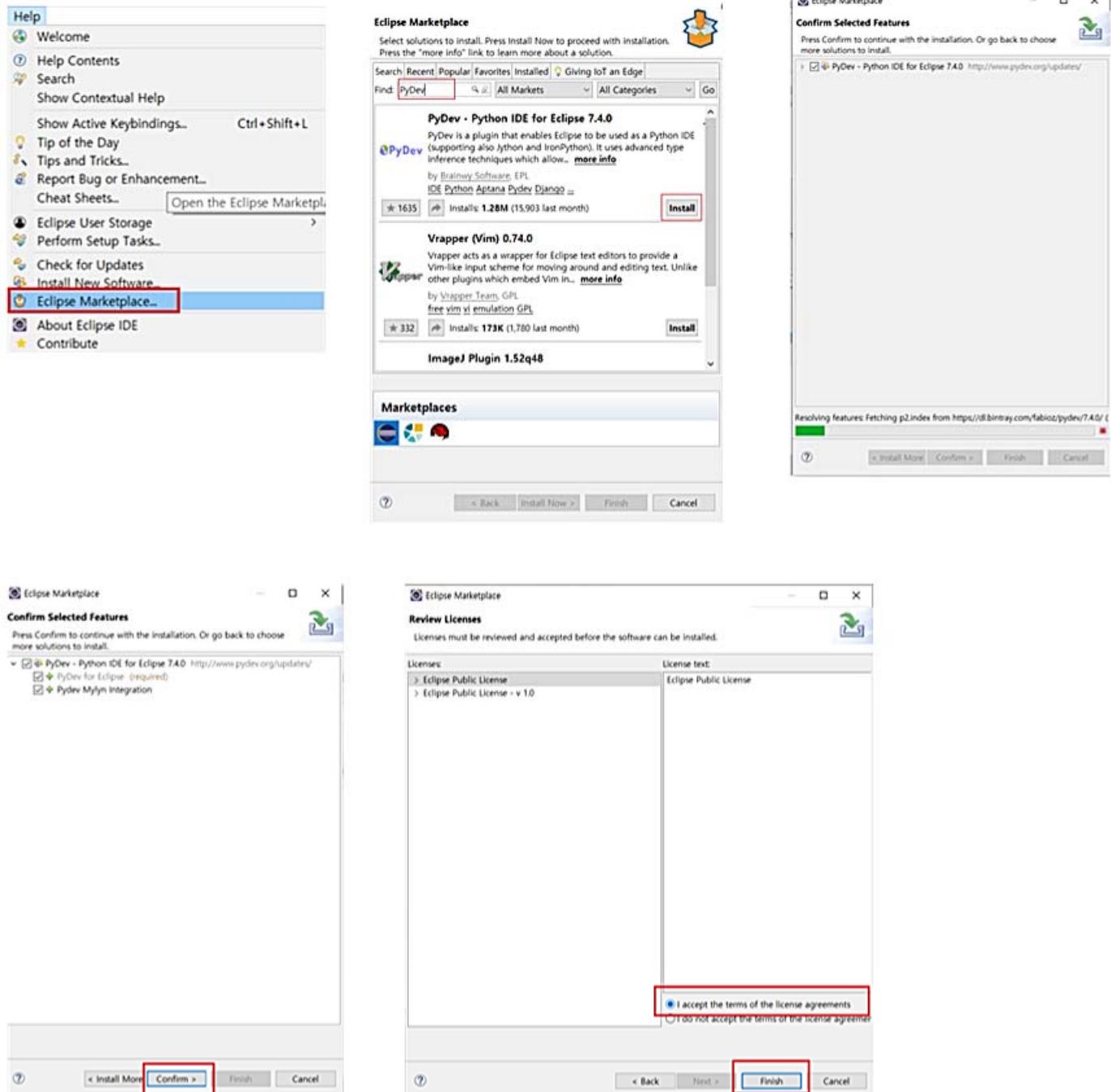


20.9 이클립스에 Django 사용하기

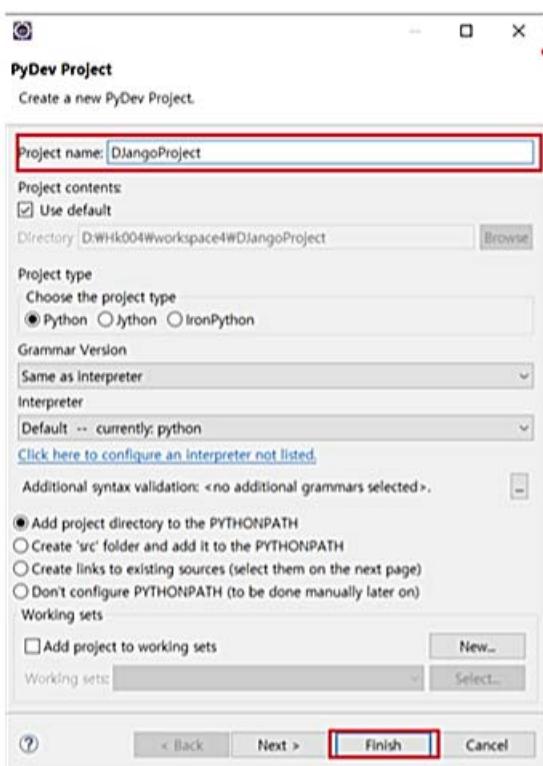
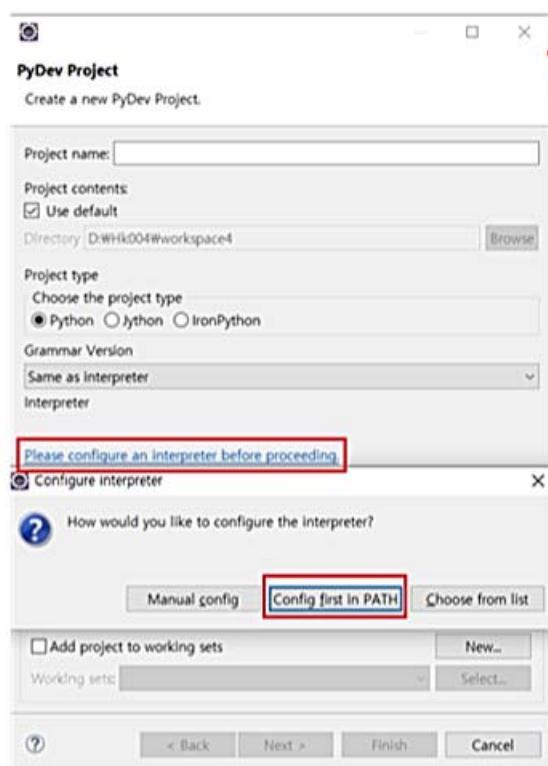
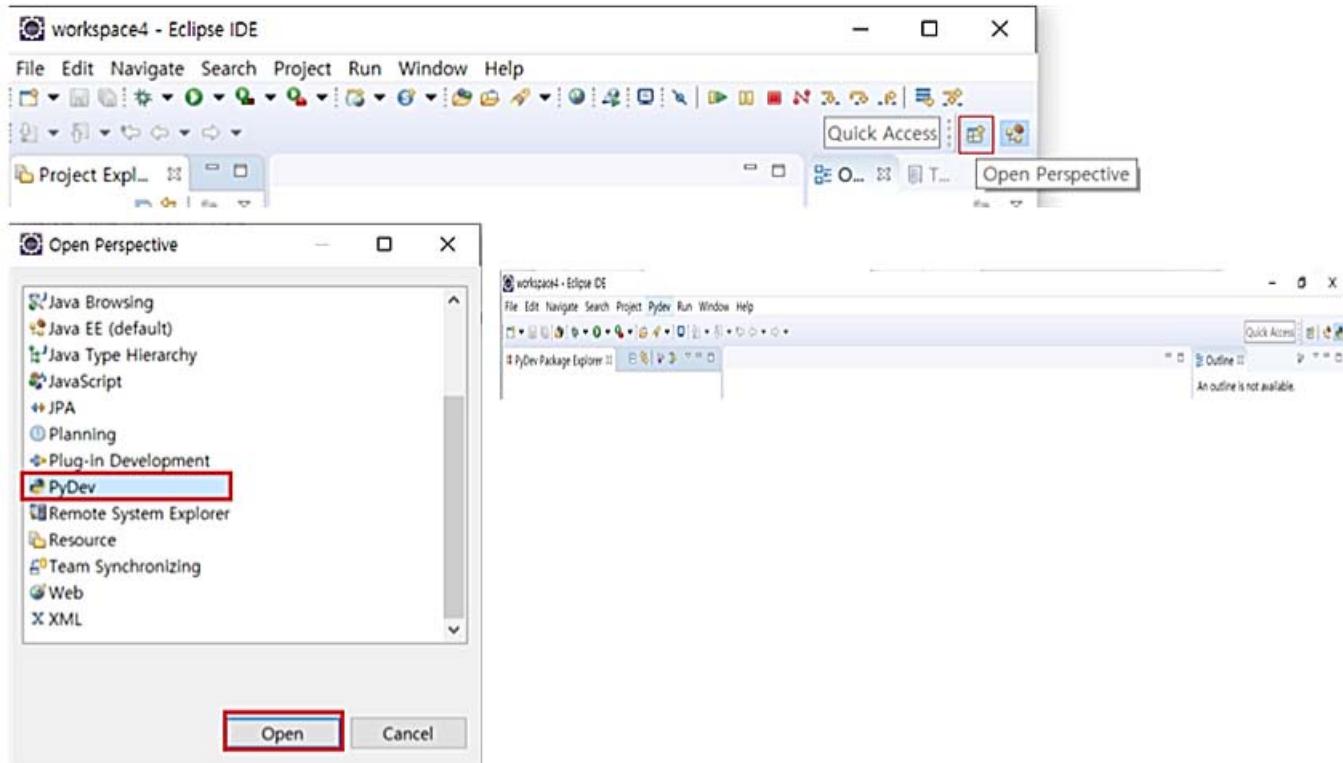
20.9.1 이클립스에 Python IDE 설정

1) 이클립스에서 Python IDE for Eclipse를 설치한다.



2) 이클립스에서 Python 프로젝트를 생성한다.

설치가 완료된 후 이클립스 우측 상단에서 Open Perspective를 선택하면 목록중에 PyDev가 있으면 준비가 완료된 것이다. 선택 후 Python 기반으로 작업을 진행하시면 된다.



3) 장고 프로젝트를 생성시킨다.

Eclipse Workspace 폴더에서 Django Project를 만들기 위해 Workspace로 이동한다.

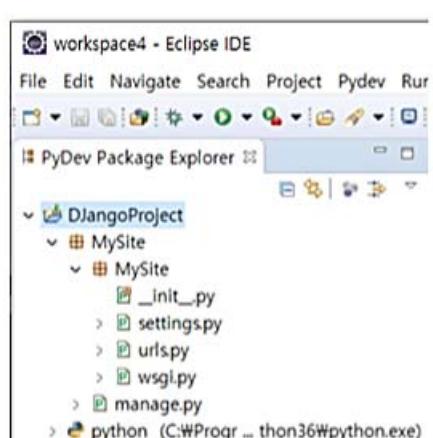
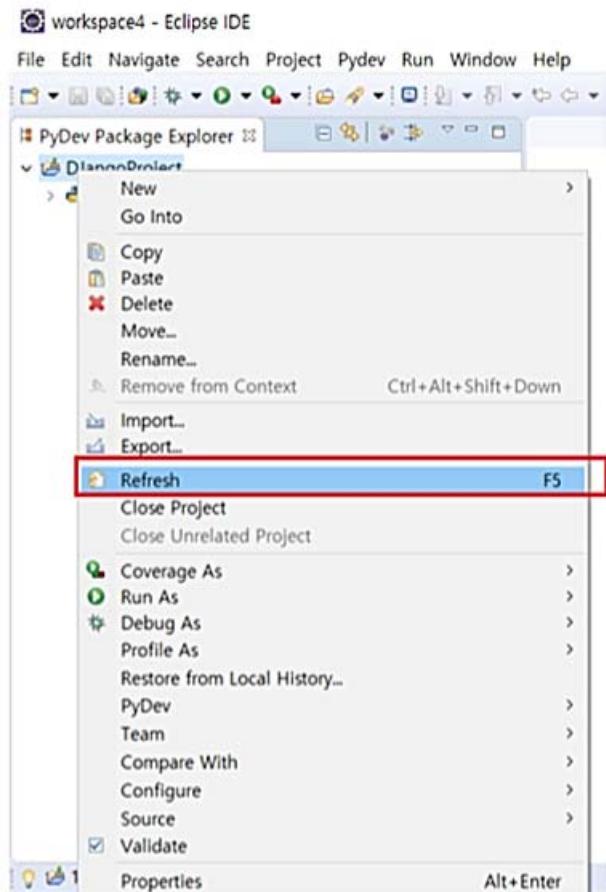
(여기서는 PowerShell을 이용한다.)

프로젝트를 생성시키기 위해 아래 명령어를 실행시킨다.

```
> django-admin.py startproject MySite  
> ls
```

디렉터리: D:\Hk004\workspace\DJangoProject

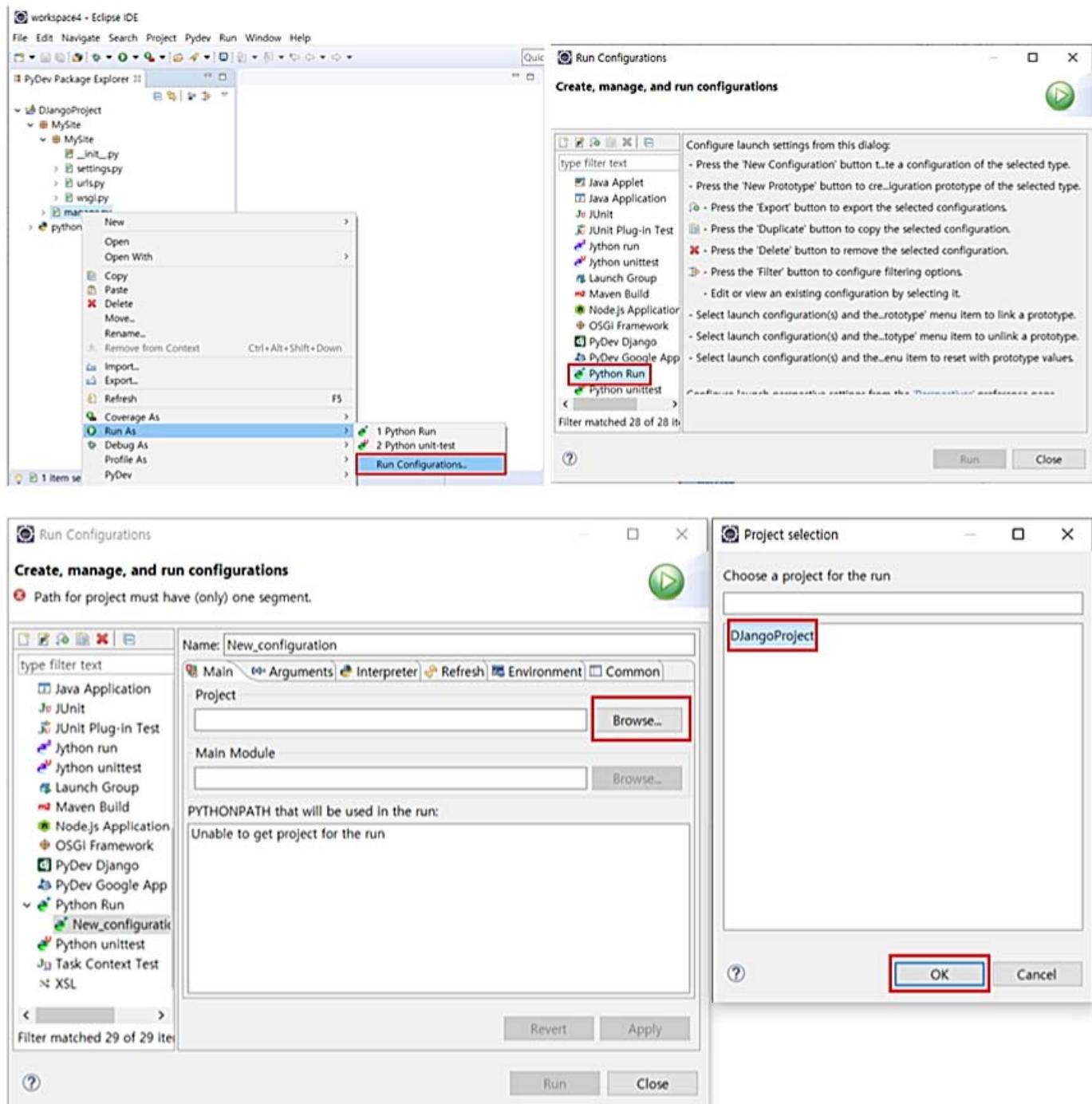
Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	2019-12-04 오후 12:50		MySite
-a---	2019-12-04 오후 12:49	384	.project
-a---	2019-12-04 오후 12:49	439	.pydevproject

