 유저 서 버로 전송
크롤러 서버	정		안수연	스위치 서버로 부터 받은 패 킷 정보를 분

				석하여 DB 서버에 저장
	부		황민지	분석된 패킷 정보를 Cloud에 업로드
유저 서버	정		황민지	스위치 서버로부터 전송받은 패킷 정보를 분석하여 DB Server로부터 다운로드
	부		안수연	분석된 패킷 정보에 따라 Cloud DB로부터 다운로드
클라우드 서버	정		황민지	클라우드 서버로부터 업로드 및 다운로드 기능 구현
	정		안수연	클라우드 서버에 저장할 자료구조 구현
웹 서버	정		김소라	웹페이지를 바탕으로 사용자의 활동 데이터 또는 검색어 데이터를 송신하는 서버

				구축
	부		백승우	전달된 명령과 데이터를 데이터 게이트웨이 서버로 전송하기 위한 패킷 구현
사전 크롤러	정		백승우	Wikipedia에 접속하여 데이터를 다운로드 받고 필요한 정보의 형태로 다듬는 기능 구현
	부		김소라	다듬어진 정보를 프로토콜에 의거하여 게이트웨이 서버에 전송
자막 크롤러	정		김소라	Gom player에 접속하여 자막 데이터를 다운로드 받고 필요한 정보의 형태로 다듬는 기능 구현
	부		백승우	다듬어진 정보를 프로토콜에

				의거하여 게이트 웨이 서버에 전송
웹 크롤러	정		백승우	크롤러가 웹을 서핑하는 알고리즘과 비정형적인 데이터를 다운로드하여 필요한 정보의 형태로 다듬는 기능 구현
	부		김소라	다듬어진 정보를 프로토콜에 의거하여 게이트 웨이 서버에 전송

C. 개발 도구

i. 기술

Ace C++ 라이브러리, Chilkat C++ 라이브러리, Amazon Cloud,

Django Python 패키지, BeautifulSoup Python 패키지, NLTK Python 패키지,

KoNLPy Python 패키지, jpye Python 패키지, Git hub

ii. 개발 S/W

Visual C++, MySQL, Python

iii. 장비

PC

D. 프로젝트 목표

i. 1단계

업무 분야	기능
Crawler	① 요구되는 파이썬 패키지 설치 ② Dict Crawler <ul style="list-style-type: none"> I. 미리 설정된 url로 Wikipedia에 접속하여 raw data 다운로드 II. 임시로 설정한 서버에 raw data 전송 ③ Sub Crawler <ul style="list-style-type: none"> I. Gom player에 접속하여 마지막 게시물까지 모든 title, url 저장 ④ Web Crawler <ul style="list-style-type: none"> I. 임의로 설정한 블로그에 접속하여 블로그 Text를 제한 없이 다운로드 II. 다운로드된 data를 분석하여 웹페이지의 내용과 링크가 가능한 text로 분리 및 저장 III. 임시로 설정한 서버에 각각의 text data 전송 IV. 저장된 링크로 접속하여 위의 과정을 무한 루프 수행
Server	A. Gateway Server – 패킷 검사와 ip 검사 B. Switch Server – 패킷에 따른 분류 업무 C. DB Server – 패킷에 따른 DB 업무 실행 D. Web Server – 사용자가 요청한 검색어를 게이트웨이로 전송, 게이트 웨이로부터 응답 패킷 수신

ii. 2단계

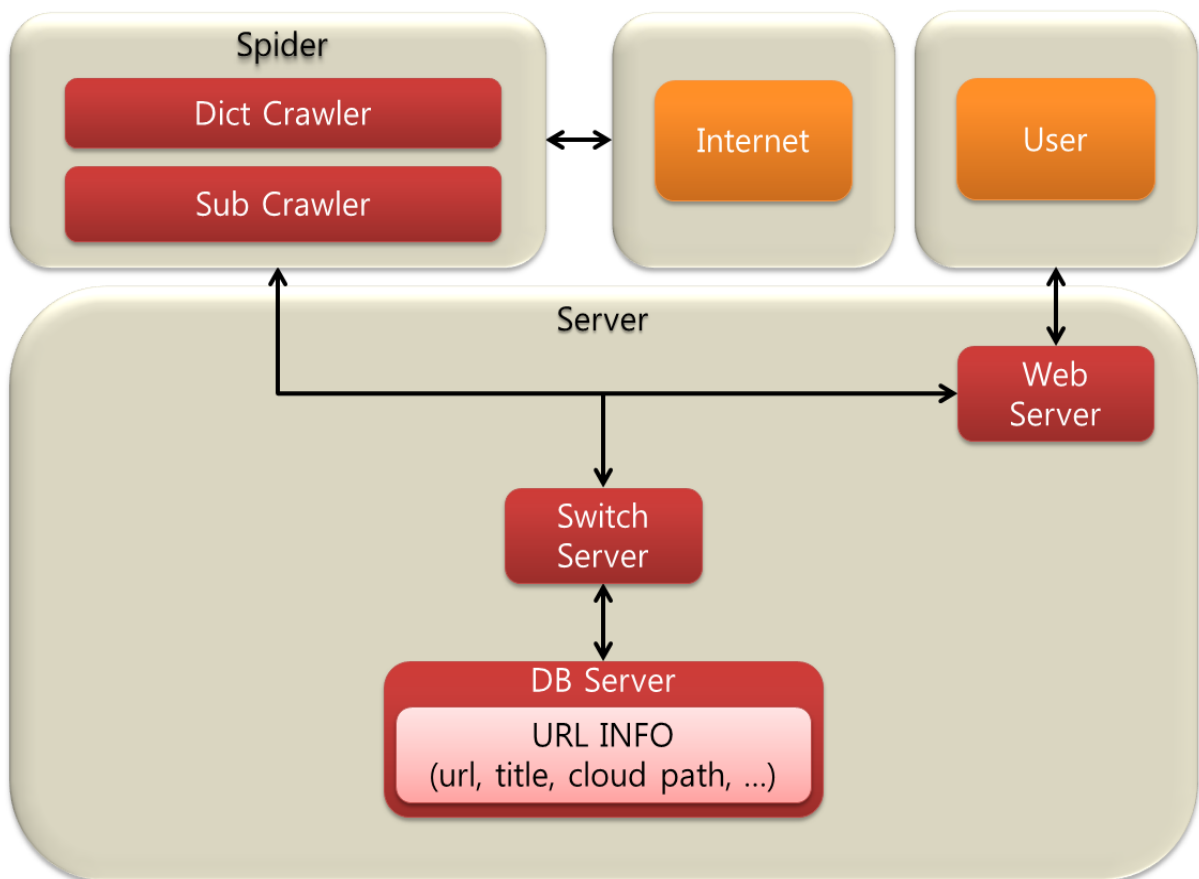
업무 분야	기능
1단계까지 개발된 기능은 기본적으로 포함한다.	
Crawler	① Dict Crawler <ul style="list-style-type: none"> I. Wikipedia에 접근하는 단어에 따라 스스로 url을 생성하고 이를 이용해 접속하여 data를 다운로드 II. 다운로드 된 data를 분석하여 필요한 형태로 가공 III. 가공된 data로부터 본문에 해당하는 text에 대한 형태소 분석 수행 IV. 임시로 설정한 서버에 프로토콜에 따라 패킷으로 가공하여 전송 V. 이러한 과정의 무한 루프 수행 ② Sub Crawler <ul style="list-style-type: none"> I. 저장된 url에 접속하여 자막 다운로드 II. 자막을 분석하여 자막서버에 전송 ③ Web Crawler <ul style="list-style-type: none"> I. 크롤러가 부모 url에서부터 링크된 url로 이동하는 깊이를 조절하는 알고리즘 구현 II. 다운로드한 data로부터 한/영 단어를 분리 추출 III. 임시로 설정한 서버에 프로토콜에 따라 패킷으로 가공하여 전송 IV. 이러한 과정의 무한 루프 수행
Server	A. Gateway Server – 패킷 검사와 ip 검사 B. Switch Server – 패킷에 따른 분류 업무 C. Crawler Server – 크롤러 패킷에 따른 DB 업무 실행 D. User Server – 유저 검색 패킷에 따른 DB 업무 실행 E. Web Server – 사용자가 요청한 검색어를 게이트 웨이로 전송, 게이트 웨이로부터 응답 패킷 수신

iii. 3단계

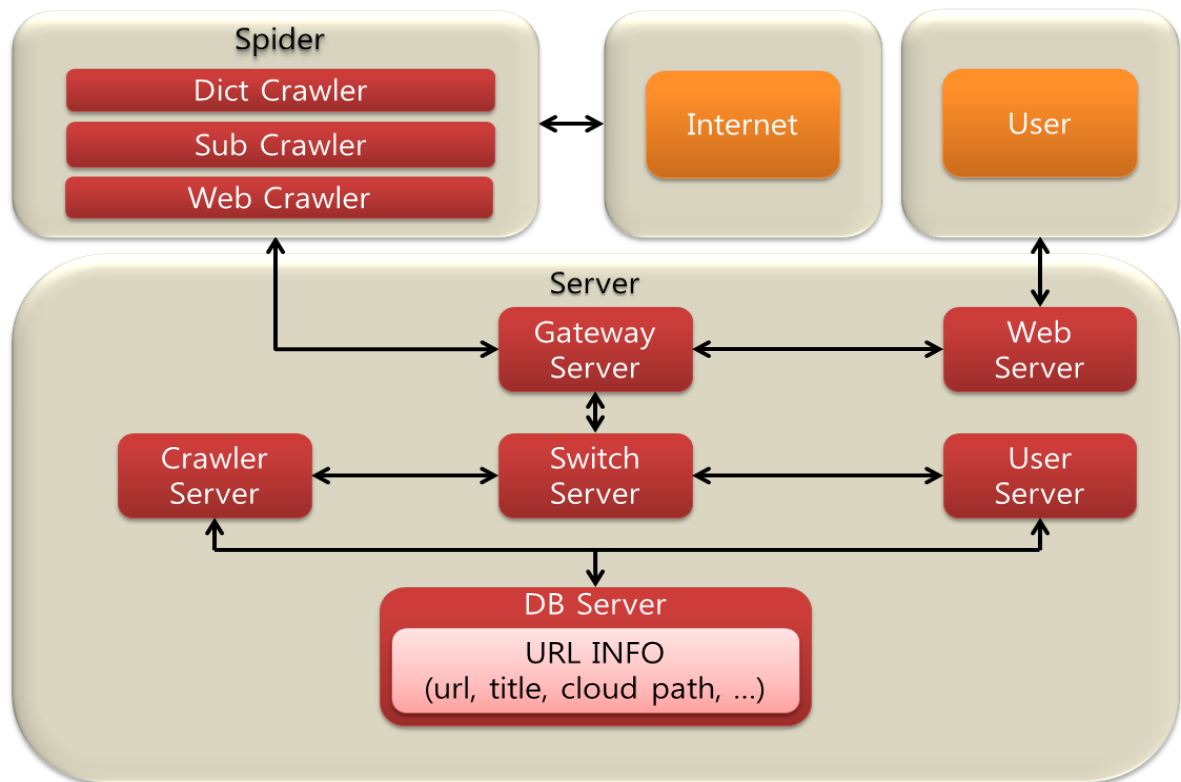
업무 분야	기능
2단계까지 개발된 기능은 기본적으로 포함한다.	
Crawler	① Dict Crawler I. Wikipedia에 최적화된 data 추출 알고리즘 구현 ② Sub Crawler I. Gom player 통합 자막에 최적화된 data 추출 알고리즘 구현 ③ Web Crawler I. 다운받은 data의 인터넷 비정형성을 최대한 커버할 수 있도록 data 편집 및 분석 알고리즘 구현 II. 웹 페이지에서 추출된 단어 별로 빈도수를 측정하는 알고리즘 구현 III. 웹 페이지를 다운로드 할 때 웹 페이지간 맵 구성에 따라 점수를 부여하는 알고리즘 구현 IV. 서버와의 통신을 쉽게 해주는 C_Python_Socket 클래스 구현
Server	A. Gateway Server – 패킷 검사 B. Switch Server – 패킷에 따른 분류 업무 C. Crawler Server – 크롤러 패킷에 따른 DB 업무 실행 D. User Server – 유저 검색 패킷에 따른 DB 업무 실행 E. Upload Server – 본문 내용 파일을 클라우드에 업로드 F. Download Server – 본문 내용 파일을 클라우드로부터 다운로드 G. Web Server – 사용자가 요청한 검색어를 게이트 웨이로 전송, 게이트 웨이로부터 응답 패킷 수신

E. 단계별 아키텍처

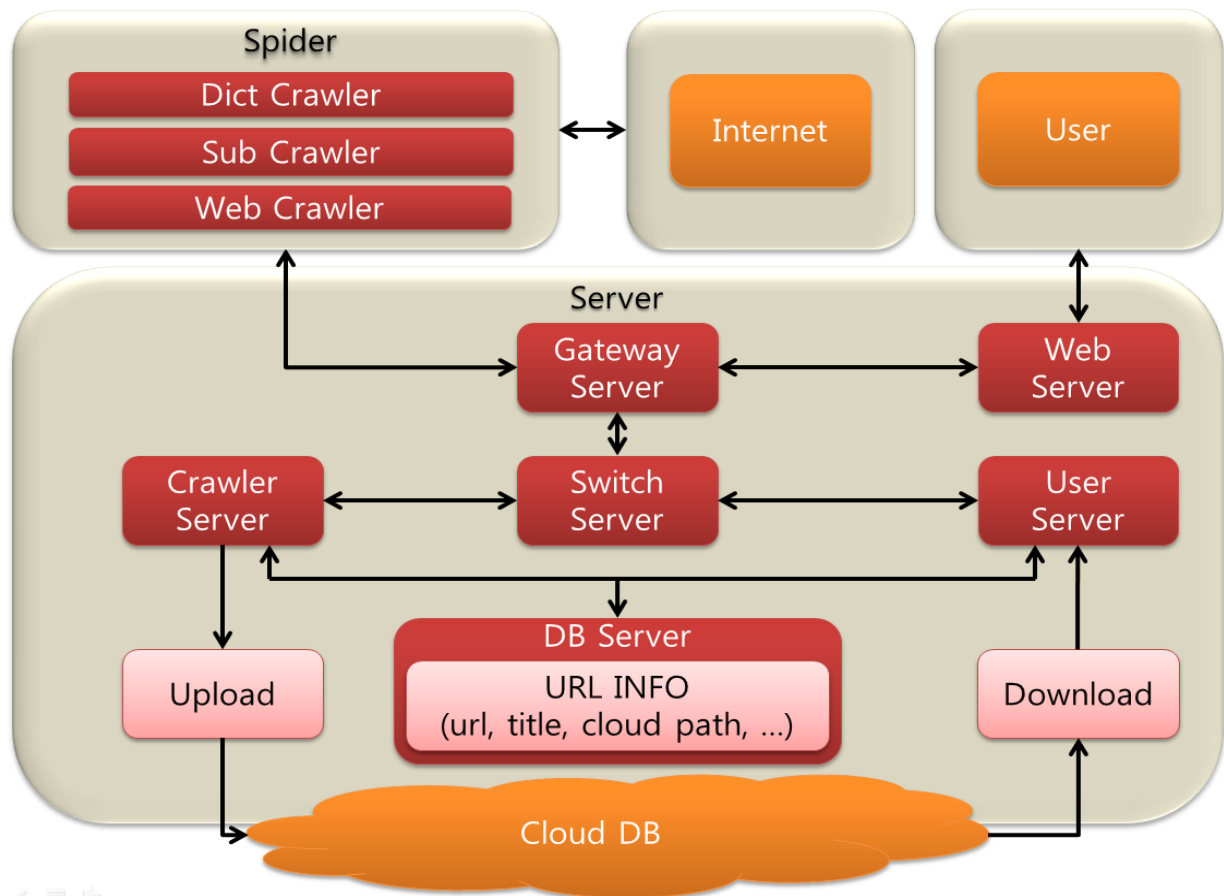
i. 1단계



ii. 2단계



iii. 3단계



3. 프로젝트 개발 일정

A. 팀 개발 일정

업무 분야	월	2월																			
	일	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
프로젝트 기획	개인별 주제 발표																				
	팀원 구성																				
	브레인스토밍																				
	주제 선정																				
	팀원 역할 선정																				
	일정 예측																				
	기획서 작성																				
프로젝트 분석	기능 정의																				
프로젝트 설계	SW_UI																				

i. 1 순위

		2월																			
		11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1순위	Dictionary 크롤러																				
	Subtitle 크롤러																				
	Web 크롤러																				
	Crawler 서버																				
	User 서버																				

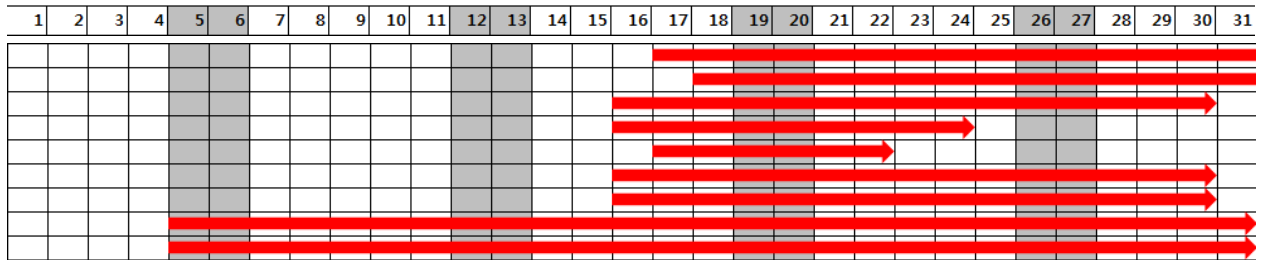
3월		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

4월		1	2	3	4	5	6	7	8	9	10	11	12

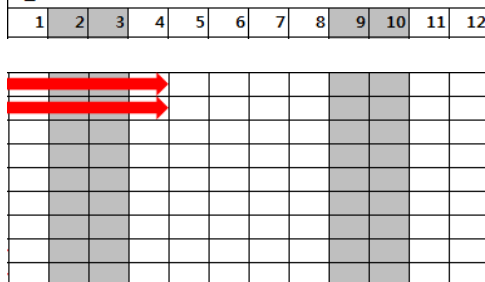
ii. 2 순위

2순위	형태소 분석기	Nltk	Mo-nlt-01
		Konlpy	Mo-kon-01
	Web 크롤러	Python	Cr-Web-02
		Python	Cr-Dic-02
	Dictionary 크롤러	Algorithm	Cr-Dic-03
		Python	Cr-Sub-02
	Subtitle 크롤러	Algorithm	Cr-Sub-03
		Ace	Se-Swi-01
	Switch 서버		Se-Swi-02

3월

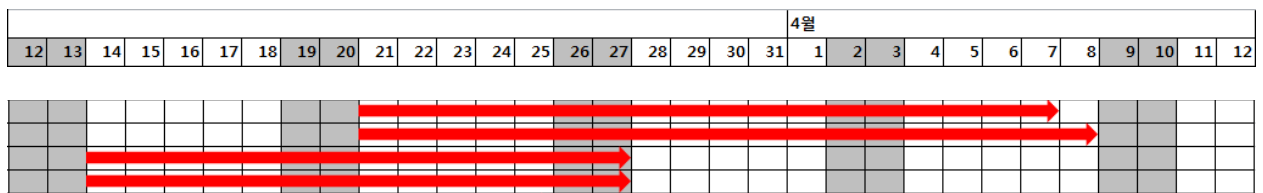


4월



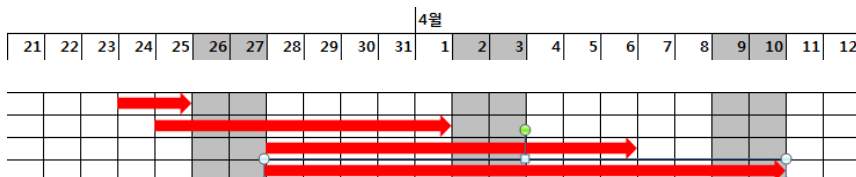
iii. 3 순위

3순위	Web 서버	Django Python	Se-Web-01
			Se-Web-02
	Gateway 서버	Ace	Se-Gate-01
			Se-Gate-02



iv. 4순위

4순위	Web 크롤러	Algorithm	Cr-Web-03
			Cr-Web-04
	upload 서버	Chilkat	Se-Clo-01
	download 서버	Chilkat	Se-Clo-02



B. 개인 개발 일정

- 개인별 일정은 별도 문서를 첨부 합니다.

2016.02.18

이름	개발과제	내용
안수연	서버 구축 관련 사전 조사 및 관련 자료 수집	1. 소켓 라이브러리 조사 (ACE & Boost.asio) 2. ACE 를 이용한 서버 구축 환경 결정 3. ACE 관련 자료 서칭 (관련 강좌: http://www.softlab365.com/wordpress/?page_id=491 관련 책 : ACE 프로그래머 가이드)
황민지	1. 서버 구축 관련 사전조사 및 자료수집 2. ACE 관련 동영상 강좌 학습	1. 서버 구축에 사용되는 소켓 라이브러리 조사 2. 소켓 라이브러리 ACE 와 Boost 비교하여 서버구축에 사용할 라이브러리 결정 3. ACE 관련 자료 서칭 (관련 강좌: http://www.softlab365.com/wordpress/?page_id=491 관련 책: ACE 프로그래머 가이드) 4. ACE 관련 강좌 학습 01. 라이브러리 포팅하기

김소라	검색엔진 개발을 위한 자료 조사, 수집	<p>1. 검색엔진개발을 위한 기술 조사</p> <p>1) crawl - 웹문서를 긁어오는 엔진으로써, 검색 결과 해당하는 웹문서 링크의 그래프를 만든 후 그래프를 방문하면서, 웹문서를 긁어온다.</p> <p>2) 색인 DB 생성 - 긁어온 문서를 이용해서 목록을 만드는 작업. 문서를 토큰을 이용해서 쪼갬 후 목록을 만들고 나중에 다시 역변환 해서 파일로 저장</p> <p>3) 검색 - 사용자 질의어를 분석후 색인 DB 를 뒤져서 해당 Term 을 포함한 문서의 정보를 얻어오는 과정. 단순히 가져오는 정도로는 원하는 품질의 검색결과를 내놓을 수 없기 때문에, 나름대로의 검색알고리즘이 필요한 영역</p> <p>2. 검색 라이브러리 루씬 & 누치에 대한 자료수집</p> <p>1) Lucene</p> <ul style="list-style-type: none"> - 아파치 최상위 Project 중에 하나로 강력한 검색기능뿐만 아니라 간단하기 때문에 IT 업계에서 많이 사용된다. - 루씬은 확장 가능한 고성능 정보검색(IR, Information Retrieval) 라이브러리이다. 꼭 필요한 몇 가지 기본 클래스들을 사용하는 방법만 익히면 색인과 검색 기능을 직접 추가할 수 있다. <p>2) Nutch</p> <ul style="list-style-type: none"> - 광고로 뒤덮인 인터넷 검색 사이트에서 상업적인 요소를 배제하고 검색 그 자체로서의 검색을 구성하고자 진행된 오픈 소스 인터넷 웹 검색엔진 프로젝트
-----	-----------------------	---

		<ul style="list-style-type: none">- 루씬을 이용하여 색인과 검색을 하며 웹에 존재하는 문서(대략 10 억에서 100 억 개 정도)를 처리할 수 있도록 만들어졌고, 물론 문서가 아주 많기 때문에 1 대보다는 여러 대의 서버에서 색인과 검색을 동시에 가동해야 한다.
--	--	--

백승우	1. 검색 엔진의 작동 방법 알아보기 2. 구글의 검색 엔진 구현 방법 알아보기 3. 파이썬 검색 엔진 구현 방법 알아보기	1. 검색 엔진은 크롤링이라는 기술로 인터넷 망을 돌며 정보를 수집한다. 그리고 각 정보를 인덱싱하여 서버에 저장한다. 2. 구글은 크롤링으로 수초, 수분 간격으로 인터넷 망을 돌며 정보를 수집한다. 정보는 링크로 되어있다. 이 간단한 아이디어는 링크의 클릭 빈도에따라 데이터의 정확성을 정렬하며 이것은 검색의 정확도를 효과적으로 높인다. 구글은 검색된 데이터에 관련된 시설에 매년 수천억달러는 투자하고 있다. 3. 오픈라이브러리로 제공되는 Lucene 을 이용해 크롤링부터 인덱싱, 그리고 웹서버까지 모두 제작할 수 있다. 하지만 Lucene 은 JAVA로만 구현 가능하다. 따라서 기존에 C++ 개발 방식은 불가능했다. 이를 위해 가용성이 높은 Python 을 대안으로 찾았고, PyLucene 라이브러리가 현존하고 있음을 알게되었다. 앞으로 PyLucene 의 사용법 및 우리 프로젝트에 최적화 시키는 기술에 대해 연구한다.
-----	--	--

2016.02.19

이름	개발과제	내용
안수연	서버 구축을 위한 자료 수집 서버 구축을 위한 학습-게임 서버 프로그래밍 입문	1. 전반적인 서버 구축 조사를 위한 책 구입 (게임 서버 프로그래밍 입문) 2. 게임 서버 정의 탐독(목차 1) 3. 개발 프로그램 설치(목차 2, 필요한 sw 설치, 윈 노트, slack etc...) 4. 어떻게 만들어야 할 것인가(목차 3, 애자일 방법론) 5. 서버 엔진 라이브러리 만들기(목차 4, 진행중)

황민지	ACE 관련 동영상 강좌 학습	ACE 관련 강좌 학습 02. 간단히 Echo 서버 만들어보기
김소라	파이썬을 이용한 검색엔진 개발을 위한 자료 조사, 수집	1. 파이썬을 활용한 개발을 위한 PyLucene(파이루씬)에 대한 조사 - Lucene 은 오픈소스 자바검색 라이브러리로서, Boogle 프로젝트에 사용할 언어로 적합하지 않기 때문에 대안을 찾아야 했으며, 그 대안으로 PyLucene 발견. - PyLucene: 자바 루씬을 파이썬으로 새롭게 작성한 것이 아니라 자바 VM 을 파이썬 프로세스에 임베딩해서 자바 루씬 코드를 파이썬으로 래핑해 호출. 2. 파이루씬의 활용내용이 들어있는 책 조사&구입 (관련도서 : 빠르게 활용하는 파이썬 3.2 프로그래밍)
백승우	1. 파이썬의 기초 문법 익히기	1-1. 파이썬의 자료형과 연산자에 대하여 복습한다. 1-2. 함수를 생성하고 사용하는 법을 복습하다. 1-3. 제어문(반복문, 조건문, 참거짓)에 관하여 복습한다.
2016.02.20		
이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 학습 -서버엔진 라이브러리 만들기(RandomMT 클래스부터 Thread 클래스까지)	1. RandomMT 클래스 2. SingleTon 클래스 3. Clock 클래스 4. XML parser 클래스 5. Lock 클래스 6. Thread 클래스
황민지		

김소라	검색엔진 개발을 위한 파이썬 공부 시작	1. 구입한 책을 토대로 파이썬 3.2 프로그래밍 공부 시작 - 파이썬 개요 및 설치 - 자료형 및 연산자에 대한 공부 - 함수 공부
백승우		

2016.02.21

이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 학습 -서버엔진 라이브러리 만들기(RandomMT 클래스부터 Thread 클래스까지)	1. RandomMT 클래스 2. SingleTon 클래스 3. Clock 클래스 4. XML parser 클래스 5. Lock 클래스 6. Thread 클래스

황민지	<p>게임 서버 프로그래밍 입문 학습</p> <p>01. 게임 서버 정의</p> <p>02. 개발 프로그램 설치</p> <p>03. 어떻게 만들어야 할 것인가</p> <p>04. 서버엔진 라이브러리 만들기</p>	<p>01. 게임 서버 정의</p> <ul style="list-style-type: none"> - 목적 - 플랫폼 정하기 <p>02. 개발 프로그램 설치</p> <ul style="list-style-type: none"> - IDE 컴파일러 - 리소스 관리 툴 - 프로그램 설계 툴 - 이슈(프로그램 문제 사항) 관리 툴 <p>03. 어떻게 만들어야 할 것인가</p> <ul style="list-style-type: none"> - 애자일 방법론 - 프로그램 설계 툴 활용하기 - 알 마인드 - STARUML - 다이어그램 - 온라인 서버 시스템 설계 <p>04. 서버엔진 라이브러리 만들기</p> <ul style="list-style-type: none"> - UTIL 클래스 만들기 - STDAFX, TYPE, UTIL - STRING - RANDOMMT 클래스
김소라	<p>검색엔진 개발을 위한 파이썬 공부 시작</p>	<p>1. 구입한 책을 토대로 파이썬 3.2 프로그래밍 공부 시작</p> <ul style="list-style-type: none"> - 파이썬 개요 및 설치 - 자료형 및 연산자에 대한 공부 - 함수 공부
백승우	<p>1. 파이썬의 기초 문법 익히기</p> <p>2. 파이썬의 클래스 개념 익히기</p>	<p>1. 파이썬의 제어구문, 조건문이 C/C++/C# 등과 어떻게 다른지 비교하며 복습한다</p> <p>2. 파이썬에서 클래스란 어떻게 작동되는지 그 개념을 파악한다.</p>

2016.02.22

이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 학습 -서버엔진 라이브러리 만들기 (Table 들부터 Monitoring 클래스 까지) 네트워크 구조 만들기 (IOCP 부터 IOCPServer 클래스까지)	1. Table 들 2. Logger 클래스 3. Assert 클래스 4. Minidump 클래스 5. SMTP Mailer 6. ThreadJobQueue 클래스 7. Memory Leak Detector 8. Low fragmentation Heap 9. MemoryPool 클래스 10. CSV parse 클래스 11. GameObject 클래스 12. Task 매니저 13. Monitoring 클래스 14. IOCP 란 15. IOCP 에코 코드 16. 기본 코드 분석 및 기초 설계 17. Server 클래스 18. IOCPServer 클래스
황민지	게임 서버 프로그래밍 입문 학습 서버엔진 라이브러리 만들기	서버엔진 라이브러리 만들기 - SINGLETON 클래스 - CLOCK 클래스 - XMLPARSER 클래스 - LOCK 클래스 - THREAD 클래스 - LOGGER 클래스 - ASSERT 클래스 - MINIDUMP 클래스

김소라	<p>검색엔진 개발을 위한 파이썬 공부 진행</p> <ul style="list-style-type: none"> - 파이썬 문법 공부 진행 	<ol style="list-style-type: none"> 제어문, 제어문과 연관된 유용 함수들 공부 <ul style="list-style-type: none"> - if 문, 참 거짓 판단, 단축 평가 - while 문, for 문, break, continue, else 파이썬에서의 클래스에 대한 전반적 개념과 사용방법 공부 <ul style="list-style-type: none"> - 클래스 객체, 인스턴트 객체 클래스 관련 메서드에 관한 공부 <ul style="list-style-type: none"> - 생성자, 소멸자 메서드 - 정적 메서드, 클래스 메서드 - 상속
백승우	<ol style="list-style-type: none"> 주말 학습 중 부족했던 부분 공부(정적 메서드, 클래스 메서드) 파이썬의 모듈 불러오기, 사용자 모듈 작성하기 연구 파이썬을 C++과 연동하는 방법 연구 데이터 베이스에 접근하는 방법 연구 	<ol style="list-style-type: none"> 비트 교재를 참고하여 정적 메서드와 클래스 메서드의 사용 방법을 익힌다. 수학 모듈, 분수 모듈, 십진법 모듈, 랜덤 모듈, 시간 모듈 등을 구현해본다. 파이썬에서 C++ API 를 사용해 본다. SQL 을 이용하여 데이터 베이스에 접근하고 테이블 레코드 조회, 레코드 정렬, 덤프 데이터 생성을 수행해 본다.
2016.02.23		
이름	개발과제	내용

안수연	<p>게임 서버 프로그래밍 입문 학습</p> <p>- 네트워크 구조 만들기(남은 부분), 패킷 처리에 대해서</p>	<ol style="list-style-type: none"> 1. Session 클래스 2. IOCPSession 클래스 3. SessionManager 클래스 4. HeartBeat 관리 5. Terminal 클래스의 정의 6. Terminal 클래스 7. 여기까지가 서버 코어 8. 두 단말기에 데이터를 주고 받기 9. Stream 클래스 10. Packet 클래스 11. PacketAnalyzer 패킷 분석기 12. ContentsProcess 13. 패킷 메이커 만들기 14. PacketMaker Project 15. DataBase 란 뭘까요?
-----	--	---

황민지	ACE 관련 동영상 강좌 학습	ACE 관련 강좌 학습 SMTP Mailer Table 들 Memory Leak Derector Low fragmemtation Heap MemoryPool 클래스 CSV parser 클래스 GameObject 클래스 Task 매니저 Monitoring 클래스 네트워크 구조 만들기(IOCP)란 IOCP 에코 코드 기본 코드 분석 및 기초 설계 Server 클래스 IOCPServer 클래스 Session 클래스 IOCPSession 클래스 SessionManager 클래스 HearBeat 관리
김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 파이썬 문법 공부 진행	1. 모듈 사용하기, 만들기, 경로, 임포트, 패키지 공부 2. 구문에러, 예외처리, raise 구문, 사용자정의 예외, assert 구문 공부 3. 표준 입출력, 파일 입출력, pickle 공부 4. C/C++와의 연동, 파이썬/C API, ctypes 공부
백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	1. C/C++과 Python 연동하기

