웹프로그래밍(게시판)

- **주요 기능**
  1) 글목록, 끌쓰기, 수정,삭제
  2) 검색기능
  3) 페이지 나누기
  4) 댓글 쓰기, 댓글 목록
  5) 파일 업로드, 다운로드

DB환경설정

- mysql 설치
- 계정생성
- 접속자만들기
- 데이터베이스 생성

```
create database pyweb charset:utf-8 설정
```

- 파이썬 mysql 패키지 설치

```
pip install mysqlclient
```

프로젝트 만들기

- **프로젝트 생성**
  - New -> Other -> PyDev Django Project
  - 프로젝트 명 : pyweb_board
  - pyweb_board디렉토리가 2개 만들어짐
  - c:₩python₩webwork₩pyweb_board
  - c:₩python₩webwork₩pyweb_board₩pyweb_board
    - python 웹프로젝트의 설정 디렉토리

기본 테이블 생성

```
cd c:₩python₩webwork₩pyweb_board
python manage.py migrate
```

슈퍼 유저 생성

```
python manage.py createsuperuser
```

username : admin
password : admin1234(비밀번호 8자 이상 영문+숫자로 구성)

애플리케이션 생성

```
python manage.py startapp board
```

eclipse에서 F5키를 누르면 board 생성

settings.py 날짜 포맷 변경

```
import os
#날짜 포맷 변경을 위한 모듈 로딩
from django.conf.locale.ko import formats as ko_formats
#날짜 포맷 설정
ko_formats.DATETIME_FORMAT = 'Y-m-d G:i:s'

import MySQLdb
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'board',
]
```

- **settings.py**

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

settings.py 데이터베이스 설정

```
# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'pyweb', # DB명
        'USER': 'pgm', # 데이터베이스 계정
        'PASSWORD': '1234', # 계정 비밀번호
        'HOST': 'localhost', # DB서버 주소(IP)
        'PORT': '3306', # 데이터베이스 포트(보통은 3306)
    }
}
```

```
LANGUAGE_CODE = 'ko'

TIME_ZONE = 'Asia/Seoul'
```

테이블 작성(models.py)-게시판

```
from django.db import models
from datetime import datetime

class Board(models.Model):
    idx=models.AutoField(primary_key=True)
    writer=models.CharField(null=False, max_length=50)
    title=models.CharField(null=False, max_length=120)
    hit=models.IntegerField(default=0)
    content=models.TextField(null=False)
    post_date=models.DateTimeField(default=datetime.now, blank=True)
    filename=models.CharField(null=True,blank=True, default="", max_length=500)
    filesize=models.IntegerField(default=0)
    down=models.IntegerField(default=0)

    def hit_up(self):
        self.hit += 1
    def down_up(self):
        self.down += 1
```

댓글 테이블 작성(models.py)

```
#댓글 테이블
class Comment(models.Model):
    #댓글 글번호
    idx=models.AutoField(primary_key=True)
    #게시물 번호
    board_idx=models.IntegerField(null=False)
    writer=models.CharField(null=False,max_length=50)
    content=models.TextField(null=False)
    post_date=models.DateTimeField(default=datetime.now,blank=True)
```

Admin 사이트에 테이블 반영(admin.py)

```
from django.contrib import admin
from board.models import Board

# Register your models here.
class BoardAdmin(admin.ModelAdmin):
    list_display=("writer","title","content")

admin.site.register(Board, BoardAdmin)
```

테이터베이스 변경 사항 반영

```
python manage.py makemigrations
python manage.py migrate
```

서버 실행

```
python manage.py runserver localhost:80
```

list 페이지 만들기

- urls.py->urlpattern 수정

```
urlpatterns = [
    #관리자용 사이트
    path('admin', admin.site.urls),

    #게시판 관련 url
    path('list/', board/views.list),

]
```

list 페이지 작성

- views.py 작성

```
def list(request):
        boardCount=Board.objects.count()
        boardList=Board.objects.all().order_by("-idx")
        return render(request,"board/ist.html",
            {"boardList":boardList,  "boardCount":boardCount})
```