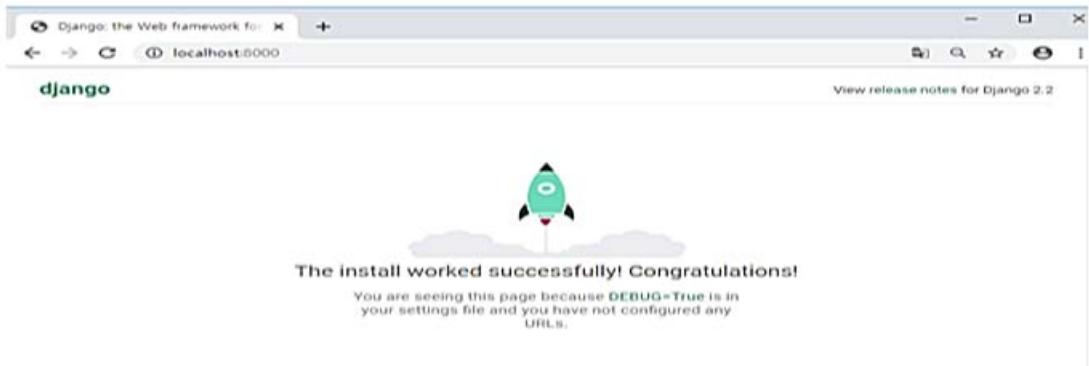
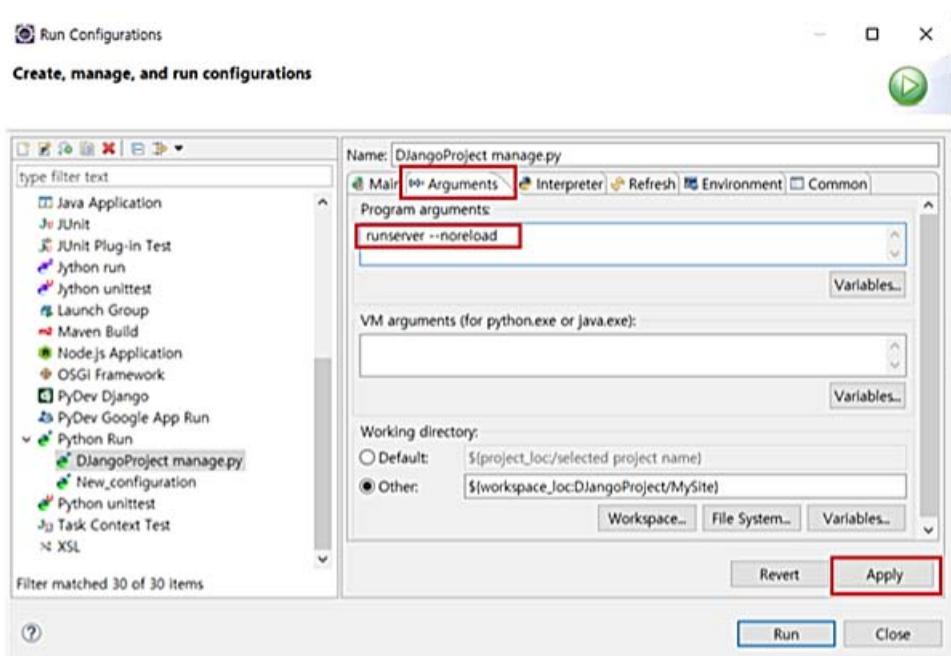
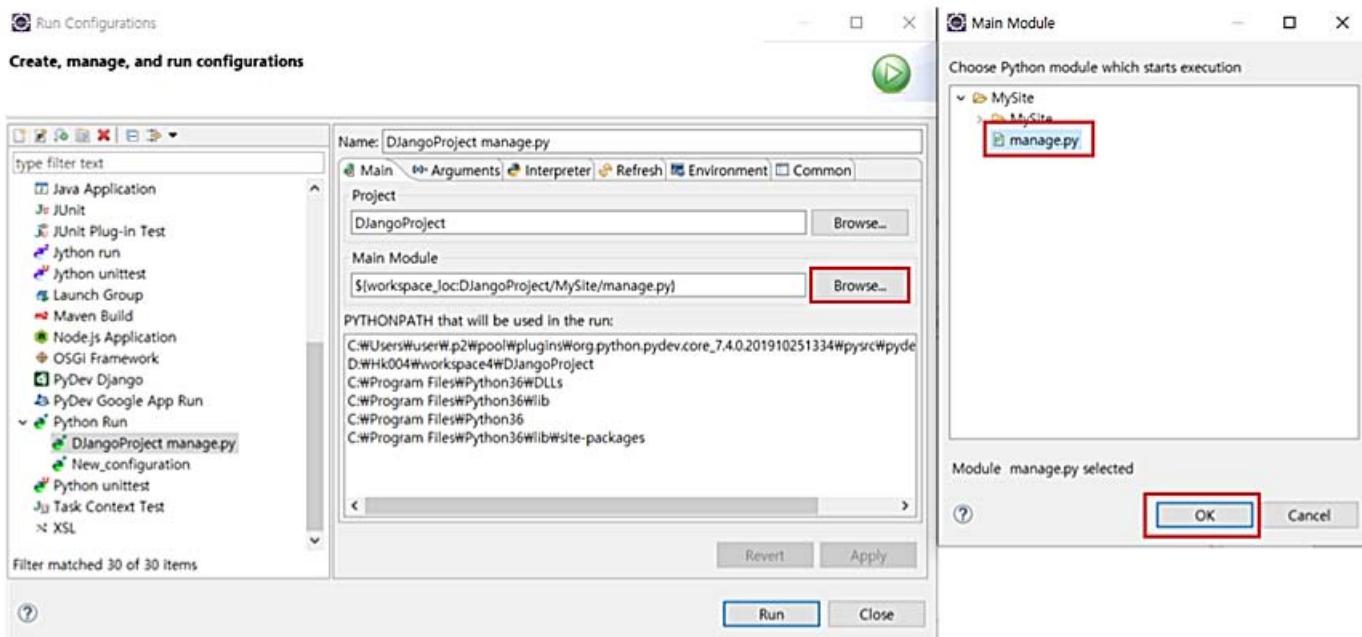


기본적으로 Django 프로젝트를 생성해봤다. 이제는 개발서버를 실행시켜보자

- MySite/manage.py를 선택
- 마우스 오른쪽 버튼으로 Run As > Run Configurations 를 선택

- c. 왼쪽메뉴에서 Python Run을 더블클릭을 한다
- d. 그럼 하위에 New_Configuration이라고 생성이 된다
- e. 오른쪽 탭에 Main 탭에서 Project 항목에서 서버를 실행하고자 하는 프로젝트를 선택한다.
- f. Main Module에서 manage.py를 선택한다.
- g. 이제 마지막으로 Arguments탭에서 Program arguments 항목에 "runserver --noreload" 를 입력하고 Apply로 적용 후 Run 을 하면 서버가 작동되는것을 볼수있다
- h. 이제는 브라우저를 실행하고 http://127.0.0.1:8000 으로 접속을 해보자





20.9.2 오라클 데이터베이스 연결

- 장고 파일의 기본구성과 역할은 다음과 같다.

- ① `__init__.py` 파일로 하여금 해당 디렉터리가 파이썬 패키지임을 인식하게 하는 빈 파일
- ② `manage.py` : 장고 프로젝트를 관리하는 커맨트-라인 스크립트
- ③ `settings.py` : 장고 프로젝트 설정 파일
- ④ `urls.py` : URL 요청을 장고 파일과 매핑하는 파일

- 데이터베이스 연결 정보 입력

`settings.py` 파일 Database 정보에 아래와 같이 입력한다.

<pre>DATABASES = { 'default': { 'ENGINE': 'django.db.backends.oracle', 'NAME': 'orcl', 'USER': 'STUDY', 'PASSWORD' : 'STUDY', 'HOST' : 'localhost', 'PORT' : '1521' } }</pre>	<p>ENGINE : 다음중 하나를 설정.</p> <p>'django.db.backends.postgresql_psycopg2' 'django.db.backends.mysql' 'django.db.backends.sqlite3' 'django.db.backends.oracle'</p> <p>'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),</p>
---	--

- 애플리케이션 생성

PS D:\Hk004\workspace\DJangoProject\MySite> python manage.py startapp member

PS D:\Hk004\workspace\DJangoProject\MySite> ls

디렉터리: D:\Hk004\workspace\DJangoProject\MySite

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	2019-12-04 오후 1:36		member
d----	2019-12-04 오후 1:04		MySite
-a----	2019-12-04 오후 1:15	0	db.sqlite3
-a----	2019-12-04 오후 12:50	647	manage.py

장고 프로젝트는 일종의 웹사이트로, 어플리케이션은 웹사이트의 서브 콤포넌트이다. 생성 후 해당 프로젝트 디렉터리 아래에 어플리케이션 디렉터리가 생성되었음을 알 수 있다.

- 모델 클래스 만들기(회원 관리 페이지 만들기)

app과 관련된 정보를 저장할 model을 정의해주기 위한 클래스를 만들어주어야 하는 데 이것을 Model이라고 한다.

모델클래스는 앱 안의 models.py에 정의해야 한다. 먼저 만든 member라는 폴더를 열어보면 "models.py"라는 파일이 있는 것을 알 수 있다.

"models.py" 파일에 모델 class를 만들기 위해서는 models.Model을 상속받아야 한다.

그리고 모델 class안의 멤버 변수의 field에 대한 정보와 Field type을 주면 된다.

django에서 보통 model이름은 대문자로 시작하고, 단수형으로 쓴다.

모델클래스에 있는 필드는 member 테이블에 있는 컬럼과 같아야 하며 각 필드의 메소드(models.CharField() : nvarchar())는 컬럼의 데이터 타입과 매핑되어야 한다.

member/models.py

```
1 from django.db import models
2 from django.core.validators import MinLengthValidator
3 # Create your models here.
4 class Member(models.Model):
5     text_validator = MinLengthValidator(12, "길이가 너무 짧습니다.")
6     pw_validator = MinLengthValidator(8, "8자 이상이어야 합니다.")
7     USER_ID = models.CharField(max_length=20, blank=False, null=False, primary_key=True)
8     USER_PW      =      models.CharField(max_length=100,      blank=False,      null=False,
9                                         validators=[pw_validator])
10    USER_NAME = models.CharField(max_length=20, blank=False, null=False)
11    USER_BIRTH = models.DateTimeField(blank=False, null=False)
12    USER_GENDER = models.CharField(max_length=1, blank=False, null=False)
13    USER_EMAIL = models.CharField(max_length=50, blank=False, null=False)
14    USER_ADDR = models.CharField(max_length=100, blank=False, null=False)
15    USER_PH1      =      models.CharField(max_length=13,      blank=False,      null=False,
16                                         validators=[text_validator])
17    USER_PH2 = models.CharField(max_length=13, blank=True, null=True)
18    USER_REGIST = models.DateTimeField(blank=False, null=False)
19    USER_CK = models.CharField(max_length=200, blank=True, null=True)
20
21    class Meta:
22        db_table = "member"
```

5행은 MinLengthValidator()를 사용하여 길이가 12자 이하이면 "길이가 너무 짧습니다."가 view(html페이지)에 출력되게 한 것이다. 14행의 USER_PH1의 길이가 12자 이상이 되도록 설정해 놓았다.

6행은 최소 길이가 8자 이상이 되도록 한것이며 8행 USER_PW에 적용해 놓았다.

Mysite/settings.py

```
1 INSTALLED_APPS = [
2     ...생략
3     'member', # 내용 추가
4 ]
```

데이터베이스를 다루기 위해서는 당연히 데이터베이스가 먼저 있어야 한다. 데이터베이스를 만드는 작업을 먼저 해보자. (이미 Database가 만들어져 있는 경우 model만 만들어서 사용하면 된다.)

- 데이터베이스에 테이블이 없는 경우 Django에서 기본적으로 제공하는 모델을 적용한다.

Django 프로젝트 폴더에서 makemigration 명령을 수행한다.

프로젝트에서 새로 생성되었거나 변경된 모델 구조를 파악하여 Database에 적용할 수 있도록 실행 스크립트를 만드는 작업이다. 이 작업을 했다고 Database에 적용되는 것은 절대 아니다. 말 그대로 실행할 수 있는 스크립트를 만들어 주는 것이다.

```
> python manage.py makemigrations
```

Migrations for 'member':

```
  member\migrations\0001_initial.py
```

- Create model Member



```
PyDev Package Explorer 0001_Initial (member.migrations) 3 from django.db import migrations, models 4 5 6 class Migration(migrations.Migration): 7 8     initial = True 9 10    dependencies = [ 11    ] 12 13    operations = [ 14        migrations.CreateModel( 15            name='Member', 16            fields=[ 17                ('USER_ID', models.CharField(blank=True, max_length=20, primary_key=True, serialize=False)), 18                ('USER_PW', models.CharField(blank=True, max_length=100)), 19                ('USER_NAME', models.CharField(blank=True, max_length=20)), 20                ('USER_BIRTH', models.DateField(blank=True)), 21                ('USER_GENDER', models.CharField(blank=True, max_length=1)), 22                ('USER_EMAIL', models.CharField(blank=True, max_length=50)), 23                ('USER_ADDR', models.CharField(blank=True, max_length=100)), 24                ('USER_PH1', models.CharField(blank=True, max_length=13)), 25                ('USER_PH2', models.CharField(blank=True, max_length=13, null=True)), 26                ('USER_REGIST', models.DateField(blank=True)), 27                ('USER_CK', models.CharField(blank=True, max_length=200, null=True)), 28            ], 29            options={ 30                'db_table': 'member', 31            }, 32        ), 33    ], 34]
```

- migrate 명령어를 실행한다.

migrate라는 것은 이 파일에 Django에서 사용할 테이블을 만드는 실제 작업이 이루어지는 명령어이다. 앞에서 만든 스크립트 파일을 실행해서 Database에 적용하는 것이다.

기본적으로 모델구조가 바뀌게 되면 위 작업을 수행해야 한다. 그래야 Database에 적용되기 때문이다.

Member를 이용해서 DB에 데이터를 넣을 수 있는 공간을 만들기 위해 "python manage.py migrate" 명령을 수행해야 한다.

```
> python manage.py migrate
```

- main 창 띄우기

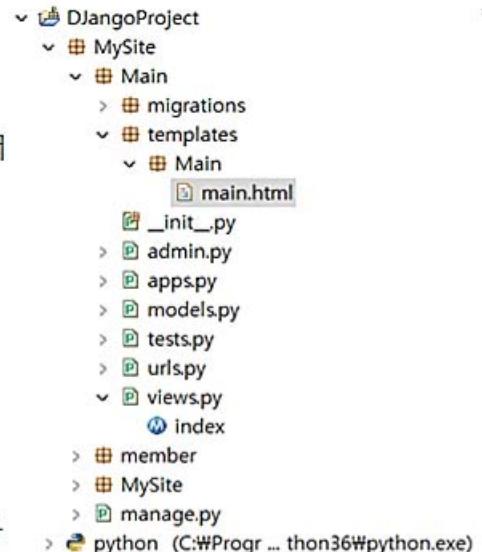
먼저 Main APP을 만들어 보자. 콘솔에서 아래와 같이 실행하여 MainAPP을 만든다.

```
> python manage.py startapp member
```

- html문서를 저장하기 위한 폴더 생성

Main app에 tempates 폴더를 만들고 아래에 app이름과 같은 Main폴더를 만들어 준다.

그리고 Main폴더에 main.html문서를 만든다.



- 첫 페이지에 main.html문서가 출력 될 수 있게 url을 잡아준다.

MySite/urls.py문서에 "path(r'^/\$', include('Main.urls')),"를 추가한다

"^"는 첫문장을 뜻하고 \$는 마지막을 뜻한다. 즉 "/"시작하여 끝난다는 뜻이다.

```
# MySite/urls.py
1 from django.urls import path, include
2 urlpatterns = [
3     path('', include('Main.urls')),
4     path('admin/', admin.site.urls),
5 ]
```

4행을 추가한다. "/"주소로 요청이 들어오면 Main app에 있는 urls.py파일을 포함시키라는 것이다.