2016.02.24

이름	개발과제	내용
----	------	----

안수연	게임 서버 프로그래밍 입문 학습 -데이터 베이스 구조 만들기 게임 서버와 database 의 만남	<ol style="list-style-type: none"> 1. Hyper-V 설치하기(Virtual Box 로 대체) 2. Database 설치 3. Database 설정(네트워크 설정) 4. SQL 의 기본 5. 게임 DB 만들기 6. 프로그램에서 DB 접근하는 방법 7. Database System 설계 8. QueryStatement 클래스 9. QueryRecord 클래스 10. Query 클래스 11. Database 클래스 12. 게임 서버 구조도
황민지	ACE 관련 동영상 강좌 학습	ACE 관련 강좌 학습 Terminal 클래스 두 단말기에 데이터를 주고 받기 Stream 클래스 Packet 클래스 PacketAnalyzer 패킷 분석기 ContentsProcess 패킷 메이커 만들기
김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 파이썬의 중요 모듈들에 관한 공부 시작	<ol style="list-style-type: none"> 1. 문자열 다루기, re 모듈 공부 2. 날짜 다루기, time, datetime 모듈 공부 3. math, fractions, decimal, random 모듈 공부 4. os.path, glob 모듈 공부 5. sqlite3 을 활용한 데이터베이스와의 연결, 함수 공부
백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	<ol style="list-style-type: none"> 1. 문자열 2. 날짜

2016.02.25

이름	개발과제	내용
----	------	----

안수연	게임 서버 프로그래밍 입문 학습 -게임 서버와 database 의 만남(남은 부분) 더미 클라이언트 만들기 서버 방어장치 간단한 채팅 엔진으로 업그레이드 마무리	1. 패킷 처리하기 2. ProgramState 구현 3. 해커로부터 방어 4. 서버 유효기간, 서버인증 서버 5. 패킷 암호화 복호화 6. 상용 보안 프로그램, 안랩 핵실드, Xtrap 7. 설계 8. DB 설정 작업 9. 패킷 추가 작업 10. 서버에서 수정할 점 11. 클라이언트에서 수정할 점 12. 서버 프로그래머로서 첫걸음
황민지	게임 서버 프로그래밍 입문 학습	PacketMaker Project(C#) DateBase 란 뭘까요? Hyper-V 설치하기 DataBase 설치 DataBase 설정 (네트워크 설정) SQL 의 기본 게임 DB 만들기 프로그램에서 DB 접근하는 방법 DataBase System 설계
김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 파이썬의 운영체제 관련 주요모듈 공부 - 파이썬에서 XML 다루는 방법 공부	1. os, sys, threading, queue 모듈 공부 2. XML 에 대해서 알아보기 3.4. os.path, glob 모듈 공부 5. sqlite3 을 활용한 데이터베이스와의 연결, 함수 공부
백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	1. 숫자 2. 파일 시스템

2016.02.26

이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 복습 - 열혈 강의 tcp/ip 동영상 학습	1. 게임 서버 프로그래밍 입문 책 소스 출력 2. 열혈 강의 overlapped IO 모델의 이해 동영상 강좌 학습
황민지	게임 서버 프로그래밍 입문 학습	Query Statement 클래스 Query Record 클래스 Query 클래스 DataBase 클래스 ADODatabase 클래스 DBManager 클래스 게임 서버 구조도 LoginServer 만들기 DB Agent 만들기 더미 클라이언트 더미 클라이언트 설계 DummyClient 프로젝트 시작
김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 파이썬에서 제공하는 네트워크, 인터넷 관련 모듈 공부	1. 파이썬에서 지원하는 네트워크, 인터넷 모듈 공부 2. OpenAPI 이용해 인터넷 상의 도서정보 가져오기 3. 파이썬으로 이메일 보내기
백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	1. 데이터 베이스 2. 운영체제 관련 기능
	2016.02.27	
이름	개발과제	내용

안수연	열혈 강의 tcp/ip 동영상 강좌 학습	1. 열혈강의-overlapped IO 에서의 입출력 완료의 확인 2. 열혈강의-overlapped IO 를 기반으로 IOCP 이해하기-1 3. 열혈강의-overlapped IO 를 기반으로 IOCP 이해하기-2 4. IOCP 의 단계적 구현
황민지		
김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 더 견고한 코드 만들기 위한 라이브러리, 모듈 공부 - disutils 를 통한 배포 - 파이썬 3 을 활용한 통계분석, 기계학습	1. 웹서버 만드는 법 공부 2. 단위테스트, pydoc, doctest 3. disutils 를 이용해 간단한 프로그램 배포 4. 통계분석 - 선형회귀 관련 모듈 SciPy, Numpy, Matplotlib 공부 5. 기계학습 - 인공신경망 관련 모듈 bpnn 모듈 공부 6. pypy
백승우		

2016.02.28

이름	개발과제	내용
안수연	열혈 강의 tcp/ip 동영상 강좌 학습	1. 열혈강의-overlapped IO 에서의 입출력 완료의 확인 2. 열혈강의-overlapped IO 를 기반으로 IOCP 이해하기-1 3. 열혈강의-overlapped IO 를 기반으로 IOCP 이해하기-2 4. IOCP 의 단계적 구현
황민지		

김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 도서관리 프로그램 인터넷 연결 수정	1. 책 보며 간단히 만들었던 도서관리 프로그램의 인터넷 연결이 안되던 문제 해결
백승우		

2016.02.29

이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 코드 분석	1. ServerLibrary.h 2. stdafx.h 3. Type.h 4. Util.h 5. RandomMt 6. Singleton.h 7. Clock.h 8. Clock.cpp 9. main.cpp 10. config.xml 11. config.h 12. config.cpp 13. server.cpp 14. lock.h 15. lock.cpp 16. thread.h 17. thread.cpp 18. logger.h

황민지	TCP/IP Ch22, Ch23 인터넷 강의 학습	<p>각 서버간 통신을 원활하게 하는 비동기 서버를 구현하기 위해 Overlapped IO 에 학습</p> <ul style="list-style-type: none"> - Overlapped IO 소켓 생성 - Overlapped IO 를 진행하는 WSASend 함수, WSARecv 함수 - Event 오브젝트 사용 : WSASend, WSARecv 함수의 여섯 번째 매개변수 활용 IO 완료 및 결과 확인 - Completion Routine 사용 : IO 완료 되었을 때, 자동으로 호출될 함수를 등록하여 IO 완료 후의 작업을 처리하는 방식 <p>Overlapped IO 를 기반으로 IOCP 이해</p> <ul style="list-style-type: none"> - 년-블로킹 모드의 소켓 구성하기 - Overlapped IO 만 가지고 에코 서버 구현하기 - Overlapped IO 모델에서 IOCP 모델로 바꾸기 - Completion Port 생성 및 오브젝트와 소켓 연결 - Completion Port 의 완료된 IO 확인과 스레드의 IO 처리 - IOCP 기반의 에코 서버의 구현
김소라	<p>검색엔진 개발을 위한 파이썬 공부 진행</p> <ul style="list-style-type: none"> - 구글 앱 엔진에 대해 공부 4. 검색 엔진 만들기 위한 준비 	<ol style="list-style-type: none"> 1. 구글 앱 엔진 다뤄보기 2. 본격적인 검색엔진 개발에 앞선 준비 위해 필요한 자료 수집
백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	1. XML
2016.03.01		
이름	개발과제	내용
안수연		
황민지		

김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 검색 엔진 만들기 위한 준비	1. 본격적인 검색엔진 개발에 앞선 준비 위해 필요한 자료 수집 이어 진행
백승우		

2016.03.02

이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 코드 분석	1. logger.cpp 2. main.cpp 3. assert.h 4. assert.cpp 5. minidump.h 6. minidump.cpp 7. smtpmail.h 8. smtpmail.cpp 9. threadjobqueue.h 10. table.h 11. memoryleak.h 12. memory_lowfragmentationheap.h 13. gameobject.h 14. task.h 15. task.cpp 16. monitoring.h 17. server.h 18. server.cpp 19. serveriopc.h 20. serveriopc.cpp

황민지	게임 서버 프로그래밍 입문 학습	Clock 클래스 XML 클래스 Lock 클래스 Thread 클래스 Logger 클래스 Assert 클래스 MiniDump 클래스 SMTP mailer ThreadJobQueue Table 들
김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 인덱싱 관련 공부	1. 파일 인덱싱에 관해 공부 2. 루씬과 PyLucene 에 대해 알아보기
백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	1. 파이썬과 인터넷(계속)

2016.03.03

이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 코드 분석	1. session.h 2. session.cpp 3. iocpsession.h 4. iocpsession.cpp 5. sessionmanager.h 6. sessionmanager.cpp 7. sessionmonitor.h 8. sessionmonitor.cpp 9. terminalsession.h 10. terminalsession.cpp

황민지	게임 서버 프로그래밍 입문 학습	Memory Leak Detector Low fragmentation heap MemoryPool 클래스 CSV parser 클래스 GameObject 클래스 Task 매니저 Monitoring 클래스 IOCP 에코 코드
김소라	검색엔진 개발을 위한 파이썬 공부 진행 - 인덱싱 관련 공부	1. 파일 인덱싱에 관해 어제에 이어 공부 2. 루씬과 PyLucene 활용해봄
백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	1. 파이썬과 인터넷
2016.03.04		
이름	개발과제	내용

안수연	게임 서버 프로그래밍 입문 코드 분석	<ol style="list-style-type: none">1. terminal.h2. terminal.cpp3. terminalmanager.h4. terminalmanager.cpp5. stream.h6. stream.cpp7. packetHeader.h8. packetClass.h9. packetAnalyzer.h10. packetAnalyzer.cpp11. packetFactory.h12. package.h13. contentsProcess.h14. contentsProcess.cpp15. loginProcess.h16. loginProcess.cpp17. excelparser.cs18. makepacket.cs19. headermakepacket.cs20. classmakepacket.cs
-----	-------------------------	---

황민지	ACE 프로그래머 가이드 학습	<p>01. ACE 소개</p> <ul style="list-style-type: none"> - 다양한 운영체제와 여기에서 사용 가능한 여러 C++ 컴파일러를 사용하고 있는 광범위한 시스템에 이식되어있다. - ACE 의 구성 <p>OS 추상 계층(일반적인 시스템 수준 명령들에 대한 wrapper 함수들을 제공)</p> <p>Wrapper facade 계층 (데이터 타입에 안전한 객체지향 인터페이스에 함수와 데이터를 캡슐화한 여러 클래스들로 구성)</p> <p>프레임워크 계층(프레임워크는 관련 어플리케이션 제품군에서의 재사용 가능한 구조를 생산하기 위해 상호협력하는 컴포넌트의 통합 모음)</p> <p>네트워크 서비스 계층(분산 로깅 서비스를 포함한 완전히 구현된 재사용 가능한 몇 가지 서비스들을 제공)</p> <p>02. ACE 의 사용법 개요 및 빌드 방법</p> <ul style="list-style-type: none"> - ACE 베타 배포본을 다운 받은 후 설명을 보며 ACE 를 사용하기 위해 실행파일과 공유 라이브러리(DLL) 양쪽 모두로 빌드
김소라	<p>검색엔진 개발을 위한 파이썬 공부 진행</p> <ul style="list-style-type: none"> - 검색 사이트 만들어보기 	<ol style="list-style-type: none"> 1. 장고(Django)설치, 프로젝트 생성 2. 장고 프레임 워크 3. 장고 MVT 에 대한 공부 4. 간단한 검색 사이트 만들어보기 <ul style="list-style-type: none"> - 파이썬 3 으로 개발 시도했으나 검색 결과 도출이 안되는 상황 발생

백승우	1. 교재 학습(빠르게 활용하는 파이썬 3.2 프로그래밍)	2. 더 견고한 코드 3. distutils 로 배포하기 1. 통계 Linear Regression 2. 인공지능경망 3. pypy
2016.03.05		
이름	개발과제	내용
안수연	게임 서버 프로그래밍 입문 코드 분석	1. factorymakepacket.cs 2. clientclassmakepacket.cs 3. main.cs 4. querystatement.h 5. querystatement.cpp 6. queryrecord.h 7. queryrecord.cpp 8. query.h 9. query.cpp 10. database.h 11. adodatabase.h 12. adodatabase.cpp 13. dbmanager.h 14. dbmanager.cpp 15. loginserver.cpp 16. dbagentprocess.h 17. dbagentprocess.cpp 18. program.cs 19. dummyclient.cs 20. loginform.cs 21. chatwnd.cs 22. chattingform.cs 23. network.cs

		24. packet.cs 25. packetclass.cs 26. packetheader.cs 27. packetfactory.cs
--	--	--

황민지	ACE 프로그래머 가이드 학습	<p>03. ACE 로그 기능 사용하기</p> <ul style="list-style-type: none"> - 모든 프로그램은 진단 정보(오류 메시지, 디버깅 정보 출력 등) 출력 기능을 필요로 한다. 개발자는 실행흐름을 추적하거나 도움이 될만한 실행 정보를 출력하기 위해 ACE의 로깅 기능은 이러한 작업에 대한 여러 가지 방법을 제공하고 동시에 개발자에게 정보 출력 방식과 정보 출력 장치 선택에 대한 다양한 제어권을 부여한다. <p>04. 어플리케이션 실행 정보 수집하기</p> <ul style="list-style-type: none"> - 사용자가 실행 동작을 지시하거나 설정하는 방법을 제공 <p>명령행 매개변수와 옵션 받아들이기 : 각각의 어플리케이션 실행 때마다 알맞게 변경될 수 있는 정보를 얻을 경우에 사용</p> <ul style="list-style-type: none"> - 설정 파일 읽기 : 설정 파일은 어플리케이션 실행중에는 일반적으로 변경하지 않거나 알아두어야 하는 사용자 위주의 정보를 담고 있다.
김소라	<p>검색엔진 개발을 위한 파이썬 공부 진행</p> <ul style="list-style-type: none"> - 검색 사이트 만들어보기 	<p>1. 장고 프레임 워크를 이용해 검색 사이트 만들기 계속 진행</p> <ul style="list-style-type: none"> - 메인 페이지 만들기 - 검색 페이지 만들기 - 파이썬 2.6 버전으로 전날 안되던 검색 결과 도출 시도. 검색 결과는 나오나 재검색 시 실패
백승우		
2016.03.06		
이름	개발과제	내용
안수연		

황민지	ACE 프로그래머 가이드 학습	<p>05. ACE 컨테이너 클래스</p> <p>견고한 컨테이너 클래스는 ACE 툴킷에서 얻을 수 있는 가장 유용한 도구 중 하나이다.</p> <p>어플리케이션 개발에 직접 적용 가능한 효과적이고 플랫폼 독립적인 풍부한 컨테이너들을 제공</p> <p>비록 권장하는 어플리케이션 수준의 컨테이너는 (ACE 프레임워크와 같이 사용 가능한) 표준 C++ 라이브러리 컨테이너이지만, 완전한 표준 C++ 라이브러리 구현이 미흡한데다 메모리 할당과 동기화와 같은 잘 조율된 기능을 필요로 할 때에는 ACE 컨테이너가 더 유용하게 쓰일 수 있다. 많은 ACE 컨테이너는 표준에 가까운 기능들을 제공하고있으며, 이는 다른 컨테이너 타입간의 변환을 쉽게 할 수 있게 도와준다.</p>
김소라	<p>검색엔진 개발을 위한 파이썬 공부 진행</p> <p>- 검색 사이트 만들어보기</p>	<p>1. 장고 프레임 워크를 이용해 검색 사이트 만들기 계속 진행</p> <p>- 책 시연 버전인 파이썬 2.5 버전으로 재시도, 성공</p>
백승우		

2016.03.07

이름	개발과제	내용
----	------	----

안수연	게임 서버 프로그래밍 입문 코드 분석	1. packetUtil.cs 2. packetProcess.cs 3. contentsProcess.cs 4. loginpacketProcess.cs 5. loginContents.cs 6. ChattingPacketProcess.cs 7. ChattingContents.cs 8. programstate.cs 9. formstate.cs 10. programvalidation.cs 11. packetObfuscation.h 12. packetObfuscation.cpp 13. packetObfuscation.cs 14. chattingProcess.h 15. chattingProcess.cpp
황민지	ACE 프로그래머 가이드 학습	06. 기본적인 TCP/IP 소켓 사용법 - Connetor : 연결을 능동적으로 구축 - Acceptor : 연결을 수동적으로 구축 - Stream : 데이터를 전송 - Address : 주소 종단점 지정 방법 정의 - TCP/IP 프로그래밍을 위해 ACE_SOCKET_Connector ,ACE_SOCKET_Acceptor, ACE_SOCKET_Stream, ACE_INET_Addr 같은 ACE 의 소켓 wrapper 클래스를 사용 6.1 간단한 클라이언트 만들기 6.2 클라이언트를 보다 견고하게 만들기

김소라	<p>검색엔진 개발을 위한 파이썬 공부 마무리</p> <ul style="list-style-type: none"> - 검색 사이트 마무리 - 책 완독과 함께 부족한 부분 복습 	<p>1. 장고 프레임 워크를 이용한 검색 사이트 완성</p> <ul style="list-style-type: none"> - Pydev 설치, 사용 - 책의 부록인 pdb 모듈을 이용한 디버깅 부분 읽어보기 <p>2. 책 완독과 함께 부족했던 부분 복습</p> <ul style="list-style-type: none"> - C/C++ 연동 - 파이썬과 인터넷
백승우	<p>1. 웹 크롤러 시험 코딩</p> <p>2. 장시간 회의</p>	<p>1. 웹 크롤러의 개념을 잡고 기초 크롤러 예제 코드를 찾아봄</p> <p>2.</p> <ul style="list-style-type: none"> - 국내 블로그 사이트 검색 엔진 구현(크롤이 불가능한 네이버가 제외된다는 제한) - 프로그래밍 언어 코드에대한 설명을 서치해 주는 검색 엔진 구현(영어권 데이터의 도움 없이는 정보량이 적다는 제한) - 영상만을 서치해 주는 검색 엔진 구현(영상을 크롤해 오기가 어려움) - 쇼핑을 돕는 상품을 검색해주는 엔진 구현(자료에대한 최저가, 카테고리 등이 구현됨. 상품 정보를 크롤해 오기가 어려움) - 요리 레시피 검색 엔진 구현 - 국내 지역별 데이터 검색 엔진 구현 - 국사에 대한 검색 엔진 구현 - 인물에대한 검색 엔진 구현(위인이 아닌 평범한 사람. 정보 보호와 페북 API 미지원으로 한계가 있음) - 크롬 app, 인스턴트 음악과 같은 기능 구현

2016.03.08

이름	개발과제	내용
----	------	----

안수연	ACE 학습 - ACE 프로그래머 가이드 Chapter 01, 02	<p>Chapter 01</p> <ol style="list-style-type: none"> 1. ACE 의 역사 2. ACE 의 장점 3. ACE 의 구성 4. 패턴, 클래스 라이브러리, 프레임워크 5. 여러 운영체제에 개발중인 코드 이식하기 6. C++ 컴파일러간 차이 완화시키기 7. 바이트 단위 문자열 및 wide 문자열 모두 사용하기 8. ACE 에 대한 정보 및 기술 지원 <p>Chapter 02</p> <ol style="list-style-type: none"> 9. ACE 버전에 대한 주의사항 10. ACE 배포본 안내 11. ACE 빌드하기 12. 개발중인 어플리케이션 소스에 ACE 를 포함하는 방법 13. 작성중인 ACE 어플리케이션 빌드 방법
황민지	ACE 프로그래머 가이드 학습	<p>06. 기본적인 TCP/IP 소켓 사용법</p> <p>6.3 서버 구축하기</p> <ul style="list-style-type: none"> - acceptor 객체는 bind()와 accept()와 같은 내장된 세부 처리들을 다룬다 <p>ACE 객체를 사용함으로써 보다 관리하기 쉽고, 이식 가능한 어플리케이션을 만드는 것이 가능하다. 보다 상위 수준의 추상화를 사용하여 작업하기 때문에 sockaddr_in 구조체를 0 으로 초기화하거나, 이를 언제 어떻게 형 변환해야 한다는 내용과 같은 네트워크 프로그래밍의 성가신 세부사항을 다룰 필요가 없다. 개발중인 어플리케이션은 보다 타입 안전성이 강해지게 되며, 이는 오류 발생 가능성을 줄여주게 된다.</p>

		게다가, 설령 오류가 있다고 하더라도 실행중보다는 컴파일 시점에서 더 많이 검출된다.
김소라	본격적인 검색엔진 개발에 앞선 회의, 구성	<ol style="list-style-type: none"> 1. 본격적인 검색 엔진 개발을 위해 Boogle 컨셉 회의 <ul style="list-style-type: none"> - 물리적 한계로 구글과 같은 검색 엔진을 만드는 것은 무리라 판단. - 컨셉 회의 후 여러 의견이 나왔으나 통합자막 활용한 문장사전 만들기로 주제 결정 2. 서버 + 검색엔진 통합한 Boogle Architecture 구성 3. 부가 기능으로 인스턴트 음악 APP 등의 추가 의견

백승우	1. 파이썬 재설정 2. 장시간 회의	1. 기초 웹 크롤러를 만들어 보려고 했으나, 파이썬과 호환이 되지 않음. 파이썬 버전을 다운그레이드 함. 2. - 주제를 선정하여 확정함. 생활영어 검색 엔진. 각종 자막 사이트로 부터 smi 파일을 다운로드하여 영화별 한국어, 영어 자막을 추출한 후 이를 DB 에 저장하고 생활영어 교육으로 활용. - 회원 관련 기능(단어장)의 추가 가능성, 영어문장에 대한 영화 내 음성 또는(제한 될 경우) 기계번역음 지원 가능성 있음. - 부가 기능으로, 인스턴트 음악 app 의 추가 가능성 있음.
-----	-------------------------	---

2016.03.09

이름	개발과제	내용
안수연	ACE 학습 - ACE 프로그래머 가이드 Chapter 03	Chapter 03 1. 기본적인 로깅 및 추적 기능 사용법 2. 로깅 수준 설정 3. ACE 로깅 매크로 기능 수정하기 4. 로깅 출력 방식 변경하기 5. 역호출 기능 사용하기 6. 로깅 클라이언트 및 서버 데몬 7. LogManager 클래스 8. ACE_Logging_Strategy 를 사용하여 실시간으로 설정 변경하기

황민지	ACE 프로그래머 가이드 학습	<p>07. 이벤트 처리 및 다중 입/출력 스프림</p> <ul style="list-style-type: none"> - ACE Reactor 프레임워크 : 다중 스레드의 사용으로 인한 과부하 없이 여러 이벤트 발생을 동시에 처리하기 위한 매우 강력하고 융통성 있는 시스템. 단일 reactor 인스턴스는 타이머, 시그널, 입/출력 이벤트의 동작을 쉽게 처리해준다 1. ACE_Event_Handler로부터 하나 이상의 클래스를 상속받아 가상 역호출 메소드에 어플리케이션만의 이벤트 처리 동작을 추가 2. 어플리케이션의 이벤트 처리 객체들을 ACE_Reactor에 등록하고 처리한 이벤트와 이 객체들을 연관 3. ACE_Reactor 이벤트 루프를 실행 4. ACE_Reactor에 감시할 이벤트와 ACE_Event_Handler를 등록(register_handler)하고 event_loop를 호출해주면 이벤트 발생시 ACE_Event_Handler의 특정함수(handle_input, handle_close 등)를 호출해준다. Async 방식
김소라	<p>본격적인 검색 엔진 개발 돌입</p> <ul style="list-style-type: none"> - 자막 크롤러 만들기 	<p>1. 자막 크롤러 설계</p> <ul style="list-style-type: none"> - 인터넷 상 여러 자막사이트 중 파싱해오기 적합한 사이트 알아보기 - 곰tv 통합자막게시판의 주소에서 각 게시글의 링크 따온 후 db에 저장 - db에 저장된 링크들 방문하여 다운링크 통해 자막 다운
백승우	1. 파이썬 재설정	<p>1. 파이썬 설정을 변경하고 요구되는 모듈을 다운로드 및 인스톨 함. 그러나 계속 호환 되지않아 위 과정을 반복함.</p>

2016.03.10

이름	개발과제	내용
----	------	----

안수연	ACE 학습 - ACE 프로그래머 가이드 Chapter 04	<p>Chapter 04</p> <ol style="list-style-type: none"> 1. ACE_Get_Opt 기능 변경하기 2. 매개변수 순서 변경하기 3. 설정 섹션 4. 설정 저장소 5. 매개변수 배열 만들기
황민지	ACE 프로그래머 가이드 학습	<p>07. 이벤트 처리 및 다중 입/출력 스프림</p> <p>7.3 시그널</p> <ul style="list-style-type: none"> - 시그널 : 시그널에 응답하는 기능을 구현(처리할 시그널 번호와 시그널 수신시 호출될 역호출 함수 포인터를 매개변수로 하여 signal() 시스템 함수를 호출) - 단일 이벤트 핸들러상에서의 다중 시그널 처리 : 하나의 시그널이 아닌 시그널 집합을 매개변수로 지정하여 처리 <p>7.4 통지처리</p> <ul style="list-style-type: none"> - handle_signal() 안에서 실행중일 때 코드는 일반적인 실행흐름하에 놓이지 않고 시그널 상태에 놓이게 되어 많은 동작 실행이 제약을 받게된다. 현재 내부 이벤트 다중 수신 장치가 대기중인 경우, 이 이벤트를 큐에 적재하면 reactor 는 이를 바로 처리 <p>7.5 타이머</p> <ul style="list-style-type: none"> - 어플리케이션에서 주기적으로 작업이 처리되도록 할 필요가 있다. 이에 대한 접근법은 별도의 쓰레드를 만들거나, 알맞을 sleep()함수를 호출하여 이를 처리 하는 것이다. ACE에서는 timerTask()함수를 정의 : timerTack() 함수는 자식 프로세스를 생성한 다음, 나중의 소거 작업을 위해 호출한 함수에게 생성한 프로세스 ID 를

		반환한다. 자식 프로세스 안에서는 지정된 실행 주기 동안 기다린 후 작업 함수 포인터를 호출
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기 주제 수정 및 확대	1. 자막 크롤러 개발 - 자막 크롤러 개발 돌입 2. 검색 엔진의 검색 범위가 자막으로 한정된데 있어 예상 프로젝트의 규모가 과하게 축소되었다는 의견에 동의해 자막검색기능과 더불어 추가 검색기능 확장-> 웹 크롤링

백승우	1. 회의 제의 2. 파이썬 재설정 3. 웹 크롤러 제작	1. 검색 엔진의 검색 범위를 자막 검색으로 한정하고 며칠간 진행해 본 결과, 프로젝트의 규모가 너무 작다고 의의를 제기함. 이에 모두가 동의하고, 웹 검색으로 다시 확대하되 자막 검색을 부가 기능으로 추가하기로 함. 2. 어제에 이어 파이썬 설정을 변경하고 요구되는 모듈을 다운로드 및 인스톨하는데 일부 성공함. 3. 가능한 만큼 웹 크롤러를 제작해 봄. 아직 작동되지 않음
-----	---------------------------------------	--

2016.03.11

이름	개발과제	내용
안수연	ACE 학습 - ACE 프로그래머 가이드 Chapter 05	Chapter 05 1. 템플릿 기반 컨테이너 2. 객체 기반 컨테이너 3. 반복자 4. 이중 연결 리스트 5. 스택 6. 큐 7. 배열 8. 집합 9. 맵 관리자 10. 해시 맵 11. 자동 조정되는 이진 트리 12. ACE_Allocator 13. ACE_Malloc
황민지	Oracle DataBase 구축	웹, 사전, 자막 크롤러들에게 받은 데이터들을 저장하고 있기위해 Oracle 의 DataBase 를 이용하기 때문에 Oracle 설치

김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기 확대된 주제의 아키텍처 재구성	1. 회의 후 주제확대에 따라 자막 크롤러 부분 파트 전담 - 자막 크롤러와 DB 서버간 통신로 개통. - 그 후 웹서버에 자막크롤러 연결하고 웹서버와 DB 서버간 통신로 개통 2. 회의 후 확대된 아키텍처 재구성
백승우	1. 웹 크롤러 제작	1. 크롤러의 핵심 기능인 인터넷 접속을 가능케 하는 BeautifulSoup의 인스톨에 성공함. 네이버, 네이버 블로그, 다음 블로그의 글을 크롤하여 콘솔 화면에 출력함.

2016.03.12

이름	개발과제	내용
안수연		
황민지		
김소라		
백승우		

2016.03.13

이름	개발과제	내용
안수연		
황민지		
김소라		
백승우		

2016.03.14

이름	개발과제	내용
----	------	----

안수연	DB 서버 구축 시작	크롤러가 수집한 데이터를 저장하기 위한 데이터베이스(오라클) 설치 패킷을 전송 받아 데이터 베이스에 내용을 저장하기 위한 데이터 베이스 서버 구축 시작 - ODBC 를 이용하여 데이터 베이스와 접속 및 연결 상태 유지 후 데이터 송수신 - 패킷을 전송받아 각 패킷에 따른 업무를 수행하기 위한 서버 구축 시작
황민지	Oracle DataBase 구축 후 DB 와 연동하는 DB 서버 구축	DB 에 접속한 후 쿼리문을 만들어 전달하여 DB 의 데이터를 가지고 오는 서버를 구축한 후 DB 와 연동하여 정상작동하는지 확인
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기	1. 자막 크롤러 개발 - 자막 크롤링 기능 거의 완성했으나 한글 인코딩 문제 발생 -> 해결 시도
백승우	1. 사전 크롤러 제작	1. 전체 일정 상, 사전 크롤러가 우선적으로 완성되고 서버와 연계 되기 때문에 웹 크롤러 개발을 일시중단하고 사전 크롤러를 개발하기 시작. 이를 위해 전체적인 기능별 구조를 설계함.

2016.03.15

이름	개발과제	내용
----	------	----

안수연	사전, 자막, 웹 크롤러 패킷 정의 패킷에 따른 DB 구축	<p>자막 크롤러 패킷 정의</p> <ul style="list-style-type: none"> - B_C_NOTIFY_SUBTITLE = 201 - B_C_NOTIFY_SUBURL = 202 - B_C_REQ_SUBURL = 203 - B_S_ANS_SUBURL = 204 <p>사전 크롤러 패킷 정의</p> <ul style="list-style-type: none"> - B_C_NOTIFY_DIC_URL_INFO = 301 - B_C_NOTIFY_DIC_WORD = 302 - B_C_NOTIFY_DIC_CONTENTS = 303 - B_C_NOTIFY_DIC_URL_PATH = 304 - B_C_REQ_DIC_URL = 305 - B_S_ANS_DIC_URL = 306 - B_I_NOTIFY_FILE = 307 <p>웹 크롤러 패킷 정의</p> <ul style="list-style-type: none"> - B_C_REQ_WEB_URL = 401 - B_C_NOTIFY_WEB_URL_ERROR = 402 - B_C_NOTIFY_WEB_CONTENTS = 403 - B_C_NOTIFY_WEB_WORD = 404 - B_C_NOTIFY_MOTHER_URL = 405 - B_C_REQ_MOTHER_URL_EXIST = 406 - B_C_NOTIFY_WEB_URL_INFO = 407 - B_S_ANS_WEB_URL = 408 - B_S_ANS_MOTHER_URL_EXIST = 409 <p>패킷에 따른 서버 업무 작업 마무리</p>
황민지	Oracle DataBase 에 오류가 생겨 다시 구축	사전, 웹 DB 와 자막 DB 를 각각 다른 PC 의 DB 에 저장하기 때문에 다른 PC 의 DB 에 접속하여 DB 의 값을 가져오는 작업을 위해 설정

		도중오류가 생겨 PC 를 리셋한 후 다시 DB 구축 시작
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기 1 차 발표 준비	1. 자막 크롤러 개발 - 자막 크롤링해오는 사이트에서 새 게시물이 추가될 경우 갱신가능하게 하는 UPDATE 기능 구현 2. 다운 된 SMI 파일을 가공해 서버와 통신가능하게 할 예정 3. 다음날 있을 1 차 발표 위한 ppt, 발표 준비
백승우	1. 사전 크롤러 제작 2. 웹 크롤러 보완	1. 어제 설계된 내용을 바탕으로 기능 별로 함수를 구현함 2-1. 크롤된 데이터를 저장장치에 파일 형태로 저장 할 수 있게 구현함. 2-2. 크롤러가 깊이 우선 탐색 알고리즘으로 작동하는데 인터넷 공간을 깊이 우선 방식으로 전부 크롤하기 위한 컴퓨터 스펙이 되지 않으므로, 크롤의 깊이를 제한하는 기능을 설계함.