- list.html 작성

글쓰기

- urls.py

```
urlpatterns = [
    #관리자용 사이트
    path('admin', admin.site.urls),
    #게시판 관련 url
    path('list/', views.list),
    path('write/', views.write),
]
```

- views.py

```
def write(request):
    return render(request, "board/write.html")
```

- write.html 작성

글 저장

- urls.py

```
from django.contrib import admin
from django.urls import path
from board import views


urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/',views.list),
    path('write/',views.write),
    path('insert/',views.insert),
]
```

글저장(views.py)

```
UPLOAD_DIR= 'c:/Temp/'
@csrf_exempt                    #Cross Site Request Forger
def insert(request):
    fname= ''
    fsize=0
    if 'file' in request.FILES:
        file=request.FILES['file']
        fname=file.name
        fsize=file.size

        fp=open('%s%s' %(UPLOAD_DIR,fname),'wb')
        for chunk in file.chunks():
            fp.write(chunk)
        fp.close()

    dto=Board(writer=request.POST['writer'], title=request.POST['title'],
              content=request.POST['content'], filename=fname, filesize=fsize)
    dto.save()
    print(dto)
    return redirect('/list/')
```

이미지 표시하기

- Settings.py: 수정

```
STATIC_URL = '/board/static/'
```

- static 폴더 작성

```
'/pyweb_board/board/static/'
```

- Images 폴더 작성

```
'/pyweb_board/board/static/images'
```

  - 표시할 이미지 images 폴더에 복사 html 파일에 다음 내용 추가

```
{% load static %}
<!DOCTYPE html>
...
<img src="{% static "images/file.gif" %}"></a>
```

파일다운로드

- urls.py

```
urlpatterns = [
    #관리자용 사이트
    path('admin', admin.site.urls),
    #게시판 관련 url
    path('list/', views.list),
    path('write/', views.write),
    path('insert/', views.insert),
    path("download/", views.download),
]
```

파일 다운로드(views.py)

```
Import os

def download(request):
    id=request.GET['idx']
    dto=Board.objects.get(idx=id)
    path=UPLOAD_DIR+dto.filename
    filename=os.path.basename(path)
    filename=urlquote(filename)
    with open(path, 'rb') as file:
        response=HttpResponse(file.read(), content_type='application/actet-stream')
        response['Content-Disposition']="attachment;filename*=UTF-8''{0}".format(filename)
        dto.down_up()
        dto.save()
        return response
```

상세보기(urls.py)

```
urlpatterns = [
    #관리자용 사이트
    path('admin', admin.site.urls),
    #게시판 관련 url
    path('', views.list),
    path('write', views.write),
    path('insert', views.insert),
    path('detail', views.detail),
    path("download", views.download),
```

상세보기(views.py)

```
def detail(request):
    #조회수 증가 처리
    id=request.GET["idx"]
    dto=Board.objects.get(idx=id)
    dto.hit_up()
    dto.save()

    print("filesize:",dto.filesize)
    #filesize = "%0.2f" % (dto.filesize / 1024)
    filesize = "%.2f" % (dto.filesize)
    return render(request, "detail.html", {"dto": dto, "filesize":filesize, })
```

- detail.html작성

댓글쓰기(urls.py)

```
urlpatterns = [
    #관리자용 사이트
    path('admin', admin.site.urls),
    #게시판 관련 url
    path('', views.list),
    path('write', views.write),
    path('insert', views.insert),
    path('detail', views.detail),
    path("download", views.download),
    path("reply_insert", views.reply_insert)
```