- Main APP있는 urls.py에서 통해 views.py파일을 통해 main.html파일을 출력할 수 있는 함수가 무엇인지 정의준다.

```
# Main/urls.py
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'Main'
5 urlpatterns = [
6     url(r'^$', views.index),
7 ]
```

6행은 views.py에서 index함수를 실행시킨다.

- forms.py에 main.html에 들어갈 내용을 먼저 정의한다.

```
# Main/forms.py
1 from django.forms import ModelForm
2 from member.models import Member
3 from django import forms
4 from django.utils.translation import gettext as _
```

```

5
6 class SigninForm(ModelForm): #로그인을 제공하는 class이다.
7     class Meta:
8         model = Member
9         widgets = {'USER_PW':forms.PasswordInput}
10        fields = ['USER_ID','USER_PW']
11        labels = {
12             'USER_ID': _('사용자 아이디'),
13             'USER_PW': _('사용자 비밀번호'),
14         }

```

- views.py에 index함수를 정의 해준다.

```

# Main/views.py
1 from django.shortcuts import render
2
3 # Create your views here.
4 def index(request):
5     return render(request, 'Main/main.html', { 'form':form})

```

5행에서 render함수를 이용하여 templates/Main/main.html의 파일을 전송하라는 것이다.

```

# Main/templates/Main/main.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js" ></script>
7 <script type="text/javascript">
8 $(function(){
9     $("#id_USER_ID").focus();
10 });
11 </script>
12 </head>
13 <body>
14
15 <!-- 로그인 되었을 때 -->
16 <a href = "member/memberDetail">내 정보</a>
17 <a href = "logout" >로그아웃</a>
18 <a href = "member/list">회원리스트</a>
19 <a href = "comment/commentList">댓글 게시판</a>
20 <a href = "board/answerBoardList">답변형 자료실</a>
21 <a href = "goodsList">상품목록</a>

```

```

22 <a href = "memberAllMail">단체 메일보내기</a>
23 <a href = "survey" >설문지</a>
24
25 <!-- 로그인 되지 않았을 때 -->
26 <form id="frm" name="frm" action="loginAction" method="post">{% csrf_token %}
27 <table border =1>
28 <tr><td>
29             자동로그인<input type="checkbox" name="autoLogin" />
30         </td>
31         <td>
32
33             <input type="checkbox" name="idStore" />아이디체크
34
35
36         </td>
37 </tr>
38 {{ f.as_table }}
39 <tr>
40     <td colspan=2 ><input type="submit" id="btn" value="로그인" /></td>
41 </tr>
42 <tr><td colspan=2>
43             <a href="#">아이디 찾기</a>|<a href="#">비밀번호 찾기</a>
44             &nbsp;&nbsp;&nbsp;&nbsp;
45             <a href="member/agree">회원가입</a>
46         </td>
47 </tr>
48 </table>
49 </form>
50
51 </body>
52 </html>

```

51행은 forms.py로부터 받아온 값이다. 소스 보기를 하면 TextInput으로 받아온 것을 확인할 수 있다.

자동로그인	
아이디	<input type="text" value="Text"/>
비밀번호	<input type="password" value="Text"/>
<input type="checkbox"/> 아이디체크 <input type="button" value="로그인"/>	
아이디 찾기 비밀번호 찾기 회원가입	

내 정보 [로그아웃](#) [회원리스트](#) [댓글 게시판](#) [답변형 자료실](#) [상품목록](#) [단체 메일보내기](#) [설문지](#)

- “회원가입”을 누르면 회원가입 창이 열리도록 설정한다.

main.html에 링크된 “member/agree”으로 요청을 받을 수 있게 프로젝트 app밑에 있는 urls.py파일에 먼저 “member”주소로 들어오는 모든 주소를 처리할 수 있게 정의를 한다.

```
# MySite/urls.py
1 from django.urls import path, include
2 urlpatterns = [
3     path(r'^$', include('Main.urls')),
4     path('member/', include('member.urls')),
5     path('admin/', admin.site.urls),
6 ]
```

4행을 추가하여 “localhost:8000/member/”로 들어오는 주소를 member app에 있는 urls.py에서 처리하도록 정의해 놓았다.

- member app에 있는 urls.py에서 views.py에 있는 함수를 호출할 수 있도록 정의한다.

```
# member/urls.py
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     url('agree', views.agree),
7 ]
```

6행에서 views.py파일에 있는 agree함수를 호출하라고 정의 해놓는다.

- main.html에 링크된 “member/agree”를 호출하게 되면 member app에 있는 agree.html문서를 출력하도록 views.py에 함수를 정의한다.

```
# member/views.py
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from django.views.decorators.csrf import csrf_exempt
4 from .models import Member
5 from datetime import datetime
6
7 # Create your views here.
8 def agree(request):
9     return render(request, 'member/agree.html')
```

8행서 함수를 정의하고 9행에서 member app에 있는 agree.html문서를 render()함수를 이용해 전송한다.

먼저 member app에 templates폴더를 만들어 주고 하위에 member 폴더를 만든 후 agree.html문서를 만들어 준다.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8   <h2>약관 내용</h2>
9   <p>약관</p>
10  <form action="regist" method="post">
11    <label>
12      <input type="checkbox" name="agree" >
13    </label>
14    <input type="submit" value="동의" />
15  </form>
16 </body>
17 </html>

```

member/agree.html	
1	<!DOCTYPE html>
2	<html>
3	<head>
4	<meta charset="utf-8">
5	<title>Insert title here</title>
6	</head>
7	<body>
8	<h2>약관 내용</h2>
9	<p>약관</p>
10	<form action="regist" method="post">
11	<label>
12	<input type="checkbox" name="agree" >
13	</label>
14	<input type="submit" value="동의" />
15	</form>
16	</body>
17	</html>

- 약관 동의 페이지에서 checkbox에 check를 하면 가입페이지로 넘어가도록 한다.

먼저 MySite에 있는 urls.py에서 "localhost:8000/member/" 주소를 포함하는 주소가 오면 member app에 있는 urls.py로 이동하도록 "path('member/', include('member.urls')),"를 정의 해주었으므로 member app에 있는 urls.py에서 가입 페이지가 전송되도록 정의한다.

# member/urls.py	
1	from django.conf.urls import url
2	from . import views
3	
4	app_name = 'member'

```

5  urlpatterns = [
6      url('agree', views.agree),
7      url('regist', views.regist),
8 ]

```

8행은 “localhost:8000/member/”주소 뒤에 “regist” 주소가 있으면 views.py에 있는 regist함수를 실행시키라는 것이다.

- member app에 있는 views.py에서 regist()함수를 이용하여 checkbox에 체크를 했는지를 확인하고 “memberForm.html”파일을 전송하도록 정의한다.
- 먼저 memberForm.html에 들어갈 테이블 내용을 정의한 forms.py을 만든다.

member/forms.py

```

1  from django.forms import ModelForm
2  from .models import Member
3  from django import forms
4  from django.utils.translation import gettext as _
5
6  class SignupForm(ModelForm): #회원가입을 제공하는 class이다.
7      USER_GENDER_CHOICES = [('M', '남자'),('F', '여자'),]
8      USER_PW_Con = forms.CharField(max_length=200, widget=forms.PasswordInput(),label =
9      사용자 비밀번호 확인',help_text = '입력한 사용자 비밀번호와 같아야 합니다.')
10     USER_GENDER = forms.ChoiceField(choices=USER_GENDER_CHOICES,
11     widget=forms.RadioSelect, label = '성별', initial='M')
12     USER_EMAIL = forms.EmailField(label = '사용자 이메일')
13
14     class Meta:
15         model=Member
16         widgets = {'USER_PW':forms.PasswordInput}
17         fields =
18     ['USER_ID','USER_PW','USER_PW_Con','USER_NAME','USER_BIRTH','USER_GENDER','USER_EM
19     AIL','USER_ADDR', 'USER_PH1', 'USER_PH2']
20
21     help_texts = {
22         'USER_BIRTH': _('yyyy-MM-dd 형식으로 입력해주세요'),
23     }
24     labels = {
25         'USER_ID': _('사용자 아이디'),
26         'USER_PW': _('사용자 비밀번호'),
27         'USER_NAME': _('사용자 이름'),
28         'USER_BIRTH': _('사용자 생년월일'),
29         'USER_ADDR': _('사용자 주소'),
30         'USER_PH1': _('사용자 연락처1'),

```

31	'USER_PH2': _('사용자 연락처2'),
32	}

6행부터 32행까지는 회원가입을 제공하는 class이다.

7행은 성별에 대해 출력될 값을 정의 하였다.

8행은 회원가입 페이지에 비밀번호 확인을 추가하기 위해 정의해 줄 필요가 있다.

10행은 성별에 대한 RadioButton으로 출력될 수 있게 정의 하였다.

12행은 email 항목을 html의 EmailInput태그를 사용하기 위해 정의해 주었다. 정의하지 않으면 기본적으로 TextInput로 정의 된다.

15행은 Member table을 사용한다는 것이다.

16행은 Member table에 있는 USER_PW의 위젯을 PasswordInput 태그로 사용한다고 정의해 놓은 것이다. 9행과 같은 의미이며 9행을 생략하면 widgets = {'USER_PW':forms.PasswordInput, 'USER_EMAIL':forms.EmailInput}으로 변경해도 된다.

17행은 Member table로 부터 사용할 컬럼을 정의하여 가입 페이지에 출력하도록 정의해 놓았다.

21행의 help_texts는 각 자료항목에 도움이 되는 내용을 출력하기 위한 것이다.

24행부터 32행은 각 자료항목에 해당되는 내용을 출력하기 위해 설정해 주었다.

- ModelForm에서 사용 되는 매서드 -

매서드	내용
CharField	CharField (** kwargs) 클래스 기본 위젯은 TextInput 이다. 기본값은 empty_value로 빈 값을 제공한다. 문자열을 값으로 받는다. max_length 및 min_length가 제공되는 경우 MaxLengthValidator 및 MinLengthValidator를 사용한다. 오류 메시지 키 : required, max_length, min_length max_length : 문자열이 주어진 길이 이상인지 확인 strip : True (기본값)이면 값이 선행 및 후행 공백에서 제거 empty_value : 기본값은 빈 문자열, “빈”값을 나타내는데 사용
ChoiceField	ChoiceField (** kwargs) 클래스 기본 위젯은 SelectBox 이다. Empty value는 빈 문자열 값이다. 문자열을 값으로 받는다. 오류 메시지 키는 required, invalid_choice이다.
TypedChoiceField	TypedChoiceField (** kwargs) 클래스 ChoicedField와 마찬가지로 TypedChoiceField는 coerce 및 empty_value라는 두 개의 추가 인수를 사용한다. 기본 위젯은 SelectBox 이다. 오류 메시지 키는 required, invalid_choice이다 추가 인수 : coerce, empty_value
DateField	DateField (** kwargs) 클래스 기본 위젯 : DateInput Python datetime.date 객체를 이용하여 데이터를 정규화한다.

	<p>제공된 값이 datetime.date, datetime.datetime 또는 특정 날짜 형식으로 형식화 된 문자열인지 확인한다.</p> <p>오류 메시지 키 : required, invalid</p> <p>하나의 선택적 인수를 취한다.</p> <p>input_formats은 datetime.date 객체로 변환하는 데 사용되는 형식의 목록을 제공 한다.</p> <pre>['% Y- % m- % d', # '2006-10-25' '% m / % d / % Y', # '10 / 25 / 2006 ' '% m / % d / % y'] # '10 / 25 / 06 '</pre> <p>USE_L10N = False를 지정하면 다음도 기본 입력 형식에 포함한다.</p> <pre>['% b % d % Y', # '2006 년 10 월 25 일' '% b % d, % Y', # '2006 년 10 월 25 일' '% d % b % Y', # '2006 년 10 월 25 일' '% d % b, % Y', # '2006 년 10 월 25 일' '% B % d % Y', # '2006 년 10 월 25 일' '% B % d, % Y', # '2006 년 10 월 25 일' '% d % B % Y', # '2006 년 10 월 25 일' '% d % B, % Y'] # '2006 년 10 월 25 일'</pre>
DateTimeField	<p>DateTimeField (** kwargs) 클래스</p> <p>기본 위젯은 DateTimeInput이다.</p> <p>Python datetime.datetime를 이용하여 데이터를 정규화한다.</p> <p>제공된 값이 datetime.datetime, datetime.date 또는 특정 날짜 / 시간 형식으로 형식화 된 문자열인지 확인합니다.</p> <p>오류 메시지 키 : required, invalid</p> <p>input_formats은 문자열을 유효한 datetime.datetime 객체로 변환하는 데 사용되는 형식 목록이다.</p> <pre>['% Y- % m- % d % H : % M : % S', # '2006-10-25 14:30:59' '% Y- % m- % d % H : % M', # '2006-10-25 14:30' '% Y- % m- % d', # '2006-10-25' '% m / % d / % Y % H : % M : % S', # '10 / 25 / 2006 14:30:59 ' '% m / % d / % Y % H : % M', # '10 / 25 / 2006 14:30 ' '% m / % d / % Y', # '10 / 25 / 2006 ' '% m / % d / % y % H : % M : % S', # '10 / 25 / 06 14:30:59 ' '% m / % d / % y % H : % M', # '10 / 25 / 06 14:30 ' '% m / % d / % y'] # '10 / 25 / 06 '</pre>
EmailField	<p>EmailField (** kwargs) 클래스</p> <p>기본 위젯은 EmailInput이다.</p> <p>값이 문자열인지 확인한다.</p> <p>EmailValidator를 사용하여 적당히 복잡한 정규식을 사용하여 지정된 값이 유효한 전자 메일 주소인지 확인한다.</p>

	<p>오류 메시지 키 : required, invalid</p> <p>유효성 검사를 위한 max_length 및 min_length의 두 가지 선택적 인수가 있다.</p> <p>제공되는 경우 이러한 인수는 문자열이 주어진 길이 이상인지 확인한다.</p>
FileField	<p>FileField 클래스 (** kwargs)</p> <p>기본 위젯은 ClearableFileInput이다.</p> <p>파일 내용과 파일 이름을 단일 객체로 하는 UploadedFile 객체.</p> <p>파일 데이터가 양식에 바인딩되었는지 검증 할 수 있다.</p> <p>오류 메시지 키 : required, invalid, missing, empty, max_length</p> <p>max_length 및 allow_empty_file의 두 가지 선택적 인수가 있다.</p> <p>FileField를 사용할 때는 파일 데이터를 폼에 바인딩해야한다.</p> <p>max_length 오류는 파일 이름의 길이를 나타낸다.</p>
MultipleChoiceField	<p>MultipleChoiceField (** kwargs) 클래스</p> <p>기본 위젯은 SelectMultiple이다.</p> <p>주어진 값 목록의 모든 값이 선택 목록에 있는지 확인한다.</p> <p>오류 메시지 키 : required, invalid_choice, invalid_list</p>
TimeField	<p>TimeField (** kwargs) 클래스</p> <p>기본 위젯은 TimeInput이다.</p> <p>Python datetime.time를 이용하여 형식화 된 문자열인지 확인한다.</p> <p>오류 메시지 키 : required, invalid</p> <p>input_formats 인수가 제공되지 않으면 기본 입력 형식은 다음과 같다.</p> <p>'% H : % M : % S', # '14 : 30 : 59 '</p> <p>'% H : % M', # '14 : 30 '</p>
IntegerField	<p>IntegerField (** kwargs) 클래스</p> <p>기본 위젯은 NumberInput 또는 TextInput.</p> <p>Python Integer를 이용하여 주어진 값이 정수인지 확인한다.</p>
GenericIPAddressField	<p>GenericIPAddressField (** kwargs) 클래스</p> <p>IPv4 또는 IPv6 주소를 포함하는 필드입니다.</p> <p>주어진 값이 유효한 IP 주소인지 확인합니다</p>
FloatField	<p>FloatField (** kwargs) 클래스</p> <p>기본 위젯은 NumberInput 또는 TextInput.</p> <p>Python Float을 이용하여 주어진 값이 부동 소수점인지 확인한다.</p>
DecimalField	<p>DecimalField (** kwargs) 클래스</p> <p>기본 위젯은 NumberInput 또는 TextInput.</p> <p>주어진 값이 10 진수인지 확인한다. max_value 및 min_value가 제공되는 경우</p> <p>.MaxValueValidator 및 MinValueValidator를 사용합니다. 선행 및 후행 공백은 무시됩니다</p> <p>오류 메시지 키 : required, invalid, max_value, min_value, max_digits, max_decimal_places, max_whole_digits</p>
field arguments	
required	name = forms.CharField(required=False)
label	name = forms.CharField(label='Your name')
label_suffix	captcha_answer = forms.IntegerField(label='2 + 2', label_suffix=' = ')
initial	>>> class CommentForm(forms.Form): ... name = forms.CharField()

	<pre> ... url = forms.URLField() ... comment = forms.CharField() >>> default_data = {'name': 'Your name', 'url': 'http://'} >>> f = CommentForm(default_data, auto_id=False) >>> print(f) <tr><th>Name:</th><td><input type="text" name="name" value="Your name" required></td></tr> <tr><th>Url:</th><td><ul class="errorlist">Enter a valid URL.<input type="url" name="url" value="http://" required></td></tr> <tr><th>Comment:</th><td><ul class="errorlist">This field is required.<input type="text" name="comment" required></td></tr> </pre>
help_text	subject = forms.CharField(max_length=100, help_text='100 characters max.')
error_messages	name = forms.CharField(error_messages={'required': 'Please enter your name'})
localize	revenue = forms.DecimalField(max_digits=4, decimal_places=2, localize=True)
Validators	even_field = models.IntegerField(validators=[validate_even])

member/views.py