2016.03.16

이름	개발과제	내용
----	------	----

안수연	데이터 베이스 서버에서 패킷을 주고 받기 위한 송수신 함수 구현	<p>구현 함수</p> <ul style="list-style-type: none"> - int subTable(PK_C_NOTIFY_SUBTITLE& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 영화 제목에 따른 테이블을 생성 후 해당 테이블에 데이터 삽입하는 함수 - void subWord(PK_C_NOTIFY_SUBTITLE& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc, int id) : 각 문장에 따른 키워드 테이블을 생성 후 해당 테이블에 삽입하는 함수 - void urlTable(PK_C_NOTIFY_SUBURL& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 자막을 크롤링 해오기 위해 받은 URL 을 저장하는 함수 - PK_S_ANS_SUBURL urlAnsAll(SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 자막크롤링을 위해 자막을 다운받기 위한 URL 을 요청했을 때 아직 다운로드 하지 않은 URL 을 응답해 주는 함수
황민지	DB 오류 해결 및 Web 서버와의 통신 패킷 정의	<p>리스너가 없다는 오류가 생겨 인터넷에서 해결 방법을 계속 찾으며 오류해결을 위해 계속 시도하였지만 되지않아 다시 한 번 컴퓨터를 리셋 후 Oracle 설치 정상작동이 되는 것을 확인</p> <p>Web 서버와의 통신을 위한 패킷을 정의</p> <ul style="list-style-type: none"> - B_C_REQ_SEARCH = 501, //웹페이지에서 검색 할 때 검색어와 검색모드 등 처음 검색했을 때 상태 정보를 보낼 때 사용하는 패킷 - B_S_ANS_DIC_CONTENTS = 506, //사전 본문 내용을 보낼 때 사용하는 패킷

김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기 1 차 발표	1. 자막 크롤러 개발 - 다운로드한 SMI 파일을 서버에 보내기 전 검색이 쉽게 문장을 단어별로 분할하여 인덱스로 사용하게 하기 위한 작업 돌입 2. 블로그 검색엔진, 사전 검색엔진, 자막 검색엔진 크롤링과 서버 패킷정의, DB 서버 시연
백승우	1. 사전 크롤러 제작 2. 웹 크롤러 보완	1-1. 어제에 이어서 함수를 보완 및 구현 함. 1-2. 크롤된 내용으로 부터 한글 단어를 추출할 수 있는 파이썬 모듈 KoNLPy 를 발견했으나 내부 함수를 이용하는데 실패함.(initVM Error 786432KB memory required) 2-1. 크롤된 url 을 저장할때 다중 리스트 방식을 적용하여, 현 url 외에도 해당 url 의 링크가 존재했던 부모 url 과 현 url 에 기록된 url 들을 자식 url 을 추가로 저장함. 2-2. url 로 부터 데이터가 올바르게 적용됐을 경우를 대비해서 페이지가 이전 페이지로 부터 링크 개수에 따른 점수를 상속 받는 재귀 함수를 설계함.

2016.03.17

이름	개발과제	내용
안수연	패킷 PK_C_NOTIFY_SUBTITLE, 함수 subWord 함수 수정 서버 테스트를 위한 자막 크롤러 클라이언트 구축	기존 한 문장에 대한 키워드를 한 패킷에 전부 받는 방식에서 패킷마다 키워드를 하나씩만 삽입하여 전달해 주는 방식으로의 변경에 따른 패킷(PK_C_NOTIFY_SUBTITLE)과 함수(subWord) 수정 서버 테스트를 위한 자막 크롤러 클라이언트 구축 - PK_C_NOTIFY_SUBTITLE 클라이언트 구축 - PK_C_NOTIFY_SUBURL 클라이언트 구축

		- PK_S_ANS_SUBURL 클라이언트 구축
황민지	사용자 DB 연결 서버를 이용해, 사용자 요청에 따라(자막 검색) 웹서버에 결과 패킷 보내기.	Web 서버의 요청 패킷을 분석한 후 보내줄 데이터를 DB 에서 가져오기위해 쿼리문을 만든 후 DB 와 연결을 하여 데이터를 가져온 후 정의한 패킷에 해당 데이터를 넣은 후 보내주는 것을 임시 client 서버와 연동 후 실험
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기	1. 자막 크롤러 개발 - 서버팀과 회의를 거쳐 프로토콜 수정. DB 서버와 자막 크롤러간의 송수신 가능하게 데이터 가공해 게이트웨이에 보낼 패킷 구성함
백승우	1. 사전 크롤러 제작 2. 프로토콜 설계	1-1. 웹상에있는 모든 사전 서비스들에 대해 크롤을 실시할 수 없다고 판단하고 사전 크롤러의 대상을 Wikipedia 로 한정함. 1-2. 파이썬 모듈 KoNLPy 의 initVM error 를 여전히 처리하지 못함. (initVM Error 786432KB memory required) 2. 사전 크롤러와 연계하게 될 서버를 담당할 안수연 팀장과 프로토콜 패킷 모양에 대해 간단히 추상화 시켜봄.

2016.03.18

이름	개발과제	내용
----	------	----

안수연	클라우드에 파일 업로드를 위한 서버 학습	<p>클라우드 서버 서칭 및 선정</p> <ul style="list-style-type: none"> - 아마존의 클라우드 스토리지 서비스(AWS S3) 로 선정 <p>아마존 클라우드 스토리지 서비스에 업로드하기 위한 라이브러리 서칭</p> <ul style="list-style-type: none"> - chilkat 라이브러리로 선정 후 학습
황민지	사용자 DB 연결 서버를 이용해, 사용자 요청에 따라(사전 검색) 웹서버에 결과 패킷 보내기. 컴퓨터 스펙을 극복하기 위해 클라우드 서버에 관해 알아봄.	<p>사용자 자막 검색의 따라 Web 서버가 보내주는 패킷을 분석하는 것이 완성되지 않아 완성 후 다시 임시 Client(Web 서버)서버를 이용하여 테스트 해봄</p> <p>웹과 사전간의 패킷을 정의</p> <ul style="list-style-type: none"> - B_S_ANS_DIC_INFO = 505, //사전 링크, 제목에 대한 정보를 보낼 때 사용하는 패킷 - B_S_ANS_DIC_CONTENTS = 506, //사전 본문 내용을 보낼 때 사용하는 패킷 <p>사전 본문 내용을 txt 파일로 저장하기에는 많은 용량을 차지하여 다른 곳에 저장하는 방법과 서버간 파일을 공유하는 방법을 생각하던 중 클라우드 서비스를 사용하면 좋겠다는 의견이 있어 클라우드 서비스와 서버를 연동하는 방법을 찾아봄</p>
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기	<p>1. 자막 크롤러 개발</p> <ul style="list-style-type: none"> - 자막 크롤러 파일 최적화 위해 영문장을 단어별로 분할해 인덱스 만드는 과정을 진행하며 영어 형태소 분석 패키지(nltk) 필요. 패키지 다운받는 중 에러 발생해 해결 시도

백승우	1. 사전 크롤러 제작	<p>1-1. 웹상에있는 모든 사전 서비스들에 대해 크롤을 실시할 수 없다고 판단하고 사전 크롤러의 대상을 Wikipedia 로 한정함.</p> <p>1-2. KoNLPy 의 initVM 이 자바 가상머신의 초기화 함수라는 것을 알았고 한글 단어를 추출하는데 성공함. 이를 이용해 한글 단어를 추출하는 기능을 구현함. 아직 정확한 예러 발생 원인을 파악 하진 못함.</p> <p>1-3. 영어 단어를 추출하기 위해 파이썬 모듈 NLTK 패키지를 사용하기로 함. 하지만 NLTK DownLoad Server 에 접속 오류가 발생. 이를 해결하지 못함.</p>
-----	--------------	---

2016.03.19

이름	개발과제	내용
안수연		
황민지		
김소라		
백승우		

2016.03.20

이름	개발과제	내용
안수연		
황민지		
김소라		
백승우		

2016.03.21

이름	개발과제	내용
----	------	----

안수연	자막 크롤러와 연동 테스트 후 오류 해결 클라우드 서버 구축 중	<p>자막 크롤러와 연동 테스트</p> <ul style="list-style-type: none"> - 테이블명의 길이 문제 발생 - 문장 내에 잘못된 문자 값 입력 발생 <p>연동 테스트 후 오류 해결</p> <p>chilkat 라이브러리를 이용한 클라우드 업로드 서비스 테스트 및 구축</p> <ul style="list-style-type: none"> - aws 가입 - api 사용을 위한 access key, secret key 발급
황민지	사용자 요청에 따라(사전 검색) 웹서버로 결과 패킷 보내기를 완성시키고, 사용자 요청에 따라(기본 웹 검색) 결과 데이터 패킷 보내기. 클라우드 서버 구축하는 방법 알아보기.	<p>사용자 사전 검색의 따라 Web 서버가 보내주는 패킷을 분석하는 것이 완성되지 않아 완성 후 다시 임시 Client(Web 서버)서버를 이용하여 테스트 해봄</p> <p>웹검색의 패킷을 정의</p> <ul style="list-style-type: none"> - B_S_ANS_WEB_INFO = 507, //웹 링크, 제목에 대한 정보를 보낼 때 사용하는 패킷 - B_S_ANS_WEB_CONTENTS = 508, //웹 본문 내용을 보낼 때 사용하는 패킷 <p>웹 검색에 따른 해당 데이터 패킷을 보내는 것을 테스트</p> <p>아마존 클라우드 서비스를 사용하기로 결정한 후 서버와 클라우드 서비스를 연동시킬 수 있는 Chilkat 소켓 라이브러리를 공부</p>
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기	<p>1. 자막 크롤러 개발</p> <ul style="list-style-type: none"> - 자막 크롤러와 DB 서버 간 통신 장애 문제 해결 - 문제되었던 nltk 패키지 설치 성공

백승우	<p>1-1. 웹상에있는 모든 사전 서비스들에 대해 크롤을 실시할 수 없다고 판단하고 사전 크롤러의 대상을 Wikipedia 로 한정함.</p> <p>1-2. KoNLPy 의 다운로드와 인스톨에 성공함. 이를 이용해 한글 단어를 추출하는 기능을 구현함.</p> <p>1-3. NLTK 필수 파일을 수동으로 다운로드하는데 성공함.</p>	<p>1-1. 웹상에있는 모든 사전 서비스들에 대해 크롤을 실시할 수 없다고 판단하고 사전 크롤러의 대상을 Wikipedia 로 한정함.</p> <p>1-2. KoNLPy 의 다운로드와 인스톨에 성공함. 이를 이용해 한글 단어를 추출하는 기능을 구현함.</p>
-----	---	---

2016.03.22

이름	개발과제	내용
안수연	클라우드 서버 구축 완료	<p>chilkat 라이브러리를 이용한 클라우드 업로드 서비스 구축 완료</p> <ul style="list-style-type: none"> - 패킷 생성 : B_I_NOTIFY_FILE = 307 - access key, secret key 를 이용해 aws 접속 완료 - 해당 bucket 에 파일 업로드 완료

황민지	자막과 사전 검색에 따른 DB 서버에 접속후 데이터를 가져오기 위해 쿼리문을 만들고 가져온 데이터들을 저장하는 함수를 정의	<pre>vector<PK_S_ANS_SUBTITLE> findWord(PK_C_REQ_SEARCH& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc);</pre> : Web 서버에서 보내온 패킷을 전달하여 쿼리문을 만들고 영화 제목과 영어/한글자막 데이터를 DB 에서 가져와 응답패킷을 만들어주는 함수 <pre>vector<PK_S_ANS_DIC_INFO> findDictionary(PK_C_REQ_SEARCH& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc);</pre> : Web 서버에서 보내온 패킷을 전달하여 쿼리문을 만들고 url 과 title 데이터를 DB 에서 가져와 응답 패킷을 만들어주는 함수
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기	1. 자막 크롤러 개발 - 자막 크롤러의 기능 구현 완료 - 독자적으로 자막 크롤러의 통신 기능 테스트
백승우	1. 사전 크롤러 제작	1. NLTK 필수 파일을 수동으로 다운로드하는데 성공함.

2016.03.23

이름	개발과제	내용
----	------	----

안수연	사전 크롤러와 패킷 조율	회의를 통해 사전 크롤러 정의 및 구축 B_C_NOTIFY_DIC_URL_INFO = 301, PK_C_NOTIFY_DIC_URL_INFO B_C_NOTIFY_DIC_WORD = 302, PK_C_NOTIFY_DIC_WORD B_C_NOTIFY_DIC_CONTENTS = 303, PK_C_NOTIFY_DIC_CONTENTS B_C_NOTIFY_DIC_URL_PATH = 304, PK_C_NOTIFY_DIC_URL_PATH B_C_REQ_DIC_URL = 305, PK_C_REQ_DIC_URL B_S_ANS_DIC_URL = 306, PK_S_ANS_DIC_URL B_I_NOTIFY_FILE = 307, PK_I_NOTIFY_FILE
황민지	사전 DB 를 사용자 DB 연결서버와 통신 할 수 있도록 패킷 조율	사전 DB 에서 데이터가 저장되어 있는 테이블간의 상관관계를 이용하여 데이터를 가져올때 사용할 쿼리문 작성을 위해 패킷을 분석한 후 쿼리문 작성
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기	1. 자막 크롤러 개발 - 자막크롤러의 통신 기능 구현 완료. - 서버팀과 회의 후 통신 테스트 예정
백승우	1. 사전 크롤러 통신 기능 구현	1. 크롤러 담당 서버와 패킷을 조율함.

2016.03.24

이름	개발과제	내용
----	------	----

안수연	사전 크롤러와 데이터 베이스 서버 구축	<p>사전 크롤러 연동을 위한 함수 구축</p> <ul style="list-style-type: none"> - void dicUrl(PK_C_NOTIFY_DIC_URL_INFO& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 전달 받은 url 저장, 전체 정보 - void insertUrl(PK_C_NOTIFY_DIC_URL_INFO& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 해당 url 이 존재 하지 않을 경우 insert - void updateUrl(PK_C_NOTIFY_DIC_URL_INFO& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 해당 url 이 존재 할 경우 update - void dicWord(PK_C_NOTIFY_DIC_WORD& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 사전 내용에서 뽑은 키워드로 테이블 생성 후 삽입하는 함수 - void dicContents(PK_C_NOTIFY_DIC_CONTENTS& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 전달받은 사전 본문 내용을 파일로 저장 후 upload 서버로 패킷 전송 - void dicSaveUrl(PK_C_NOTIFY_DIC_URL_PATH & data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 전달 받은 url 저장, url 만 - void dicGetUrl(PK_S_ANS_DIC_URL& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 방문하지 않은 url 요청할 경우 방문하지 않은 url 을 추출하기 위한 함수
-----	-----------------------	--

황민지	Switch 서버 구축	크롤러에게 받은 패킷과 Web 서버에서 받은 패킷을 각 크롤러 DB 서버와 사용자 DB 서버에게 나눠서 보내주는 Switch 서버를 구축 각 패킷을 분석한 후 크롤러 DB 서버에게 보낼 패킷인지 사용자 DB 서버에게 보낼 패킷인지 구분한 후 해당 서버에게 전송시키는 역할
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기 마무리 단계	1. 자막 크롤러 개발 - 서버팀과 통신하기 위해 일차적으로 만들었던 패킷 조율 필요. - 패킷 재정의
백승우	1. 사전 크롤러 제작 완료 1 차	1-1. 사전 크롤러가 Wikipedia 의 특정 단어에 대한 웹 페이지 내용을 다운로드하고 data 가공함. 1-2. 프로토콜에 따라 가공된 data 를 송신 하는데 성공함.

2016.03.25

이름	개발과제	내용
안수연	크롤러 서버 최적화	기존 reactor(동기)로 만들었던 데이터 베이스 서버를 ace proactor 를 사용해 비동기 서버로 변경 - ace proactor 로 변경 하기 위한 ace proactor 학습 - ace proactor client 만들기 학습 (참조 사이트 : http://www.softlab365.com/wordpress/?p=931) - ace proactor server 만들기 학습 (참조 사이트 : http://www.softlab365.com/wordpress/?p=923) - 학습 후 실제 적용

황민지	Switch 서버와 DB 서버 연동	임시 client 서버에서 보낸 패킷을 Switch 서버에서 구분하여 DB 에게 보내는 것을 테스트 한 후 Switch 서버와 DB 서버간 연동에 오류를 고침
김소라	본격적인 검색 엔진 개발 돌입 - 자막 크롤러 만들기 마무리	1. 자막 크롤러 개발 완료 - 서버팀과 통신 테스트. 자막 크롤러 완성
백승우	1. 웹 크롤러 제작	1-1. 큐스택 방식을 이용한 웹 크롤러를 예제로 하여 기존의 웹 크롤러를 업그레이드 함. 1-2. 검색의 정확도를 위한 웹 페이지 점수 연산 재귀 함수를 구현 하는 중.

2016.03.26

이름	개발과제	내용
안수연		
황민지		
김소라		
백승우		

2016.03.27

이름	개발과제	내용
안수연		
황민지		
김소라		
백승우		

2016.03.28

이름	개발과제	내용
안수연	게이트 웨이 서버 구축	게이트 웨이 서버 구축 - 게이트 웨이 서버로의 구현을 위해 기존 ace proactor 소스 코드 수정 - 패킷 검사해 pass / non pass 해주는 기능 구현 - 패킷을 받아 connect 한 서버로의 전송하는 기능 구현

황민지	동기화로 스위치 서버랑 디비서버 완성	서버간의 연동 오류 수정 후 동기화로 스위치 서버와 디비서버간 연동 성공
김소라	검색 사이트 만들기 - Boogle 웹페이지 구상	1. Boogle 사이트 만들기 위해 장고 프레임워크 사용해 웹서버 구상. 구축 시작
백승우	1. 웹 크롤러 제작	1. 웹 크롤러의 크롤 깊이를 제한하는 기능을 일시 중지했던 설계를 바탕으로 재구성함.

2016.03.29

이름	개발과제	내용
안수연	게이트웨이와 각 스위치, 클라이언트와 연결 서비스 기능 구현	<p>게이트웨이와 각 스위치, 클라이언트와 연결 서비스 기능 구현</p> <ul style="list-style-type: none"> - 클라이언트가 접속 가능하게 서버 동작 기능 구현 - 스위치와 연결하여 스위치로 패킷 전송 가능하게 구현 - 클라이언트가 보낸 패킷을 그대로 스위치로 전달해주는 기능 구현 - 스위치가 응답 패킷을 보낼 경우 클라이언트로 전송해 주는 기능 구현

황민지	Switch 서버와 DB 서버를 비동기로 수정작업	<p>동기화로 구현되어있어 한번에 한가지 일을 못했던 서버들여러가지일을 동시에 할 수 있게 하기위해 비동기로 바꾸는 작업 진행</p> <ul style="list-style-type: none"> - Proactor 는비동기 작업을 지시하고 완료 이벤트를 받을 completion handler 를 등록한다. - 비동기 프로세스가 가능한 작업을 대기한다. 혹은 작업이 발생하면 깨어나 처리한다. - 가능한 작업들이 생기면 비동기 프로세스가 작업을 분리하여 비동기적으로 처리한다. - 작업이 완료되면 비동기 프로세스는 completion dispatcher 에게 완료 정보를 넘기고 dispatcher 는 정보를 이벤트로 만들어서 적절한 completion handler 에게 발송한다. - completion handler 는 받은 이벤트의 정보를 토대로 정해진 콜백을 호출하여 process event 를 처리한다.
김소라	검색 사이트 만들기 - Boogle 웹서버 구축	<ol style="list-style-type: none"> 1. 장고 프레임워크 사용해 웹서버 구축 2. url 연결과 검색결과 도출에 필요한 함수 생성
백승우	1. 웹 크롤러 제작	<ol style="list-style-type: none"> 1. 웹 크롤러 크롤 깊이 제한 기능을 완성하여 컴퓨터 스펙 한계를 넘지 않는 선에서 넓이 중심 크롤이 가능하도록 함. 이를 통해 검색된 데이터의 신뢰도가 다소 하락하지만 정보를 인터넷 상에서 넓게 크롤해 올 수 있도록 함.

2016.03.30

이름	개발과제	내용
----	------	----

안수연	웹 크롤러와 통신을 위한 패킷 구성	<p>웹 크롤러와 통신을 위한 패킷 구성</p> <ul style="list-style-type: none"> - B_C_REQ_WEB_URL = 401, PK_C_REQ_WEB_URL - B_C_NOTIFY_WEB_URL_ERROR = 402, PK_C_NOTIFY_WEB_URL_ERROR - B_C_NOTIFY_WEB_CONTENTS = 403, PK_C_NOTIFY_WEB_CONTENTS - B_C_NOTIFY_WEB_WORD = 404, PK_C_NOTIFY_WEB_WORD - B_C_NOTIFY_MOTHER_URL = 405, PK_C_NOTIFY_MOTHER_URL - B_C_REQ_MOTHER_URL_EXIST = 406, PK_C_REQ_MOTHER_URL_EXIST - B_C_NOTIFY_WEB_URL_INFO = 407, PK_C_NOTIFY_WEB_URL_INFO - B_S_ANS_WEB_URL = 408, PK_S_ANS_WEB_URL - B_S_ANS_MOTHER_URL_EXIST = 409, PK_S_ANS_MOTHER_URL_EXIST
황민지	비동기서버로 변환하는 것을 계속	Switch 서버는 게이트웨이와의 Accept, 패킷에 따라 크롤러 DB 서버와 UserDB 서버 둘 중 connect 할 대상을 정해서 해야하는데 이벤트 등록하는 과정에서 문제가 생겨 인터넷을 찾아보며 오류 수정을 하는것을 계속 하고 있다.
김소라	검색 사이트 만들기 - Boogle 웹페이지 디자인	<p>1. Google 웹페이지를 본따 Boogle 의 웹페이지 디자인 시작</p> <p>- javascript/html, css 로 템플릿 생성</p>
백승우	1. 웹 크롤러 제작	<p>1-1. 웹 크롤러의 통신 패킷을 제작함.</p> <p>1-2. 기존 사전 크롤러의 패킷과 그 함수들을 이용하여 범용 송수신 소켓 클래스를 완성시킴.</p>

2016.03.31

이름	개발과제	내용
----	------	----

안수연	웹 크롤러와 통신을 위한 함수 구축	<p>웹 크롤러 연동을 위한 함수 구축</p> <ul style="list-style-type: none"> - void webInsert(PK_C_NOTIFY_WEB_URL_INFO& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 웹 url 저장하는 함수 - void webReadUrl(PK_S_ANS_WEB_URL& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 방문하지 않은 url 을 찾아주는 함수 - void webContents(PK_C_NOTIFY_WEB_CONTENTS& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : url 의 본문 내용을 파일로 저장하는 함수 - void webWord(PK_C_NOTIFY_WEB_WORD& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 본문에서 추출한 키워드를 테이블로 생성 후 삽입하는 함수 - void webMother(PK_C_NOTIFY_MOTHER_URL& data, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : mother url 을 받아 저장하는 함수 - void webReadMother(PK_C_REQ_MOTHER_URL_EXIST& data, PK_S_ANS_MOTHER_URL_EXIST& sendData, SQLRETURN& ret, SQLHSTMT& hstmt, SQLHDBC& hDbc) : 전달받은 mother url 이 존재하는지 여부를 리턴해주는 함수
황민지	비동기서버로 Switch 서버 변환	<p>패킷에 따라 크롤러 DB 서버와 UserDB 서버 둘 중 connect 할 대상을 정해서 해야하는데 이벤트 등록하는 과정이 오류가 있었는데 패킷을 보내줄 때 마다 connect 를 하며 끊지 않고 계속</p>

		연결하는 방법으로 바꾸어 오류 해결
김소라	검색 사이트 만들기 - Boogle 웹페이지 디자인 완성	1. javascript, css 를 사용해 디자인 완성. - 임시 데이터 생성해 웹페이지의 검색 결과 도출 시도
백승우	1. 사전 크롤러 완료 2 차 2. 웹 크롤러 제작 완료 1 차 3. 웹 크롤러 점수 계산 기능 업그레이드	1. 사전 크롤러 수신 기능 구현. 2-1. 깊이 알고리즘 구현 성공. 2-2. 통신 소켓 클래스를 이용하여 통신 기능 구현. 3-1. url 다중 리스트 방식으로 부터 나름의 점수 계산법을 얻어내려 했으나, 무한 루프에 빠지게 됨. 3-2. 구글의 페이지 랭크 알고리즘에 관해 조사함.

2016.04.01

이름	개발과제	내용
안수연	웹 크롤러와 통신 테스트	웹 크롤러와 통신 테스트 - 완성된 웹 크롤러와 통신 테스트 - 각 패킷에 따라 정확한 기능이 작동되는지를 테스트 해 봄
황민지	DB 서버를 비동기서버로 변환	DB 서버를 비동기화 시킨 후 Switch 서버와 연동 사전크롤러, 게이트웨이서버, Switch 서버와 크롤러 DB 서버를 다 같이 연동시켜 DB 에 데이터 넣는 작업 시행 후 오류 수정
김소라	검색 사이트 만들기 - Boogle 웹페이지 연동	1. 웹서버와 DB 서버간의 연동을 위한 프로토콜 재설계
백승우	1. 웹 크롤러 점수 계산 기능 업그레이드	1-1. 나름의 점수 계산법을 개선하여 무한 루프를 벗어나려 시도했으나 실패함. 1-2. 구글의 페이지랭크 알고리즘에 관해 조사함.

2016.04.02

이름	개발과제	내용
----	------	----

안수연		
황민지		
김소라	검색 사이트 만들기 - 검색 모드 추가	1. 웹서버와 DB 서버간 패킷 설계, 2. 자막 검색 모드에 모든 검색모드의 결과물을 한데 모아 출력해주는 AllSearch 모드 추가
백승우		

2016.04.03

이름	개발과제	내용
안수연		
황민지		
김소라		
백승우		

2016.04.04

이름	개발과제	내용
안수연	전체 적인 서버 테스트	지금까지 구현한 서버들과 크롤러들을 연동해 테스트 -크롤러 Part - 자막 크롤러 - 사전 크롤러 - 웹 크롤러 - 서버 Part - 게이트웨이 - 스위치 - 크롤러 DB 서버 - upload 서버 테스트 후 오류 발견 - upload 서버에 패킷이 여러 개 올 경우 제대로 upload 가 되지 않음

황민지	2 차 발표 PPT와 발표대본 연습 Switch 서버 최적화 작업	2 차 발표 PPT 작업 후 발표 대본 작성 및 연습 Web 서버와 UserDB 서버연동을 위해 변경된 패킷 정의 B_S_ANS_COUNT = 503, //검색에 대한 결과물에 대한 개수를 보낼 때 사용하는 패킷 B_S_ANS_WEB_INFO = 507, //웹 링크, 제목에 대한 정보를 보낼 때 사용하는 패킷 B_S_ANS_WEB_CONTENTS = 508, //웹 본문 내용을 보낼 때 사용하는 패킷
김소라	검색 사이트 만들기 - 발표 준비	1. 웹서버와 DB 서버간 패킷 재설계, 2. 다음날 발표 준비 - 웹서버와 DB 서버간의 연동이 아직 준비 안되어 크롤링해 DB 서버에 넣어주는 기능만 시연하기로 함
백승우	1. 웹 크롤러 점수 계산 기능 업그레이드	1-1. 구글의 페이지랭크 알고리즘에 관해 조사함. 1-2. 조사 내용을 바탕으로 구글의 알고리즘과 같지는 않지만 같은 결론에 이르는 Markov Chain 을 활용한 확률 점수를 설계함. 하지만 Markov Chain 에서 사용되는 행렬에서 문제가 발생. - url 이 100 만개가 있다고 하면 Markov Chain 의 행렬 P 는 100 만 x100 만이 되는데 이 행렬을 파이썬 numpy 모듈로 계산하기위해서는 메모리 상에 20GB 를 저장 시켜야 한다는 문제.

2016.04.05

이름	개발과제	내용
안수연	발표 준비	발표 준비 - 발표를 위한 전체적인 서버 테스트 - 피피티 대본 검토 - 발표 후 미흡한 점 회의 후 보완 방향 회의
황민지	2 차 발표 및 발표 후 보완점	2 차 발표 후 보완할점 상의

	상의	
김소라	Boogle 프로젝트 2 차 발표	1. Boogle 프로젝트 2 차 발표 - 웹, 사전, 자막 크롤러가 크롤링한 데이터 DB 에 저장하는 기능 시연
백승우	1. 웹 크롤러 점수 계산 기능 업그레이드	1-1. 메모리 공간 부족 문제를 해결 방안 : 파이썬 모듈 PyTables 를 이용하여 100 만 x100 만 행렬을 HDD 에 txt 파일로 저장하고 1 행 씩 읽어서 계산을 처리하는 방안으로 문제 극복이 가능하다는 결론은 얻음 1-2. PyTables 의 해결 방안을 사용 하려면 컴퓨터에 큰 부하가 오기 때문에 크롤러가 수행되는 CPU 에서는 작업을 병행 할 수 없음. 또한 행렬의 제곱을 40 회 이상 반복해야 하기 때문에, 연산 속도의 문제로 이 점수 계산을 수행하지 않기로 함. 1-3. Marcov Chain 점수와 합쳐지게 될 Word Frequency(단어 빈도) 점수를 구현 함. 1-4. Word Frequency 점수를 표준화 하여 Marcov Chain 점수와 가중치의 차이가 없도록 만들고 두 점수를 합한 최종 형태를 얻어냄. $\text{Page_Score} = \text{F_SCORE} / (1 - \text{M_SCORE}^3)$ [Page_Score : 해당 웹페이지 점수, F_SCORE : Word Frequency Score, M_SCORE : Marcov Chain Score]

2016.04.06

이름	개발과제	내용
----	------	----

안수연	upload 서버 오류 해결 자막 크롤러와 연동	upload 서버 오류 해결 - 기존에 패킷이 여러 개 도착할 경우 upload 가 되지 않던 문제 해결 자막 크롤러와 연동 - 자막 크롤러로 부터 문장과 키워드를 받았을 때 제대로 동작하지 않던 문제 해결
황민지	빈도수를 이용하여 우선순위로 보내줄 데이터 결정	해당 Url 에 들어있는 단어 빈도수를 이용하여 높은 순으로 Web 서버에 보내주기위해 쿼리문 변경 검색한 단어수가 두가지 이상이면 해당 단어들이 가장 많이 들어가있는 순서대로 보내주기 위해 쿼리문 변경
김소라	검색 사이트 만들기 - 패킷 설계, 독자적 테스트	1. 웹서버와 DB 서버간 송수신할 패킷 설계 완성 - 독자적으로 연동 테스트 시도
백승우	1. 웹 크롤러 기능 최적화	1. 일부 웹페이지에 대해 본문 내용이 올바르게 파싱 되지 않는 문제를 해결 하기 위해, 일부 함수를 수정함.

4. 프로젝트 개발 내용

A. 프로젝트 배경 지식/기술/알고리즘

i. Server

일반적으로 서버 프로그램이 실행되고 있는 컴퓨터 하드웨어를 서버라고 부르며 다른 프로그램에게 서비스를 제공하는 컴퓨터 프로그램을 말하기도 한다.

서버는 프린터 제어나 파일 관리 등 네트워크 전체를 감시·제어하거나, 메인프레임이나 공중망을 통한 다른 네트워크와의 연결, 데이터·프로그램·파일 같은 소프트웨어 자원이나 모뎀·팩스·프린터 공유, 기타 장비 등 하드웨어 자원을 공유할 수 있도록 도와주는 역할을 한다.

서버는 사용자(클라이언트)의 요청에 의하여 서비스를 하는데 이와 같이 구성된 시스템을 클라이언트/서버 시스템이라고 하며, 이는 하나 이상의 응용 프로그램을 상호 협력적인 환경에서 운용하는 분산처리 형태를 의미한다.

즉, 서비스를 요청하는 클라이언트와 클라이언트의 요청을 처리하는 서버와의 협동작업을 통해서 사용자가 원하는 바람직한 결과를 얻는 처리방식이 클라이언트/서버 시스템이다.

클라이언트의 수가 5~20대 정도인 소규모 LAN의 경우에는 한 대의 서버로 충분히 모든 서비스를 소화할 수 있으나, 대규모 LAN의 경우에는 여러 대의 서버를 배치하고, 파일 관리는 파일 서버, 프린터 제어는 프린터 서버, 인터넷 등의 외부와의 교환은 커뮤니케이션 서버가 담당하는 등 각각 역할을 세분하게 된다.

인터넷에서 서버는 특수한 형태로 자신의 하드디스크에 담겨진 정보들을 외부에 공개해주는 컴퓨터를 이른다. 일반적으로 인터넷에서는 여러 정보들을 서버에서 관리하고, 일반 사용자들은 자신들의 컴퓨터(클라이언트)를 이용하여 서버에 접속하고 서버에서 제공하는 정보를 이용하게 된다.

ii. Crawler

웹크롤러·스파이더(spider)·로봇·웹수집기·로봇에이전트 등 다양한 이름으로 불린다. 인터넷에서의 정보검색시스템은 정보수집·정보가공·정보제공의 세 가지 기능으로 구성되는데 정보수집은 크롤러, 정보가공은 인덱서(indexer, 문서색인기) 그리고 정보제공은 사용자 인터페이스가 맡게 된다.

다시 말하면 구글·네이버·다음·야후 등 인터넷포털사이트에서 정보검색 서비스를 제공하기 위하여 운영하는 검색엔진은 수많은 웹페이지에서 다양한 정보를 수집해오는 크롤러(crawler)라는 프로그램과 크롤러가 수집한 정보를 검색하기 쉬운 형태로 가공한 후 색인을 만들어 서버에 보존하는 인덱서, 사용자가 특정 검색어를 입력하면 쿼리(query; 데이터를 조회하기 위한 명령)를 통해 인덱서 서버 안에 축적되어 있는 정보를 꺼내어 검색결과를 제공하는 시스템으로 구성되어 있다.

크롤러(crawler)란 사전적으로 '기어가는 사람' 또는 '포복동물'이라는 의미로, 방대한 수의 웹페이지(웹사이트)를 두루 돌아다니며 웹문서의 위치(URL)와 링크정보, 문서내용 등 각종 정보들을 수집해오는 기능으로 인하여 이런 이름이 붙었다. 검색엔진의 근간이 되는 크롤러가 수행하는 작업은 크롤링(crawling) 또는 스파이더링(spidering)이라고 부른다. 종류로는 앤츠(ants)·보츠(bots)·웜즈(worms)·웹스파이더(web spider)·웹로봇(web robot)·웹스쿠터(web scooter) 등이 있다.

크롤러는 사용자가 웹페이지의 각 링크를 일일이 따라가 정보를 얻는 작업을 대신하여 자동적으로 웹서버를 순회하며 웹페이지의 내용을 분석하고, 그 안에 포함되어 있는 URL들을 추출한 후 그 URL들로 하나씩 이동하면서 텍스트·수치·그림·멀티미디어 정보 등 수많은 정보를 수집하게 된다.

iii. Ace

The ADAPTIVE Communication Environment (ACE)는 동시 통신 소프트웨어의 많은 핵심 패턴을 구현하는 무료 제공 오픈 소스 객체 지향 (OO) 프레임 워크이다. ACE는 재사용 가능한 풍부한 C ++ wrapper facades 세트 및 OS 플랫폼의 범위에 걸쳐 일반적인 통신 소프트웨어 작업을 수행하는 프레임 워크 구성 요소를 제공한다.