댓글쓰기(views.py)

```
@csrf_exempt
def reply_insert(request):
    id=request.POST["idx"] #게시물 번호
    #댓글 객체 생성
    dto = Comment(board_idx=id,
            writer=request.POST["writer"],content=request.POST["content"])
    #insert query 실행
    dto.save()
    # detail?idx=글번호 페이지로 이동
    return HttpResponseRedirect("detail?idx="+id)
```

detail.html 에 댓글 폼 작성

댓글 목록 가져오기

```
def detail(request):
    #조회수 증가 처리
    id=request.GET["idx"]
    dto=Board.objects.get(idx=id)
    dto.hit_up()
    dto.save()

    commentList = Comment.objects.filter(board_idx = id).order_by("idx")

    print("filesize:",dto.filesize)
    #filesize = "%0.2f" % (dto.filesize / 1024)
    filesize = "%.2f" % (dto.filesize)
    return render(request,"detail.html",
            {"dto": dto, "filesize":filesize, "commentList":commentList})
```

detail.html 에 댓글 폼 아래에 댓글 목록 출력 작성

수정, 삭제(urls.py)

```
urlpatterns = [
    #관리자용 사이트
    path('admin', admin.site.urls),
    #게시판 관련 url
    path('', views.list),
    path('write', views.write),
    path('insert', views.insert),
    path('detail', views.detail),
    path('update', views.update),
    path('delete', views.delete),
    path("download", views.download),
    path("reply_insert", views.reply_insert)
```

수정(views.py)

```
@csrf_exempt
def update(request):
    id=request.POST["idx"] #글번호
    #select * from board_board where idx=id
    dto_src=Board.objects.get(idx=id)
    fname=dto_src.filename #기존 첨부파일 이름
    fsize=0 #기존 첨부파일 크기
    if "file" in request.FILES: #새로운 첨부파일이 있으면
        file=request.FILES["file"]
        fname=file._name #새로운 첨부파일의 이름
        fp = open("%s%s" % (UPLOAD_DIR, fname), "wb")
        for chunk in file.chunks():
            fp.write(chunk) #파일 저장
        fp.close()
        #첨부파일의 크기(업로드완료 후 계산
        fsize=os.path.getsize(UPLOAD_DIR+fname)
    #수정 후 board의 내용
    dto_new = Board(idx=id, writer=request.POST["writer"], title=request.POST["title"],
                    content=request.POST["content"], filename=fname, filesize=fsize)
    dto_new.save() #update query 호출
    return redirect("/") #시작 페이지로 이동
```

삭제(views.py)

```
@csrf_exempt
def delete(request):
    id=request.POST["idx"] #삭제할 게시물의 번호
    Board.objects.get(idx=id).delete() #레코드 삭제
    return redirect("/list/") #시작 페이지로 이동
```

페이지 나누기 및 검색- 검색폼

```
<form method="post">
                        {% csrf_token %} <!-- 크로스사이트스크립팅 공격방지코드 -->
                        <select name="search_option">
{% if search_option == "writer" %}
                                <option value="writer" selected>이름</option>
                                <option value="title">제목</option>
                                <option value="content">내용</option>
                                <option value="all">이름+제목+내용</option>
{% elif search_option == "title" %}
                                <option value="writer">이름</option>
                                <option value="title" selected>제목</option>
                                <option value="content">내용</option>
                                <option value="all">이름+제목+내용</option>
{% elif search_option == "content" %}
                                <option value="writer">이름</option>
                                <option value="title">제목</option>
                                <option value="content" selected>내용</option>
                                <option value="all">이름+제목+내용</option>
{% elif search_option == "all" %}
                                <option value="writer">이름</option>
                                <option value="title">제목</option>
                                <option value="content">내용</option>
                                <option value="all" selected>이름+제목+내용</option>
{% endif %}
                        </select>
                        <input type="text" name="search" value="{{search}}">
                        <input type="submit" value="검색">
            </form>
```

페이지 나누기 및 검색 - 페이지

```
{% if start_page >= page_list_size %} <!--  [이전] 표시 -->
  <a href="?start={{prev_list}}">[이전]</a>
{% endif %}

{% autoescape off %}
{% for link in links %} <!-- 페이지 링크 표시 -->
            {{link}}
{% endfor %}
{% endautoescape %}

{% if total_page > end_page %} <!-- [다음] 표시 -->
  <a href="?start={{next_list}}">[다음]</a>
{% endif %}
```