```

1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from django.views.decorators.csrf import csrf_exempt
4 from .models import Member
5 from datetime import datetime
6 from .forms import SignupForm
7 # Create your views here.
8 def agree(request):
9     return render(request, 'member/agree.html')
10
11 @csrf_exempt
12 def regist(request):
13     if request.method == "GET":
14         return HttpResponseRedirect("agree")
15     elif request.method == "POST" :
16         agree = request.POST.get("agree")
17         if agree is not None :
18             return render(request, 'member/memberForm.html',{'f':SignupForm()})
19         else :
20             return HttpResponseRedirect("agree")

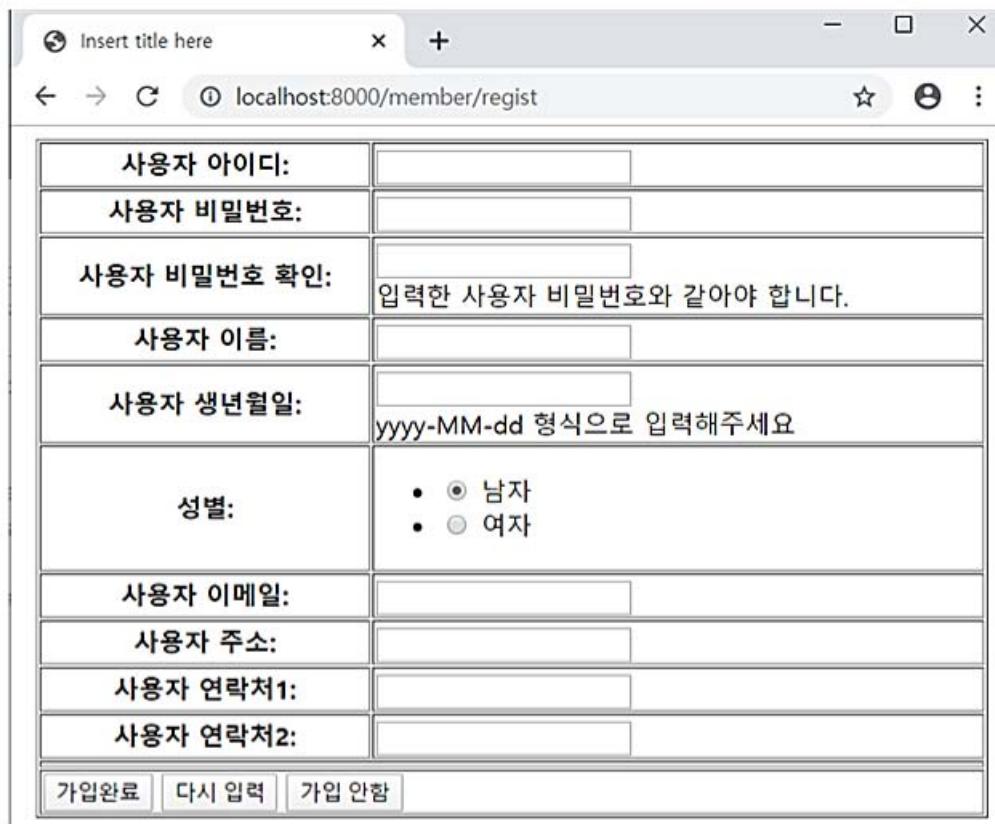
```

15행은 "POST"방식으로 전송되었을 때 실행하도록 조건을 준 것이다.

16행은 request로 받아온 데이터가 POST일때 tag이름이 "agree"로부터 값이 agree변수에 저장한다.

18행은 agree변수에 값이 저장되었는지 확인하고 16행에서 "가입창"을 전송해 준다. {'f':SignupForm()}는 forms.py에 정의된 클래스에 있는 내용을 회원가입 페이지에 전송해 주기 위해 적어주었다.

20행은 checkbox를 체크하지 않았다면 다시 agree 페이지로 이동하라는 것이다.



사용자 아이디:	<input type="text"/>
사용자 비밀번호:	<input type="password"/>
사용자 비밀번호 확인:	<input type="password"/> 입력한 사용자 비밀번호와 같아야 합니다.
사용자 이름:	<input type="text"/>
사용자 생년월일:	<input type="text"/> yyyy-MM-dd 형식으로 입력해주세요
성별:	<input checked="" type="radio"/> 남자 <input type="radio"/> 여자
사용자 이메일:	<input type="text"/>
사용자 주소:	<input type="text"/>
사용자 연락처1:	<input type="text"/>
사용자 연락처2:	<input type="text"/>
<input type="button" value="가입완료"/> <input type="button" value="다시 입력"/> <input type="button" value="가입 안함"/>	

member/memberForm.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Insert title here</title>
6 <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js" ></script>
7 </head>
8 <body>
9 <form name = "frm" id="frm" method = "post" action="memberJoinAction">{{ csrf_token }}
10 <input type="hidden" name = "confirmNum" id = "confirmNum" value="2">
11 <table width = 600 align = "center" border = 1 >
12     {{ f.as_table }}
13     <tr>
14         <td colspan = 2>
15             {% if error %}
16                 <h3 style="color:red">{{error}}</h3>
17             {% endif %}
18         </td>
19     </tr>
```

```

20      </tr>
21      <tr>
22          <td colspan = 2>
23              <input type= "submit" value="가입완료" />
24              <input type= "reset" value="다시 입력" />
25              <input type= "button" value="가입 안함" />
26      </td>
27      </tr>
28  </table>
29 </form>
30 </body>
31 </html>

```

12행은 forms.py에 있는 SignupForm클래스의 내용을 테이블(table)로 받아오겠다는 것이다,
 23행에서 submit을 하면 MemberJoinAction를 요청하고 요청한 페이지에서 디비에 저장하게 된다.
 15행 ~ 17행은 전송한 데이터에 오류가 있을 경우 오류 메시지를 views.py로부터 받아온다는 것이다.

요청한 주소가 “localhost:8000/member/MemberJoinAction”이므로 프로젝트 app에있는 urls.py에
 “localhost:8000/member”가 있는 경우 member app에 있는 urls.py로 가도록 되었으므로 member app에
 있는 urls.py를 수정해 주면 된다.

- member app에 있는 urls.py에서 views.py에 있는 함수를 실행 시키도록 설정

```
# member/urls.py
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     url('agree', views.agree),
7     url('regist', views.regist),
8     url('MemberJoinAction', views.MemberJoinAction),
9 ]
```

8행을 추가하며 “MemberJoinAction”으로 주소를 요청했을 때 views.py파일에 있는 memberJoinAction()함수를 실행시키라는 것이다.

- member app에 있는 views.py를 수정한다.

```
# member/views.py
1 from django.http.response import HttpResponseRedirect
2 ... (중략)
3 @csrf_exempt
4 def memberJoinAction(request):
5     if request.method == "GET":
```

```

6         return HttpResponseRedirect("agree")
7     elif request.method == "POST":
8         form = SignupForm(request.POST)
9         if form.is_valid():
10             if form.cleaned_data['USER_PW'] == form.cleaned_data['USER_PW_Con']:
11                 member = Member()
12                 member.USER_ID = form.cleaned_data['USER_ID']
13                 member.USER_PW = form.cleaned_data['USER_PW']
14                 member.USER_NAME = form.cleaned_data['USER_NAME']
15                 member.USER_BIRTH = form.cleaned_data['USER_BIRTH']
16                 member.USER_GENDER = form.cleaned_data['USER_GENDER']
17                 member.USER_EMAIL = form.cleaned_data['USER_EMAIL']
18                 member.USER_ADDR = form.cleaned_data['USER_ADDR']
19                 member.USER_PH1 = form.cleaned_data['USER_PH1']
20                 member.USER_PH2 = form.cleaned_data['USER_PH2']
21                 member.USER_REGIST = datetime.now()
22             try:
23                 member.save()
24                 print("저장되었습니다.")
25                 return HttpResponseRedirect("/")
26             except :
27                 return HttpResponseRedirect("가입되지 않았습니다.")
28             #return render(request, 'member/memberForm.html',{'f':form,'error':'가입'
29 되지 않았습니다.'})
30         else :
31             return render(request, 'member/memberForm.html',{'f':form,'error':'비밀번호'
32 확인이 비밀번호와 다릅니다.'})
33         else :
34             return render(request, 'member/memberForm.html',{'f':form,'error':'가입되지 않았'
35 습니다.'})

```

1행은 27행의 오류 내용을 회원가입 페이지로 보내주기 위해 import했다.

7행은 POST방식으로 넘어온 데이터가 있는 경우 오류가 발생하지 않도록 한다.

8행은 가입페이지로부터 전송된 데이터를 SignupForm()에게 전송하였으므로 SignupForm클래스를 이용하여 데이터를 받아오도록 한다.

11행은 member 클래스의 객체를 생성한다.

12행~ 20행은 SignupForm()에 있는 데이터를 member객체에 저장한다.form.cleaned_data()메서드는 form으로부터 데이터를 받아올 때 사용한다.

23행은 member객체에 있는 데이터를 디비에 저장하게 한다. insert문을 안 적어줘도 저장이 된다.

POST방식으로 보내진 데이터를 받을 수 있도록 "request.POST[]"를 사용하며 "[]"안에 있는 문자는 html에 있는 tag들의 이름이다. member객체에 있는 멤버 필드에 맞게 대입해주면 된다.

21행에서 datetime.now()는 Timestamp로 현재 날짜를 가져온다. 그러면 datetime을 import해야 한다.

27행은 23행에서 오류가 발생하면 가입페이지에 오류메세지를 전송한다.

- main화면에서 아이디와 비번을 입력하여 로그인되도록 한다.

main.html에서 로그인 버튼을 누르면 “loginPro”를 요청하고 id와 비번이 있는지 확인하고 있으면 session에 저장하도록 한다.

먼저 main화면에서 작성한 id와 비번을 “loginPro”를 통해 전달한다.

```
# Main/urls.py
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'Main'
5 urlpatterns = [
6     url(r'^$', views.index),
7
8     url('loginAction', views.loginAction),
9 ]
```

```
# Main/views.py
1 ... (중략)
2 @csrf_exempt
3 def loginAction(request):
4
5     response = HttpResponseRedirect("/")
6     if request.method == "GET":
7         return response
8     elif request.method == "POST":
9         id1 = request.POST['USER_ID']
10        pw = request.POST['USER_PW']
11        member = Member.objects.get(USER_ID=id1);
12        if member is not None:
13            if member.USER_PWD == pw :
14                try :
15                    request.session['member'] = {}
16                    request.session['member']['USER_ID'] = member.USER_ID
17                    request.session['member']['USER_PWD'] = member.USER_PWD
18                    request.session['member']['USER_NAME'] = member.USER_NAME
19                    request.session['member']['USER_EMAIL'] = member.USER_EMAIL
20                except :
21                    print("로그인 되지 않았습니다.")
22            else :
23                print("비밀번호가 틀립니다.")
24        return render(request, 'Main/main.html', {'f':SigninForm(),'error':'비밀번호가'})
```

	틀립니다.'})
25	else :
26	return render(request, 'Main/main.html', {'f':SigninForm(),'error':'아이디가 존재하지 않습니다.'})

4행은 forms.py의 SigninForm()클래스를 이용하여 가입창에서 데이터를 전송받아 온다.

9행과 10행은 SigninForm()클래스로부터 받은 데이터를 id1과 pw에 저장한다.

11행은 member 테이블에서 USER_ID가 id1에 해당하는 데이터가 있으면 가져온다.

12행은 11행에서 데이터를 가져왔는지 확인한다.

13행은 데이터베이스로부터 가져온 비밀번호와 로그인 창으로부터 받아온 비밀번호가 같은지를 비교한다.

15행은 member라는 이름의 매트릭스를 session으로 만들고 member에 각각 'USER_ID', 'USER_PW', 'USER_NAME', 'USER_EMAIL'을 저장하는 session을 만들어 주었다.

24행은 비밀번호가 틀렸을 경우 main페이지로 돌아가고 비밀번호가 틀렸다는 오류메시지를 전달한다.

26행은 11행에서 데이터를 가져오지 못한 경우 main페이지로 돌아가면서 아이디가 존재하지 않는다는 오류 메세지를 전달한다.

- session 사용하기

main 페이지에 session을 적용시켜 로그인되었을 때와 안되었을 때를 구분할 수 있게 한다.

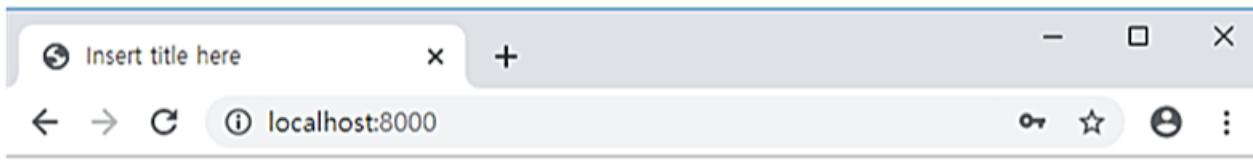
# Main/templates/Main/main.html	
1	<!DOCTYPE html>
2	<html>
3	<head>
4	<meta charset="UTF-8">
5	<title>Insert title here</title>
6	<script type="text/javascript" src="http://code.jquery.com/jquery-latest.js" ></script>
7	<script type="text/javascript">
8	\$(function(){
9	\$("#id_USER_ID").focus();
10	});
11	</script>
12	</head>
13	<body>
14	{% if request.session.member %}
15	<!-- 로그인 되었을 때 -->
16	내 정보
17	로그아웃
18	회원리스트
19	댓글 게시판
20	답변형 자료실
21	상품목록
22	단체 메일보내기
23	설문지
24	{% else %}

```

25 <!-- 로그인 되지 않았을 때 -->
26 <form id="frm" name="frm" action="loginAction" method="post">{% csrf_token %}
27 <table border =1>
28 <tr><td>
29         자동로그인<input type="checkbox" name="autoLogin" />
30     </td>
31     <td>
32
33         <input type="checkbox" name="idStore" />아이디체크
34
35     </td>
36 </tr>
37 {{ f.as_table }}
38 <tr>
39     <td colspan=2 ><div id="idmsg">
40         {% if error %}
41             <h3 style="color:red">{{error}}</h3>
42         {% endif %}
43     </div></td>
44 </tr>
45 <tr>
46     <td colspan=2 ><input type="submit" id="btn" value="로그인" /></td>
47 </tr>
48 <tr><td colspan=2>
49         <a href="#">아이디 찾기</a>|<a href="#">비밀번호 찾기</a>
50         &nbsp;&nbsp;&nbsp;
51         <a href="member/agree">회원가입</a>
52     </td>
53 </tr>
54 </table>
55 </form>
56 {% endif %}
57 </body>
58 </html>

```

14행은 member라는 이름의 session이 있는지 확인한다. member이름의 session이 존재하면 16행부터 23행을 출력한다. 아니면 26행부터 55행까지 출력한다.



[내 정보](#) [로그아웃](#) [회원리스트](#) [댓글 게시판](#) [답변형 자료실](#) [상품목록](#) [단체 메일보내기](#) [설문지](#)

- 로그아웃하기

main페이지에서 “로그아웃”에 링크된 주소를 보면 “<http://localhost:8000/logout>”로 로그아웃요청을 하도록 되어 있다.