또 ACE는 통신 소프트웨어 작업을 제공하는데, 이는 이벤트 다중화 및 이벤트 핸들러 디스패치, 신호 처리, 서비스 초기화, 상호 통신, 공유 메모리 관리, 메시지 라우팅, 분산 서비스 동적 (재)구성, 동시 실행과 동기화를 포함한다. ACE는 고성능의, 실시간 통신 서비스, 응용 프로그램의 개발자를 대상으로 한다. 그리고 이벤트 분리, 프로세스 간 통신, 명시적 동적 링크 및 동시성을 활용하는 OO 네트워크 응용 프로그램 및 서비스의 개발을 단순화한다.

또한, 런타임 애플리케이션의 동적 링크 서비스와 하나 또는 그 이상의 프로세스나 스레드 서비스를 실행함으로써 시스템 구성 및 재구성을 자동화한다. ACE는 오픈 소스 비즈니스 모델을 사용하여 여러 회사에서 시판, 지지된다

iv. Chilkat

Chilkat Inc.에서 개발한 소켓 라이브러리이다.

참고 사이트 : <https://www.chilkatsoft.com/>

v. Cloud computing

정보가 인터넷 상의 서버에 영구적으로 저장되고, 데스크톱·태블릿컴퓨터·노트북·넷북·스마트폰 등의 IT 기기 등과 같은 클라이언트에는 일시적으로 보관되는 컴퓨터 환경을 뜻한다. 즉 이용자의 모든 정보를 인터넷 상의 서버에 저장하고, 이 정보를 각종 IT 기기를 통하여 언제 어디서든 이용할 수 있다는 개념이다.

다시 말하면 구름(cloud)과 같이 무형의 형태로 존재하는 하드웨어·소프트웨어 등의 컴퓨팅 자원을 자신이 필요한 만큼 빌려 쓰고 이에 대한 사용요금을 지급하는 방식의 컴퓨팅 서비스로, 서로 다른 물리적인 위치에 존재하는 컴퓨팅 자원을 가상화 기술로 통합해 제공하는 기술을 말한다. 클라우드로 표현되는 인터넷상의 서버에서 데이터 저장, 처리, 네트워크, 콘텐츠 사용 등 IT 관련 서비스를 한번에 제공하는 혁신적인 컴퓨팅 기술인 클라우드 컴퓨팅은 '인터넷을 이용한 IT 자원의 주문형 아웃소싱 서비스'라고 정의되기도 한다.

클라우드 컴퓨팅을 도입하면 기업 또는 개인은 컴퓨터 시스템을 유지·보수·관리하기 위하여 들어가는 비용과 서버의 구매 및 설치 비용, 업데이트 비용, 소프트웨어 구매 비용 등 엄청난 비용과 시간·인력을 줄일 수 있고, 에너지 절감에도 기여할 수 있다.

또 PC에 자료를 보관할 경우 하드디스크 장애 등으로 인하여 자료가 손실될 수도 있지만 클라우드 컴퓨팅 환경에서는 외부 서버에 자료들이 저장되기 때문에 안전하게 자료를 보관할 수 있고, 저장 공간의 제약도 극복할 수 있으며, 언제 어디서든 자신이 작업한 문서 등을 열람·수정할 수 있다. 하지만 서버가 해킹당할 경우 개인정보가 유출될 수 있고, 서버 장애가 발생하면 자료 이용이 불가능하다는 단점도 있다.

구글·다음·네이버 등의 포털에서 구축한 클라우드 컴퓨팅 환경을 통하여 태블릿컴퓨터나 스마트폰 등 휴대용 IT 기기로도 손쉽게 각종 서비스를 사용할 수 있게 되었다. 이용편리성이 높고 산업적 파급효과가 커 차세대 인터넷 서비스로

주목받고 있는 클라우드 컴퓨팅은 2000 년 대 후반에 들어 새로운 IT 통합관리모델로 등장하였다.

vi. BeautifulSoup

BeautifulSoup는 HTML 및 XML 파일에서 data를 끌어내기 위한 파이썬 라이브러리이다. BeautifulSoup는 탐색, 검색, parse tree를 생성하는 관용적 방법 중 사용자가 원하는 parser로 작동한다.

vii. NLTK

NLTK는 Natural Language Toolkit의 약자이며 파이썬 모듈 패키지로 배포된다. NLTK는 파이썬 프로그래밍 언어를 위한 기호 및 통계 자연어 처리(Natural Language Processing)에 대한 라이브러리 및 프로그램의 모음이다. NLTK는 그래픽 데모와 샘플 데이터를 포함하며 툴킷의 자연어 처리 기본개념 설명서와 해설서를 지원한다.

NLTK는 연구, 교육 NLP 또는 이와 밀접한 분야인 경험 언어학, 인지 과학, 인공 지능, 정보 검색, 기계 학습을 지원하기 위해 사용된다.

viii. KoNLPy

KoNLPy는 NLP의 한국어 버전이다.

ix. Markov Chain

마르코프 체인(Markov chain)은 어떤 시스템의 미래 움직임을 예측하기 위하여 그 시스템의 현재 형태를 분석하는 절차를 말한다. 즉, 과거의 정보가 아닌 오직 현재의 정보에 입각하여 단기간의 미래와 장기간의 미래를 예측하는 방법이다.

B. 프로젝트 상세 개발 내용

i. 기능 정의

대분류	중간항목	상세항목	항목번호	설명	담당자	우선 순위
서버	Gateway 서버	Ace	Se-Gate-01	패킷을 받아 스위치로 전달	황민지	3
		Ace	Se- Gate -02	전송한 패킷의 타입을 검사하여 적합한 패킷인지 검사	안수연	3
	Switch 서버	Ace	Se-Swi-01	게이트 웨이로부터 데이터를 전송 받아 DB 서버로 전송	황민지	2
		Ace	Se- Swi -02	게이트 웨이로부터 받은 패킷의 타입을 검사하여 적합한 DB 서버로 전송	안수연	2
	Crawler 서버	Ace	Se-Cra-01	스위치로부터 받은 패킷의 타입에 따라 적합한 크롤러 DB 서비스 업무를 실행	안수연	1
	User 서버	Ace	Se-Use-01	스위치로부터 받은 패킷의 타입에 따라 적합한 유저 DB 검색 서비스 업무를 실행	황민지	1
	Upload 서버	Chilkat	Se-Clo-01	크롤러 패킷으로부터 upload 해야하는 파일의 위치와 이름을 전달받아 해당 파일을 클라우드에 업로드	안수연	4
	Download 서버	Chilkat	Se-Clo-02	크롤러 패킷으로부터 upload 해야하는 파일의 위치와 이름을 전달받아 해당 파일을 클라우드로 부터 다운로드	황민지	4
	웹 서버	Django Python	Se-Web-01	웹 UI 구성, 사용자의 데이터 입력에 대한 처리	김소라	4
		Django Python	Se-Web -02	Gateway 서버 – Web 서버 간 데이터 송수신	백승우	4

크롤러	Web 크롤러	Python	Cr-Web-01	The_Boogle의 이름으로 웹사이트를 돌며 웹페이지의 Html을 parsing	김소라	1
		Python	Cr-Web-02	프로토콜에 따라 Python, C++ 간 통신을 위한 다양한 자동화 기능을 가지는 클래스(C_Python_Socket)	백승우	2
		Algorithm	Cr-Web-03	깊이 우선 탐색 알고리즘을 활용하여 웹 검색 봇의 웹 서칭 루트 최적화	김소라	5
		Algorithm	Cr-Web-04	페이지 점수 알고리즘을 이용하여 페이지 별 검색 정확도 가중치 부여.	백승우	5
	Dictionary 크롤러	Python	Cr-Dic-01	검색어에 맞는 Wikipedia Url을 생성한 후 The_Boogle의 이름으로 Wikipedia 해당 페이지의 Html을 parsing	김소라	1
		Python	Cr-Dic-02	C_Python_Socket 클래스로 송수신이 불가능한 특수 형태의 사전 고유 형식 패킷들을 송수신	백승우	2
		Algorithm	Cr-Dic-03	약 65525 글자수를 넘는 장문의 패킷을 설정 길이만큼으로 분할 생성	김소라	2
	Subtitle 크롤러	Python	Cr-Sub-01	The_Boogle의 이름으로 Gom player 자막 자료실에 접속한 후, Html을 파싱하고 자막 게시물 url을 추출	백승우	1

		Python	Cr-Sub-02	크롤러가 추출한 자막 url을 DB로 송신 및 url을 수신 받아 data를 다운로드 한 후, data를 가공하여 DB로 송신	김소라	2
		Algorithm	Cr-Sub-03	깊이 우선 탐색 알고리즘으로 자막 게시판 페이지 url을 한 싸이클에 모두 parsing	백승우	2
	Morphological Analyzer	Nltk	Mo-nlt-01	영어 텍스트의 형태소를 분석	김소라	3
		Konlpy	Mo-kon-01	한글 텍스트의 형태소를 분석	백승우	3

ii. 주요 소스

가) 사전 크롤러

```
# coding: cp949
#사전
import urllib, urllib.parse, urllib.robotparser #웹 페이지를 파싱 하기 위해 사용하는 모듈
import traceback, time, re, sys, os, os.path, codecs, random, pprint #re : 정규표현식 모듈
#import socket
from bs4 import BeautifulSoup
import konlpy, nltk
import C_Python_Socket #자체 프로토콜을 지원하는 통신 클래스로 C, Python간 통신을 가능하게 하며
가변인자로 여러 데이터를 패킷으로 이어 붙일 수 있다.
#os.environ['PATH'] = os.path.join(os.environ['JAVA_HOME'], r'jre\bin\client')+ ';' + os.environ['PATH']

#크롤러 이름
Crawler = 'The_Boogle'
#위키백과(영), 위키백과(한)
dicUrlContainer = {'enwiki':'http://en.wikipedia.org/wiki/', 'kowiki':'http://ko.wikipedia.org/wiki/'}
savepath = 'C:\\Seungwu\\Finalproject\\20160310\\webcrawler\\webcrawler\\data\\'
tokenizer = None #NLTK 모듈에서 사용하는 토큰분석용 변수를 메모리에 할당
tagger = None #NLTK 모듈에서 사용하는 형태소 분석용 변수를 메모리에 할당

def CanFetch(url): #각 웹페이지에 게시된 Robots.txt 검사
    robot = urllib.robotparser.RobotFileParser(url+'robots.txt')
    return robot.can_fetch(Crawler, url)

def getUrlContent(url, delay=1): #페이지 html 소스 다운로드
    time.sleep(delay)
    print(url)
    try:
        opener = urllib.request.build_opener()
```

```

        opener.addheaders = [('User-agent', Crawler)] #웹페이지에 내 크롤러의 이름을 알려줌
        print("Downloading...")

        content = opener.open(urllib.request.quote(url.encode('utf-8'), '/:')).read() #type:bytes
        opener.close()

    except:

        traceback.print_exc()

        return None

    return content

def getBody(contents): # 정규 표현식을 이용해 불필요한 Text를 줄여 <body>부분 따오기
    soup = BeautifulSoup(contents, "lxml")
    content = soup('div', {'id':'mw-content-text'})
    content = content[0].encode()
    content = content.decode()
    content = content.encode('cp949', 'ignore')
    content = content.decode('cp949')
    content = re.sub('<(.*?)>', '', content, flags=re.DOTALL|re.IGNORECASE)
    content = re.sub('[\t\r\n]', ' ', content)
    content = re.sub('\n', ' ', content)
    content = re.sub(' {2}', ' ', content)
    content = re.sub('&', '&', content)
    content = re.sub('&lt;', '<', content)
    content = re.sub('&gt;', '>', content)
    content = re.sub('^ ', '', content)
    content = re.sub(' $', '', content)

    return content

def makeDictUrl(key): #사전 url 생성
    if isKor(key): wikiUrl = dicUrlContainer['kowiki']
    else: wikiUrl = dicUrlContainer['enwiki']

    url = wikiUrl+key

    return url

```

```
def makeDirNameFromUrl(Url): #url에서 불가능 특수문자 제거하고 디렉터리 이름 형태로 만들기
    delhttp = re.split('http://', Url)
    getUrl = re.split('/', delhttp[1])
    finUrl = re.sub('[./]', '_', getUrl[0])
    finUrl = finUrl.rstrip('_')
    return finUrl

def isKor(key): return not bool(re.search('[a-zA-Z]', key)) #단어의 한영을 구분하여 true(한글) false(영어) 값을 반환

def makeLinkFromDictUrl(body, key, url_from): #본문에서 얻어낸 단어로부터 url을 생성하여 저장
    kor_linkword = getKorNoun(body)
    eng_linkword = getEngNoun(body)
    for word in kor_linkword:
        socketCls.sendWORD(url_from, 'dict_'+word)
        url = makeDictUrl(word)
        socketCls.sendNextURL(url)
    for word in eng_linkword:
        socketCls.sendWORD(url_from, 'dict_'+word)
        url = makeDictUrl(word)
        socketCls.sendNextURL(url)

def dicParsing(url): #사전 파싱하기
    content = getUrlContent(url)
    if not content == None:
        content = content.decode('utf-8')
        content = content.encode('cp949', 'ignore')
        content = content.decode('cp949')
        sp = content.find('<title>')
        if sp > -1:
            ep = content[sp:].find('</title>')
            title = content[sp+7:sp+ep]
        else:
```

```
        title = ""
        body = getBody(content)
        socketCls.sendURL(title, url)
        key = url.split('/')[1]
        print('key : '+key)
        makeLinkFromDictUrl(body, key, url)
        socketCls.sendContents(url, body, 10000)
    else:
        return None

def getKorNoun(body): #태그없앤 content에서 한글 명사만 추출하기
    contents = re.sub('[^가-힣 \n]+', '', body)
    print("Get nouns from contents...")
    kkma = konlpy.tag.Kkma()
    noun = kkma.nouns(contents)
    noun = list(set(noun))
    return noun

def init_nltk(): #NLTK에 사용하는 변수들을 초기화함
    global tokenizer
    global tagger
    tokenizer = nltk.tokenize.RegexpTokenizer(r'[^\Ww+][^\WwWs]+')
    tagger = nltk.UnigramTagger(nltk.corpus.brown.tagged_sents())

def tag(text): #인자로 받은 text의 형태소를 분석하고 리스트 of 리스트로 반환
    global tokenizer
    global tagger
    print("Get nouns from contents...")
    if not tokenizer:
        init_nltk()
    tokenized = tokenizer.tokenize(text)
    tagged = tagger.tag(tokenized)
    return tagged
```

```
def getEngNoun(body):#태그없엔 content에서 영어 명사만 추출하기
    contents = tag(body)
    contents = list(set(contents))
    noun = []
    for factor in contents:
        if factor[1] == 'NN' and factor[0].__len__() > 1: #명사인 형태소만 얻어냄
            noun.append(factor[0])
    noun.sort()
    return noun

socketCls = C_Python_Socket.C_Python_Socket()
key = 'cat'
url = makeDictUrl(key) #url생성
socketCls.HOST = '192.168.0.185'
socketCls.PORT = 50000
while(True):
    try:
        dicParsing(url)
    except KeyboardInterrupt: #사용자 의지에 따른 정상 종료
        socketCls.s.close()
        break
    except:
        traceback.print_exc()
        pass
    url = socketCls.recvDictURL()
```

나) 웹 크롤러

```
#coding: cp949
#웹 크롤러 page spider
import urllib, urllib.parse, urllib.robotparser #웹 페이지의 내용을 parsing 하기위한 모듈을 임포트 함
import sys, os, os.path, re, traceback, time, pprint, copy, fileinput #copy 레퍼런스 기반 복사가 이루어지는
파이썬 기본 설정외에 레퍼런스가 완전히 독립되어 data를 넘겨 받고 싶을 때 copy.deepcopy 함수 사용
from bs4 import BeautifulSoup
#from bs4 import SoupStrainer
from C_Python_Socket import *
import nltk, konlpy

tokenizer = None
tagger = None
Crawler = 'The_Boogle'
crawlerRecvData = ''
max_depth = 4 # 크롤러가 해당 서버에서 최대 4 깊이까지 파고 들어감
urls = []
def getBody(contents):#<body>부분 따오기
    sp = contents.find('<body>')
    ep = contents.rfind('</body>')
    content = contents[sp+6:ep]
    content = re.sub('<style(.*)>(.*?)</style>', '', content, flags=re.IGNORECASE|re.DOTALL|re.MULTILINE)
    content = re.sub('<script(.*)>(.*?)</script>', '', content,
                    flags=re.MULTILINE|re.DOTALL|re.IGNORECASE)
    content = re.sub('<(.*?)>', '', content, flags=re.DOTALL|re.IGNORECASE)
    content = re.sub('[\WtWrWfWv]', '', content)
    content = re.sub('\n', ' ', content)
    content = re.sub('{2,}', ' ', content)
    content = re.sub(' \n', '\n', content)
    content = re.sub('&', '&', content)
    content = re.sub('<', '<', content)
```

```
content = re.sub('&gt;', '>', content)
content = re.sub('^ ', '', content)
content = re.sub(' $', '', content)
return content

def getKorNoun(body): #content에서 한글 명사 추출하기
    print("=> K-Morphological Analyzer Initializing & Noun Collecting...")
    kkma = konlpy.tag.Kkma()
    body = re.sub('[^가-힣]', '', body)
    posList = kkma.pos(body)
    nounlist = koNounfilter(posList)
    nounWithCount = countNoun(nounlist)
    #noun = kkma.nouns(body)
    #noun = list(set(noun))
    return nounWithCount

def getEngNoun(body): #태그없엔 content에서 영어 명사만 추출하기
    contents = tag(body)
    #contents = list(set(contents))
    #noun = []
    #for factor in contents:
    #    if factor[1] == 'NN' and factor[0].__len__() > 1:
    #        noun.append(factor[0])
    #noun.sort()
    nounlist = enNounfilter(contents)
    nounWithCount = countNoun(nounlist)
    return nounWithCount

def koNounfilter(poslist): #한글 단어의 명사만 추출
    filteredList = []
    for item in poslist:
        if item[1] == 'NNG' or item[1] == 'NNP' or item[1] == 'NNB' or item[1] == 'NNM':
            filteredList.append(item[0])
```

```
    return filteredList

def enNounfilter(taglist): #영어 단어의 명사만 추출
    filteredList = []
    for item in taglist:
        if item[1] == 'NN' and item[0].__len__() > 1: #단순 알파벳 제거
            filteredList.append(item[0])

    return filteredList

def countNoun(filteredList): #단어 총갯수
    score_set=[]
    for item in set(filteredList):
        unit = [item, ]
        cnt = 0
        for word in filteredList:
            if item == word:
                cnt += 1

        unit.append(cnt)
        score_set.append(unit)

    return score_set

def getWord(contents): #이 함수에서 본문에 대한 한/영 단어 추출 과정이 모두 이루어짐
    korContents = getKorNoun(contents)
    engContents = getEngNoun(contents)
    wholeWord = korContents
    wholeWord.extend(engContents)

    return wholeWord

def saveContent(url, soup):
    content = soup.decode('utf-8').encode('cp949', errors = 'ignore').decode('cp949')
    sp = content.find('<title>')
    if sp > -1:
        ep = content[sp:].find('</title>')
        title = content[sp+7:sp+ep]
```



```

        title = title.strip()
        title = re.sub('[W/:.*?"<>]', '_', title)
        title = re.sub('&', '&', title)
        title = re.sub('&lt;', '<', title)
        title = re.sub('&gt;', '>', title)
    else:
        title = 'NoTitled'

    body = getBody(content)
    #filename = re.sub('http://', '', url[0])
    #filename = re.sub('[WWW|/|:|W*|W?|'|<|>|W|=|W.]', '_', filename)
    socketCls.sendWebContents(url[0], title, body, unit=5000)
    wordList = getWord(body)
    for word in wordList:
        socketCls.generalSend(socketCls.PK_C_NOTIFY_WEB_WORD, (4, url[0]), (4, 'web_'+word[0]), (4,
word[1]), delay=0.3) # 가변인자를 활용한 소켓 송신
def checkMotherUrls(motherUrl):
    """인자(motherUrl)가 새로운 url이면 0 반환, 이미 mouther urls에 있으면 0 보다 큰 수 반환"""
    socketCls.generalSend(socketCls.PK_C_REQ_MOTHER_URL_EXIST, (4, motherUrl), autoDisconnect=0)
    rcvData = socketCls.generalRecv()
    boolData = int(treatRcvData(rcvData))
    return boolData
def addMotherUrls(motherUrl, check=True):
    """motherUrl 중복 체크 후, 삽입
    if check is True, call function, checkMotherUrls, first.
    if check is False, do not call function, checkMotherUrls."""
    if check is True:
        if checkMotherUrls(motherUrl) == 0 :
            socketCls.generalSend(socketCls.PK_C_NOTIFY_MOTHER_URL, (4, motherUrl))
        else: socketCls.generalSend(socketCls.PK_C_NOTIFY_MOTHER_URL, (4, motherUrl))
def analyData(startIndex, usingData):

```

"""usingData(패킷)의 startIndex 뒤로 명시된 길이 만큼의 내용(4자리)을 읽고,
다음으로 읽을 Index 위치 값과 읽은 Data를 튜플로 반환함"""

```
lengthData = ""
for oneChar in usingData[startIndex:]:
    if oneChar == '\0':
        break
    lengthData += oneChar
lengthData = int(lengthData)
if lengthData == 0: #url이 없을때 날아옴!
    print(">> [DB Error] No more URLs in DB : ", end="")
    print('\t>> Received Data : '+usingData)
    socketCls.s.close()
    exit()

prevIndex = startIndex+4
nextIndex = prevIndex+lengthData
return (nextIndex, usingData[prevIndex:nextIndex])

def treatRecvData(recvData): # 받은 패킷 분석
    cmdData = ""; lengthData = ""; realData = ""; prevIndex = 0
    realUrl = ""; realMotherUrl = ""; realDepth = ""
    print(">> recv data handling...")
    for oneChar in recvData[8:]: #cmd 받아오기
        if oneChar == '\0':
            break
        cmdData += oneChar
    cmdData = int(cmdData)
    #print('cmdData : '+str(cmdData))
    if cmdData == socketCls.PK_S_ANS_WEB_URL:
        prevIndex, realUrl = analyData(12, recvData)
        #prevIndex, realMotherUrl = analyData(prevIndex, recvData)
        ##prevIndex, realDepth = analyData(prevIndex, recvData)
```

```

        for oneChar in recvData[prevIndex:]:
            if oneChar == '\0':
                break

            realDepth += oneChar

        #print('real url : '+realUrl+'\n'+real mother url : '+realMotherUrl+'\n'+real depth :
'+realDepth)

        print('\treal url : '+realUrl+'\n'+\treal depth : '+realDepth)
        realDepth = int(realDepth)
        #realData = [realUrl, realMotherUrl, realDepth]
        realData = [realUrl, realDepth]
        return realData

    elif cmdData == socketCls.PK_S_ANS_MOTHER_URL_EXIST:
        boolData = recvData[13:14]
        #print('\tboolData : ' + boolData)
        boolData = int(boolData)
        return boolData

    else:
        print("=> wrong cmd! : "+str(cmdData))
        return None

def normalUrlspop(urls): #해당 urls가 정상적으로 크롤이 완료되었으며 메모리에서 삭제함
    socketCls.generalSend(socketCls.PK_C_REQ_WEB_URL, autoDisconnect=0)
    crawlerRecvData = socketCls.generalRecv()
    recvedUrl = treatRecvData(crawlerRecvData)
    urls.append(recvedUrl)
    urls.pop(0)
    print("=> old url popped!")

def errorUrlspop(urls): #해당 urls는 크롤러 에러 또는 홈페이지 에러로 인해 크롤이 불가능하므로
메모리에서 삭제함
    socketCls.generalSend(socketCls.PK_C_NOTIFY_WEB_URL_ERROR, (4, urls[0][0]))
    socketCls.generalSend(socketCls.PK_C_REQ_WEB_URL, autoDisconnect=0)

```

```
crawlerRecvData = socketCls.generalRecv()
recvdedUrl = treatRecvData(crawlerRecvData)
urls.append(recvdedUrl)
urls.pop(0)
print("=>> old url popped!")

def updateDepth(childUrl, depth): # 해당 웹페이지(childUrl)의 깊이를 계산함
    childUrl2 = getMotherUrl(childUrl)
    if checkMotherUrls(childUrl2):
        childDepth = copy.deepcopy(depth)
        childDepth += 1
        return childDepth
    else:
        childDepth = copy.deepcopy(depth)
        childDepth = 0
        return childDepth

def childUrlFilter(childUrl): #본문의 불필요한 텍스트를 제거
    if childUrl.startswith('javascript') or childUrl.startswith(':') or childUrl.endswith('.js') or
    childUrl.endswith('.css'):
        return None
    return childUrl

def completeChildUrl(childUrl, motherUrl): #HTML 문법을 소화하기위한 함수
    if childUrl.startswith('//'):
        childUrl = 'http:' + childUrl
    elif childUrl.startswith('/'):
        childUrl = 'http://' + motherUrl + childUrl
    elif childUrl.startswith('#'):
        childUrl = 'http://' + motherUrl + '/' + childUrl
    return childUrl

def getTags(soup, preMotherUrl, depth): #url에서 링크들을 추출하기
    motherUrl = getMotherUrl(preMotherUrl)
```

```
localVisited = [preMotherUrl]
urlList = []
for tag in soup.find_all(src=True):
    try:
        childUrl = trim(tag['src']) #src 추출
        childUrl = completeChildUrl(childUrl, motherUrl)
        if childUrl in localVisited:
            continue
        else :
            childUrl = childUrlFilter(childUrl)
            if childUrl is None: #Invalid Url은 childUrlFilter에서 None으로 출력
                continue
            if not bool(re.match("http://", childUrl)):
                continue
            else:
                localVisited.append(childUrl)
                childDepth = updateDepth(childUrl, depth)
                socketCls.generalSend(socketCls.PK_C_NOTIFY_WEB_URL_INFO,
                                     (4, childUrl), depth=str(childDepth))

    except:
        traceback.print_exc()
        continue
for tag in soup.find_all(href=True):
    try:
        childUrl = trim(tag['href']) #href 추출
        childUrl = completeChildUrl(childUrl, motherUrl)
        if childUrl in localVisited:
            continue
        else:
```

```
        childUrl = childUrlFilter(childUrl)
        if childUrl is None: #Invalid Url은 childUrlFilter에서 None으로 출력
            continue
        if not bool(re.match('http://', childUrl)):
            continue
        else:
            localVisited.append(childUrl)
            childDepth = updateDepth(childUrl, depth)
            socketCls.generalSend(socketCls.PK_C_NOTIFY_WEB_URL_INFO,
                                (4, childUrl), depth=str(childDepth))
    except:
        traceback.print_exc()
        continue

    return urlList

def getContent(urls): # 웹 페이지의 본문을 크롤하는 함수. 크롤러의 코어 기능
    time.sleep(1)
    print('=>> Crawling : '+urls[0][0])
    if int(urls[0][1]) < max_depth: # 크롤 중단 한계선
        if not bool(re.match('http://', urls[0][0])):
            print("=>> [Crawling Canceled] Invalid URL")
            errorUrlspop(urls)
            return None
        else:
            try:
                opener = urllib.request.build_opener(urllib.request.HTTPCookieProcessor)
                #CookieProcessor를 이용해야 특정 url 접속했을때 서버가 다른 url
                주소를 넘겨주고 그곳으로 이동해야 하는 경우, Cookie data를 제공해줌
                htmlcontent = opener.open(urls[0][0]).read()
                opener.close()
                soup = BeautifulSoup(htmlcontent, "lxml")
```

```

        saveContent(urls[0], soup)
        motherUrl = getMotherUrl(urls[0][0])
        addMotherUrls(motherUrl)
        #print("#mother_url : " + motherUrl)
        if soup is not None: urlList = getTags(soup, urls[0][0], int(urls[0][1]))
        else: print('soup is None!'); return None
        #urls.extend(urlList)
        normalUrlspop(urls)
    except SystemExit:
        exit()
    except:
        traceback.print_exc()
        print(">>> [Crawling Canceled] Error Occured")
        errorUrlspop(urls)
        return None
    return True

else:
    print(">>> [Crawling Canceled] Out of Depth : " + str(max_depth))
    errorUrlspop(urls)

socketCls = C_Python_Socket()
socketCls.HOST = '192.168.0.185'
socketCls.PORT = 10000
##### 웹 크롤러가 맨 처음 시작 될 때 사용 #####
socketCls.generalSend(socketCls.PK_C_NOTIFY_WEB_URL_INFO, (4,
"http://blog.daum.net/simhsook48?t_nil_recomm=vipimg"), depth='0')
#socketCls.generalSend(socketCls.PK_C_NOTIFY_WEB_URL_INFO, (4,
"http://cfile232.uf.daum.net/T225x225/227F183956FDC537339A988"), depth='0')
socketCls.generalSend(socketCls.PK_C_REQ_WEB_URL, autoDisconnect=0)
crawlerRecvData = socketCls.generalRecv()

```

```
temp_recvdUrl = treatRecvData(crawlerRecvData)
```

```
#####
```

```
urls.append(temp_recvdUrl)
```

```
while urls.__len__() > 0:
```

```
    getContent(urls)
```

```
    print(urls.__len__())
```


다) C_Python_Socket 클래스

```

#coding: cp949
import socket, re, time, os
class C_Python_Socket:
    """C와 Python의 소켓 통신을 위한 클래스.
    내부에 소켓 s를 만드므로 s를 이용해 send 및 recv를 한다.
    socket, re, time이 C_Python_Socket '모듈'에 임포트 되어 있다.
    (C_Python_Socket '클래스'와 다름에 주의)"""
    s = socket.socket()
    PK_C_NOTIFY_DIC_URL_INFO = 301; PK_C_NOTIFY_DIC_WORD = 302; PK_C_NOTIFY_DIC_CONTENTS =
303
    PK_C_NOTIFY_DIC_URL_PATH = 304; PK_C_REQ_DIC_URL = 305; PK_S_ANS_DIC_URL = 306

    PK_C_REQ_WEB_URL = 401; PK_C_NOTIFY_WEB_URL_ERROR = 402; PK_C_NOTIFY_WEB_CONTENTS =
403
    PK_C_NOTIFY_WEB_WORD = 404; PK_C_NOTIFY_MOTHER_URL = 405
    PK_C_REQ_MOTHER_URL_EXIST = 406; PK_C_NOTIFY_WEB_URL_INFO = 407
    PK_S_ANS_WEB_URL = 408; PK_S_ANS_MOTHER_URL_EXIST = 409
    data = ""; sendData = ""; recvData = ""; sendConData = ""; dataType = "
    sendConLen = 0; titLen = 0; urlLen = 0; dataLen = 0; fileNameLen = 0
    generalData = ""; generalSize = 0; generalLen = 0
    HOST = ""; PORT = 0
    def __init__(self): # 미리 초기화 된 상태로 할당이 필요한 변수이거나, 또 하나의 객체에서 같은
                        변수를 중복해서 사용하는 경우 등이 발생 할 수 있으므로 매 객체 함수의
                        시작에 앞서 __init__()을 호출해 줘야 함
        self.data = ""; self.sendData = ""; self.recvData = ""; self.sendConData = ""; self.dataType = "
        self.sendConLen = 0; self.titLen = 0; self.urlLen = 0; self.dataLen = 0; self.fileNameLen = 0
        self.generalData = ""; self.generalSize = 0; self.generalLen = 0
    def connect(self, ip, port): #미리 세팅된 ip, port로 통신 스트림을 여는 함수
        """이 함수는 자동으로 소켓 s를 불러와 해당 인자로 connect 한다.
        사용하기 전 주의 사항
        ip는 문자열로, port는 int로 입력해주세요"""
        #global s
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

```

```

self.HOST = ip                # The remote host
self.PORT = port              # The same port as used by the server
self.s.connect((self.HOST, self.PORT))
def FillSpacePacket(self, dataLen, index):
    """패킷의 자릿수를 채워야 할때 공백(' ')으로 빈공간을 채워줌
dataLen : 분석을 원하는 변수의 길이
index : space로 채울 인덱스 최대값
ex)

    data = 'abcd'
    data += FillSpacePacket(data.__len__(), 5)
    print(data)
    결과 : 'abcd  '
    빈칸 두개 생성"""
    index += 1
    space = "
    if dataLen < index:
        for count in range(0, index-dataLen):
            space += ' '

    return space
def isASCII(self, text): # C언어는 아스키 코드 외의 문자의 길이를 2로 계산하는 반면, 파이썬 3+
버전에서는 유니코드를 사용하여 모든 문자의 길이를 1로 계산한다. 따라서 패킷에 보낼 문자들의 길이 값
계산에서 이 함수가 필요함
    """ASCII문자인지 판별. text에 ASCII가 아닌 문자가 한개라도 있으면 FALSE, 없으면
TRUE"""

    return not bool(re.search('[^\x00-\x7F]', text))
def __Len_Cstyle__(self, text):
    """text를 C 스타일 길이로 구함"""
    CLen = 0
    for i in text:
        if self.isASCII(i):
            CLen += 1
        else: CLen += 2
    return CLen
def sendURL(self, title, url, delay=0.1): #사전 크롤러 url 전송
    #global s#data = "#sendData = "

```

```

time.sleep(delay)
self.connect(self.HOST, self.PORT)
self.__init__()
self.titLen = self.__Len_Cstyle__(title)
self.urlLen = self.__Len_Cstyle__(url)
self.data += str(self.PK_C_NOTIFY_DIC_URL_INFO)
self.data += 'W0'
self.data += self.FillSpacePacket(self.data.__len__(), 3)
self.data += str(self.titLen)
self.data += 'W0'
self.data += self.FillSpacePacket(self.data.__len__(), 7)
self.data += title
self.data += str(self.urlLen)
self.data += 'W0'
self.data += self.FillSpacePacket(self.data.__len__(), 7+title.__len__()+4)
self.data += url
self.dataLen = self.__Len_Cstyle__(self.data)+8
self.dataLen = str(self.dataLen)
self.dataLen += 'W0'
self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
self.sendData += self.dataLen
self.sendData += self.data
self.sendData = self.sendData.encode('cp949')
print(self.sendData.decode('cp949'))
self.s.send(self.sendData)
self.s.close()

def sendNextURL(self, url, delay=0.1): #사전 크롤러에서 단어 추출후 새로 url을 생성하여 전송
    #global s#data = ""#sendData = ""
    time.sleep(delay)
    self.connect(self.HOST, self.PORT)
    self.__init__()
    self.urlLen = self.__Len_Cstyle__(url)
    self.data += str(self.PK_C_NOTIFY_DIC_URL_PATH)
    self.data += 'W0'
    self.data += self.FillSpacePacket(self.data.__len__(), 3)

```

```

self.data += str(self.urlLen)
self.data += '\0'
self.data += self.FillSpacePacket(self.data.__len__(), 7)
self.data += url
self.dataLen = self.__Len_Cstyle__(self.data)+8
self.dataLen = str(self.dataLen)
self.dataLen += '\0'
self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
self.sendData += self.dataLen
self.sendData += self.data
self.sendData = self.sendData.encode('cp949')
print(self.sendData.decode('cp949'))
self.s.send(self.sendData)
self.s.close()

def sendWORD(self, url, word, delay=0.1): #사전 크롤러 단어 전송
    #global s#data = "#sendData = "
    time.sleep(delay)
    self.connect(self.HOST, self.PORT)
    self.__init__()
    self.urlLen = self.__Len_Cstyle__(url)
    self.wordLen = self.__Len_Cstyle__(word)
    self.data += str(self.PK_C_NOTIFY_DIC_WORD)
    self.data += '\0'
    self.data += self.FillSpacePacket(self.data.__len__(), 3)
    self.data += str(self.urlLen)
    self.data += '\0'
    self.data += self.FillSpacePacket(self.data.__len__(), 7)
    self.data += url
    self.data += str(self.wordLen)
    self.data += '\0'
    self.data += self.FillSpacePacket(self.data.__len__(), 7+self.url.__len__()+4)
    self.data += word
    self.dataLen = self.__Len_Cstyle__(self.data)+8
    self.dataLen = str(self.dataLen)
    self.dataLen += '\0'

```

```

self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
self.sendData += self.dataLen
self.sendData += self.data
self.sendData = self.sendData.encode('cp949')
print(self.sendData.decode('cp949'))
self.s.send(self.sendData)
self.s.close()

```

```
def sendContents(self, url, contents, unit, delayOverUnit=0.1, delayUnderUnit=0.1):
```

#사전 크롤러 내용 전송. ACE 라이브러리를 이용한 데이터 송신 테스트 결과, 한 패킷은 최대 65525 길이를 가질 수 있는 것으로 판단된다.