페이징 및 검색(views.py)

```
@csrf_exempt
def list(request):
    #검색옵션, 검색값
    try: #예외가 발생할 가능성이 있는 코드
        search_option = request.POST["search_option"]
    except: #예외가 발생했을 때의 코드
        search_option = "writer"

    try:
        search= request.POST["search"]
    except:
        search = ""

    print("search_option:",search_option)
    print("search:",search)
```

페이징 및 검색(views.py)

```
#레코드 갯수 계산
# 필드명__contains=값 => where 필드명 like '%값%'
# count() => select count(*)
    if search_option == "all": #이름+제목+내용
        boardCount=Board.objects.filter(₩
            Q(writer__contains=search) | Q(title__contains=search)|₩
            Q(content__contains=search)).count()
    elif search_option == "writer": #이름
        boardCount = Board.objects.filter(writer__contains=search)).count()
    elif search_option == "title": #제목
        boardCount = Board.objects.filter(title__contains=search)).count()
    elif search_option == "content": #내용
        boardCount = Board.objects.filter(content__contains=search)).count()
```

```
# limit start,레코드갯수
    try:
        start = int(request.GET['start'])
    except:
        start = 0

    page_size = 10; # 페이지당 게시물수
    page_list_size = 10; # 한 화면에 표시할 페이지의 갯수
    end=start+page_size
    #전체 페이지 갯수 , math.ceil() 올림함수
    total_page = math.ceil(boardCount / page_size)
    #start 레코드시작번호 => 페이지번호
    current_page = math.ceil( (start+1) / page_size )
    #페이지 블록의 시작번호, math.floor() 버림함수
    start_page = math.floor((current_page - 1) / page_list_size ) * page_list_size+ 1;
    #페이지 블록의 끝번호
    end_page = start_page + page_list_size - 1;
    #마지막 페이지가 범위를 초과하지 않도록 처리
    if total_page < end_page:
        end_page = total_page
```

...계속

```
...계속
# 1 ~ 10
   [이전] 11 ~ 20
   if start_page >= page_list_size:
       prev_list = (start_page - 2) * page_size;
   else:
       prev_list = 0
   # 91 ~ 100 ,
   # 81 ~ 90 [다음]
   if total_page > end_page:
       next_list = end_page * page_size
   else:
       next_list = 0
```

```
#레코드 내용
# 리스트[start:end] => start~end-1
    if search_option=="all":
        boardList = Board.objects.filter(₩
            Q(writer__contains=search) | Q(title__contains=search) |
            Q(content__contains=search)).order_by("-idx")[start:end]
    elif search_option=="writer":
        boardList = Board.objects.filter(writer__contains=search).order_by("-idx")[start:end]
    elif search_option=="title":
        boardList = Board.objects.filter(title__contains=search).order_by("-idx")[start:end]
    elif search_option=="content":
        boardList = Board.objects.filter(content__contains=search).order_by("-idx")[start:end]

print("start_page:",start_page)
    print("end_page:",end_page)
    print("page_list_size:",page_list_size)
    print("total_page:",total_page)
    print("prev_list:",prev_list)
    print("next_list:",next_list)
```

```
links=[]
# range(start_page,end_page+1) start_page~end_page
# str(숫자변수) => 숫자변수를 스트링변수로
for i in range(start_page,end_page+1):
    page = (i - 1) * page_size
    links.append("<a href='?start="+str(page)+"'>"+str(i)+"</a>")

return render(request,"list.html",{"boardList": boardList, "boardCount": boardCount,
        "search_option": search_option,
        "search": search,
        "range":range(start_page-1,end_page),
        "start_page":start_page,
        "end_page":end_page,
        "page_list_size":page_list_size,
        "total_page":total_page,
        "prev_list":prev_list,
        "next_list":next_list,
        "links":links})
```