MySite/urls.py에 “/”은 Main/urls.py로 가게 정의되어 있으므로 여기에 로그아웃을 하기 위한 설정을 해준다.

Main/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'Main'
5 urlpatterns = [
6     url(r'^$', views.index),
7     url('logout', views.logout),
8
9     url('loginAction', views.loginAction),
10 ]
```

7행에 로그아웃 메서드를 views.py에 logout으로 해 놓았다.

Main/views.py

```
1 def logout(request):
2     response = HttpResponseRedirect("/")
3     try:
4         del request.session['member']
5     except KeyError:
6         pass
7     return response
```

4행은 member라는 이름의 session을 삭제한다.

- 쿠키 사용하기

자동 로그인과 아이디 체크를 쿠키를 이용하여

자동로그인을 체크하면 계속 로그인된 상태가 되게 하고

아이디 체크를 하면 아이디가 계속 노출되도록 한다.

쿠키를 사용하기 위해서는 먼저 response객체를 만들어야 하는데 HttpResponseRedirect()메소드를 이용하여 객체를 생성시킨다.

자동 로그인이나 로그인 체크는 로그인된 후 쿠키에 저장이 되어야 하므로 login 경로에 정의 해주어야 한다.

The screenshot shows a login interface with two checkboxes at the top: '자동로그인' (checked) and '아이디체크' (unchecked). The '자동로그인' checkbox is highlighted with a red box. Below the checkboxes are two input fields with error messages: '사용자 아이디:' and '사용자 비밀번호:', each followed by the message '• This field is required.'. At the bottom is a large blue '로그인' button. At the very bottom of the page are three blue links: '아이디 찾기', '비밀번호 찾기', and '회원가입'.

Main/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'Main'
5 urlpatterns = [
6     url(r'^$', views.index),
7     url('logout', views.logout),
8
9     url('loginAction', views.loginAction),
10 ]
```

“Main/urls.py”에서 로그인은 “views.loginAction”임을 위 예제에서 확인할 수 있다.

Main/views.py

```
1 @csrf_exempt
2 def loginAction(request):
3     ... 중략
4         try :
5             ... 중략
6                 response = HttpResponseRedirect("/")
7                 if request.POST.get("idStore") is not None:
8                     max_age = 30 * 24 * 60 * 60
9                     response.set_cookie('idStore',member.USER_ID, max_age=max_age)
10                else :
11                    response.set_cookie('idStore',member.USER_ID, max_age=0)
12                return response
13            except :
14                print("로그인 되지 않았습니다.")
15    ... 중략
16
17 def index(request):
18     form = SigninForm(request.POST)
19
20     if request.COOKIES.get('idStore') is not None :
21         context = {'idStore': request.COOKIES.get('idStore')}
22         form = SigninForm(initial={"USER_ID": context['idStore']})
23         return render(request, 'Main/main.html', {'f':form, 'context':context})
24
25     return render(request, 'Main/main.html', {'f':form})
```

6행은 response 객체를 만들기 위해 HttpResponseRedirect를 사용하는데 이때 HttpResponseRedirect를 이용하여 이동할 주소를 적어준다. 여기서는 “/”(루트)로 이동한다는 것이다.

즉, HttpResponseRedirect("/")는 “http://localhost:8000/”를 의미한다.

7행에서 아이디체크에 체크가 되었는지 비교한다.

8행은 유지시간을 정의해줬다.

9행에서 아이디체크가 체크되었다면 'idStore' 쿠키를 만들고 쿠키 값은 member.USER_ID가 된다.

11행은 아이디 체크에 체크가 풀렸을 경우 'idStore' 쿠키를 제거한다.

20행은 'idStore' 쿠키가 있는지 확인한다.

21행은 context 사전을 만들어 'idStore'를 저장하고 값을 준다.

22행은 사용자 아이디에 아이디가 출력되도록 form에 USER_ID에 초기값을 전달한다.

- 아이디 체크가 된 경우 main페이지에 사용자 아이디에 아이디가 출력되게 forms.py를 수정한다.

Main/forms.py

```
1 class SigninForm(ModelForm): #로그인을 제공하는 class이다.
2     class Meta:
3         ... 중략
4     def __init__(self, *args, **kwargs):
5         super(SigninForm, self).__init__(*args, **kwargs)
6         self.fields['USER_ID'].initial = False
```

4행에 초기화 하기 위한 __init__ 메서드를 정의한다.

6행에서 'USER_ID'에 값을 초기화한다.

Main/main.html

```
1 <form id="frm" name="frm" action="loginAction" method="post">{% csrf_token %}
2 <table border =1>
3 <tr><td>
4             자동로그인<input type="checkbox" name="autoLogin" />
5             </td>
6             <td>
7
8             <input type="checkbox" name="idStore" {% if context %} checked  {% endif %} />아이
9             디체크
10            </td>
11        </tr>
```

8행을 수정하여 context가 전달되면 checked 옵션을 사용하게 만들어 준다.

- 로그인시 쿠키를 사용하여 로그인 상태 유지

Main/views.py

```
1 @csrf_exempt
2 def loginAction(request):
3     ... 중략
4         try :
5             ... 중략
```

```

6         ... 중략
7         if request.POST.get("autoLogin") is not None:
8             max_age = 30 * 24 * 60 * 60
9             response.set_cookie('autoLogin',member.USER_ID, max_age=max_age)
10            return response
11        except :
12            print("로그인 되지 않았습니다.")
13    ... 중략
14
15    def index(request):
16        form = SigninForm(request.POST)
17
18        if request.COOKIES.get('idStore') is not None :
19            ... 중략
20        if request.COOKIES.get('autoLogin') :
21            print("index :" + request.COOKIES.get('autoLogin'))
22            member = Member.objects.get(USER_ID=request.COOKIES.get('autoLogin'));
23            request.session['member'] = {}
24            request.session['member']['USER_ID'] = member.USER_ID
25            request.session['member']['USER_PW'] = member.USER_PW
26            request.session['member']['USER_NAME'] = member.USER_NAME
27            request.session['member']['USER_EMAIL'] = member.USER_EMAIL
28
29        return render(request, 'Main/main.html', {'f':form})

```

7행에서 자동 로그인이 체크되었는지 확인한다.

9행에서 자동 로그인에 관련된 쿠키를 생성시키다.

20행은 “/(루트)로 올 때 자동 로그인 되는지 확인해야 하므로 index에서 쿠키가 있는지 확인하고 쿠키가 있으면 로그인시 session을 사용하므로 session을 정의해준다.

- 메일전송 하기

Django 에서 메일 송신

Django 에서는 Python의 내장 모듈인 smtplib 를 통해 메일을 발송한다

smtplib 로도 메일 전송을 할 수 있지만 Django에서는 좀 더 편하게 서비스를 제공한다

(1) Django 설정

Django에서는 위의 첫번째 설정이 settings.py의 몇 줄의 코드로 설정된다

MySite/settings.py	
--------------------	--

1	EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
2	EMAIL_HOST = "smtp.gmail.com"
3	EMAIL_HOST_USER = '구글이메일'

```

4 EMAIL_HOST_PASSWORD = '구글비밀번호'
5 EMAIL_PORT = 587
6 EMAIL_USE_TLS = True
7 DEFAULT_FROM_EMAIL = EMAIL_HOST_USER

```

EMAIL_BACKEND : 메일을 전송에 쓰는 백엔드 설정 (장고에서는 위같이 설정하라고 나와있다)

EMAIL_PORT : 587 포트는 GMail의 권장 사항이다

EMAIL_USE_TLS : TLS(Transport Layer Security)는 통신 내용을 암호화하는 통신 규약이다. 일반적으로 587 포트에 적용한다고 한다

Django 에서 메일 송신

Django 에서는 Python의 내장 모듈인 EmailMultiAlternatives 를 통해 메일을 발송한다

member/views.py

```

1 from django.core.mail import EmailMultiAlternatives
2 import uuid
3
4 def mailSend(member):
5     num = str(uuid.uuid1()).replace('-', '')
6     content = "<html><body>안녕하세요 가입을 환영합니다. 아래 링크를 누르셔야만 가입이 완료됩니다.<br />"
7     content += "<a href='http://localhost:8000/member/memberMail/" + num
8     content += "?email=" + member.USER_EMAIL + "'> 클릭 </a></body></html>"
9     subject = "가입환영인사";
10    from_email = 'hiland00@gmail.com'
11    to = member.USER_EMAIL
12
13
14    msg = EmailMultiAlternatives(subject, content, from_email, [to])
15    msg.attach_alternative(content, "text/html")
16    msg.send()

```

2행은 범용 고유 식별자(universally unique identifier, UUID)를 말한다.

4행은 uuid를 이용하여 고유의 값을 받아온다. 문자열에 "-가 포함되어있다면 replace()를 이용하여 지워지게 했다.

6행~ 9행은 메일로 보내질 내용을 "html"형식으로 작성하였다.

10행은 메일로 전달될 제목이다.

11행은 보낼 사람의 이메일 주소이다.

12행은 받을 사람의 이메일주소를 적은 것이다.

14행은 msg라는 변수에 EmailMultiAlternatives객체를 저장하는 데 이때 6행부터 12행까지의 내용이 저장된다.

15행은 EmailMultiAlternatives객체에 내용의 형식을 "html"형식으로 전송될 수 있게 정의한 것이다.

16행에서 smtp로 메일을 보내게 된다.

UUID란

역사

UUID는 원래 아폴로 네트워크 컴퓨팅 시스템(NCS)에서 사용되었다가 나중에 개방 소프트웨어 재단(OSF)의 분산 컴퓨팅 환경(DCE)에서 사용되었다. DCE UUID의 초기 설계는 NCS UUID에 기반을 두었으며[1], 여기서 디자인은 아폴러 컴퓨터가 설계한 운영 체제인 도메인/OS에 정의되고 사용된 64비트 고유 식별자의 영향을 받았다.

개요

네트워크 상에서 서로 모르는 개체들을 식별하고 구별하기 위해서는 각각의 고유한 이름이 필요하다. 이 이름은 고유성(유일성)이 매우 중요하다. 같은 이름을 갖는 개체가 존재한다면 구별이 불가능해 지기 때문이다. 고유성을 완벽하게 보장하려면 중앙관리시스템이 있어서 일련번호를 부여해 주면 간단하지만 동시에 다발적이고 독립적으로 개발되고 있는 시스템들의 경우 중앙관리시스템은 불가능하다. 개발주체가 스스로 이름을 짓도록 하되 고유성을 충족할 수 있는 방법이 필요하다. 이를 위하여 탄생한 것이 범용고유식별자(UUID)이며 국제기구에서 표준으로 정하고 있다.

UUID 표준에 따라 이름을 부여하면 고유성을 완벽하게 보장할 수는 없지만 실제 사용상에서 중복될 가능성은 거의 없다고 인정되기 때문에 많이 사용되고 있다.

정의

UUID는 16 옥텟 (128비트)의 수이다. 표준 형식에서 UUID는 32개의 십육진수로 표현되며 총 36개 문자(32개 문자와 4개의 하이픈)로 된 8-4-4-4-12라는 5개의 그룹을 하이픈으로 구분한다. 이를테면 다음과 같다.

550e8400-e29b-41d4-a716-446655440000

340,282,366,920,938,463,463,374,607,431,768,211,456개의 사용 가능한 UUID가 있다.

8-4-4-4-12 포맷 문자열은 16바이트의 UUID를 위한 레코드 레이아웃에 기반을 둔다

UUID 레코드 레이아웃

이름	길이 (바이트)	길이 (16진 숫자)	내용
time_low	4	8	시간의 low 32비트를 부여하는 정수
time_mid	2	4	시간의 middle 16비트를 부여하는 정수
time_hi_and_version	2	4	최상위 비트에서 4비트 "version", 그리고 시간의 high 12비트
clock_seq_hi_and_res clock_seq_low	2	4	최상위 비트에서 1-3비트, 그리고 13-15비트 클럭 시퀀스
node	6	12	48비트 노드 id

-- 회원가입시 메일이 전송될 수 있게 memberJoinAction()메소드를 수정한다.

member/views.py

```
1 @csrf_exempt
2 def memberJoinAction(request):
3     ... 중략
```

```

4         try:
5             mailSend(member)
6             member.save()
7             return HttpResponseRedirect("/")
8         except :
9             return HttpResponseRedirect("가입되지 않았습니다.")
10    ... 중략
11

```

5행을 추가하여 위에서 만든 mailSend()메소드를 실행시키니다. 이때 회원의 정보를 전달하기 위해 member를 인자로 넘긴다.

-- 메일전송이 되면 가능확인 메세가 전송이 되어 가입승인을 할 수 있도록 한다.

★ 가입환영인사

보낸사람 : <hiland00@gmail.com> 주소록추가 | 수신차단

안녕하세요 가입을 환영합니다. 아래 링크를 누르셔야만 가입이 완료됩니다.

[클릭](#)

<http://localhost:8000/member/memberMail/09e85246208211ea9f719822eff8b110/?email=hiland0@nate.com>

(1) 먼저 “member/memberMail/09e85246208211ea9f719822eff8b110”주소를 찾을 수 있도록 urls.py를 수정 한다.

member/urls.py

```

1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     url('agree', views.agree),
7     url('regist', views.regist),
8     url('memberJoinAction', views.memberJoinAction),
9
10    url('memberMail/(?P<USER_CK>\w+)/$', views.memberMail), #③
11 ]

```

10행을 추가한다. “?P<USER_CK>\w+/\$”는 “memberMail/” 다음에 오는 주소를 “USER_CK”로 받겠다는 의미이고 “\w”는 모든 문자를 의미한다. “memberMail/”뒤에는 문자가 와야하고 전송된 문자를 “USER_CK”dp 저장해서 views.py에 있는 memberMail() 전달한다.

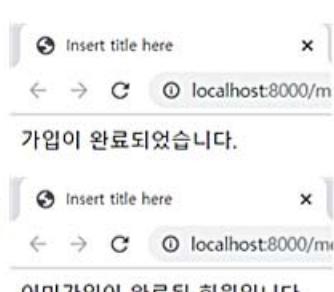
예를 들어 “USER_CK”는 “09e85246208211ea9f719822eff8b110”값을 전달 받아 views.py에 있는 memberMail()메소드에 전달하게 된다.

-- views.py파일에 memberMail()메소드를 만들어 준다.

mailSend(member)메소드에서 전송된 주소를 이용하여 "highland0@nate.com"을 가진 사용자의 USER_CK컬럼 "09e85246208211ea9f719822eff8b110"값을 저장하여야 한다.

member/views.py

```
1 def memberMail(request, USER_CK):
2     USER_EMAIL = request.GET['email']
3     member = Member.objects.get(USER_EMAIL = USER_EMAIL)
4     try:
5         if member.USER_CK == '' :
6             member.USER_CK = USER_CK
7             member.save()
8             return render(request, 'member/memberMailTrue.html')
9         else :
10            return render(request, 'member/memberMailFalse.html')
11     except:
12         pass
```



1행에서 "USER_CK"는 위 표에 있는 "member/urls.py"의 10행의 "USER_CK"를 의미한다,

urls.py의 표에서 "USER_CK"는 "09e85246208211ea9f719822eff8b110"을 전달받아 1행에 있는 "USER_CK"에 전달하게 된다.

2행은 querystring으로 날라온 email값을 USER_EMAIL에 저장하게 된다.

3행은 member 테이블로부터 USER_EMAIL에 해당되는 데이터를 필터링하여 가져온다.

5행은 member테이블로부터 받아온 값에 USER_CK 컬럼의 값이 null인 경우이다.

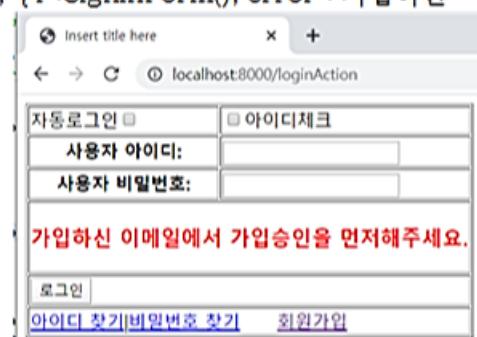
6행은 member테이블로부터 받아온 USER_CK 컬럼의 값을 USER_CK가 받아온 값으로 저장한다.

7행은 merge를 사용하여 update를 한다,

-- 로그인 할 때 USER_CK 컬럼의 값이 null인 경우 가입인증이 안되었음을 알려주자.