#하지만 사전 내용은 이보다 긴 경우가 많이 때문에, 여러 패킷으로 나누어 보낼 수 있는 기능이 필요하다.

"""주의 사항

unit이 약 32500 이상이 되면,

파이썬은 한 패킷에 약 32500개 글자를 담아 전송하게 된다.

이때, 글자가 한글이라면 실제 C에서 받아들이는 글자수는 65000개로 늘어나게 되고,

이 수치가 변수 허용 최고치를 넘으면 안된다!!

그러므로 unit을 20000 이하로 사용할 것을 권장한다.

unit : 한번에 보내는 글자수 조절."""

```

#global s#data = "#sendData = "#print(contents.encode('utf-8'))
self.__init__()
self.urlLen = self.__Len_Cstyle__(url)
contentsCLen = self.__Len_Cstyle__(contents)
contentsPLen = contents.__len__()
if contentsCLen > unit:
    count = 1
    num = contentsPLen//unit + 1
    while(count <= num):
        time.sleep(delayOverUnit)
        self.connect(self.HOST, self.PORT)
        self.data = ""
        self.sendData = ""
        self.sendConData = contents[unit*(count-1):unit*count]
        self.sendConLen = self.__Len_Cstyle__(self.sendConData)
        self.data += str(self.PK_C_NOTIFY_DIC_CONTENTS)
        self.data += 'W0'

```

```

self.data += self.FillSpacePacket(self.data.__len__(), 3)
self.data += str(self.urlLen)
self.data += '\x00'
self.data += self.FillSpacePacket(self.data.__len__(), 7)
self.data += url
self.data += str(self.sendConLen)
self.data += '\x00'
self.data += self.FillSpacePacket(self.data.__len__(), 7+url.__len__()+8)
self.data += self.sendConData
self.dataLen = self.__Len_Cstyle__(self.data)+8
self.dataLen = str(self.dataLen)
self.dataLen += '\x00'
self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
self.sendData += self.dataLen
self.sendData += self.data
self.sendData = self.sendData.encode('cp949')
#print(self.sendData.decode('cp949'))
self.s.send(self.sendData)
count += 1
self.s.close()

```

else:

```

time.sleep(delayUnderUnit)
self.connect(self.HOST, self.PORT)
self.data += str(self.PK_C_NOTIFY_DIC_CONTENTS)
self.data += '\x00'
self.data += self.FillSpacePacket(self.data.__len__(), 3)
self.data += str(self.urlLen)
self.data += '\x00'
self.data += self.FillSpacePacket(self.data.__len__(), 7)
self.data += url
self.data += str(contentsCLen)
self.data += '\x00'
self.data += self.FillSpacePacket(self.data.__len__(), 7+url.__len__()+8)
self.data += contents
self.dataLen = self.__Len_Cstyle__(self.data)+8

```

```

        self.dataLen = str(self.dataLen)
        self.dataLen += '\0'
        self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
        self.sendData += self.dataLen
        self.sendData += self.data
        self.sendData = self.sendData.encode('cp949')
        #print(self.sendData.decode('cp949'))
        self.s.send(self.sendData)
        self.s.close()

def recvDictURL(self): #받을때 keyword, url 두개를 반환하도록 설정!!
    self.connect(self.HOST, self.PORT)
    #data = "#sendData = "#recvData = "#dataType = "#urlLen = "
    url = ""
    self.__init__()
    self.urlLen = ""
    self.data += str(self.PK_C_REQ_DIC_URL)
    self.data += '\0'
    self.dataLen = self.__Len_Cstyle__(self.data)+8
    self.dataLen = str(self.dataLen)
    self.dataLen += '\0'
    self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
    self.sendData += self.dataLen
    self.sendData += self.data
    self.sendData = self.sendData.encode('cp949')
    self.s.send(self.sendData)
    self.recvData = self.s.recv(1024)
    print('=> received : ' + self.recvData.decode('cp949'))
    self.recvData = self.recvData.decode('cp949')
    for i in self.recvData[8:]:
        if i is not '\0':
            self.dataType += i
        else: break
    #print('Data type : ' + self.dataType)
    if self.dataType == str(self.PK_S_ANS_DIC_URL):
        for i in self.recvData[12:]:

```

```

        if i is not '\0':
            self.urlLen += i
        else: break
    #print(type(self.urlLen), 'Url len : ' + str(self.urlLen))
    self.urlLen = int(self.urlLen)
    url = self.recvData[16:16+self.urlLen]
else: print('=> Wrong data received!\n'+self.recvData)
self.s.close()
return url

def autoConnect(self): #연결이 끊어져 있을 경우 자동으로 재연결
    try:
        self.s.getsockname() # 소켓 s가 존재하는지 검사.
    except IOError: # 소켓이 아닌 개체를.. 어쩌구 하면 자동으로 다시 연결!
        self.connect(self.HOST, self.PORT)

def generalSend(self, cmd, *tupledata, depth='', delay=0.1, autoDisconnect=1):
    #C_Python_Socket 클래스의 코어 함수. *tupledata는 가변인자로 하나의 인자는 다음과 같이 구성된다.
    #(보낼 data의 길이를 표현하는 헤더의 크기, 보낼 data)
    """*tupledata는 (실제 data의 길이를 표현할 부분이 차지하는 크기, 실제 data)로 구성됨.
    autoDisconnect != 0 이면,
    (1)객체 소켓 s에 저장된 HOST, PORT로 연결
    (2)작업 처리
    (3)연결 해제
    autoDisconnect == 0 이면,
    (1)객체 소켓 s에 저장된 HOST, PORT로 연결
    (2)작업 처리
    """

    time.sleep(delay)
    self.autoConnect()
    self.__init__()
    self.data += str(cmd)
    self.data += '\0'
    self.data += self.FillSpacePacket(self.data.__len__(), 3)
    for tupleItem in tupledata:
        self.generalSize, self.generalData = tupleItem
        self.generalData = str(self.generalData)

```



```

        #print('size : '+str(self.generalSize)+' '+'data : '+ self.generalData)
        tempdata = ""
        self.generalLen = self.__Len_Cstyle__(self.generalData)
        tempdata += str(self.generalLen)
        tempdata += '\x00'
        tempdata += self.FillSpacePacket(tempdata.__len__(), self.generalSize-1)
        self.data += tempdata
        tempdata = ""
        tempdata += self.generalData
        self.data += tempdata
        self.data += depth

    self.dataLen = self.__Len_Cstyle__(self.data)+8
    self.dataLen = str(self.dataLen)
    self.dataLen += '\x00'
    self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
    self.sendData += self.dataLen
    self.sendData += self.data
    self.sendData = self.sendData.encode('cp949')
    print(">>> send : "+self.sendData.decode('cp949'))
    self.s.send(self.sendData)
    if not autoDisconnect == 0:
        self.s.close()

    def generalRcv(self, autoDisconnect=1):
        """rcv 상태에서 대기. data를 받으면 str로 반환.
        autoDisconnect != 0이면 rcv완료 후 자동 연결 종료.
        autoDisconnect == 0이면 rcv완료 후에도 연결 유지"""
        self.__init__()
        print(">>> rcv waiting...")
        self.rcvData = self.s.recv(1024)
        if not autoDisconnect == 0:
            self.s.close()

        self.rcvData = self.rcvData.decode('cp949')
        print(">>> rcv : "+self.rcvData)
        return self.rcvData

    def makePacketPart(self, strData, strDataSize):

```

```

        """strData의 type는 str, strDataSize의 type는 int.
        strData를 받아서 부분 패킷을 생성. 이때, strDataSize를 인덱스로 하여
        인덱스(strDataSize)까지 여유 공간을 공백으로 채움"""
        tempdata = strData
        tempdata += 'W0'
        tempdata += self.FillSpacePacket(tempdata.__len__(), strDataSize)
        return tempdata
    def sendWebContents(self, url, title, contents, unit, delayOverUnit=0.1, delayUnderUnit=0.1):
        """unit : 한번에 보내는 글자수 조절.

```

주의 사항

unit이 약 32500이면,

파이썬은 한 패킷에 약 32500개 글자를 담아 전송하게 된다.

이때, 글자가 한글이라면 실제 C에서 받아들이는 글자수는 65000개로 늘어나게 되고,

이 수치가 변수 허용 최고치를 넘으면 안된다!!

그러므로 unit을 20000 이하로 사용할 것을 권장한다.

"""

```

        #global s#data = "#sendData = "#print(contents.encode('utf-8'))
        temptxtW =
open('C:WWSeungwuWWFinalprojectWW20160310WWwebcrawlerWWwebcrawlerWWtempWWsend_data.txt', 'w')
        self.__init__()
        self.urlLen = self.__Len_Cstyle__(url)
        self.titLen = self.__Len_Cstyle__(title)
        #self.fileNameLen = self.__Len_Cstyle__(filename)
        contentsCLen = self.__Len_Cstyle__(contents)
        contentsPLen = contents.__len__()
        if contentsCLen > unit:
            count = 1
            num = contentsPLen//unit + 1
            while(count <= num):
                time.sleep(delayOverUnit)
                self.autoConnect()
                self.data = ""
                self.sendData = ""
                self.sendConData = contents[unit*(count-1):unit*count]
                self.sendConLen = self.__Len_Cstyle__(self.sendConData)

```

```

        tempdata =
            self.makePacketPart(str(self.PK_C_NOTIFY_WEB_CONTENTS), 3)
        self.data += tempdata
        tempdata = self.makePacketPart(str(self.urlLen), 3)
        self.data += tempdata
        self.data += url
        tempdata = self.makePacketPart(str(self.titLen), 3)
        self.data += tempdata
        self.data += title
        #tempdata = self.makePacketPart(str(self.fileNameLen), 3)
        #self.data += tempdata
        #self.data += filename
        tempdata = self.makePacketPart(str(self.sendConLen), 7)
        self.data += tempdata
        self.data += self.sendConData
        self.dataLen = self.__Len_Cstyle__(self.data)+8
        self.dataLen = str(self.dataLen)
        self.dataLen += 'W0'
        self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
        self.sendData += self.dataLen
        self.sendData += self.data
        self.sendData = self.sendData.encode('cp949')
        #print(self.sendData.decode('cp949'))
        #print(str(self.PK_C_NOTIFY_WEB_CONTENTS)+url+title)
        self.s.send(self.sendData)
        print("=>> send : "+self.sendData.decode('cp949'))
        count += 1
        self.s.close()
        temptxtA =
open('C:WWSeungwuWWFinalprojectWW20160310WWwebcrawlerWWwebcrawlerWWtempWWsend_data.txt', 'a')
        temptxtA.write(self.sendData.decode('cp949'))
        temptxtA.close()

    else:
        time.sleep(delayUnderUnit)
        self.autoConnect()

```

```

self.data = ""
self.sendData = ""
tempdata = self.makePacketPart(str(self.PK_C_NOTIFY_WEB_CONTENTS), 3)
self.data += tempdata
tempdata = self.makePacketPart(str(self.urlLen), 3)
self.data += tempdata
self.data += url
tempdata = self.makePacketPart(str(self.titLen), 3)
self.data += tempdata
self.data += title
tempdata = self.makePacketPart(str(contentsCLen), 7)
self.data += tempdata
self.data += contents
self.dataLen = self.__Len_Cstyle__(self.data)+8
self.dataLen = str(self.dataLen)
self.dataLen += '\0'
self.dataLen += self.FillSpacePacket(self.dataLen.__len__(), 7)
self.sendData += self.dataLen
self.sendData += self.data
self.sendData = self.sendData.encode('cp949')
#print(self.sendData.decode('cp949'))
#print(str(self.PK_C_NOTIFY_WEB_CONTENTS)+str(url)+str(title))
self.s.send(self.sendData)
print(">> send : "+self.sendData.decode('cp949'))
self.s.close()
temptxtA =
open('C:\WWSeungwu\WFinalproject\WW20160310\WWwebcrawler\WWwebcrawler\WWtemp\WWsend_data.txt', 'a')
    temptxtA.write(self.sendData.decode('cp949'))
    temptxtA.close()
temptxtW.close()
#testCls = C_Python_Socket()
#tuple1 = (4, 'sdfd')
#tuple2 = (4, 'abcde44')
#testCls.HOST='192.168.0.185'
#testCls.PORT=50000

```

```
#testCls.generalSend(C_Python_Socket.PK_C_NOTIFY_WEB_URL_INFO, tuple1, tuple2, delay=0.01)  
#testCls.generalSend(C_Python_Socket.PK_C_NOTIFY_WEB_URL_INFO)
```

라) 자막 크롤러

```
#-*- coding: cp949 -*-
from bs4 import BeautifulSoup
from urllib.parse import quote
from urllib import robotparser
import urllib
from urllib import *
import urllib.robotparser
import time, traceback, re, os, os.path
import sqlite3
import cgi
import sys
from SubDB import SubDB
import SubEditor
import socket

#import SubDB
#from SubDB import HOST, PORT, s
#####

# SOCKET NETWORKING
HOST = '192.168.0.185'          # The remote host
#HOST = '127.0.0.1'
PORT = 10000          # The same port as used by the server
s = socket.socket()

sType = dict(B_C_NOTIFY_SUBTITLE = '201', B_C_NOTIFY_SUBURL = '202', B_C_REQ_SUBURL = '203' ,
B_S_ANS_SUBURL = '204', B_C_REQ_WORD = '509', B_S_ANS_SUBTITLE = '504', B_S_ANS_COUNT = '503') # 패킷
타입

crawler_name = 'smigle_crawler'
```

```
gom_mainPage = 'http://gomtv.com/'
gom_gomMainPage = 'http://gom.gomtv.com'
gom_mainBoardPage =
'http://gom.gomtv.com/main/index.html?ch=subtitles&pt=l&menu=subtitles&lang=3&page=1&md5key='
gom_searchMainBoard = '/main/index.html?ch=subtitles&pt=l&menu=subtitles&lang=3&sWord=&page='
gom_boardURL = 'http://gom.gomtv.com/main/index.html/'
gom_DownCHPT_URL = 'ch=subtitles&pt=down&'

smiDir = './smi/'
temp_startPage = 1
rp = robotparser.RobotFileParser(gom_mainPage + 'robots.txt')
rp.read()
```

```
def canFetch(self, url):
```

```
    return rp.can_fetch(crawler_name, url)
```

```
def getGomLastBoard(url):
```

```
    GomBoard = urllib.request.urlopen(url).read()
```

```
    #unicode(GomBoard, 'utf-8').decode('utf-8').encode('utf-8')
```

```
    soup = BeautifulSoup(GomBoard, "html.parser")
```

```
    lastBoard = soup('a', {'class':'next_last'})
```

```
    temp = str(lastBoard[0])
```

```
    lastBoardNumTemp = temp.split(' ')[2]
```

```
    lastBoardNum = int(lastBoardNumTemp.rsplit('=', 1)[-1].rstrip(""))
```

```
    return lastBoardNum
```

```
def getGomAllBoardPageURL(lastpage):
    pageURLList = []
    for page in range(temp_startPage, lastpage+1):
        pageURL = gom_gomMainPage + gom_searchMainBoard + str(page)
        pageURLList.append(pageURL)

    return pageURLList

def getGomTitleLink(url):
    print ("getGomTitleLink Start!")
    #url =
    ['http://gom.gomtv.com/main/index.html?ch=subtitles&pt=1&menu=subtitles&lang=3&page=1&md5key=']
    for u in range(0, len(url)):
        time.sleep(1)
        gomPageTemp = urllib.request.urlopen(url[u]).read()
        #gomPageTemp = urllib2.urlopen(url[u]).read()
        gomPage= str(gomPageTemp).replace("$", "")
        soup = BeautifulSoup(gomPage, "html.parser")
        rTitles = re.compile('.+prepage.+')
        titlesTemp = soup('a', {'href':rTitles})
        titlesList = []
        sTitleURLListTemp = []
        sTitleURLList = []

    for i in range(0, len(titlesTemp)):
        titles = re.sub('&','&', str(titlesTemp[i]))
        titlesList.append(titles)
        titlesList[i] = titlesList[i].split(' ')[1]
        title = titlesList[i].strip('href="')
```



```
titleURL = 'http://gom.gomtv.com'+title

sTitleURLList.insert(0, sType['B_C_NOTIFY_SUBURL'])
sTitleURLList.insert(1, str(len(titleURL)))
sTitleURLList.insert(2, titleURL)
time.sleep(0.1)
db.connect(HOST, PORT)
db.sendURL(sTitleURLList)
db.closesocket()

def getGomDownLink(url):
    #url =
    ['http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=910618&prepage=1&md
    5key=&md5key=']
    print ("getGomDownLink Start!")
    rDownLinkList = []
    rIntCapSeq = re.compile('Wd+')
    rTitle = re.compile("W'(.+?).smiW'")
    down_cnt = 0
    contentsList = []

    gomDown = urllib.request.urlopen(url)
    textTemp = gomDown.read()
    textTemp = textTemp.decode('utf-8')
    text = str(textTemp).replace("$", "")
    soup = BeautifulSoup(text, 'html.parser')
    rDownLink = soup('a', {'class':'btn_type3 download'})
    if(len(rDownLink) == 0):
        return 0
```

```
rDownLink[0] = str(rDownLink[0])
rDownList.insert(0, rDownLink[0])
intSeq = rIntCapSeq.findall(str(rDownList[0]))[1]
capSeq = rIntCapSeq.findall(str(rDownList[0]))[2]
temp = rTitle.findall(str(rDownList[0]))[0]
titleTemp = str(rTitle.findall(str(rDownList[0]))[0])
fileName = str(titleTemp.split("'")[-1])
#print ("fileName : ", smart_unicode(fileName))

fullDownURL=
gom_boardURL+quote(fileName)+'?'+gom_DownCHPT_URL+'intSeq='+str(intSeq)+'&'+ 'capSeq='+str(capSeq)
req = urllib.request.Request(fullDownURL)
res = urllib.request.urlopen(req)
#fileName = smart_unicode(fileName)

#http://gom.gomtv.com/main/index.html/Femme.Fatales.S01E01.720p.HDTV.x264.smi?ch=subtitles&pt=down
&intSeq=910618&capSeq=908517
contents = res.read()
try:
    contents = contents.decode('utf-8').encode('cp949', 'ignore').decode('cp949')

except UnicodeDecodeError:
    contents = contents.decode('cp949')

contentsList.insert(0, fileName)
contentsList.insert(1, contents)
return contentsList
```

```
def checkUpdatedDownURL():
    print("checkUpdatedDownURL")
    time.sleep(10000)
    page = 1
    check_cnt = 0
    temp = 0
    updatedTitlesList = []

    while (temp != page):
        #print "check while start!"
        #print "check page : ", page
        updatedPageURL = gom_gomMainPage + gom_searchMainBoard + str(page)
        getGomTitleLink(updatedPageURL)
        recvURLNum = 0
        while True:
            db.connect(HOST, PORT)
            db.reqURL() # titleURL 요청
            updatedTitleURL = db.recvURL()

            db.closesocket()

            if(len(updatedTitleURL) == 0): # 가져온 titleURL의 길이가 0일 때 (받은 패킷의 URL 길이가
0이면 종료)
                break;
            updatedTitleList.append(updatedTitleURL)

        if(len(updatedTitleList) > 15):
            page = page+1
```

```
temp = temp+1
```

```
return updatedTitleList
```

```
def contentsEditSend(contentsList):
    print ("contentsEditSend Start!")
    sendSubList = []
    #contentsList[0] : FileName
    #contentsList[1] : Contents
    if(contentsList == 0) :
        print ("FileName - ",contentsList[0], " contents Error!")
    else:
        if(SubEditor.checkKREN(str(contentsList[1])) == True):           # 자막 정렬
            oneSortedSubList = SubEditor.sortTXT(contentsList)
            for j in range(0, len(oneSortedSubList)-1):
                sendTitle = contentsList[0]
                pStyle = re.compile(".smi")
                sendTitle = pStyle.sub("", sendTitle)
                pStyle = re.compile(r"720p.*", re.IGNORECASE)
                sendTitle = pStyle.sub("", sendTitle)
                pStyle = re.compile(r"고화질", re.IGNORECASE)
                sendTitle = pStyle.sub("", sendTitle)
                pStyle = re.compile(r"무삭제", re.IGNORECASE)
                sendTitle = pStyle.sub("", sendTitle)
                pStyle = re.compile(r"BDRip.*", re.IGNORECASE)
                sendTitle = pStyle.sub("", sendTitle)
                pStyle = re.compile(r"dvdrip.*", re.IGNORECASE)
```

```
sendTitle = pStyle.sub(' ', sendTitle)
pStyle = re.compile(r"HD.*", re.IGNORECASE)
sendTitle = pStyle.sub(' ', sendTitle)
pStyle = re.compile(r"1080p.*", re.IGNORECASE)
sendTitle = pStyle.sub(' ', sendTitle)
pStyle = re.compile(r"[W[W*W-W/W(W)W]]", re.IGNORECASE)
sendTitle = pStyle.sub('_', sendTitle)
pStyle = re.compile(r" ", re.IGNORECASE)
sendTitle = pStyle.sub(' ', sendTitle)
pStyle = re.compile(r"W.", re.IGNORECASE)
sendTitle = pStyle.sub('_', sendTitle)

pRmvSpace = re.compile(" ")
print ("sendTitle: ", sendTitle)
sendEN = oneSortedSubList[j][1]
sendEN = sendEN.strip()
sendEN = pRmvSpace.sub(" ", sendEN)
sendKO = oneSortedSubList[j][2]
sendKO = sendKO.strip()
sendKO = pRmvSpace.sub(" ", sendKO)
if(len(sendEN) == 0):
    print("len(sendEN) is null")
    continue
if(len(sendKO) == 0):
    print("len(sendKO) IS NULL")
    continue
if(sendEN == sendKO):
    continue

#print ("sendEN : ", sendEN)
```

```
sendENKeywords = SubEditor.getEngNoun(sendEN)
sendENKeywordsLen = len(sendENKeywords)
```

```
sendKOKeywords = SubEditor.getKorNoun(sendKO)
sendKOKeywordsLen = len(sendKOKeywords)
```

```
pStyle = re.compile("")
sendEN = pStyle.sub("", sendEN)
sendKO = pStyle.sub("", sendKO)
```

```
sendSubList.insert(0, sendTitle) # 0 : 타이틀
sendSubList.insert(1, sendEN) # 1 : 영문장
sendSubList.insert(2, sendKO) # 2 : 한국어문장
sendSubList.insert(3, str(sendENKeywordsLen+sendKOKeywordsLen)) # 3 : 영단어+한단어
```

개수

```
for k in range(4, 4+sendENKeywordsLen):
    sendSubList.insert(k, sendENKeywords[k-4])

for k in range(0, sendKOKeywordsLen):
    sendSubList.insert(k+4+sendENKeywordsLen, sendKOKeywords[k])

#time.sleep(0.5)
#db.connect(HOST, PORT)
#print("sendSubtitle!")
if(len(sendSubList) == 0):
    print ("SendSubList Null!")
else:
    print("db.sendSubTitle(sendSubList);")
    db.sendSubtitle(sendSubList)
#db.closesocket()
```

```
    else:
        print ("Wrong KREN SMI")

db = SubDB()

class AllStatesFetched(Exception):
    def __init__(self):
        pass

if __name__ == '__main__':
    print ('starting Gom Crawl.py...')
    lastBoardNum = getGomLastBoard(gom_mainBoardPage)
    lastBoardNum = 3
    #print ("lastBoardNum : ", lastBoardNum)
    pageURLList = []
    pageURLList = getGomAllBoardPageURL(lastBoardNum)
    titleList = []

    getGomTitleLink(pageURLList)

    while True:
        time.sleep(0.5)
        db.connect(HOST, PORT)
        db.reqURL() # titleURL 요청
        titleURL = db.recvURL()
        db.closesocket()
```

```
if(len(titleURL) == 0): # 가져온 titleURL의 길이가 0일 때 (받은 패킷의 URL 길이가 0이면 종료)
    break;
titleList.append(titleURL)

#titleList = [

#'http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=910913&prepage=1&md
5key=&md5skey=',

#'http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=910903&prepage=1&md
5key=&md5skey=',

#'http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=910882&prepage=1&md
5key=&md5skey=',

#'http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=910844&prepage=1&md
5key=&md5skey=',
    #
'http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=910834&prepage=1&md5
key=&md5skey='
    #]

#print (titleList[0])
#titleList =
['http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=911016&prepage=1&md
5key=&md5skey=']
for i in range(0, len(titleList)):
    contentsList = getGomDownLink(titleList[i])
    contentsEditSend(contentsList)
```



```
while 1:
    print ("checkUpdatedDownURL START! ")
    updatedTitleList = []
    time.sleep(100)
    updatedTitleList = checkUpdatedDownURL()
    #updatedTitleList =
['http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=908441&prepage=5&md5key=&md5skey=',
'http://gom.gomtv.com/main/index.html?ch=subtitles&pt=v&menu=subtitles&seq=907827&prepage=7&md5key=&md5skey=']

    for i in range(0, len(updatedTitleList)):
        updatedContentsList = getGomDownLink(updatedTitleList[i])
        contentsEditSend(updatedContentsList)
```

마) 자막 통신

```

#-*- coding:cp949 -*-
import socket
import re
import pprint

sType = dict(B_C_NOTIFY_SUBTITLE = '201', B_C_NOTIFY_SUBURL = '202', B_C_REQ_SUBURL = '203' ,
B_S_ANS_SUBURL = '204', B_C_REQ_WORD = '509', B_S_ANS_SUBTITLE = '504', B_S_ANS_COUNT = '503') # 패킷
타입

sMode = dict(MODE_PAGE_URL = '1', MODE_TITLE_URL = '2') # URL 타입
from C_Python_Socket import C_Python_Socket
import time

#####
# SOCKET NETWORKING
HOST = '192.168.0.185'          # The remote host
#HOST = '127.0.0.1'
PORT = 10000          # The same port as used by the server
s = socket.socket()

#####

def FillSpacePacket(dataLen, index):
    """패킷의 자릿수를 채워야 할때 공백(' ')으로 빈공간을 채워줌
    dataLen : 분석을 원하는 변수의 길이
    index : space로 채울 인덱스 최대값
    ex)
    data = 'abcd'
    data += FillSpacePacket(data.__len__(), 5)
    print(data)
    결과 : 'abcd  '
    빈칸 두개 생성"""

```

```
index += 1
space = ''
if dataLen < index:
    for count in range(0, index-dataLen):
        space += ' '
return space

def isASCII(text):
    """ASCII문자인지 판별. text에 ASCII가 아닌 문자가 한개라도 있으면 False, 없으면 True"""
    return not bool(re.search('[^Wx00-Wx7F]', text))

def __Len_Cstyle__(text):
    """text를 C 스타일 길이로 구함"""
    CLen = 0
    for i in text:
        if isASCII(i):
            CLen += 1
        else: CLen += 2

    return CLen

class SubDB():
    def connect(self, ip, port):
        """사용하기 전 주의 사항
        1) import socket
        2) 전역변수 s = socket.socket()를 선언"""
        global s
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        HOST = ip                # The remote host
        PORT = port              # The same port as used by the server
```

```
s.connect((HOST, PORT))
```

```
def closesocket(self):
```

```
    s.close()
```

```
def sendSubtitle(self, sendSubList):
```

```
    data = ""
```

```
    dataLen = ""
```

```
    sendData = ""
```

```
    eachWordLenList = []
```

```
    subEnLen = 0
```

```
    subKOLen = 0
```

```
    wordNumLen = 0
```

```
    data2 = ""
```

```
    titleLen = __Len_Cstyle__(sendSubList[0])
```

```
    subENLen = __Len_Cstyle__(sendSubList[1])
```

```
    subKOLen = __Len_Cstyle__(sendSubList[2])
```

```
    #wordNumLen = __Len_Cstyle__(str(sendSubList[3]))
```

```
    for i in range(0, int(sendSubList[3])):
```

```
        eachWordLen = __Len_Cstyle__(sendSubList[4+i])
```

```
        eachWordLenList.append(eachWordLen)
```

```
    #wordTupleList = []
```

```
    #titleTuple = (titleLen, sendSubList[0])
```

```
#subENTuple = (subENLen, sendSubList[1])
#subKOTuple = (subKOLen, sendSubList[2])
#wordNumTuple = (wordNumLen, str(sendSubList[3]))
#for i in range(0, wordNumLen):
#    wordTupleList.append((eachWordLenList[i], sendSubList[4+i]))
#sendSub = C_Python_Socket.generalSend(B_C_NOTIFY_SUBTITLE,)
```

```
data += sType['B_C_NOTIFY_SUBTITLE']
data += '\0'
data += FillSpacePacket(data.__len__(), 3)
```

```
data += str(titleLen)
data += '\0'
data += FillSpacePacket(data.__len__(), 3+4)
data += sendSubList[0]
```

```
data += str(subENLen)
data += '\0'
subENLenSpace = FillSpacePacket(str(subENLen).__len__(), 2)
data += subENLenSpace
data += sendSubList[1]
```

```
data += str(subKOLen)
data += '\0'
subKOLenSpace = FillSpacePacket(str(subKOLen).__len__(), 2)
data += subKOLenSpace
data += sendSubList[2]
```

```
#data += sendSubList[3]
#data += '\0'
#wordNumLenSpace = FillSpacePacket(str(sendSubList[3]).__len__(), 2)
#data += wordNumLenSpace
data2 = data

for i in range(0, int(sendSubList[3])):
    sendData = ""
    data += str(eachWordLenList[i])
    data += '\0'
    eachWordSpace = FillSpacePacket(str(eachWordLenList[i]).__len__(), 2)
    data += eachWordSpace
    data += sendSubList[4+i]

    dataLen = __Len_Cstyle__(data) + 8
    dataLen = str(dataLen)
    dataLen += '\0'
    dataLen += FillSpacePacket(dataLen.__len__(), 7)
    sendData += dataLen
    sendData += data
    sendData = sendData.encode('cp949')

pprint.pprint (sendData.decode('cp949'))

f = open('./smi/edit6.txt', 'a')
f.write(sendData.decode('cp949'))
f.write('\n\n')
f.close()
```

```
try:
    #pprint.pprint (sendSubList[0])
    self.connect(HOST, PORT)
    time.sleep(0.5)
    s.sendall(sendData)
    #data = data2
    self.closesocket()
except:
    print("packet send error!")
    pass

data = data2

def sendURL(self, sendURLList):
    data = ""
    dataLen = ""
    sendData = ""
    #print ("sendURLList[0] : ", sendURLList[0]) # 패킷 타입
    #print ("sendURLList[2] : ", sendURLList[1]) # url 길이
    #print ("sendURLList[3] : ", sendURLList[2]) # url

    #sendURLLen = __Len_Cstyle__(str(sendURLList[1]))
    sendURLLen = __Len_Cstyle__(sendURLList[1])

    data += sendURLList[0] # 패킷 타입
    data += '\x0'
    data += FillSpacePacket(data.__len__(), 3)

    data += sendURLList[1] # url 길이
    data += '\x0'
```

```
subPageURLLenSpace = FillSpacePacket(sendURLLen, 2)
data += subPageURLLenSpace
```

```
data += sendURLList[2] # url
```

```
dataLen = __Len_Cstyle__(data) + 8
dataLen = str(dataLen)
dataLen += 'W0'
dataLen += FillSpacePacket(dataLen.__len__(), 7)
sendData += dataLen
sendData += data
sendData = sendData.encode('cp949')
```

```
pprint.pprint (sendData)
s.sendall(sendData)
```

```
def reqURL(self):
```

```
    print("reqURL!")
    data = ""
    dataLen = ""
    sendData = ""
```

```
    data += sType['B_C_REQ_SUBURL']
    data += 'W0'
    data += FillSpacePacket(data.__len__(), 3)
```

```
    dataLen = __Len_Cstyle__(data) + 8
    dataLen = str(dataLen)
```



```
dataLen += 'W0'
dataLen += FillSpacePacket(dataLen.__len__(), 7)
sendData += dataLen
sendData += data
sendData = sendData.encode('cp949')

print ("send req url data")
s.sendall(sendData)

def recvURL(self):
    b_size = ""
    b_type = ""
    b_urllen = ""
    b_url = ""
    data = s.recv(1024)
    data = data.decode('cp949')

    if not data:
        print ("no data received")
        pass

    print ('Received URL-----', data)
    b_listTemp = []

    for i in range(0, 7):
        if(data[i] != 'W0'):
            b_size += data[i]

    for i in range(8, 11):
        if(data[i] != 'W0'):
```

```
        b_type += data[i]

    for i in range(12, 15):
        if(data[i] != '\0'):
            b_urlen += data[i]

    for i in range(0, 0+int(b_urlen)):
        b_url += data[i+16]

    #print ("b_type : ", b_type)
    #print ("b_urlen : ", b_urlen)
    #print ("b_url : ", b_url)

    #b_list = "".join(b_listTemp)

    #print ("b_list : ", b_list)

    print ("-----")

    return b_url
```