Main/views.py

```
1 @csrf_exempt
2 def loginAction(request):
3     ... 중략
4     if member is not None:
5         if member.USER_CK == '' :
6             return render(request, 'Main/main.html', {'f':SigninForm(),'error':'가입하신 이
7               메일에서 가입승인을 먼저해주세요.'})
8
9         if member.USER_PW == pwEncrypt(pw):
10            ... 중략
11        else :
12            ... 중략
13        else :
14            ... 중략
```



5행~ 6행을 추가하여 이메일을 통해 가입승인을 하지 않았음을 알려주도록 한다.

- 암호화하기

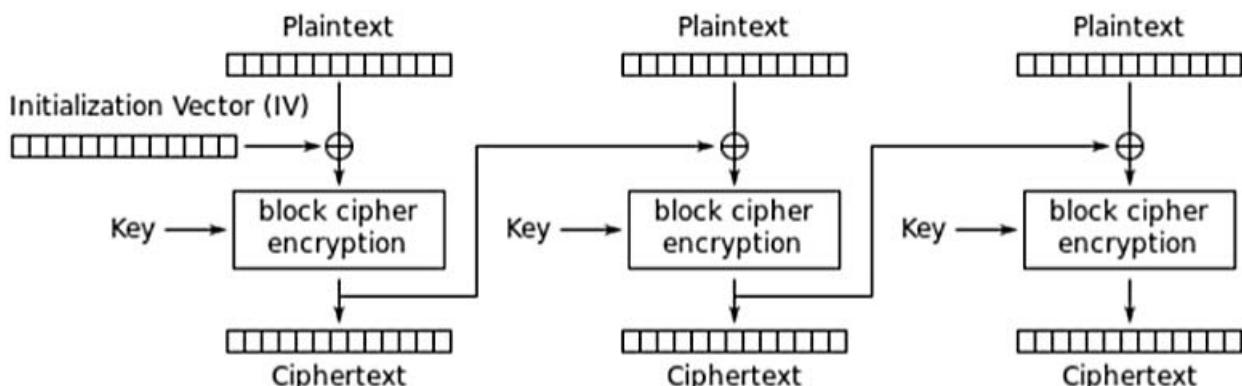
업무를 하다보면 민감한 문서를 다뤄야 할 때가 있다.

이럴때 암호화/복호화 기술을 활용하면 용이하다.

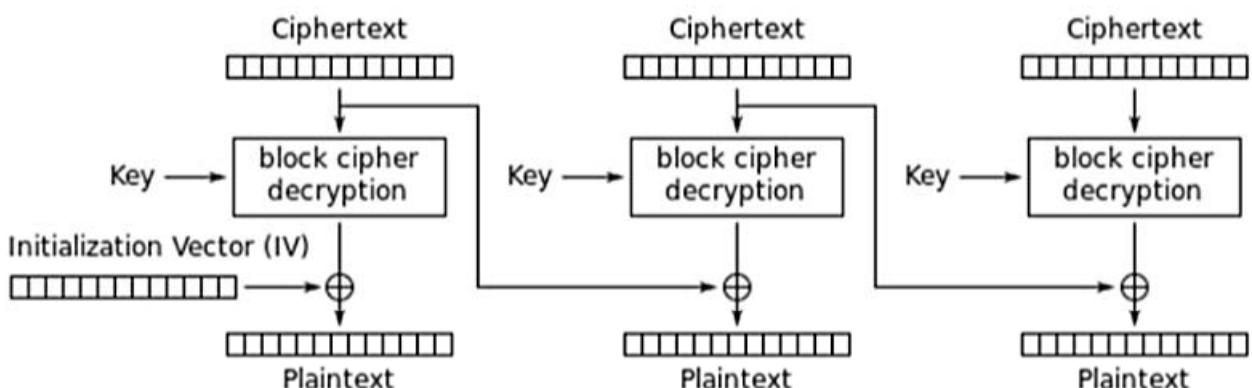
파이썬 모듈을 활용해 쉽게 적용할 수 있다.

암호블록체인(Cipher-block chainingm CBC) 방식은 1976년 IBM에 의해 개발되었다. 각 블록은 암호화되기 전에 이전 블록의 암호화 결과와 XOR되며, 첫 블록의 경우에는 초기화 벡터가 사용된다. 초기화 벡터가 같은 경우 출력결과가 항상 같기 때문에 매 암호화마다 다른 초기화 벡터를 사용해야 한다.

코딩시 IV(Initial Vector) 변수가 초기화벡터이다. 즉, IV를 초기 설정해 암호화하고 복호화할때도 동일한 IV값이 있어야 한다. IV가 제 2의 키역할을하게 된다.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

CBC방식은 현재 가장 널리 사용되고 있는 방식으로 암호화 입력값이 이전 결과에 의존하기 때문에 암호화시에는 병렬화가 불가능하지만 복호화시에는 병렬화가 가능하다.

코드에서 사용한 모듈은 AES CBC 모드로 암호화/복호화 하는 클래스이다. 그럼 AES란 또 무엇일까?

표준대칭키 알고리즘 중 하나이다. 암호문을 생성(암호화)할 때 사용하는 키와 암호문으로부터 평문을 복원(복호화)할 때 사용하는 키가 동일한 암호 시스템이다.

전체적인 암호화과정은 암호블록체인 알고리즘을 활용하고 체인별 암호화/복호화시에는 대칭키알고리즘 을 활용하는 것이다.

대칭키의 대표적인 알고리즘은 DES(Data Encryption Standard)였다. 하지만 DES는 54bit의 키를 사용하여 현재 컴퓨팅 기술에 보안에 취약하였다. 이에 미국의 표준기술연구소(NIST)는 1998년 DES를 대체할 최고 보안 규격의 알고리즘을 공모하였고 2001년 Rijndael(Submitted by Joan Daeman and Vincent Rijman)이 선정되어 AES알고리즘이 탄생하게 된다.

[AES의 특징]

- 입력 평문의 길이는 128비트로 고정
- AES의 기본 연산은 Byte 단위로 수행
- 사용하는 암호화 키의 길이는 128, 192, 256비트 중 하나를 선택할 수 있음
참고) Rijndael 알고리즘은 키의 길이와 입력 평문의 길이를 128, 192, 256 비트 중 선택 가능
- 알려진 모든 공격법에 대해 안전하도록 설계됨
- 모든 플랫폼에서 속도와 code compactness 면에서 효율적
Design simplicity
Suited for Smart cards

대칭키 암호 시스템은 알고리즘이 상대적으로 단순한 장점이 있지만 키 관리에 어려움이 많다.

시스템에 가입한 사용자들 사이에 매 두 사용자 마다 하나의 서로 다른 키를 공유해야 하기 때문에 n 명이 가입한 시스템에는 $\sim n \sim C \sim 2 \sim = n(n-1)/2$ 개의 키가 필요하다. 또 각 사용자는 $n-1$ 개의 키를 관리해야 하는 부담이 있다. 이는 매우 큰 단점으로 키 관리가 상대적으로 용이한 공개키 암호 시스템의 출현 의 계기가 되었다.

대표적인 공개키 암호 시스템이 RSA 이다. 암호화뿐만 아니라 전자서명이 가능한 최초의 알고리즘 으로 알려져 있다. RSA가 갖는 전자서명 기능은 인증을 요구하는 전자 상거래 등에 RSA의 광범위한 활용을 가능하게 하였다. RSA 암호체계의 안정성은 큰 숫자를 소인수 분해하는 것이 어렵다는 것에 기반을 두고 있다. 그려므로 큰 수의 소인수 분해를 획기적으로 빠르게 할 수 있는 알고리즘이 발견된다면 이 암호 체계는 가치가 떨어질 것이다. 1993년 피터 쇼어는 쇼어 알고리즘을 발표하여, 양자 컴퓨터를 이용하여 임의의 정수를 다항 시간 안에 소인수 분해하는 방법을 발표하였다. 따라서 양자 컴퓨터가 본격적으로 실용화되면 RSA 알고리즘은 무용지물이 될 수도 있으나 양자 컴퓨터가 이 정도 수준으로 실용화되려면 아직 여려 해가 더 필요할 것으로 보인다.

RSA는 두 개의 키를 사용 한다. 일반적으로 공개키(public key)는 모두에게 알려져 있으며 메시지를 암호화(encrypt)하는데 쓰이며, 암호화된 메시지는 개인키(private key)를 가진 자만이 복호화(decrypt)하여 열어볼 수 있다. 하지만 RSA 공개키 알고리즘은 이러한 제약조건이 없다. 즉 개인키로 암호화하여 공개키로 복호화할 수 있다.

공개키 알고리즘은 누구나 어떤 메시지를 암호화할 수 있지만, 그것을 해독하여 열람할 수 있는 사람은 개인키를 지닌 단 한 사람만이 존재한다는 점에서 대칭키 알고리즘과 차이를 가진다.

RSA는 소인수 분해의 난해함에 기반하여, 공개키만을 가지고는 개인키를 쉽게 짐작할 수 없도록 디자인되어 있다.

메시지와 공개키 모두를 알 수 있다면 변조된 메시지를 보낼 수 있기 때문에, 실제로는 수신측의 공개키만을 사용하여 암호화하는 경우는 드물다. 송수신 양측의 키쌍을 사용하는 방법으로는 A의 개인키로 암호화 -> B의 공개키로 암호화 한 메시지를 전달하고 복호화 과정은 B의 개인키로 복호화 -> A의 공개키로 복호화로 구성된 방식이 일반적이다.

즉, 사용목적에 따라 대칭키 방식 공개키 방식을 사용할 수 있다. 다수의 참여자가 활동하는 환경에서는 공개키 방식을 사용하는 것이 실용적이지만, 개인적인 업무 혹은 소수집단에서의 활용측면에서는 대칭키 방식도 유용하다.

① 모듈 로드

콘솔에서 아래 명령을 통해 라이브러리를 설치한다.

```
> pip install pycryptodomex
```

```
Collecting pycryptodomex
  Downloading https://files.pythonhosted.org/packages/12/0d/c11d5fdc304b38968a530f87ab8bf6167111d1c346c0090bd4f493f09a4f
/pycryptodomex-3.9.4-cp38-cp38-win_amd64.whl (10.1MB)
|████████████████████████████████████████████████████████████████████████████████████████████████████████████████████| 10.1MB 6.8MB/s
Installing collected packages: pycryptodomex
  Running setup.py install for pycryptodomex ... done
```

② 블럭사이즈에 대한 패딩로직

스트링 문자열을 encrypt에 인자로 전달시 입력받은 데이터의 길이가 BLOCK_SIZE의 배수가 아닐때 패딩개념이 필요하다. AES에서는 BLOCK_SIZE가 128bit 즉 16바이트로 고정되는데, 아래 코드를 통해 자동 패딩처리한다. 한글에 대한 처리를 위해 pad시에 len(s.encode('utf-8')) 처리가 반드시 필요하다. 영문과 기호는 문자당 1바이트지만 한글은 문자당 2바이트이기 때문이다. len()함수를 활용해 길이를 통해 바이트 계산을 하는 방식이므로 'utf-8' 변환하지 않을 경우 오류가 발생하게 된다.

```
BS = 16
pad = lambda s: s + (BS - len(s.encode('utf-8')) % BS) * chr(BS - len(s.encode('utf-8')) % BS)
unpad = lambda s : s[:-ord(s[len(s)-1:])]
```

③ 암호화알고리즘 클래스 생성

```
class AESCipher:
    def __init__( self, key ):
        self.key = key

    def encrypt( self, raw ):
        raw = pad(raw)
        iv = Random.new().read( AES.block_size )
        cipher = AES.new( self.key, AES.MODE_CBC, iv )
        return base64.b64encode( iv + cipher.encrypt( raw.encode('utf-8') ) )
```

```

def decrypt( self, enc ):
    enc = base64.b64decode(enc)
    iv = enc[:16]
    cipher = AES.new(self.key, AES.MODE_CBC, iv )
    return unpad(cipher.decrypt( enc[16:] ))

```

encrypt 함수의 내부를 보면 입력받은 raw데이터를 raw.encode('utf-8') 16바이트 바이너리로 변환 후, 다시 base64.b64encode() 인코딩 한다. 즉 문자열을 기준으로 해서 byte화시키는 문자인코딩과는 다르게 bytes를 기준으로 문자열화 시키는 인코딩 방식 이 필요하다.

④ KEY설정

AES256을 구현하기 위해 32바이트의 키를 생성한다. ($2^{32} = 256$) KEY값은 아래와 같이 리스트 배열로 바로 지정하거나, time함수와 hashlib를 통해 임의로 생성해볼 수 있다.

(1) 리스트에 셋팅하는 방식

```

key = [0x10, 0x01, 0x15, 0x1B, 0xA1, 0x11, 0x57, 0x72, 0x6C, 0x21, 0x56, 0x57, 0x62, 0x16, 0x05,
0x3D, 0xFF, 0xFE, 0x11, 0x1B, 0x21, 0x31, 0x57, 0x72, 0x6B, 0x21, 0xA6, 0xA7, 0x6E, 0xE6, 0xE5,
0x3F]

```

(2) 임의로 생성하는 방식

```

import base64
import hashlib

def make_pass():
    timekey = int(time.time())
    return str(timekey)

password = make_pass().encode('utf-8')

```

⑤ 암호화/복호화

데이터 스트링 셋팅시 '표현에 유의하자' 자주 사용되는 '를 문자열 내에서 표현하기 위해서는 문자열을 감싸는 표현은 쌍따옴표"를 사용한다. 역으로 문자열 내에서 "를 표현하기 위해서는 문자열을 '를 사용해야 한다.

```

data = "Iran has seized a foreign oil tanker in the Persian Gulf that was smuggling fuel to some
Arab states, according to a state television report on Sunday. The report said that the tanker
had been detained and the ship's foreign crew held by the country's elite Islamic Revolutionary
Guards Corps."

```

데이터암호화

```

encrypted_data = AESCipher(bytes(key)).encrypt(data)
encrypted_data

```

```
b'643crQRSMHsahr+JNXabtD9yNhdFs9+4UmWr8amNR1JGZHbqsYSh/+eDSFRmkuTTuEXTL5a6VzbIG2f1MGcYW89OGmiwBs/1X/7QcX2V+SDyvwDUJ6CZBncSpQ30vSZftWSXIDtnuuhCF8M4CvuH1SgsHyhc e0JEpVeisNr9fwV0PYmzYp0QETZ4V2yzwaBToOVrcgJOCzigWGT4JoMSbtX0Z5T5oFnwCzIVULYAvRtsxW8 MSh4e5lznEBGnM1tpVSMpmmdETLPyXDoAp/y5N+6jk0YZ5RcetSA83iO407iR4DpNLLVbd+T+3i2Icw7Zq DZeilyxbBFepGvqJAg6qgNKbLobpTB+/BugzDjiSVUX6JWTT1XVynxagktXWPPRmIWfbBjd/QI3wZSOVXHp 6WtlTloWY6Zj+1wuaGk='
```

데이터복호화

```
decrypted_data = AESCipher(bytes(key)).decrypt(encrypted_data)  
decrypted_data.decode('utf-8')
```

```
b'Iran has seized a foreign oil tanker in the Persian Gulf that was smuggling fuel to some Arab states, according to a state television report on Sunday. The report said that the tanker had been detained and the ship's foreign crew held by the country's elite Islamic Revolutionary Guards Corps.'
```

-- 회원 가입 페이지에서 비밀번호를 암호화하기

먼저 암호화 시키기 위한 메소드를 만든다.

```
member/views.py  
1 import hashlib  
2 def pwEncrypt(USER_PW):  
3     result = hashlib.sha256(USER_PW.encode())  
4     return result.hexdigest()
```

1행에서 암호화를 하기 위한 모듈을 import한다.

2행에서 사용자 비밀번호를 매개변수가 받는다.

3행에서 암호화 방식은 sha이고 256비트로 USER_PW(사용자 비밀번호)를 암호화한다.

암호화 메소드를 회원정보에 적용시키기 위해 memberJoinAction()메소드에 pwEncrypt()메소드를 적용시킨다.

```
member/views.py  
1 def memberJoinAction(request):  
2     ... 중략  
3         if form.cleaned_data['USER_PW'] == form.cleaned_data['USER_PW_Con']:  
4             member = Member()  
5             member.USER_ID = form.cleaned_data['USER_ID']  
6             member.USER_PW = pwEncrypt(form.cleaned_data['USER_PW'])  
7             member.USER_NAME = form.cleaned_data['USER_NAME']  
8             member.USER_BIRTH = form.cleaned_data['USER_BIRTH']  
9             member.USER_GENDER = form.cleaned_data['USER_GENDER']  
10            member.USER_EMAIL = form.cleaned_data['USER_EMAIL']  
11            member.USER_ADDR = form.cleaned_data['USER_ADDR']  
12            member.USER_PH1 = form.cleaned_data['USER_PH1']
```

13	member.USER_PH2 = form.cleaned_data['USER_PH2']
14	member.USER_REGIST = datetime.now()
15	print(pwEncrypt(form.cleaned_data['USER_PW']))
16	... 중략

6행에서 암호화하기 위한 pwEncrypt()를 적용시킨다.