바) 자막 텍스트 편집

```
#-*- coding:cp949 -*-
from bs4 import BeautifulSoup
#from django.utils.encoding import smart_str, smart_unicode
from urllib import *
import urllib.robotparser
from ctypes import *
import urllib
import time, traceback, re, os
import sqlite3
import codecs
import cgi
import sys
import konlpy
import pprint
import nltk
from konlpy.tag import Komoran
from konlpy.tag import Twitter
from nltk.tokenize import RegexpTokenizer
from SubDB import *

tokenizer = None
tagger = None

dir = './smi/'

def allFiles(dir):
    tempFileList = []
    filenames = os.listdir(dir)
```

```
for filename in filenames:
    full_filename = os.path.join(dir, filename)
    tempFileList.append(full_filename)

return tempFileList

def checkKREN(contents):
    "다운받은 smi 파일이 정상인지 체크"
    fileList = []

    #contents = contents.decode('cp949')
    pKRCC = re.compile(r'<P Class=KR.*>', re.IGNORECASE)
    pENCC = re.compile(r'<P Class=EN.*>', re.IGNORECASE)

    #print (contents)
    try:
        if(bool(re.search(pKRCC, contents))):
            if(bool(re.search(pENCC, contents))):
                return True

        else:
            print ("File contents error!!!")
            return False
    except:
        print ("File import error!!!")
        return False

def sortTXT(contentsList):
```

"smi파일내용 정렬"

```
contents = contentsList[1][:]
```

```
print(type(contents))
```

```
#contents = contents.decode('utf-8').encode('cp949', 'ignore').decode('cp949')
```

```
#contents = contents.decode('cp949', 'ignore')
```

```
#contents = contentstemp
```

```
print (type(contents))
```

```
pStyle = re.compile('<br>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub(' ', contents)
```

```
pStyle = re.compile('<font color=.*?>', re.IGNORECASE)
```

```
contents = pStyle.sub(' ', contents)
```

```
pStyle = re.compile('</font>', re.IGNORECASE)
```

```
contents = pStyle.sub(' ', contents)
```

```
pStyle = re.compile('<HEAD(.*)>(.*)</HEAD>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub("", contents)
```

```
pStyle = re.compile('<!--(.*)-->', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub("", contents)
```

```
pStyle = re.compile('<br>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub(' ', contents)
```

```
pStyle = re.compile('<SAMI>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub("", contents)
```

```
pStyle = re.compile('<BODY>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub("", contents)
```

```
pStyle = re.compile('</SAMI>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub("", contents)
```

```
pStyle = re.compile('</BODY>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub("", contents)
```

```
pStyle = re.compile('<i>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
```

```
contents = pStyle.sub("", contents)
```

```

pStyle = re.compile('</i>', re.IGNORECASE | re.MULTILINE | re.DOTALL)
contents = pStyle.sub("", contents)
#print (contents)
pStyle = re.compile(r'<SYNC Start=₩d+><P Class=KR.*>&nbsp;')
contents = pStyle.sub("", contents)
pStyle = re.compile(r'<SYNC Start=₩d+><P Class=EN.*>&nbsp;')
contents = pStyle.sub("", contents)

pStyle = re.compile(r'(\r\n{1}$\r\n)', re.IGNORECASE | re.MULTILINE | re.DOTALL)
contents = pStyle.sub(r'', contents)
#print (contents)
pStyle = re.compile(r'(\r\n{2,})', re.IGNORECASE | re.MULTILINE | re.DOTALL)
contents = pStyle.sub(r'\r\n', contents)

#print (contents)
pStyle = re.compile(r'\r\n$', re.IGNORECASE | re.MULTILINE | re.DOTALL)
contents = pStyle.sub(r'', contents)
#print (contents)
pStyle = re.compile(r'{2,}', re.MULTILINE | re.DOTALL)
contents = pStyle.sub(r' ', contents)
pStyle = re.compile(r'\r\n', re.IGNORECASE | re.MULTILINE | re.DOTALL)
contents = pStyle.sub(r' ', contents)

#print (contents)
#print ("-----")
contentsKRTemp = []

```

```
contentsENTemp = []
contentsKR = []
contentsEN = []

pStyleKR = re.compile(r'<SYNC Start=(\Wd+)><P Class=KR.*?>(.*?)<', re.IGNORECASE | re.MULTILINE |
re.DOTALL)

pStyleEN = re.compile(r'<SYNC Start=(\Wd+)><P Class=EN.*?>(.*?)<', re.IGNORECASE | re.MULTILINE |
re.DOTALL)

contentsKRTemp = pStyleKR.findall(contents)
contentsENTemp = pStyleEN.findall(contents)

#print (contentsKRTemp)
#pTime = re.compile(r'WS{2}WZ')
pTime = re.compile('[0-9]{2}WZ')

for i in range(0, len(contentsKRTemp)):
    #print ("contentskrtemp")
    #if(contentsKRTemp[i][1] == ' '):
    #    continue;
    #contentsKRTemp[i][1] = str(contentsKRTemp[i][1]).strip()
    contentsKR.append(list(contentsKRTemp[i]))
    contentsKR[i][0] = pTime.sub("", contentsKR[i][0])
    #smart_str(contentsKR[i][1])
    #print (contentsKR[i][0])
    #print (contentsKR[i][1])

for i in range(0, len(contentsENTemp)):
    contentsEN.append(list(contentsENTemp[i]))
    contentsEN[i][0] = pTime.sub("", contentsEN[i][0])
```

```
#print (contentsKR[0][1])
#print contentsEN
oneTotalSubList = joinKREN(contentsKR, contentsEN)
#print.pprint(oneTotalSubList)
return oneTotalSubList
```

```
def joinKREN(list_kr, list_en):
```

```
    totalSubList = []
    totalSubListTemp = []
    totalSubListTemp2 = []
```

```
    if (len(list_kr) <= len(list_en)):
```

```
        for i in range(0, len(list_kr)):
```

```
            for j in range(0, len(list_en)):
```

```
                if(list_kr[i][0] != list_en[j][0]):
```

```
                    pass
```

```
                else:
```

```
                    totalSubListTemp.insert(0, list_kr[i][0])
```

```
                    totalSubListTemp.insert(1, list_en[j][1])
```

```
                    totalSubListTemp.insert(2, list_kr[i][1])
```

```
                    totalSubList.append(list(totalSubListTemp[0:3]))
```

```
                    break
```

```
    else:
```

```
        for i in range(0, len(list_en)):
```

```
            for j in range(0, len(list_kr)):
```

```
                if(list_en[i][0] != list_kr[j][0]):
```

```
                    pass
```

```
                else:
```



```
totalSubListTemp.insert(0, list_en[i][0])
totalSubListTemp.insert(1, list_en[i][1])
totalSubListTemp.insert(2, list_kr[j][1])
totalSubList.append(list(totalSubListTemp[0:3]))

return totalSubList

def makeFileName(_FileList):
    mkFileNameList = _FileList[:]
    pStyle = re.compile('./smi/')
    for i in range(0, len(FileList)):
        mkFileNameList[i] = pStyle.sub("", mkFileNameList[i])

    return mkFileNameList

def getKorNoun(body):#태그없앤 content에서 명사만 추출하기
    contents = body[:]

    contents = re.sub('[^가-힣 \n]+', '', contents)
    #print (contents)

    kkma = konlpy.tag.Kkma() #-Xmx128m 로 바꾸기
    #print("Get nouns from contents...")
    keywords = kkma.nouns(contents)
    #print (type(keywords))
    #print ("kkma   :", keywords)
    pSub = re.compile("sub_")
    for i in range(0, len(keywords)):
        keywords[i] = re.sub(keywords[i], "sub_"+str(keywords[i]), keywords[i])
```

```
#keywords = list(set(keywords))
print ("Keywords : ", keywords)
return keywords
```

```
def getEngNoun(contents):
```

```
    tokenizer = RegexpTokenizer("[Ww']+")
    keywords = tokenizer.tokenize(contents)
    pStyle = re.compile("")
    pStyle2 = re.compile(r"W.")
    #pSub = re.compile("sub_")
    for i in range(0, len(keywords)):
        keywords[i] = pStyle.sub("", keywords[i])
        keywords[i] = pStyle2.sub("", keywords[i])
        keywords[i] = re.sub(keywords[i], "sub_"+str(keywords[i]), keywords[i])
    keywords = list(set(keywords))
    print ("keywords : ", keywords)
    return keywords
```

```
def FillSpacePacket(dataLen, index):
```

```
    """패킷의 자릿수를 채워야 할때 공백(' ')으로 빈공간을 채워줌
    dataLen : 분석을 원하는 변수의 길이
    index : space로 채울 인덱스 최대값
    ex)
    data = 'abcd'
    data += FillSpacePacket(data.__len__(), 5)
    print(data)
    결과 : 'abcd  '
    빈칸 두개 생성"""
    index += 1
    space = "
```

```
if dataLen < index:
    for count in range(0, index-dataLen):
        space += ' '
    return space

def isASCII(text):
    """ASCII문자인지 판별. text에 ASCII가 아닌 문자가 한개라도 있으면 False, 없으면 True"""
    return not bool(re.search('[^Wx00-Wx7E]', text))

def __Len_Cstyle__(text):
    """text를 C 스타일 길이로 구함"""
    CLen = 0
    for i in text:
        if isASCII(i):
            CLen += 1
        else: CLen += 2
    return CLen
```

사) 서버 공통

Packet.h 파일

```
#pragma once
```

```
enum PacketType {  
    //자막 크롤러 관련 패킷  
    B_C_NOTIFY_SUBTITLE = 201, //자막 크롤러가 서버로 자막 데이터를 보낼 때 사용하는 패킷  
    B_C_NOTIFY_SUBURL = 202,  
    B_C_REQ_SUBURL = 203,  
    B_S_ANS_SUBURL = 204,  
  
    //사전, 웹 크롤러, 웹 검색 모드 이하 생략  
};
```

PacketClass.h 파일

```
#pragma once  
#include "functionTable.h"  
#include "Packet.h"  
#include <iostream>  
#include <vector>  
  
using namespace std;  
  
class Packet {  
public:  
    virtual PacketType type() = 0;  
    virtual void encoding() {}  
    virtual void decoding(char* buf) {}  
};  
//대표로 PK_C_NOTIFY_SUBTITLE 패킷 정의 소스  
class PK_C_NOTIFY_SUBTITLE : public Packet  
{
```

private:

```
string title;  
string kor;  
string eng;  
vector<string> word;
```

```
char stream[10240];  
size_t offset;
```

public:

```
PacketType type() { return B_C_NOTIFY_SUBTITLE; }
```

```
void setTitle(string s) { title = s; }  
string getTitle() { return title; }
```

```
void setKor(string s) { kor = s; }  
string getKor() { return kor; }
```

```
void setEng(string s) { eng = s; }  
string getEng() { return eng; }
```

```
size_t getSize() { return offset; }  
char* getStream() { return stream; }
```

```
vector<string> getWord() { return word; }
```

```
void encoding()  
{  
    offset = 0;  
    _itoa(this->type(), stream, 10);  
    offset += sizeof(int);  
  
    _itoa(title.length(), stream + offset, 10);  
    offset += sizeof(int);  
    memcpy_s(stream + offset, title.length(), title.c_str(), title.length());  
}
```

```
offset += title.length();

_itoa(kor.length(), stream + offset, 10);
offset += sizeof(int);
memcpy_s(stream + offset, kor.length(), kor.c_str(), kor.length());
offset += kor.length();

_itoa(eng.length(), stream + offset, 10);
offset += sizeof(int);
memcpy_s(stream + offset, eng.length(), eng.c_str(), eng.length());
offset += eng.length();

_itoa(word.size(), stream + offset, 10);
offset += sizeof(int);
for (int i = 0; i < word.size(); i++)
{
    _itoa(word.at(i).length(), stream + offset, 10);
    offset += sizeof(int);
    memcpy_s(stream + offset, word.at(i).length(), word.at(i).c_str(), word.at(i).length());
    offset += word.at(i).length();
}
}

void decoding(char* buf)
{
    title.clear();
    kor.clear();
    eng.clear();
    word.clear();

    offset = sizeof(int);

    size_t len = atoi(buf + offset);
    offset += sizeof(int);
```

```
char temp[128];

memcpy_s(temp, len, buf + offset, len);
offset += len;
temp[len] = 'W0';
title = temp;

len = atoi(buf + offset);
offset += sizeof(int);

memcpy_s(temp, len, buf + offset, len);
offset += len;
temp[len] = 'W0';
kor = temp;

len = atoi(buf + offset);
offset += sizeof(int);

memcpy_s(temp, len, buf + offset, len);
offset += len;
temp[len] = 'W0';
eng = temp;

int count = atoi(buf + offset);
offset += sizeof(int);

for (int i = 0; i < count; i++)
{
    len = atoi(buf + offset);
    offset += sizeof(int);

    memcpy_s(temp, len, buf + offset, len);
    offset += len;
    temp[len] = 'W0';
    word.push_back(temp);
}
```

```
        }  
    }  
  
};
```

PacketFactory.h

```
#pragma once  
#include "Packet.h"  
#include "PacketClass.h"  
#include "aceWSingleton.h"  
  
class packetFactory  
{  
public:  
    Packet* getPacket(int packetType)  
    {  
        switch (packetType) {  
            //자막 크롤러에 대한 정의  
            case B_C_NOTIFY_SUBTITLE: return new PK_C_NOTIFY_SUBTITLE();  
            case B_C_NOTIFY_SUBURL: return new PK_C_NOTIFY_SUBURL();  
            case B_C_REQ_SUBURL: return new PK_C_REQ_SUBURL();  
            case B_S_ANS_SUBURL: return new PK_S_ANS_SUBURL();  
            //사전, 웹 크롤러 웹 검색 모드에 따른 것은 이하 생략  
            default: return NULL;  
        }  
    }  
};
```


아) 게이트 웨이

- 웹 서버로부터 데이터를 수신 한 패킷에 대해 패킷 검사를 수행 후 존재 하지 않는 패킷이면 패스 하고 적합한 패킷일 경우에만 스위치에게 전달해 주는 소스

```
void
```

```
EchoService::handle_read_stream(const ACE_Asynch_Read_Stream::Result &result) {
```

```
    if (!result.success()) {
```

```
        std::cout << std::endl << "Error:" << result.error();
```

```
        delete this;
```

```
        return;
```

```
    }
```

```
    if (0 == result.message_block().length()) {
```

```
        std::cout << std::endl << "Client Exist";
```

```
        delete this;
```

```
        return;
```

```
    }
```

```
    cout << endl << "client recv : " << endl;
```

```
    static char recvBuf[67000];
```

```
    char buf[67000];
```

```
    static char packetSize[8] = "0";
```

```
    int bufOffset = 0;
```

```
    static int recvOffset = 0;
```

```
    memcpy_s(buf, result.bytes_transferred(), result.message_block().rd_ptr(), result.bytes_transferred());
```

```
    while (bufOffset < result.bytes_transferred())
```

```
    {
```

```
        if (strcmp(packetSize, "0") == 0)
```

```
            memcpy_s(packetSize, sizeof(long long), buf + bufOffset, sizeof(long long));
```

```
        if (bufOffset + atoi(packetSize) > recvOffset + result.bytes_transferred())
```

```
        {
            memcpy_s(recvBuf + recvOffset, result.bytes_transferred() - bufOffset, buf +
bufOffset, result.bytes_transferred() - bufOffset);
            recvOffset = result.bytes_transferred() - bufOffset;
            break;
        }

        if (atoi(packetSize) < result.bytes_transferred())
            memcpy_s(recvBuf + recvOffset, atoi(packetSize), buf + bufOffset, atoi(packetSize));
        else
            memcpy_s(recvBuf + recvOffset, atoi(packetSize) - recvOffset, buf + bufOffset,
atoi(packetSize) - recvOffset);

        bufOffset += atoi(packetSize);

        Packet* p = PacketFactory::instance()->getPacket(atoi(recvBuf + sizeof(long long)));

        if (p == NULL)
        {
            ZeroMemory(recvBuf, 67000);
            strcpy(packetSize, "0");
            recvOffset = 0;
            break;
        }

        svc.send(recvBuf, atoi(packetSize));

        ZeroMemory(recvBuf, 67000);
        strcpy(packetSize, "0");
        recvOffset = 0;
    }

    msg_.reset();
    this->read_require_.read(msg_, msg_.space() - 1);
}
```

자) 스위치

- 게이트 웨이로부터 수신 받은 데이터의 패킷을 검사하여 해당 DB 서버에게로 데이터를 전송해 주는 소스

void

```
EchoService::handle_read_stream(const ACE_Asynch_Read_Stream::Result &result) {
```

```
    if (!result.success()) {
```

```
        std::cout << std::endl << "Error:" << result.error();
```

```
        delete this;
```

```
        return;
```

```
    }
```

```
    if (0 == result.message_block().length()) {
```

```
        std::cout << std::endl << "Client Exist";
```

```
        delete this;
```

```
        return;
```

```
    }
```

```
static char recvBuf[67000];
```

```
char buf[67000];
```

```
static char packetSize[8] = "0";
```

```
int bufOffset = 0;

static int recvOffset = 0;

static char packetHeader[8];

memcpy_s(buf, result.bytes_transferred(), result.message_block().rd_ptr(), result.bytes_transferred());

while (bufOffset < result.bytes_transferred())
{
    if (strcmp(packetSize, "0") == 0)

        memcpy_s(packetSize, sizeof(long long), buf + bufOffset, sizeof(long long));

    if (bufOffset + atoi(packetSize) > recvOffset + result.bytes_transferred())
    {
        memcpy_s(recvBuf + recvOffset, result.bytes_transferred() - bufOffset, buf + bufOffset,
result.bytes_transferred() - bufOffset);

        recvOffset = result.bytes_transferred() - bufOffset;

        break;
    }

    if (atoi(packetSize) < result.bytes_transferred())

        memcpy_s(recvBuf + recvOffset, atoi(packetSize), buf + bufOffset, atoi(packetSize));
```

```
else

    memcpy_s(recvBuf + recvOffset, atoi(packetSize) - recvOffset, buf + bufOffset, atoi(packetSize) -
recvOffset);

    bufOffset += atoi(packetSize);

    Packet* p = PacketFactory::instance()->getPacket(atoi(recvBuf + sizeof(long long)));

    if (p == NULL)

    {

        ZeroMemory(recvBuf, 67000);

        strcpy(packetSize, "0");

        recvOffset = 0;

        return;

    }

    memcpy_s(packetHeader, sizeof(int), recvBuf + sizeof(long long), sizeof(int));

    cout << "*****gateway recv packetHeader: " << packetHeader << "*****" << endl;

    int SW = atoi(packetHeader) / 100;

    cout << SW << endl;
```

```
if (SW == 2 || SW == 3 || SW == 4)

{

    C_svc.send(recvBuf, atoi(packetSize)); // 선택

}

else if (SW == 5)

{

    svc.send(recvBuf, atoi(packetSize)); // 및

}


ZeroMemory(recvBuf, 67000);

ZeroMemory(packetHeader, 8);

strcpy(packetSize, "0");

recvOffset = 0;


}


msg_.reset();

this->read_require_.read(msg_, msg_.space() - 1);

}
```

차) 크롤러 DB 서버

- 스위치로부터 수신 한 패킷 타입에 따라 적합한 DB 업무를 수행하는 각 함수를 호출 하는 소스

```
int packetProcess(Packet* p, char* recvData)
{

    switch (p->type())
    {
        //사전 시작
        case B_C_NOTIFY_DIC_URL_INFO:
        {
            cout << "NOTIFY_DIC_URL_INFO" << endl << endl;
            p->decoding(recvData);
            PK_C_NOTIFY_DIC_URL_INFO* packet = (PK_C_NOTIFY_DIC_URL_INFO *)p;
            dicUrl(*packet, ret, hstmt, hDbc);
            //upload file
        }
        break;

        case B_C_NOTIFY_DIC_WORD:
        {
            cout << "NOTIFY_DIC_WORD" << endl << endl;
            p->decoding(recvData);
            PK_C_NOTIFY_DIC_WORD* packet = (PK_C_NOTIFY_DIC_WORD *)p;
            dicWord(*packet, ret, hstmt, hDbc);
        }
        break;

        case B_C_NOTIFY_DIC_CONTENTS:
        {
            cout << "NOTIFY_DIC_CONTENTS" << endl << endl;
            p->decoding(recvData);
            PK_C_NOTIFY_DIC_CONTENTS* packet = (PK_C_NOTIFY_DIC_CONTENTS *)p;
            dicContents(*packet, ret, hstmt, hDbc);
        }
    }
}
```

```
}
break;

case B_C_NOTIFY_DIC_URL_PATH:
{
    cout << "NOTIFY_DIC_URL_PATH" << endl << endl;
    p->decoding(recvData);
    PK_C_NOTIFY_DIC_URL_PATH* packet = (PK_C_NOTIFY_DIC_URL_PATH *)p;
    dicSaveUrl(*packet, ret, hstmt, hDbc);
}
break;

case B_C_REQ_DIC_URL:
{
    cout << "REQ_DIC_URL" << endl << endl;
    PK_S_ANS_DIC_URL packet;
    dicGetUrl(packet, ret, hstmt, hDbc);

    //packet을 보내기

    eSvc->send(packet.getStream(), packet.getSize());
}
break;
//사전 끝

//자막 시작
case B_C_NOTIFY_SUBTITLE:
{
    cout << "NOTIFY_SUBTITLE" << endl << endl;
    p->decoding(recvData);
    PK_C_NOTIFY_SUBTITLE* packet = (PK_C_NOTIFY_SUBTITLE *)p;
    subWord(*packet, sret, shstmt, shDbc, subTable(*packet, sret, shstmt, shDbc));
}
break;
case B_C_NOTIFY_SUBURL:
```



```
{
    cout << "NOTIFY_SUBURL" << endl << endl;
    p->decoding(recvData);
    PK_C_NOTIFY_SUBURL* packet = (PK_C_NOTIFY_SUBURL *)p;

    urlTable(*packet, sret, shstmt, shDbc);
}
break;

case B_C_REQ_SUBURL:
{
    cout << "B_C_REQ_SUBURL" << endl << endl;
    PK_S_ANS_SUBURL sendPacket = urlAnsAll(sret, shstmt, shDbc);

    eSvc->send(sendPacket.getOutputStream(), sendPacket.getSize());
}
break;
//자막 끝

//웹 시작
case B_C_NOTIFY_WEB_URL_INFO:
{
    cout << "NOTIFY_WEB_URL_INFO" << endl << endl;
    p->decoding(recvData);
    PK_C_NOTIFY_WEB_URL_INFO* packet = (PK_C_NOTIFY_WEB_URL_INFO *)p;

    webInsert(*packet, ret, hstmt, hDbc);
}
break;

case B_C_REQ_WEB_URL:
{
    cout << "B_C_REQ_WEB_URL" << endl << endl;

    PK_S_ANS_WEB_URL packet;
```

```
webReadUrl(packet, ret, hstmt, hDbc);

//packet을 보내기
eSvc->send(packet.getStream(), packet.getSize());
}

case B_C_NOTIFY_WEB_URL_ERROR:
{
    cout << "B_C_NOTIFY_WEB_URL_ERROR" << endl << endl;
}
break;

case B_C_NOTIFY_WEB_CONTENTS:
{
    cout << "B_C_NOTIFY_WEB_CONTENTS" << endl << endl;
    p->decoding(recvData);
    PK_C_NOTIFY_WEB_CONTENTS* packet = (PK_C_NOTIFY_WEB_CONTENTS *)p;
    cout << packet->getUrl() << endl << endl;
    webContents(*packet, ret, hstmt, hDbc);
}
break;

case B_C_NOTIFY_WEB_WORD:
{
    cout << "B_C_NOTIFY_WEB_WORD" << endl << endl;
    p->decoding(recvData);
    PK_C_NOTIFY_WEB_WORD* packet = (PK_C_NOTIFY_WEB_WORD *)p;
    webWord(*packet, ret, hstmt, hDbc);
}
break;

case B_C_NOTIFY_MOTHER_URL:
{
    cout << "B_C_NOTIFY_MOTHER_URL" << endl << endl;
    p->decoding(recvData);
    PK_C_NOTIFY_MOTHER_URL* packet = (PK_C_NOTIFY_MOTHER_URL *)p;
```

```
        webMother(*packet, ret, hstmt, hDbc);
    }
    break;

    case B_C_REQ_MOTHER_URL_EXIST:
    {
        cout << "B_C_REQ_MOTHER_URL_EXIST" << endl << endl;
        p->decoding(recvData);
        PK_C_REQ_MOTHER_URL_EXIST* packet = (PK_C_REQ_MOTHER_URL_EXIST *)p;
        PK_S_ANS_MOTHER_URL_EXIST sendPacket;
        webReadMother(*packet, sendPacket, ret, hstmt, hDbc);

        //packet을 보내기
        eSvc->send(sendPacket.getOutputStream(), sendPacket.getSize());

        //cout << "보냄!" << endl;
    }
    break;
    //웹 끝
}
return 0;
}
```

카) 유저 DB 서버

- 스위치로부터 패킷을 전달 받은 패킷 헤더 검사 후 MODE 에 따라 DB 검색 쿼리를 결정해 데이터를 추출 하는 각 함수 호출

```
int packetProcess(Packet* p, char* recvData)
{
    if (p->type() == B_C_REQ_SEARCH)
    {
        cout << "_REQ_SEARCH" << endl << endl;

        p->decoding(recvData);

        PK_C_REQ_SEARCH* packet = (PK_C_REQ_SEARCH *)p;

        PK_S_ANS_COUNT* cntpacket;

        switch (packet->getMode())
        {
            case 1:
            {
                vector<PK_S_ANS_WEB_INFO> sendPacket_Web = findWeb(*packet, ret, hstmt, hDbc);

                for (int i = 0; i < 3; i++)
                {
                    sendPacket_Web.at(i).encoding();
                }
            }
        }
    }
}
```

```
eSvc->send(sendPacket_Web.at(i).getStream(), sendPacket_Web.at(i).getSize());

}

vector<PK_S_ANS_DIC_INFO> sendPacket_Dic = findDictionary(*packet, ret, hstmt, hDbc);

for (int i = 0; i < 3; i++)

{

    sendPacket_Dic.at(i).encoding();

    eSvc->send(sendPacket_Dic.at(i).getStream(), sendPacket_Dic.at(i).getSize());

}

vector<PK_S_ANS_SUBTITLE> sendPacket_Subtitle = findWord(*packet, ret, hstmt, hDbc);

for (int i = 0; i < 3; i++)

{

    sendPacket_Subtitle.at(i).encoding();

    eSvc->send(sendPacket_Subtitle.at(i).getStream(), sendPacket_Subtitle.at(i).getSize());

}

break;
```

```
}

case 2:

{

    vector<PK_S_ANS_WEB_INFO> sendPacket = findWeb(*packet, ret, hstmt, hDbc);

    cntpacket->setCnt(sendPacket.size());

    cntpacket->encoding();

    eSvc->send(cntpacket->getStream(), cntpacket->getSize());

    if (sendPacket.size() > packet->getPacketCnt() * 100)

    {

        for (int i = (packet->getPacketCnt() - 1) * 100; i < packet->getPacketCnt() * 100; i++)

        {

            sendPacket.at(i).encoding();

            eSvc->send(sendPacket.at(i).getStream(), sendPacket.at(i).getSize());

        }

    }

    else

    {

        for (int i = (packet->getPacketCnt() - 1) * 100; i < packet->getPacketCnt(); i++)

        {
```

```
        sendPacket.at(i).encoding();

        eSvc->send(sendPacket.at(i).getStream(), sendPacket.at(i).getSize());

    }

}

break;

}

case 3:

{

    vector<PK_S_ANS_DIC_INFO> sendPacket = findDictionary(*packet, ret, hstmt, hDbc);

    cntpacket->setCnt(sendPacket.size());

    cntpacket->encoding();

    eSvc->send(cntpacket->getStream(), cntpacket->getSize());

    if (sendPacket.size() > packet->getPacketCnt() * 100)

    {

        for (int i = (packet->getPacketCnt() - 1) * 100; i < packet->getPacketCnt() * 100; i++)

        {

            sendPacket.at(i).encoding();

            eSvc->send(sendPacket.at(i).getStream(), sendPacket.at(i).getSize());

        }

    }

}
```

```
    }

    else

    {

        for (int i = (packet->getPacketCnt() - 1) * 100; i < packet->getPacketCnt(); i++)

        {

            sendPacket.at(i).encoding();

            eSvc->send(sendPacket.at(i).getStream(), sendPacket.at(i).getSize());

        }

    }

    break;

}

case 4:

{

    vector<PK_S_ANS_SUBTITLE> sendPacket = findWord(*packet, ret, hstmt, hDbc);

    cntpacket->setCnt(sendPacket.size());

    cntpacket->encoding();

    eSvc->send(cntpacket->getStream(), cntpacket->getSize());

    int cnt = sendPacket.size();
```



```
if (sendPacket.size() > packet->getPacketCnt() * 100)
{
    for (int i = (packet->getPacketCnt() - 1) * 100; i < packet->getPacketCnt() * 100; i++)
    {
        sendPacket.at(i).encoding();

        eSvc->send(sendPacket.at(i).getStream(), sendPacket.at(i).getSize());
    }
}

else
{
    for (int i = (packet->getPacketCnt() - 1) * 100; i < packet->getPacketCnt(); i++)
    {
        sendPacket.at(i).encoding();

        eSvc->send(sendPacket.at(i).getStream(), sendPacket.at(i).getSize());
    }
}

break;
}

}
```

```
}  
  
return 0;  
  
}
```

타) Upload 서버

- 전달 받은 파일의 위치와 파일 명에 따라 업로드 하는 함수

```
void uploadWebFile(char *dir, char* name)  
{  
    CkHttp http;  
  
    bool success;  
    success = http.UnlockComponent("Anything for 30-day trial");  
    if (success != true) {  
        printf("%s\n", http.lastErrorText());  
        return;  
    }  
  
    http.put_AwsAccessKey("AKIAJZG2XTUMWW57V6GQ");  
  
    http.put_AwsSecretKey("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");  
  
    const char * bucketName;  
    bucketName = "boogle-web-file";  
  
    const char * contentType;  
    contentType = "text/txt";  
  
    success = http.S3_UploadFile(dir, contentType, bucketName, name);
```

```

        if (success != true) {
            printf("%s\n", http.lastErrorText());
        }
        else {
            printf("File uploaded.\n");
        }
    }
}

```

파) Download 서버

- 전달 받은 파일의 위치와 파일 명에 따라 다운로드 하는 함수
-

```
#include <CkHttp.h>
```

```
#include <iostream>
```

```
void main(void)
```

```
{
```

```
    CkHttp http;
```

```
    bool success = http.UnlockComponent("Anything for 30-day trial");
```

```
    if (success != true) {
```

```
        std::cout << http.lastErrorText() << "\n\n";
```

```
        return;
```

```
}
```

```
// Insert your access key here:
```

```
http.put_AwsAccessKey("AKIAJZG2XTUMWW57V6GQ");
```

```
// Insert your secret key here:
```

```
http.put_AwsSecretKey("ybieSdpbTXbvhetCi9guM1oNfKQ7RX+Pj6K8eqhd");
```

```
const char *bucketName = "my-boogle-crawler-db";
```

```
const char *objectName = "1.txt";
```

```
const char *charset = "euc-kr";
```

```
const char *fileContents = 0;
```

```
fileContents = http.s3_DownloadString(bucketName, objectName, charset);
```

```
if (fileContents == 0) {
```

```
    // Failed
```

```
    std::cout << http.lastErrorText() << "WrWn";
```

```
    }

    else {

        // Success

        std::cout << fileContents << "r\n";

    }

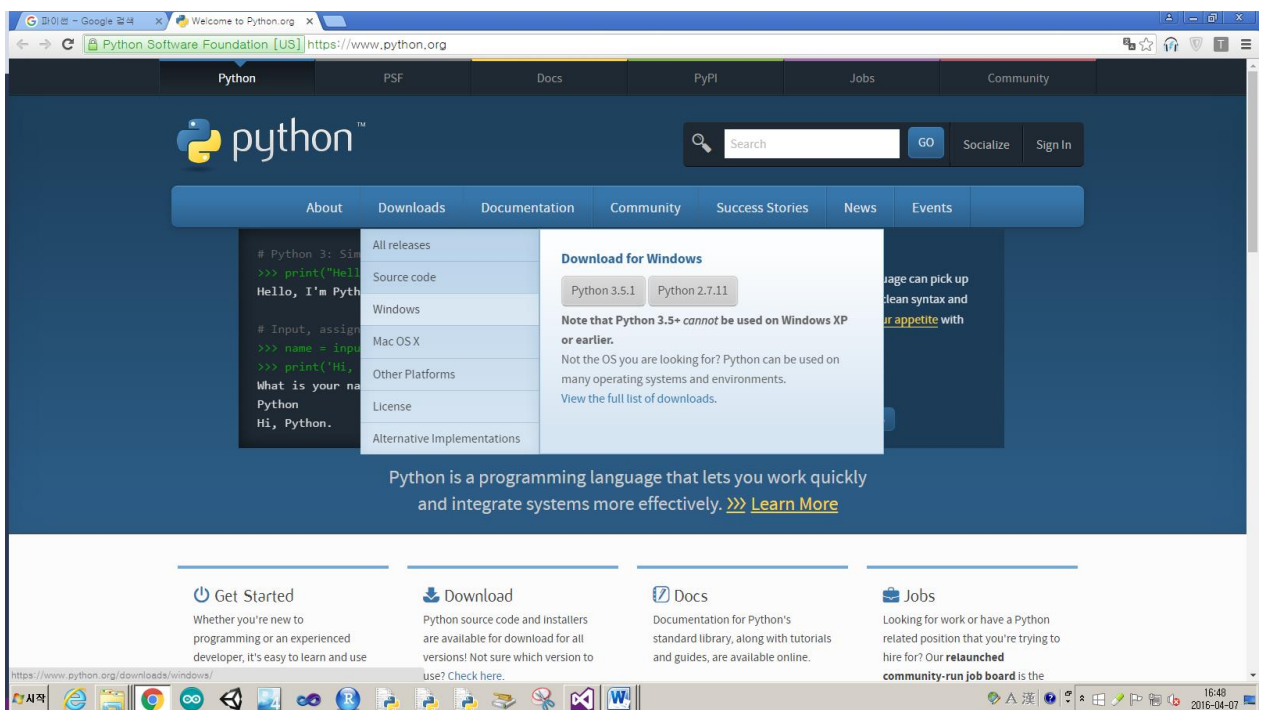
}
```

5. 개발 환경 설치 매뉴얼

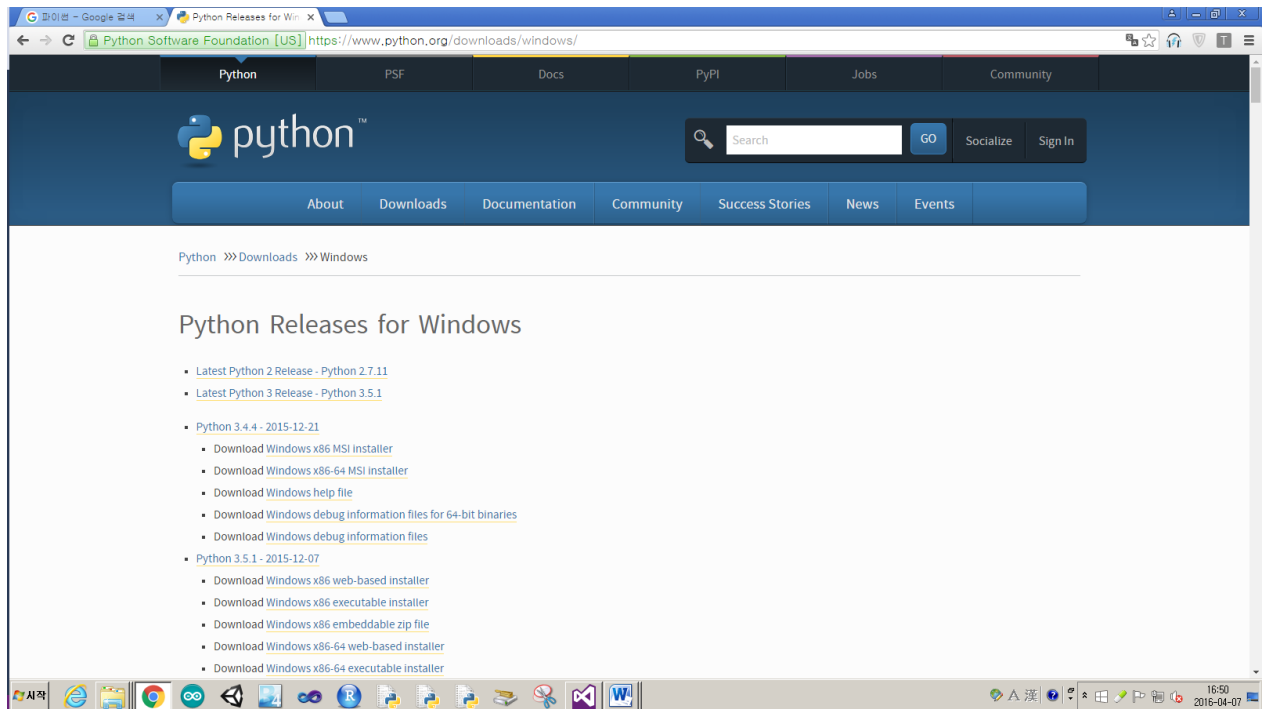
- 모든 내용은 윈도우 OS에 기반합니다.

A. 크롤러

(가) 파이썬3.4 설치



- <http://www.python.org>에 접속
- Downloads 메뉴의 Windows 클릭



- iii. 가장 최신의 3.4 버전 파이썬 Windows x86 MSI Installer 혹은 64비트의 경우 Windows x86-64 MSI Installer 다운로드

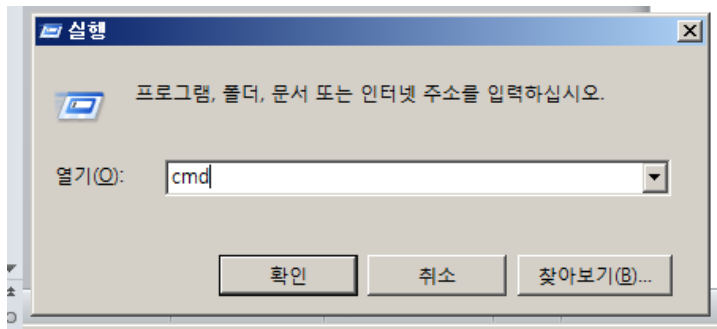
- iv. 파이썬 3.x 버전은 설치된 python 폴더의 scripts 하위 폴더에 pip.exe가 존재한다. 만약 존재 하지 않는 경우, google에 pip 검색 후 다운로드 혹은 <https://pypi.python.org/pypi/pip>

이곳에서 pip-8.1.1.tar.gz(혹은 최신 버전의 pip)를 다운받아 압축을 푼다.

이미 설치가 되어있는 경우 v. 에서부터 진행

- i. 압축을 푼 파일을 (기본 설치 경로)C:\python34\Scripts로 이동한다.

II. 명령 프롬프트를 호출



III. cd c:\python34\scripts\pip-8.1.1를 입력하여 C:\Python34\scripts\pip-8.1.1 디렉터리로 이동

IV. 명령 프롬프트에 setup.py build 수행

V. 명령 프롬프트에 setup.py install 수행

VI. 설치 완료

v. BeautifulSoup 모듈 설치

I. 환경 변수->사용자변수->path에 C:\Python34\Scripts가 있는 경우,

- 명령 프롬프트에 pip install bs4 수행

```
C:\Users\bit>pip install bs4
```

환경 변수 설정이 되어있지 않은 경우,

- 명령 프롬프트에 C:\Python34\Scripts\pip.exe install bs4 수행

II. 설치 완료

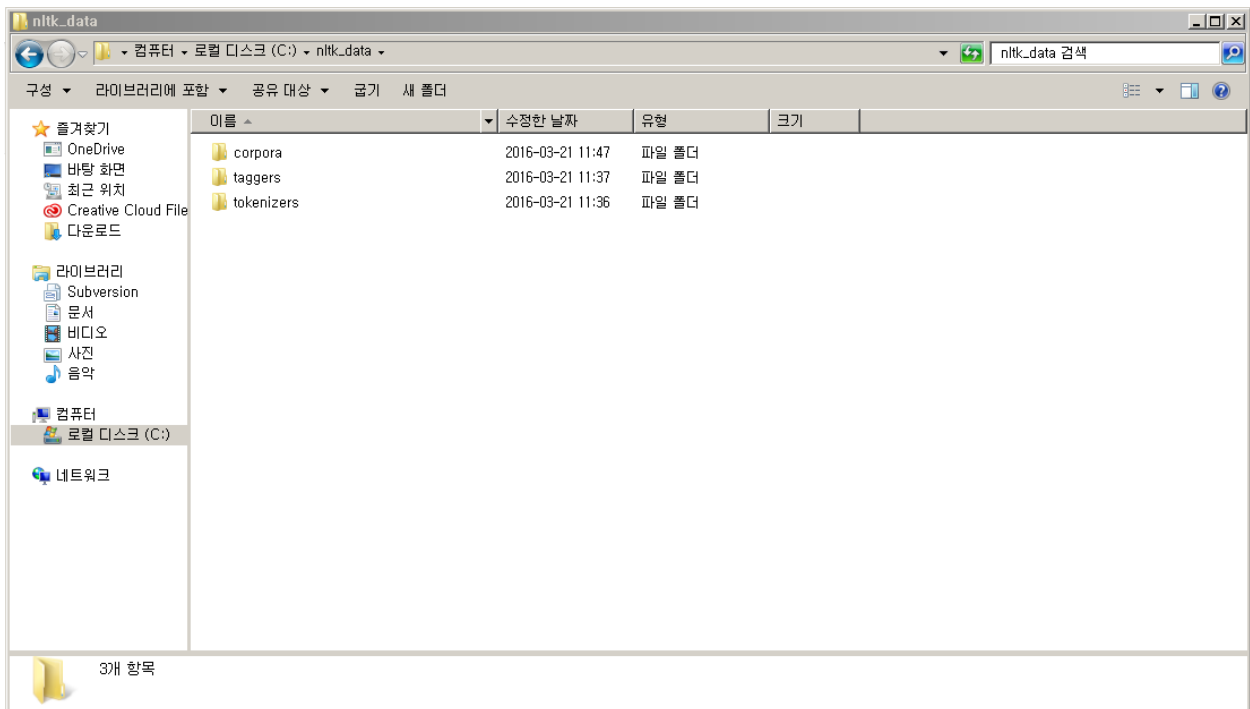
vi. NLTK 모듈 설치

I. 'v.'의 방법과 마찬가지로 설치

- 명령 프롬프트에 `pip install nltk` 수행
- Python IDLE을 이용해 `nltk.download()` 수행

II. `Nltk.download()` 접속 에러가 발생하는 경우, 수동 설치

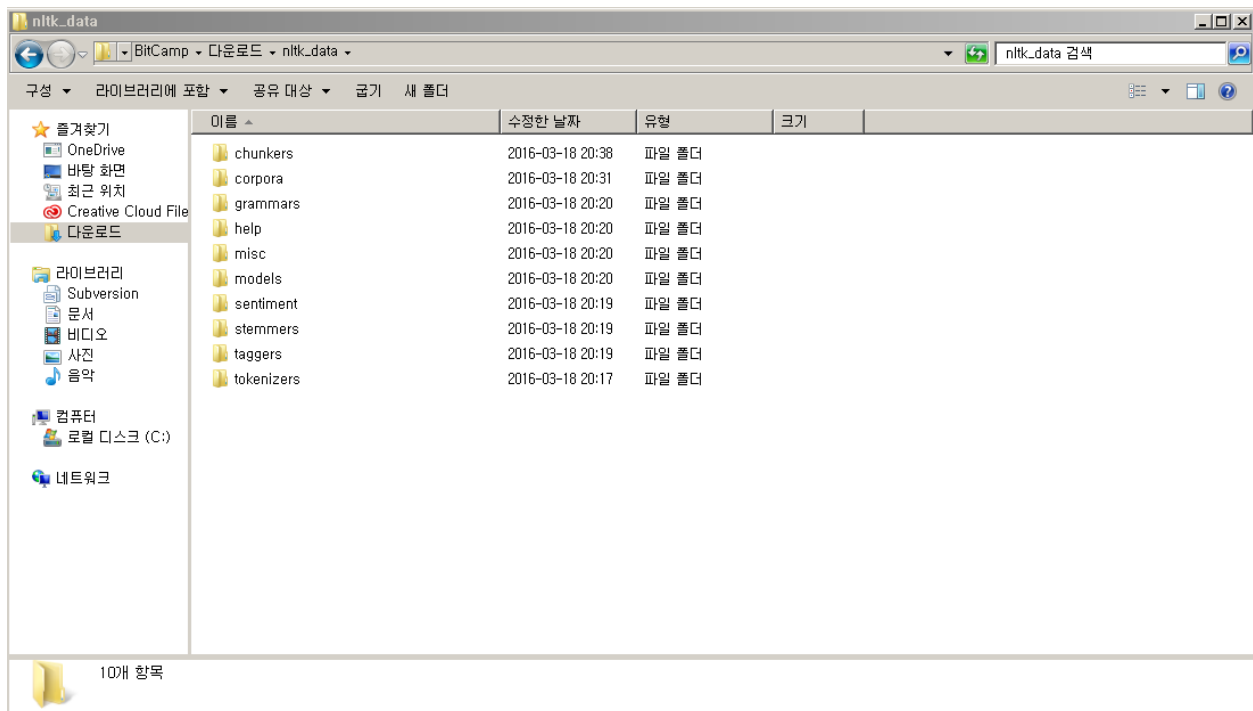
- `C:\nltk_data` 디렉터리 생성. 초기엔 `nltk_data` 폴더 안에 아무런 `data`도 들어 있지 않음.



- https://github.com/nltk/nltk_data/tree/gh-pages

에 접속하여 DownloadZIP 클릭

- 다운 받은 `nltk_data` zip 파일을 압축 해제



- NLTK를 임포트 한후 파이썬 코드를 짜고 실행
- C:\nltk_data에 특정 파일이 없다는 에러가 발생하면 다운 받은 nltk_data 중 이름이 같은 특정 파일을 찾아서 C:\nltk_data에 넣어줌
- 설치 완료

vii. KoNLPy 모듈 설치

l. <http://konlpy.readthedocs.org/ko/v0.4.3/install/#id2>

에 접속하여 지시하는대로 설치 수행.

- 가장 최신의 Java JDK 설치(v1.7 이상)
- Java JDK가 설치되면 환경변수->시스템 변수->새로만들기

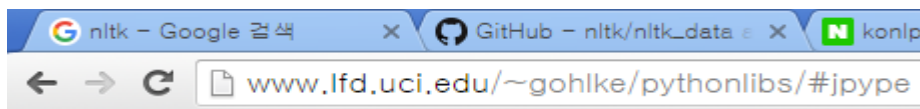
변수 : JAVA_HOME

값 : JDK 설치 경로(예: C:\Program Files\Java\jdk1.8.0_74)

- 최신 버전의 JPytype 설치(v0.5.7 이상)

<http://www.lfd.uci.edu/~gohlke/pythonlibs/#jpytype>

에 접속하여, JPytype1-0.6.1-cp34-none-win32.whl 또는
JPytype1-0.6.1-cp34-none-win32_amd64.whl 다운로드



JPytype allows full access to Java class libraries.

[JPytype1-0.6.1-cp27-none-win32.whl](#)

[JPytype1-0.6.1-cp27-none-win_amd64.whl](#)

[JPytype1-0.6.1-cp34-none-win32.whl](#)

[JPytype1-0.6.1-cp34-none-win_amd64.whl](#)

[JPytype1-0.6.1-cp35-none-win32.whl](#)

[JPytype1-0.6.1-cp35-none-win_amd64.whl](#)

명령 프롬프트에 pip install (다운로드파일경로)\JPytype파일이름).whl 수행

- 명령 프롬프트에 pip install konlpy 수행
- 설치 완료

B. 서버

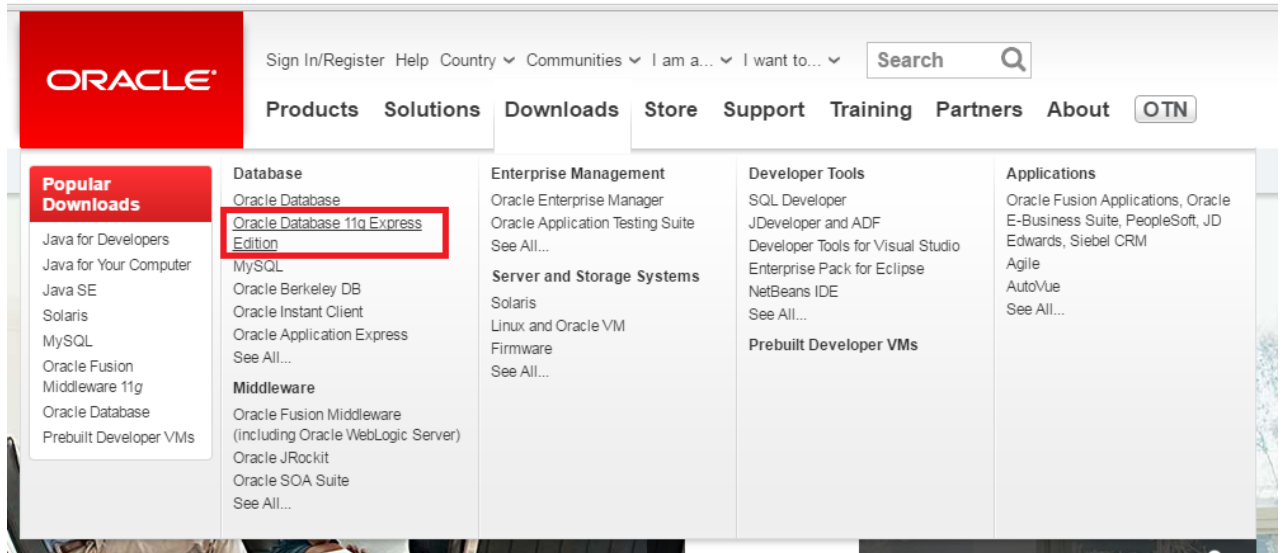
I. 오라클

i. 오라클 홈페이지 접속

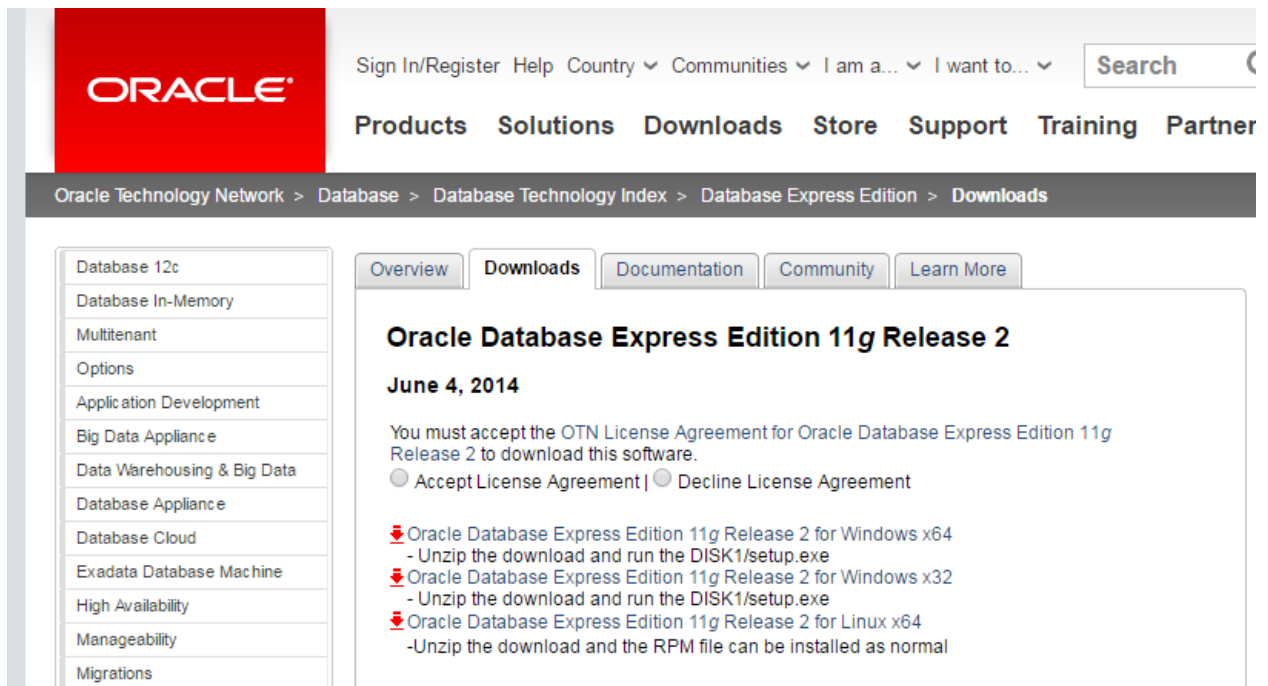
<http://www.oracle.com/index.html>





ii. Download의 Oracle Database 11g Express Edition 클릭



iii. Accept License Agreement 클릭 후 os에 맞는 오라클을 다운로드

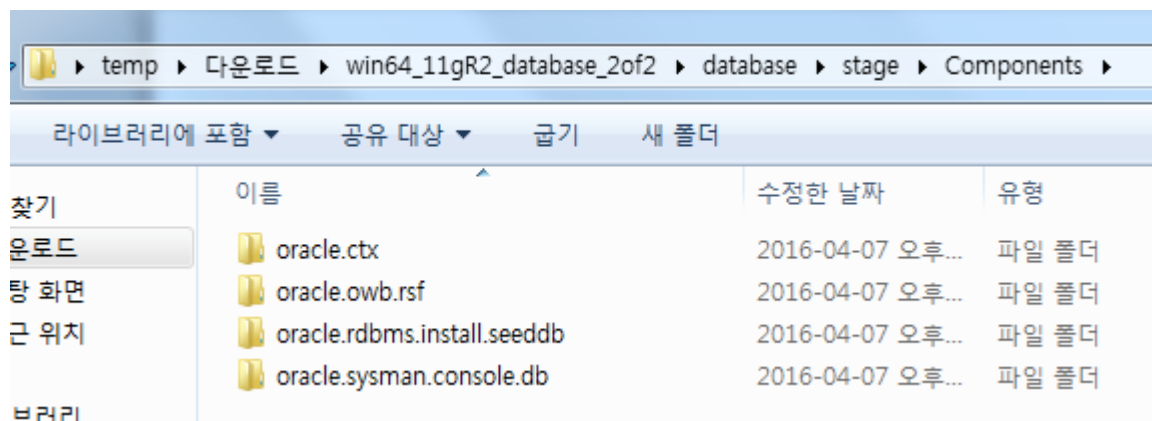


iv. 다운로드 한 파일의 압축을 각 각 푼다

	win64_11gR2_database_1of2.zip	2016-03-10 오후...	압축(ZIP) 파일	1,185,061...
	win64_11gR2_database_2of2.zip	2016-03-10 오후...	압축(ZIP) 파일	984,365KB

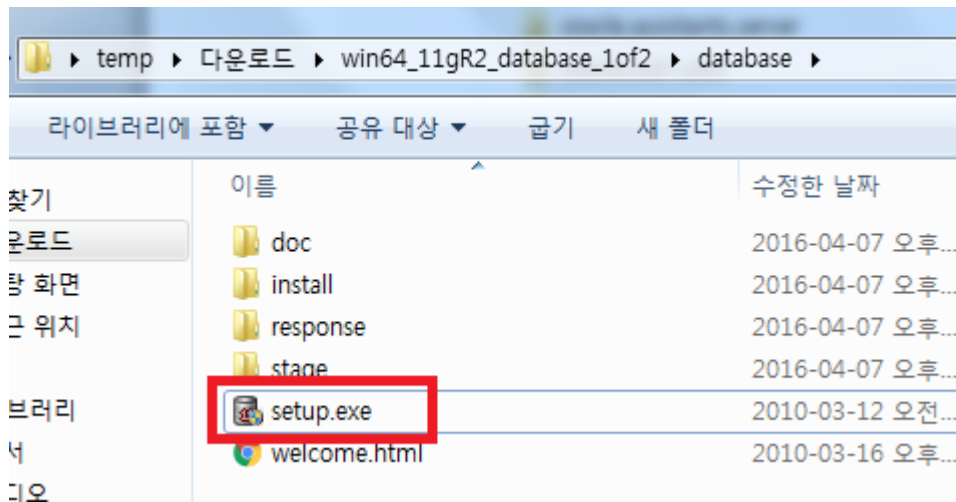
v. 압축을 각 각 툰 후 2번째 파일의 4개의 폴더 들을(첫번째 사진)

복사하여 첫 번째 폴더의 database -> stage -> components 폴더 아래에
붙여넣는다



temp > 다운로드 > win64_11gR2_database_1of2 > database > stage > Components >			
라이브러리에 포함 > 공유 대상 > 굽기 새 폴더			
	이름	수정한 날짜	유형
드	oracle.aspnet_2	2016-04-07 오후...	파일 폴더
사면	oracle.assistants.acf	2016-04-07 오후...	파일 폴더
사치	oracle.assistants.deconfig	2016-04-07 오후...	파일 폴더
	oracle.assistants.netca.client	2016-04-07 오후...	파일 폴더
리	oracle.assistants.oemlt	2016-04-07 오후...	파일 폴더
	oracle.assistants.server	2016-04-07 오후...	파일 폴더
	oracle.bali.cabo	2016-04-07 오후...	파일 폴더
	oracle.bali.dbui	2016-04-07 오후...	파일 폴더
	oracle.bali.dbui4	2016-04-07 오후...	파일 폴더
	oracle.bali.displayFonts	2016-04-07 오후...	파일 폴더
	oracle.bali.ewt	2016-04-07 오후...	파일 폴더
	oracle.bali.help	2016-04-07 오후...	파일 폴더
	oracle.bali.ice	2016-04-07 오후...	파일 폴더
스크 (C:)	oracle.bali.jewt	2016-04-07 오후...	파일 폴더
스크 (D:)	oracle.bali.jle	2016-04-07 오후...	파일 폴더
	oracle.bali.kodiak	2016-04-07 오후...	파일 폴더
	oracle.bali.ohw	2016-04-07 오후...	파일 폴더
	oracle.bali.regexp	2016-04-07 오후...	파일 폴더
	oracle.bali.share	2016-04-07 오후...	파일 폴더

vi. 그 후 첫번째 폴더의 database -> setup.exe 실행

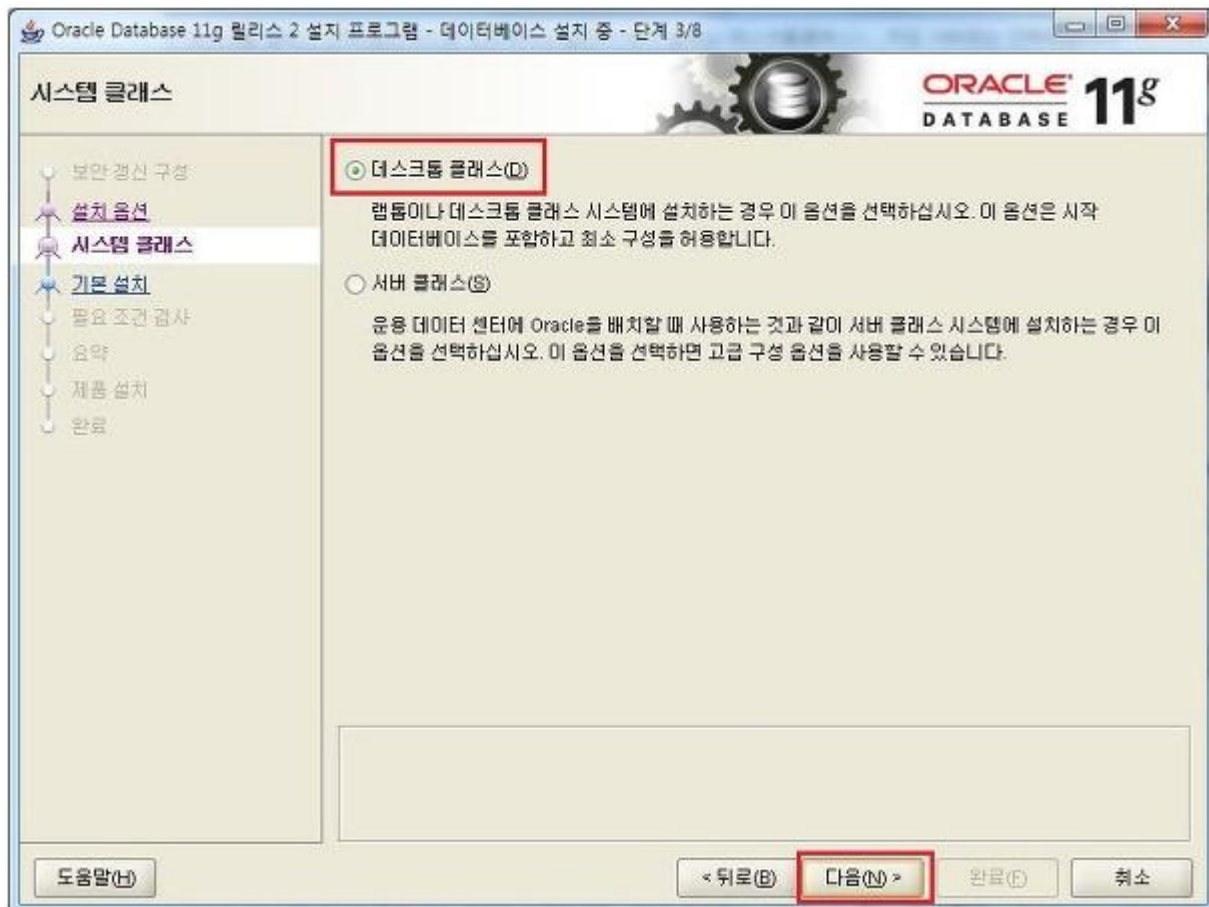


- vii. 빨간 박스의 체크 박스를 해제 한 후 다음 버튼 클릭, 경고 창이 나와도 무시한다.



viii. 빨간 박스를 순차적으로 클릭(첫 번째 사진, 두 번째 사진)