-- 로그인할 시 입력한 비밀번호를 암호화를 하여 비교해야 한다.

Main/views.py

1	from member.views import pwEncrypt
2	
3	def loginAction(request):
4	... 중략
5	pw = request.POST['USER_PW']
6	member = Member.objects.get(USER_ID=id1);
7	if member.USER_PWD == pwEncrypt(pw):
8	try :
9	request.session['member'] = {}
10	request.session['member']['USER_ID'] = member.USER_ID
11	request.session['member']['USER_PWD'] = member.USER_PWD
12	request.session['member']['USER_NAME'] = member.USER_NAME
13	request.session['member']['USER_EMAIL'] = member.USER_EMAIL
14	response = HttpResponseRedirect("/")
15	... 중략
16	except :
17	print("로그인 되지 않았습니다.")
18	else :
19	... 중략

5행은 로그인 창으로부터 받아온 값을 POST로 받아 pw변수에 저장

7행을 변경한다. 6행에서 데이터베이스로부터 가져온 member값의 사용자 비밀번호가 암호화가 되어 있으므로 로그인 창에서 입력한 사용자 비밀번호를 pwEncrypt()메소드를 이용하여 암호화한 후 비교한다.

-- 내 정보 상세보기

회원이 가입할 때 입력한 정보를 보고 수정하거나 탈퇴 할 수 있도록 구현해 보자.

main.html에서 내정보 보기를 클릭하면 "member/memberDetail"로 이동하도록 되어있다.

"member/urls.py"에서 해당경로를 받아 views.py에서 함수를 실행할 수 있게 한다.

member/urls.py

1	from django.conf.urls import url
2	from . import views
3	
4	app_name = 'member'
5	urlpatterns = [

```

6 ... 중략
7     url('memberDetail', views.memberDetail),
8 ]

```

7행에서 main.html로부터 “내 정보”를 클릭한 경로를 받아 views.py에 있는 memberDetail()메소드를 실행한다.

member/views.py

```

1 ... 중략
2 def memberDetail(request):
3     USER_ID = request.session['member']['USER_ID']
4     member = Member.objects.get(USER_ID=USER_ID);
5     context = {'member': member}
6     return render(request, 'member/memberDetail.html', context)

```

2행에서 urls.py에서 실행시킬 memberDetail()메소드를 선언한다.

3행은 현재 접속중인 사용자의 정보를 session에 저장되어 있는 USER_ID값을 받아온다.

4행은 MEMBER 테이블에서 USER_ID에 해당하는 정보를 받아와 MEMBER에 저장한다.

5행에서 context 딕셔너리를 만들어 6행에서 HTML문서에 전달한다.

member/memberDetail.html

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>
7 <script type="text/javascript">
8 $(function(){
9     $("#modify").click(function(){
10         location.href="memberModify";
11     })
12     $("#memDel").click(function(){
13         location.href="memberDelete";
14     })
15 });
16 </script>
17 </head>
18 <body>
19 이름 : {{ member.USER_ID }} <br />
20 아이디 : {{ member.USER_NAME }} <br />
21 이메일 : {{ member.USER_EMAIL }} <br />
22 생년월일 : {{member.USER_BIRTH | date:"Y-m-d"}} <br />
23 {% if member.USER_GENDER == 'M' %}<br />

```

```

24 성별 : 남 <br />
25 {% else %}
26 성별 : 여 <br />
27 {% endif %}
28 연락처 1 : {{ member.USER_PH1 }} <br />
29 연락처 2 : {{ member.USER_PH2 }} <br />
30 등록일 : {{member.USER_REGIST | date:"Y-m-d f"}} <br />
31 주소 : {{ member.USER_ADDR }} <br />
32
33 <input type="button" name="modify" id = "modify" value="수정">
34 <input type="button" value="취소" onclick="javascript:history.back();">
35 <input type="button" name="memDel" id = "memDel" value="탈퇴">
36
37 </body>
38 </html>

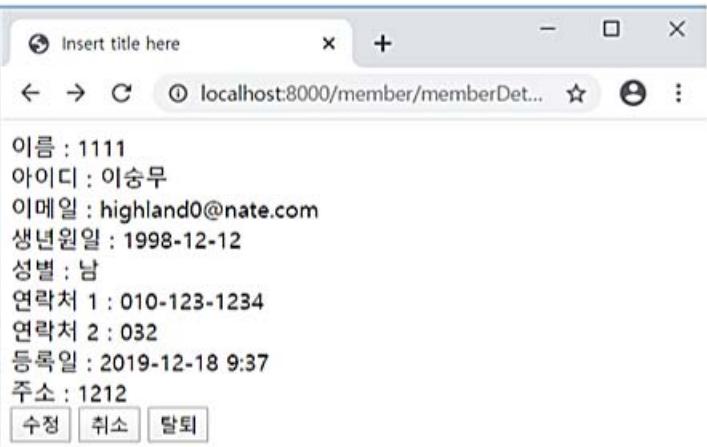
```

context딕셔너리에 있는 member에 저장된 사용자정보를 받아온다.

member에는 models.py에 만든 Member모델에 저장되어 있으므로 모델에 있는 딜드명으로 데이터를 가져올 수 있다.

22행과 30행에서는 시간을 사용자가 원하는 형태로 가져오기 위해 template을 사용했다.

33행에서는 9행에서 11행에서 수정페이지로 이동하기 위해 “memberModify”주소로 이동한다.



'날짜' Template

{date자료형|date:"표현식"} 을 넣으면 되는데.. 기준 시간: 2012-11-26 05:53:04 (KST)

- a, A: 오전, 오후(해당 언어를 따라가는 듯 하다)
- b: 월 (11월 이런식으로 표기됨)
- c: 년-월-일T시:분:초 로 표기됨
- d: 날 만 나옴
- D: 요일(월요일)
- E: 월(11월)
- f:[시]:분
- F: 또 월이 나옴(11월)
- g,G: 시간이 숫자만 나옴(5시면 5)
- h,H: 시간
- i:분

```
j: 일이 나옴(숫자만)
l: 월요일
n,m: 월 숫자만
N,M: 월 뒤에'월'글자 붙음. 11월
o: 년도 나옴
O: GTC +몇인지 (+09:00이 나오므로 추측)
P: 시:분 오전/오후 (5:53 오전 이렇게 나옴)
```

```
r: 요일, 일 월 년 시:분:초 GTC
```

```
s: 초
```

```
T: KST(시간 대역)
```

```
U: 타임스탬프 출력됨(1353876784)
```

```
y: 12 (년도의 뒷 2개 숫자인듯?)
```

```
Y: 2012(년도 4자리 인듯)
```

```
ps)
```

```
년-월-일 로 표기하고 싶다면, {{date|date:"Y-m-d"}}
```

참고 사이트

<https://docs.djangoproject.com/en/dev/ref/templates/builtins/?from=olddocs>

-- memberDetail페이지에서 memberModify페이지로 이동하기 위해 먼저 member/urls.py에서 views.py파일을 실행하도록 설정해야한다.

member/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     ... 중략
7     url('memberDetail', views.memberDetail),
8     url('memberModify', views.memberModify),
9 ]
```

8행에서 views.py에 있는 memberModify()메소드를 실행시킨다

member/views.py

```
1 ... 중략
2 def memberModify(request):
3     USER_ID = request.session['member']['USER_ID']
4     member = Member.objects.get(USER_ID=USER_ID);
5     context = {'member': member}
```

```
6     return render(request, 'member/memberModify.html', context)
```

2행은 member.urls.py에서 실행할 수 있게 memberModify()메소드를 정의한다.

3행에서 session에 있는 USER_ID값을 USER_ID에 저장한다.

4행에서 USER_ID에 해당하는 데이터를 MEMBER테이블로부터 받아온다.

5행에서 member를 딕셔너리로 만들어 context 저장한다.

6행에서 context를 memberModify.html페이지로 전송한다.

member/memberModify.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js" ></script>
7 <script type="text/javascript">
8 $(function(){
9     $("#memPw").click(function(){
10         location.href="changePassword";
11     })
12 });
13 </script>
14 </head>
15 <body>
16 <form name = "frm" id="frm" method = "post" action="memberModifyPro">{% csrf_token %}
17 <table width = 600 align = "center" border = 1 >
18 <tr><th><label for="id_USER_ID">사용자 아이디:</label></th><td>{{ member.USER_ID }}</td></tr>
19 <tr><th><label for="id_USER_PW">사용자 비밀번호:</label></th><td><input type="password" name="USER_PW" maxlength="100" required id="id_USER_PW"></td></tr>
20 <tr><th><label for="id_USER_NAME">사용자 이름:</label></th><td>{{ member.USER_NAME }}</td></tr>
21 <tr><th><label for="id_USER_BIRTH">사용자 생년월일:</label></th><td><input type="text" name="USER_BIRTH" required id="id_USER_BIRTH" value="{{member.USER_BIRTH | date:'Y-m-d'}}"><br>
<span class="helptext">yyyy-MM-dd 형식으로 입력해주세요</span></td></tr>
22 <tr><th><label for="id_USER_GENDER">성별:</label></th><td>{% if member.USER_GENDER == 'M' %} 남 {% else %} 여 {% endif %}</td></tr>
23 <tr><th><label for="id_USER_EMAIL">사용자 이메일:</label></th><td><input type="email" name="USER_EMAIL" required id="id_USER_EMAIL" value="{{ member.USER_EMAIL }}"></td></tr>
24 <tr><th><label for="id_USER_ADDR">사용자 주소:</label></th><td><input type="text"
```

```

name="USER_ADDR"      maxlength="100"      required      id="id_USER_ADDR"      value="{{ member.USER_ADDR }}">></td></tr>
25 <tr><th><label    for="id_USER_PH1">사용자      연락처1:</label></th><td><input    type="text"
name="USER_PH1"      maxlength="13"      required    id="id_USER_PH1"    value="{{ member.USER_PH1
}}}></td></tr>
26 <tr><th><label    for="id_USER_PH2">사용자      연락처2:</label></th><td><input    type="text"
name="USER_PH2"      maxlength="13"      id="id_USER_PH2"    value="{{ member.USER_PH2
}}}></td></tr>
27     <tr>
28         <td colspan = 2>
29             <input type= "submit" value="수 정" />
30             <input type= "reset" value="다시 입력" />
31             <input type= "button" value="비밀번호변경" id="memPw"/>
32         </td>
33     </tr>
34 </table>
35 </form>
36 </body>
37 </html>

```

18행부터 26행에 있는 “{{}}”(더블 중괄호)안에 있는
값은 member딕셔너리에 있는 Member모델에 저
장된 값을 출력한 것이다.

29행은 수정을 완료한후 “수정”버튼을 클릭하면 16
행에 있는 “memberModifyPro”페이지로 이동하게
된다.

사용자 아이디:	1111
사용자 비밀번호:	(empty)
사용자 이름:	이승무
사용자 생년월일:	1998-12-12 yyyy-MM-dd 형식으로 입력해주세요
성별:	남
사용자 이메일:	highland0@nate.com
사용자 주소:	1212
사용자 연락처1:	010-123-1234
사용자 연락처2:	032
<input type="button" value="수 정"/> <input type="button" value="다시 입력"/> <input type="button" value="비밀번호변경"/>	

member/urls.py

```

1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     url('memberModifyPro', views.memberModifyPro),
7     ... 중략
8     url('memberDetail', views.memberDetail),
9     url('memberModify', views.memberModify),

```

[]
6행을 추가하여 views.py에서 memberModifyPro()메소드를 실행할 수 있도록 정의한다.

member/views.py	
1	... 중략
2	@csrf_exempt
3	def memberModifyPro(request):
4	userId = request.session['member']['USER_ID']
5	userPw = pwEncrypt(request.POST['USER_PW'])
6	try:
7	member = Member.objects.get(USER_ID = userId, USER_PW = userPw)
8	member.USER_BIRTH = request.POST['USER_BIRTH']
9	member.USER_EMAIL = request.POST['USER_EMAIL']
10	member.USER_ADDR = request.POST['USER_ADDR']
11	member.USER_PH1 = request.POST['USER_PH1']
12	member.USER_PH2 = request.POST['USER_PH2']
13	member.save()
14	return HttpResponseRedirect("memberDetail")
15	except:
16	return HttpResponseRedirect("memberModify?num=1")

4행에서 session에 저장된 USER_ID값을 받아온다.

5행은 memberModify.html페이지로부터 받아온 패스워드를 암호화하여 저장한다.

6행은 try는 7행에서 데이터를 가져오지 못한 경우 exception이 발생하므로 오류를 처리하기 위해 사용한다.

7행은 Member 테이블로부터 USER_ID와 USER_PW에 해당하는 데이터를 불러와 Member 모델에 저장하고 member에 대입한다.

8행부터 12행은 memberModify.html페이지로부터 받아온 값을 member변수에 있는 Member모델에 각각의 값을 저장한다.

13행은 Member 테이블에 있는 값을 Member모델에 있는 값으로 변경하기 위해 전송한다.

14행은 update가 완료 되면 detail페이지로 이동한다.

16행은 7행에서 데이터를 가져 오지 못한 경우 exception이 발생하면 memberModify페이지로 이동할 때 num에 1값을 저장하여 전송하게 한다. 1을 전송하므로 비밀번호가 틀렸음을 알려준다.

-- memberModify페이지로 전송할 때 num에 있는 1값을 받을 수 있어야 하므로 views.py파일을 수정한다.

member/views.py	
1	... 중략
2	def memberModify(request):
3	USER_ID = request.session['member']['USER_ID']
4	member = Member.objects.get(USER_ID=USER_ID);
5	context = {}
6	try:
7	request.GET['num']

```

8     context = {'member': member, 'error': '비밀번호가 틀렸습니다.'}
9 except:
10    context = {'member': member}
11 return render(request, 'member/memberModify.html', context)

```

5행부터 11행으로 수정한다.

6행은 7행에서 GET방식으로 num 값이 오지 않은 경우 exception이 발생하게 된다.

7행은 GET방식으로 num값이 넘어온 것을 의미한다.

8행은 member와 error을 딕셔너리로 만들어 context에 저장하여 error은 비밀번호가 틀렸음을 memberModify.html에 전달한다.

11행은 member/memberModify.html페이지를 열면서 context값을 전달한다.

-- member/memberModify.html에 error를 받을 수 있게 수정한다.

member/memberModify.html

```

1 ... 중략
2 <tr><th><label for="id_USER_PH2">사용자 연락처2:</label></th><td><input type="text"
3 name="USER_PH2" maxlength="13" id="id_USER_PH2" value="{{ member.USER_PH2
4 }}"></td></tr>
5     {% if error %}
6         <tr>
7             <td colspan = 2>
8                 <h3 style="color:red">{{error}}</h3>
9             </td>
10        </tr>
11    {% endif %}
12    <tr>
13        <td colspan = 2>
14            <input type= "submit" value="수 정" />
15            <input type= "reset" value="다시 입력" />
16            <input type= "button" value="비밀번호변경" id="memPw"/>
17        </td>
18    </tr>
19 </table>
20 </form>
21 </body>
22 </html>

```

6행부터 10행은 context에 있는 error를 출력할 수 있게 추가한다.

8행에서 error를 출력할 수 있게 한다.

-- 비밀번호 변경

member/memberModify.html페이지에서 “비밀번호 변경”버튼을 클릭하면 “changePassword”로 이동하게 된다. 비밀번호를 변경하기 위한 확인 페이지로 먼저 이동해보자.

“changePassword”으로 이동하기 위해서는 member/urls.py에 추가한다.

member/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     ... 중략
7     url('memberModify', views.memberModify),
8     url('changePassword', views.changePassword),
9 ]
```

8행을 추가한다,

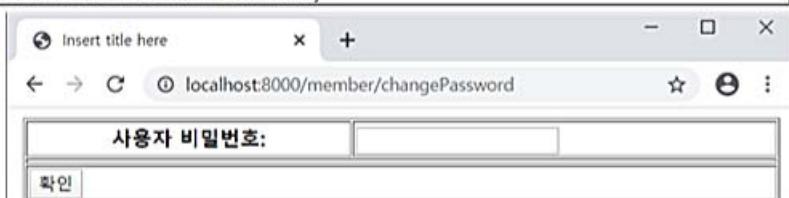
8행에서 “member/changePassword” 주소가 넘어오면 views.py에 있는 changePassword()메소드를 실행시켜도록 한다.

member/views.py

```
1 ... 중략
2 def changePassword(request):
3     return render(request, 'member/memberPassword.html')
```

2행에서 changePassword()메소드를 선언한다.