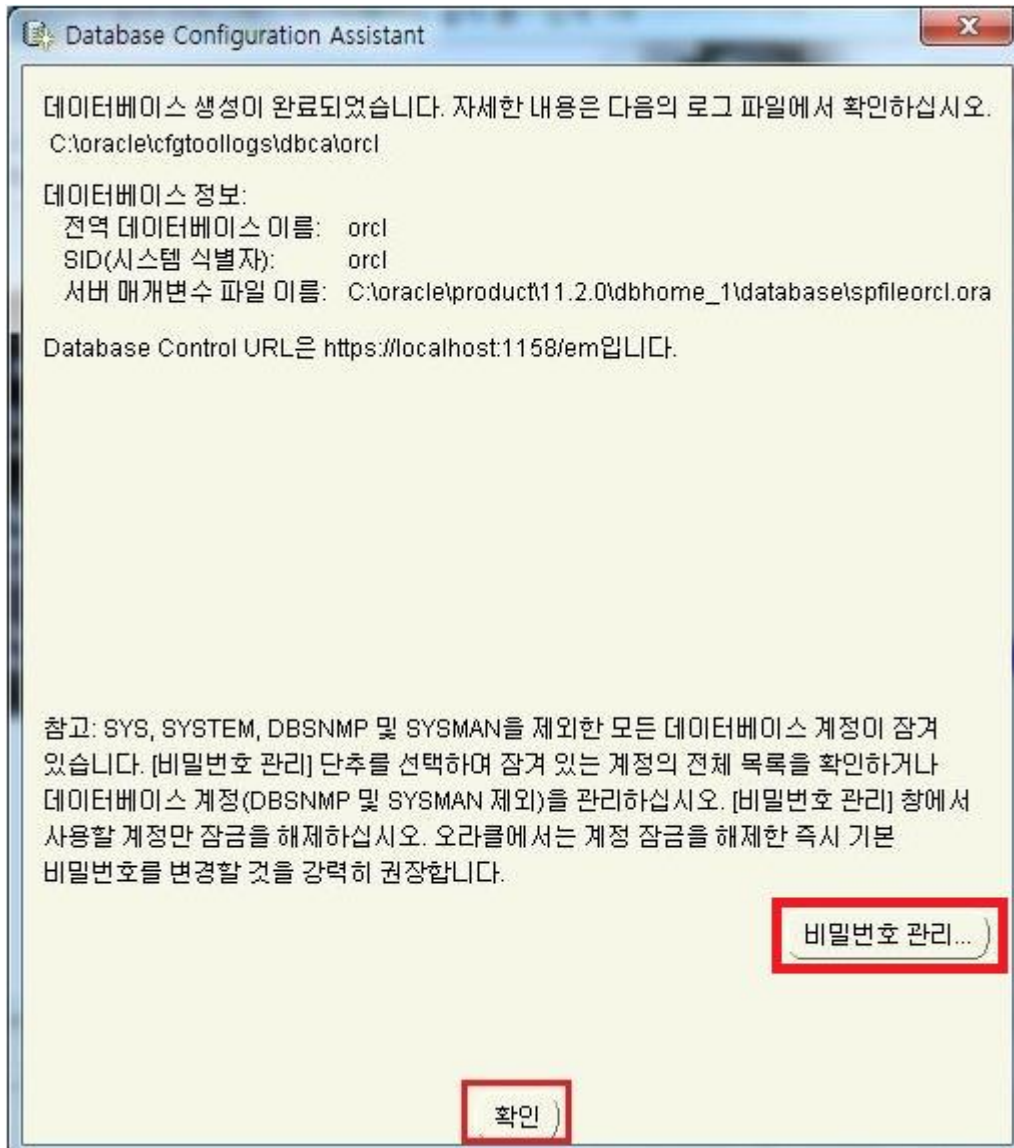




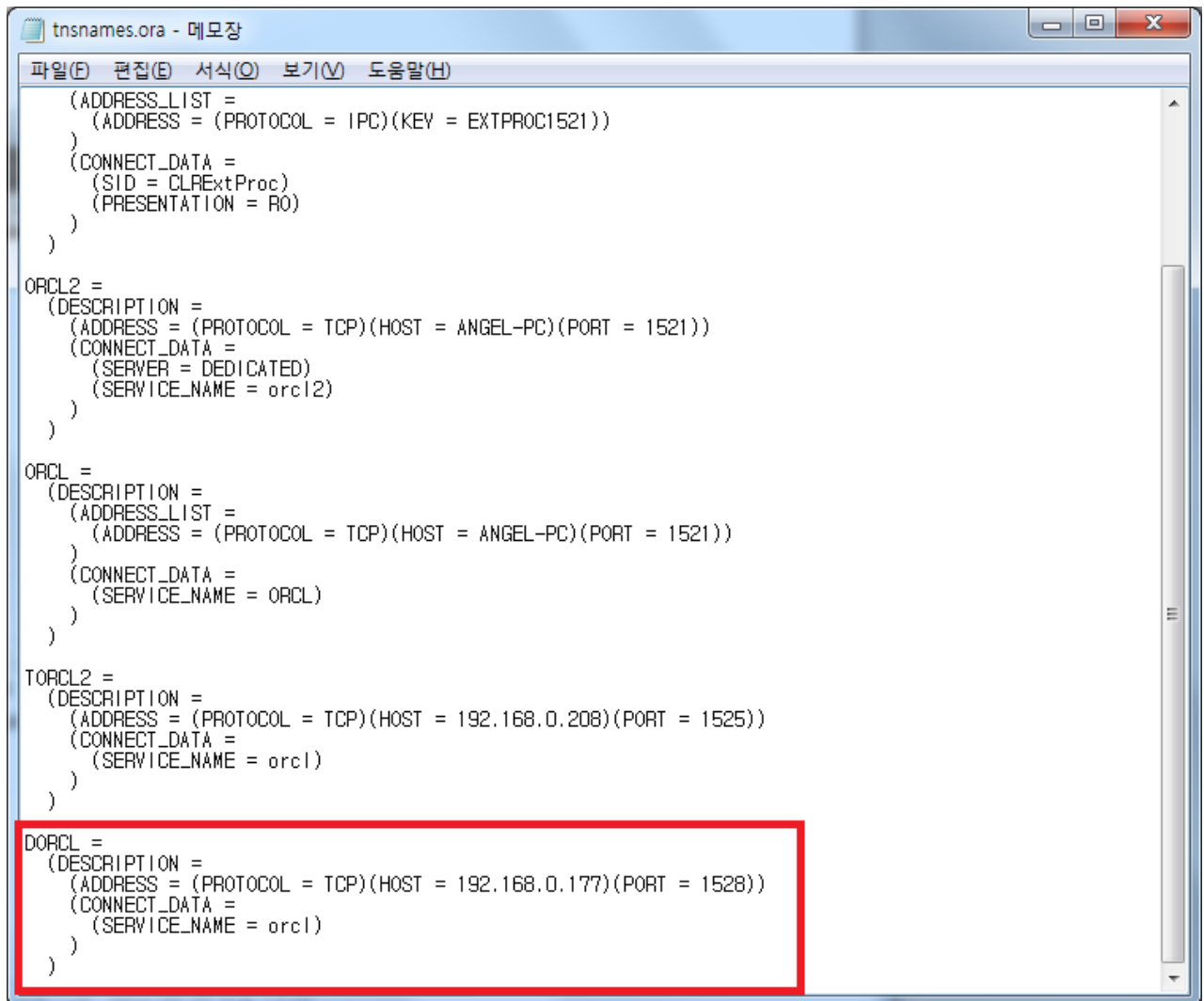
- ix. 설치할 위치를 설정, 관리 비밀번호를 적어준 후 다음 버튼 클릭



- x. 다음 버튼을 누르다 해당 화면이 나타나면 비밀번호 관리 버튼을 클릭 후 system, scott 계정의 계정 잠금을 선택 해제하고 각 비밀 번호를 적어준다



- xi. 외부 DB에 접속 할 경우 오라클을 설치 한 폴더의 product -> 11.2.0(버전) -> dbhome_1 -> NETWORK -> ADMIN -> tnsnames.ora 파일에 접속 할 데이터 베이스를 추가 한다.(DORCL은 임의로 선정 가능, 해당 데이터 베이스의 IP와 PORT를 적어주면 된다)



```
tnsnames.ora - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

(ADDRESS_LIST =
  (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
)
(CONNECT_DATA =
  (SID = CLRExtProc)
  (PRESENTATION = RO)
)
)

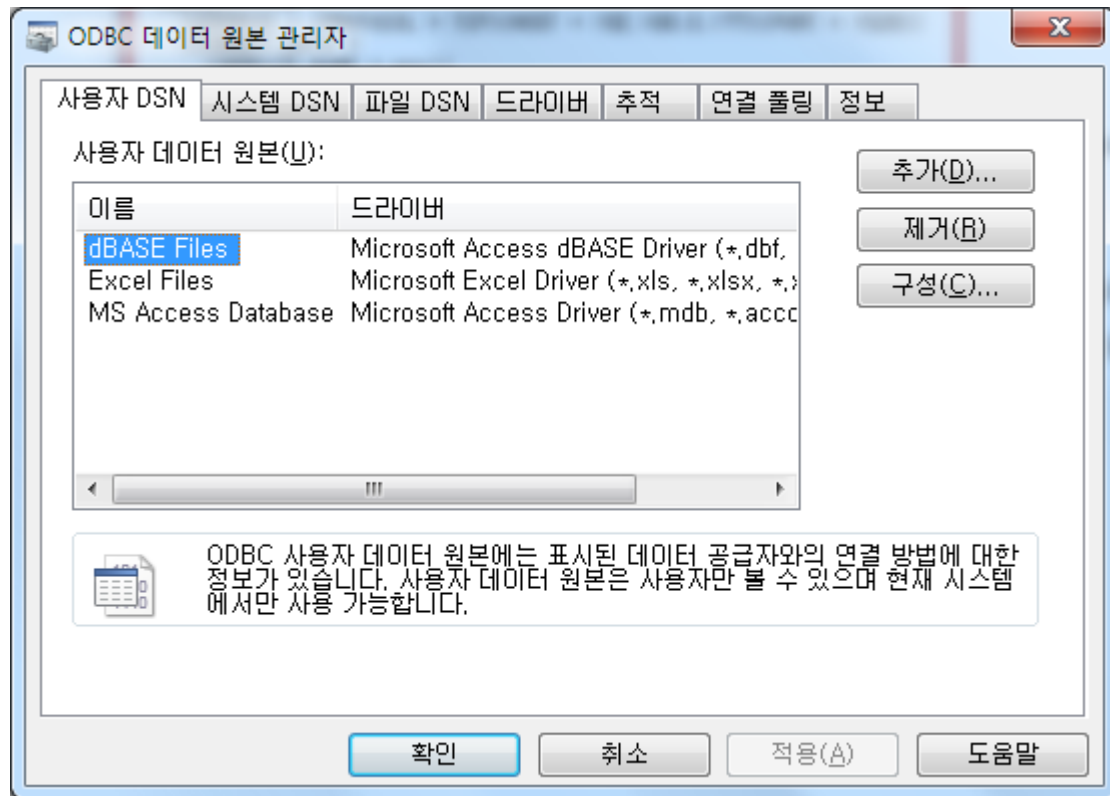
ORCL2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = ANGEL-PC)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl2)
    )
  )
)

ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = ANGEL-PC)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ORCL)
    )
  )
)

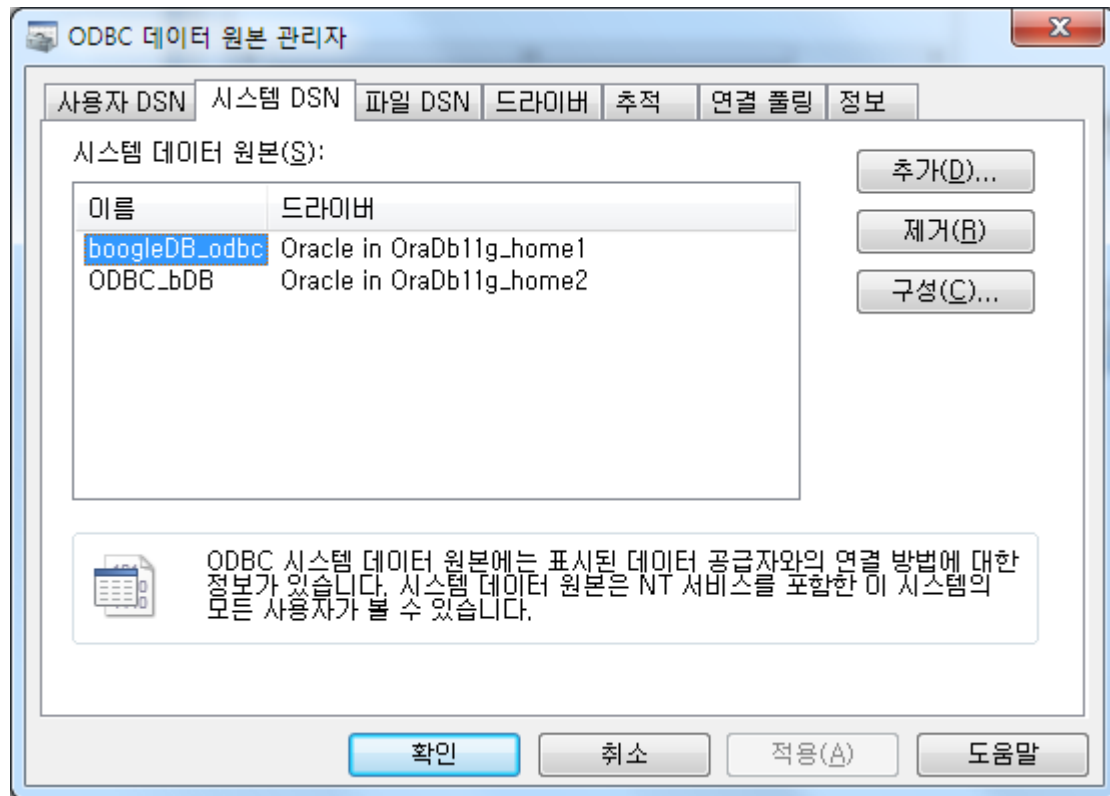
TORCL2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.0.208)(PORT = 1525))
    (CONNECT_DATA =
      (SERVICE_NAME = orcl)
    )
  )
)

DORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.0.177)(PORT = 1528))
    (CONNECT_DATA =
      (SERVICE_NAME = orcl)
    )
  )
)
```

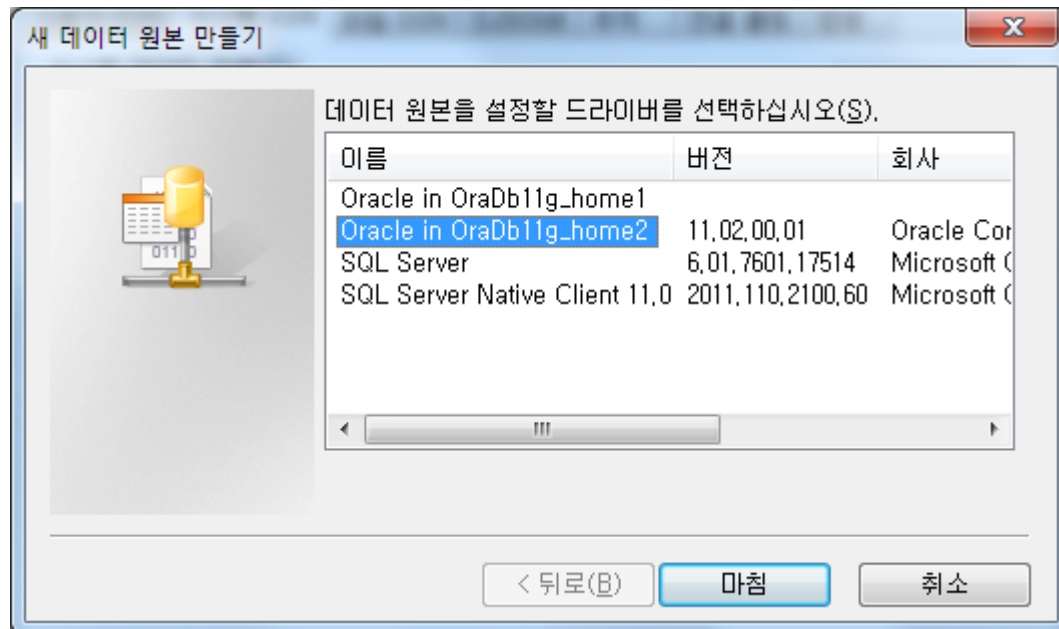
- xii. 오라클을 C++ 에서 사용하기 위해 데이터 원본에 오라클 드라이브를 등록해 준다. 제어판 -> 관리 도구 -> 데이터 원본(ODBC) 실행



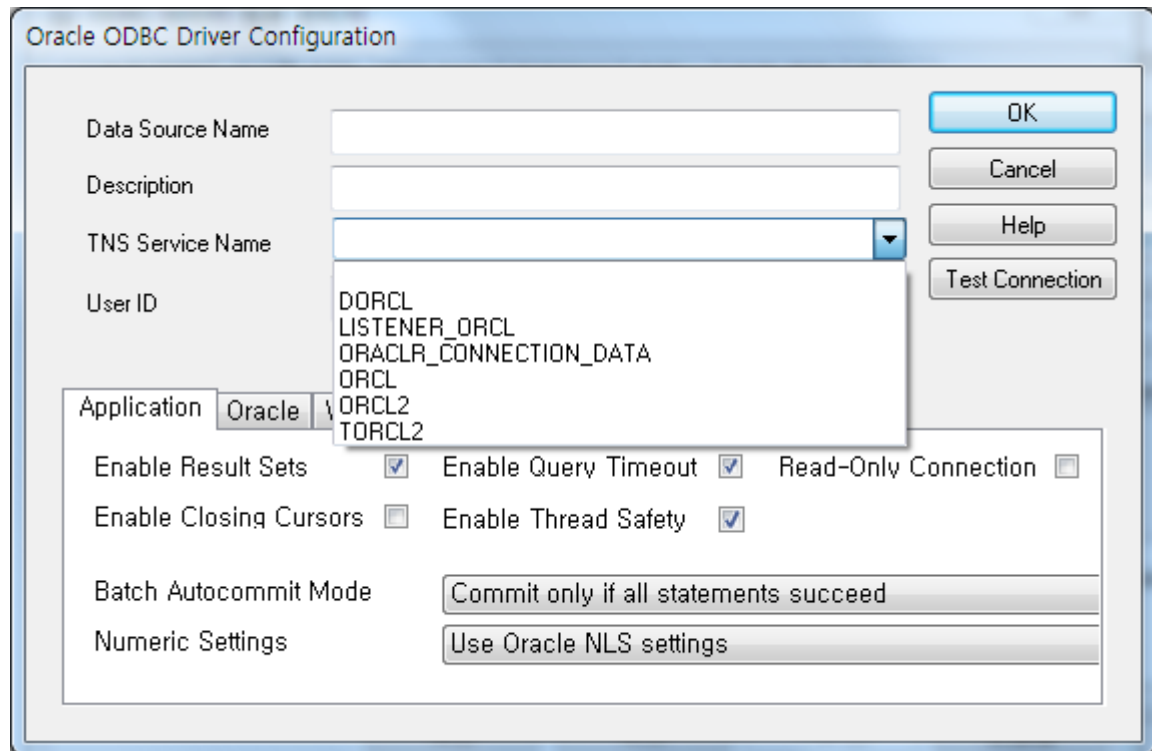
xiii. ODBC 데이터 원본 관리자의 시스템 DSN 클릭 후 추가 버튼 클릭



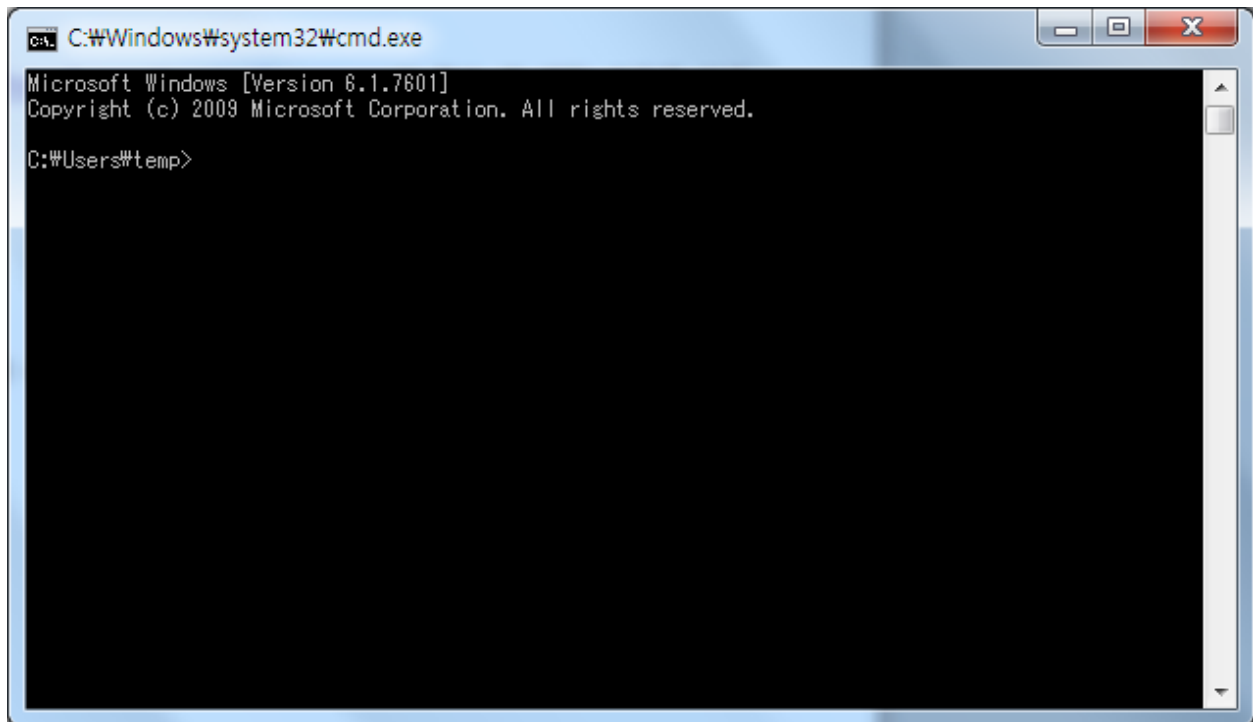
- xiv. Oracle in OraDb11g_home1 클릭 후 마침(2는 다시 설치하지 않는 한 생성되지 않는다)



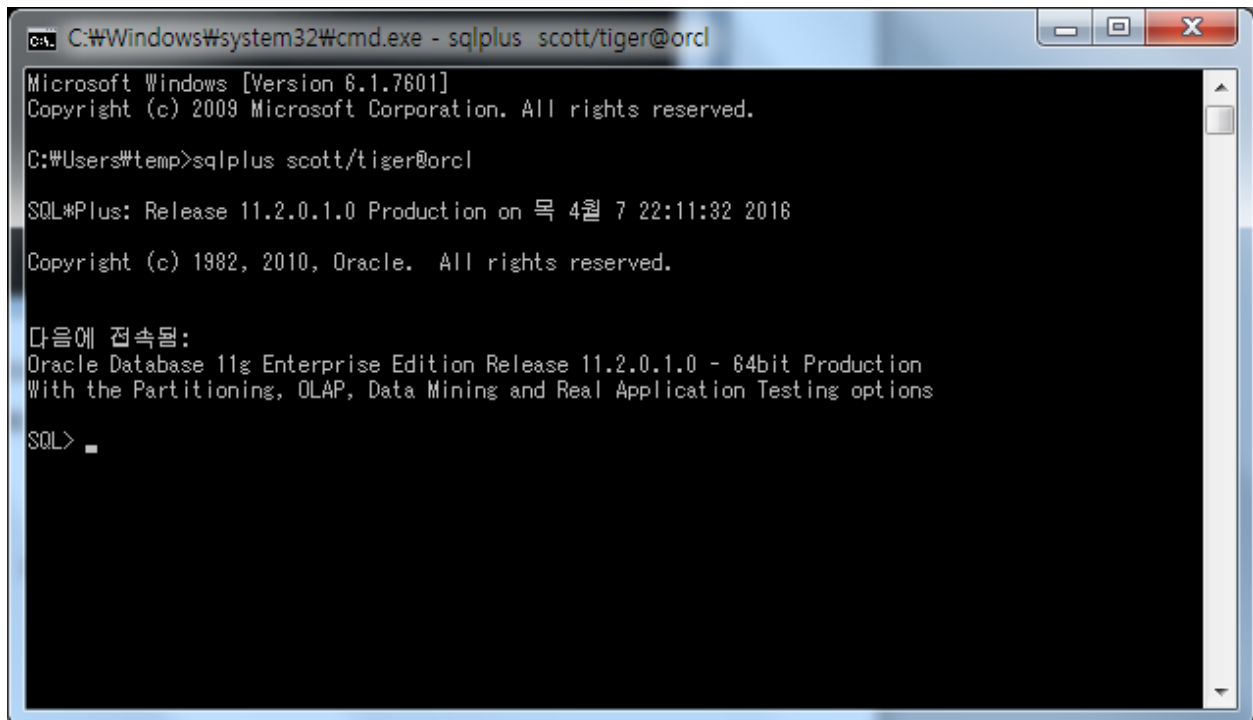
- xv. Data Source Name에 ODBC_원하는이름을 적어준다 (EX.ODBC_DB). TNS Service Name의 화살표 버튼을 눌러 연결하기 원하는 DB를 선택한다 (tnsnames.ora 파일 참고) 적어준 후 OK 버튼 클릭(32비트로 컴파일 할 경우 32비트 버전을 깔고 32비트 데이터 원본 관리자에 등록해 줘야 한다)



xvi. 실행 : 윈도우 키 + r 키를 눌러 cmd를 실행



- xvii. 실행 2 : Sqlplus soctt/tiger@orcl(orcl은 설치 당시 입력한 전역 데이터 베이스 이름, 기본적으로 orcl로 적혀 있다) 입력 후 엔터를 입력하면 데이터 베이스를 실행 할 수 있다



```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger@orcl
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\temp>sqlplus scott/tiger@orcl

SQL*Plus: Release 11.2.0.1.0 Production on 목 4월 7 22:11:32 2016

Copyright (c) 1982, 2010, Oracle. All rights reserved.

다음에 접속됨:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
```

B. ACE library

- i. <http://download.dre.vanderbilt.edu/> 에서 ace 다운로드
- ii. 압축 해제
- iii. 압축 해제 한 폴더 내의 ACE_wrappers -> ace 폴더에 config.h 파일을 생성한다.
- iv. 생성 한 config.h 폴더에 아래 내용 추가 후 저장

```
#define ACE_HAS_STANDARD_CPP_LIBRARY 1

#include "config-win32.h"

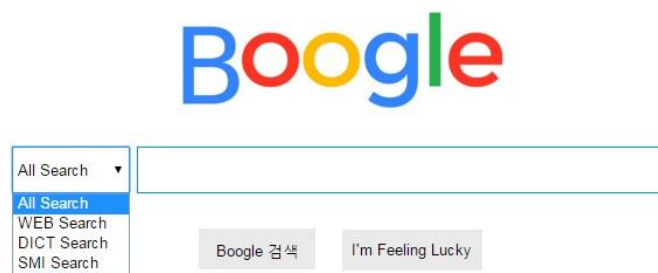
#include "config-win32-msvc.h"
```
- v. 압축 해제 한 폴더 내의 ACE_wrappers 의 ACE_wrappers_vc_14.sln 폴더를 실행 후 컴파일 한다.
- vi. Visual studio에 프로젝트 속성의 C/C++ -> 일반 -> 추가 포함 디렉터리에 다운로드 받은 ACE 폴더내의 ACE_wrappers -> ace 위치를 추가
- vii. Visual studio에 프로젝트 속성의 링커 -> 일반 -> 추가 라이브러리 디렉터리에 다운로드 받은 ACE 폴더내의 ACE_wrappers -> lib 위치를 추가
- viii. Visual studio에 프로젝트 속성의 링커 -> 입력 -> 추가 종속성에 ACE.lib 와 ACEd.lib를 추가한다
- ix. 소스코드를 작성 해 디버깅 한 후 실행 파일이 위치한 곳에 ACE_wrappers -> ace -> bin의 ACE.lib 파일 ACEd.lib 파일을 복사 -> 붙여 넣기 해준다

C. Chilkat 라이브러리

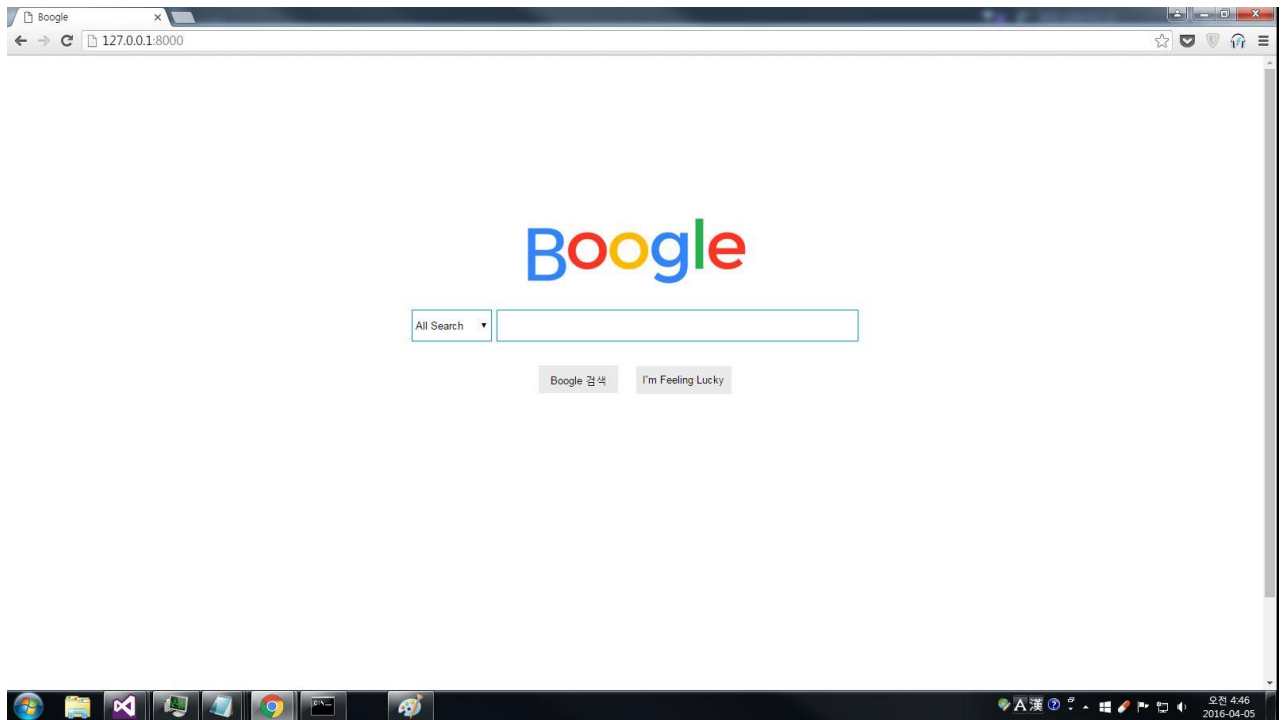
- i. https://www.chilkatsoft.com/downloads_vcpp.asp에 접속 후 해당 파일을 다운로드 받는다.
- ii. Visual studio에 프로젝트 속성의 C/C++ -> 일반 -> 추가 포함 디렉터리에 다운로드 받은 chilkat 폴더내의 include 위치를 추가
- iii. Visual studio에 프로젝트 속성의 링커 -> 일반 -> 추가 라이브러리 디렉터리에 다운로드 받은 chilkat 폴더내의 libs 위치를 추가
- iv. Visual studio에 프로젝트 속성의 링커 -> 입력 -> 추가 종속성에 ChilkatDbgDll.lib 를 추가한다

6. 사용자 매뉴얼

A. 웹 서버



- 검색어 입력란 옆에 검색 테마 설정



- 검색어 입력
- 엔터키 입력 혹은 Boogle 검색 버튼 클릭

7. 시행 착오 및 개선 방안

A. 크롤러

i. 모듈 설치 오류 - 버전 오류

모듈이 파이썬에서 올바르게 작동하지 않는 오류가 발생. 'module' has no attribute '(함수 이름)'.

해결방안 : 파이썬의 버전과 모듈 버전 사이의 호환성에 대한 문서를 읽고 알맞은 버전의 모듈을 선택해야 한다.

ii. 모듈 설치 오류 - 설치 방법1

Pip을 이용해 모듈을 설치 할 수 없음.

해결방안 : 설치하려는 모듈의 배포 형태에의해 pip이 모듈을 읽을 수 없는 경우가 발생한다. 이때, pip의 대안으로 easy_install을 사용한다. 이 프로젝트에서 사용하는 한 사용 방법은 같다.

iii. 모듈 설치 오류 - 설치 방법2

올바른 설치 파일을 가지고 pip install을 이용했는데 install이 불가능한 형태의 파일이라는 에러메세지가 출력됨.

해결방안 : 자체 배포 가능 패키지인지 확인한다. 소스 파일 구성 중 setup.py를 찾는다. 명령 프롬프트를 이용하여 2단계에 걸쳐 설치를 수행 할 수 있다.

1단계 - setup.py build

2단계 - setup.py install

iv. 파일 압축 해제 오류

tar.gz 확장자의 압축 해제가 불가함.

해결방안 : 반디집을 이용하면 tar.gz 확장자 압축파일도 해제 할 수 있다.

v. Init_VM 오류

VM을 초기화 할 수 없다는 오류가 발생하고 786432KB의 메모리 공간이 필요하다는 메시지와 함께 프로그램이 중단됨.

해결방안 : KoNLPy와 NLTK의 경우 자바 코드로 작성되어 있다. 때문에 자바 가상 머신을 이용해 파이썬 모듈 형태로 변형되는데, 자바 가상 머신이 작동하기 위해서는 초기화 과정이 필요하다.

이런 오류가 발생하는 경우 (내컴퓨터 속성 -> 고급 시스템 설정 -> 고급 탭 -> 성능란의 설정 -> 고급 탭 하단의 변경 -> 사용자 지정 크기) 경로에서 메모리 공간을 더 크게 할당 함으로써 해결 할 수 있다.

B. 서버

i. 전반적인 시행 착오

(가) 라이브러리가 충돌 해 같이 사용하지 못했음

(나) 자료 부족으로 인한 구현의 어려움

(다) 오라클 리스너의 정보 부족으로 인한 연동의 어려움

ii. 개선 방안

(가) 게이트 웨이 : 접속 한 클라이언트의 ip를 검사해 데이터 pass / non pass 를 구현
하지 못함

(나) 각 서버들간의 통신 기록 저장을 위한 로그 서버 구축

8. 참고 문헌 및 논문

i. 참고 문헌

- 빠르게 활용하는 파이썬 3.2 프로그래밍 | 신호철, 이상정, 최동진 | 위키북스
- 루씬 인 액션 | 오티스 고스포트네티치 저 | 강철구 역 | 에이콘출판
- ACE 프로그래머 가이드 | Stephen D. Huston 외 2명 저 | 권태인 역 | 인포북
- 열혈 TCP/IP 소켓 프로그래밍 | 윤성우 | 오렌지 미디어
- 게임 서버 프로그래밍 입문 | 김동성 | 퍼플

ii. 참고 사이트

- <https://www.youtube.com/watch?v=SFas42HBtMg>
WebCrawler 만들기
- <http://www.nextree.co.kr/p4327/>
정규표현식 문법
- <http://blog.daum.net/creazier/15309380>
정규표현식 기본 문법 정리표
- <http://sens.tistory.com/454>
NLTK 영어 형태소 리스트
- <http://markov.tistory.com/14>
구글 검색 알고리즘의 원리

- <http://edoli.tistory.com/46>
파이썬2과 파이썬3에 포함된 urllib 모듈 사용법의 차이
- http://tutorial.djangogirls.org/ko/django_start_project/index.html
장고로 웹 서버 만들기
- http://www.softlab365.com/wordpress/?page_id=491
ACE 소켓 통신 TCP/IP 서버와 클라이언트 소스