3행에서 member/memberPassword.html 페이지를 보내준다.



member/memberPassword.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <form name = "frm" id="frm" method = "post" action="memberPasswordPro">{% csrf_token %}
9 <table width = 600 align = "center" border = 1 >
10    <tr><th><label for="id_USER_PW">사용자 비밀번호:</label></th><td><input
11 type="password" name="USER_PW" maxlength="100" required id="id_USER_PW"></td></tr>
12    <tr>
13        <td colspan = 2>
14            {% if error %}
15                <h3 style="color:red">{{error}}</h3>
16            {% endif %}
17        </td>
```

```

18    </tr>
19    <tr><td colspan=2><input type="submit" value="확인" /></td></tr>
20  </table>
21  </form>
22  </body>
23  </html>

```

15행은 “memberPasswordPro”로 이동할 때 비밀번호가 틀릴 경우 오류 메세지가 출력된다.

19행에서 “submit”을 하면 비밀번호 변경 페이지인 “memberPasswordPro”로 이동하게 된다.

-- 비밀번호 변경 페이지로 이동

“memberPasswordPro”로 이동하기 위해 member/urls.py파일에 추가한다.

member/urls.py

```

1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     ... 중략
7     url('changePassword', views.changePassword),
8     url('memberPasswordPro', views.memberPasswordPro),
9 ]

```

8행을 추가한다.

8행에서 views.py에 있는 memberPasswordPro()메소드를 실행한다.

member/views.py

```

1 ... 중략
2 def memberPasswordPro(request):
3     userId = request.session['member']['USER_ID']
4     userPw = pwEncrypt(request.POST['USER_PW'])
5     try:
6         member = Member.objects.get(USER_ID = userId, USER_PW = userPw)
7         return render(request, 'member/memberPasswordPro.html')
8     except:
9         return render(request, 'member/memberPassword.html',{'error':'비밀번호가 틀렸습니다.'})
10

```

2행은 memberPasswordPro()메소드를 선언한다.

3행은 session으로부터 USER_ID값을 받아온다.

4행은 “memberPassword.html”로부터 받아온 비밀번호를 저장한다.

5행은 6행에서 데이터를 가져오지 못해 발생할 exception을 처리한다.

6행은 USER_ID와 USER_PW가 있는지 데이터베이스로부터 가져오고 데이터를 가져오지 못하면 9행이 실행됨

다.

9행은 비밀번호가 틀린 경우 “memberPassword.html”를 사용자에게 보내주면서 error메세지도 같이 보내준다.

현재 비밀번호:	<input type="password"/>
변경할 비밀번호:	<input type="password"/>
변경할 비밀번호 확인:	<input type="password"/>
<input type="button" value="확인"/>	

member/memberPasswordPro.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <form name = "frm" id="frm" method = "post" action="memberPasswordModifyPro">{%
9 csrf_token %}
10 <table width = 600 align = "center" border = 1 >
11     <tr><th><label for="id_USER_PW">현재 비밀번호:</label></th><td><input type="password" name="USER_PW_CRRUNT" id="USER_PW_CRRUNT"></td></tr>
12     <tr><th><label for="id_USER_PW">변경할 비밀번호:</label></th><td><input type="password" name="USER_PW_NEW" id="USER_PW_NEW"></td></tr>
13     <tr><th><label for="id_USER_PW">변경할 비밀번호 확인:</label></th><td><input type="password" name="USER_PW_RE_NEW" id="USER_PW_RE_NEW"></td></tr>
14     <tr>
15         <td colspan = 2>
16             {% if error %}
17                 <h3 style="color:red">{{error}}</h3>
18             {% endif %}
19         </td>
20     </tr>
21 <tr><td colspan=2><input type="submit" value="확인" /></td></tr>
22 </table>
23 </form>
24 </body>
25 </html>
```

11행은 현재 비밀번호를 입력한다.

12행은 새로운 비밀번호를 입력한다.

13행은 12행과 같은 비밀번호를 다시 한번 더 입력한다.

21행에서 “memberPasswordModifyPro”로 전송한다.

member/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     url('memberPasswordModifyPro', views.memberPasswordModifyPro),
7     ... 중략
8     url('changePassword', views.changePassword),
9     url('memberPasswordPro', views.memberPasswordPro),
10 ]
```

6행을 추가하한다. “memberPasswordModifyPro”주소로 들어오면 views.py에 있는 memberPasswordModifyPro메소드를 실행한다.

member/views.py

```
1 ... 중략
2 @csrf_exempt
3 def memberPasswordModifyPro(request):
4     userId = request.session['member']['USER_ID']
5     userPw = pwEncrypt(request.POST['USER_PW_CRRUNT'])
6     USER_PW_NEW = request.POST['USER_PW_NEW']
7     USER_PW_RE_NEW = request.POST['USER_PW_RE_NEW']
8     if USER_PW_NEW != USER_PW_RE_NEW :
9         return render(request, 'member/memberPasswordPro.html', {'error':'새 비밀번호와 비밀번호 확인이 다릅니다.'})
10    else:
11        try:
12            member = Member.objects.get(USER_ID = userId, USER_PW = userPw)
13            userNewPw = pwEncrypt(USER_PW_NEW)
14            member.USER_PW = userNewPw
15            member.save()
16            return HttpResponseRedirect("memberDetail")
17        except:
18            return render(request, 'member/memberPasswordPro.html', {'error':'비밀번호를 잘 못 입력하였습니다.'})
```

4행은 session으로부터 USER_ID값을 가져온다.

5행은 “memberPasswordPro.html”로 넘어온 현재 비밀번호를 받아 암호화한다.

6행과 7행은 각각 새로운 비밀번호와 비밀번호 확인을 받아온다.

8행은 비밀번호와 비밀번호 확인이 다른지를 확인한다.

9행은 비밀번호와 비밀번호 확인이 서로 다르면 “memberPasswordPro.html”페이지를 보내주면서 “오류메세지” 같이 보내준다.

12행에서는 USER_ID와 USER_PW에 해당하는 데이터가 MEMBER 테이블에 있으면 가져온다.

13행에서 새 비밀번호를 암호화시킨다.

14행에서 USER_PW에 암호화된 새로운 비밀번호를 저장한다.

15행에서 update가 된다.

16행에서 Detail 페이지로 이동한다.

18행에서 update가 되지 않은 경우 “memberPasswordPro.html”를 사용자에게 다시 보내주면서 오류메세지도 같이 보낸다.

-- 회원리스트

아이디	이메일	이름	가입일
2222	highland0@	이승무	2019-12-23 9:22
1111	highland0@	이승무	2019-12-18 9:37

[이전] [1] [이후]

회원리스트를 출력시 한 페이지에 원하는 갯만큼만 출력하기 위해서는 페이징처리를 해야한다.

페이징이란 프로그램이 한 번에 처리할 수 있는 적당한 크기(페이지)로 분할하여 페이지 단위로 처리하는 것을 말한다.

* main.html페이지에서 “회원리스트”를 클릭하면 “member/memberlist”주소를 요청하게 된다.

member/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     ... 중략
7     url('memberPasswordPro', views.memberPasswordPro),
8     url('memberlist', views.memberlist),
9 ]
```

8행을 추가하여 “member/memberlist”주소를 요청하게 되면 views.py에 있는 memberlist()메소드를 실행시킨다.

member/views.py

```
1 ... 중략
2 from django.core.paginator import Paginator
```

```

3 import math
4 def memberlist(request):
5     page = request.GET.get('page',1)
6     limit = 10
7     page_range = 10
8     members = Member.objects.all().order_by('-USER_REGIST')      # order_by('-field_name')
: 내림차순
9
10    paginator = Paginator(members, limit)
11    contacts = paginator.get_page(page)
12
13    current_block = math.ceil(int(page)/page_range)
14    start_block = (current_block-1) * page_range
15    end_block = start_block + page_range
16    p_range = paginator.page_range[start_block:end_block]
17
18    context = {'contacts': contacts,'p_range' : p_range,}
19    return render(request, 'member/memberList.html',context)

```

2행은 페이징을 하기 위해서는 paginator를 import를 해주어야 한다.

paginator는 게시판과 같은 목록이 주어져있을 때, 페이지 당 몇 개의 글을 보여줄지 지정해줄 수 있도록 도와주는 모듈입니다

5행은 'page'가 null이면 1값을 가지도록 한다.

6행은 한 페이지에 출력될 개수를 10개로 정한다.

7행은 페이지에 페이지 번호가 몇 개를 출력할 것인지 정한다.(10개 : 1 2 3 4 5 6 7 8 9 10)

8행에서 order_by('-USER_REGIST')는 "USER_REGIST"컬럼을 기준으로 내림차순으로 정렬하라는 것이다.

즉 select * from MEMBER order by USER_REGIST desc

오름 차순이 되려면 "order_by('USER_REGIST')"로 "-"를 제외하면 된다.

10행은 Paginator모듈을 이용하여 8행의 members로부터 limit(10개) 만큼씩 페이징을 한다.

11행은 현 페이지에 맞는 리스트를 가져온다.

“page”변수가 1이면 요청 페이지가 1페이지이므로 1번째부터 10번째까지

“page”변수가 2이면 요청 페이지가 2페이지이므로 11번째부터 20번째까지

“page”변수가 3이면 요청 페이지가 3페이지이므로 21번째부터 30번째까지

13행은 요청한 페이지 번호를 이용하여 첫페이지 번호를 가져오기 위한 연산이다.

ceil()함수는 소수점 올리고 정수로 반환한다. (0.3 => 1)

floor() : 소수점 버리고 정수로 반환한다 (1.6 => 1)

round() : 소수점 반올림하고 정수 반환한다. (0.5 => 1, 0.4 => 0)

14행은 페이징 번호의 첫 번째 번호를 가져온다.

15행은 페이징 번호의 마지막 번호를 가져온다.

16행은 페이징 번호를 팩터로 저장

17행은 출력 할 데이터와 페이징번호를 딕셔너리에 저장

18행은 “member/memberList.html”로 딕셔너리를 전송

```

member/memberList.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <table>
9 <tr><th>아이디</th><th>이메일</th><th>이름</th><th>가입일</th></tr>
10 {% for member in contacts %}
11 <tr><th><a href="memberInfo/{{member.USER_ID}}">
{{member.USER_ID}}</a></th><th>{{member.USER_EMAIL}}</th><th>{{member.USER_NAME}}</th>
<th>{{member.USER_REGIST | date:"Y-m-d f"}}</th></tr>
12 {% endfor %}
13 <tr align="center" height=20 >
14     <td colspan = 3>
15             {% if not contacts.has_previous  %}
16                 [이전]&nbsp;
17             {% else  %}
18                 [ <a href="memberlist?page={{ contacts.previous_page_number }}">이전
</a> ]&nbsp;
19             {% endif %}
20
21             {% for i in p_range %}
22                 [ <a href="memberlist?page={{i}}">{{ i }}</a> ]
23             {% endfor %}
24
25             {% if not contacts.has_next %}
26                 [이후]&nbsp;
27             {% else %}
28                 [ <a href="memberlist?page={{contacts.next_page_number }}">이후
</a>]&nbsp;
29             {% endif %}
30         </td>
31     <tr>
32     </table>
33     </body>
34     </html>

```

10행부터 12행은 "contacts"에 있는 값을 member에 담아 반복하여 출력한다.

11행은 회원의 아이디와 이메일 및 이름 그리고 등록일 등을 member로 부터 출력한다.

15행은 페이지 번호가 이전번호가 있으면 true를 반환하고 없으면 false를 반환한다.

18행에서 “previous_page_number”은 현재 페이지에서 -1된 페이지 번호를 가져온다.

21행부터 23행은 출력되어야 하는 페이지 번호를 반복문을 이용해서 출력한다.

25행에서 “contacts.has_next”는 다음 페이지 번호가 있으면 true, 없으면 false를 반환한다.

28행에서 “next_page_number”는 현 페이지번호에서 +1된 값을 가져온다.

-- 리스트 페이지에서 회원의 상세보기 보기

리스트에서 상세보기를 하기 위해서는 아이디를 클릭하면 그 아이디를 가진 회원의 상세보기 페이지로 넘어간다. 그러기 위해서는 “memberInfo/{{member.USER_ID}}”주소로 이동해야한다. 여기서 “member.USER_ID”는 회원의 아이디가 된다. 즉 아이디가 “1111”이라는 회원이라면 “memberInfo/1111”이 된다.

member/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     ... 중략
7     url('memberlist', views.memberlist),
8     url('memberInfo/(?P<USER_ID>\w+)/$', views.memberInfo),
9 ]
```

8행을 추가한다. ‘memberInfo/(?P<USER_ID>\w+)/\$’에서 “?P<USER_ID>”는 예를 들어 url이 “memberInfo/1111”라면 “1111”을 받아오는 변수이다. “\w”는 문자가 와야한다는 것이다. “+”모두 그리고 “views.py”에 있는 “memberInfo()”메소드를 실행시킨다.

“memberInfo()”메소드로 “<USER_ID>”값을 인자로 전달하게 된다.

member/views.py

```
1 ... 중략
2 def memberInfo(request, USER_ID):
3     member = Member.objects.get(USER_ID = USER_ID)
4     context = {'member': member, 'num' : 1}
5     return render(request, 'member/memberDetail.html', context)
```

2행은 memberInfo()메서드를 선언했고 아이디를 인자로 받는 매개변수
USER_ID가 선언되어 있다.

3행은 아이디가 “USER_ID”에 해당하는 회원을 출력해 온다.

4행은 딕셔너리를 만들어 “memberDetail.html” 전달하는 데 num이 이름 : 1111
“1”이면 “memberInfo()”메소드를 사용한 것을 알리기 위해 전송한다. 아이디 : 이승무
“memberDetail.html”은 “memberDetail()”메소드에서도 사용하는 페이
지이다. 이메일 : highland0@nate.com
생년원일 : 1998-12-12
성별 : 남

연락처 1 : 010-123-1234
연락처 2 : 032
등록일 : 2019-12-18 9:37
주소 : aaaaaaaaaaa

```
member/memberDetail.html
```

```
1 ... 중략
2
3 <script type="text/javascript">
4 $(function(){
5     $("#modify").click(function(){
6         location.href="memberModify";
7     })
8     $("#memDel").click(function(){
9         location.href="memberDelete";
10    })
11 });
12 </script>
13
14 ... 중략
15
16 {% if num != 1 %}
17 <input type="button" name="modify" id = "modify" value="수정">
18 <input type="button" value="취소" onclick="javascript:history.back();">
19 <input type="button" name="memDel" id = "memDel" value="탈퇴">
20 {% endif %}
21
22 </body>
23 </html>
```

16행과 20행을 추가 해주므로 “member/views.py”에서 memberInfo()메서드로부터 넘어온 num이 없어야 출력하도록 조건문을 사용했다.

-- 회원 탈퇴

main.html파일에서 “내 정보”를 클릭하면 “member/memberDetail.html”페이지로 넘어온다.

여기서 탈퇴 버튼을 누르면 회원삭제가 되도록 만든다.

위 표에서 19행을 클릭하면 8행에서 “memberDelete”로 이동하도록 했다.

```
member/urls.py
```

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'member'
5 urlpatterns = [
6     url('memberPasswordModifyPro', views.memberPasswordModifyPro),
7     url('memberModifyPro', views.memberModifyPro),
8     url('memberDelete', views.memberDelete),
9     ... 중략
10 ]
```

8행을 추가하여 views.py에 있는 memberDelete()메소드가 실행하도록 한다.

member/views.py	
1	... 중략
2	def memberDelete(request):
3	userId = request.session['member']['USER_ID']
4	member = Member.objects.get(USER_ID = userId)
5	member.delete ()
6	return HttpResponseRedirect("/logout")

3행에서 session에 있는 사용자를 userId변수에 저장한다.

4행에서 userId에 해당하는 사용자를 디비로부터 받아온다.

5행은 해당 사용자를 데이터베이스로부터 삭제한다.

6행은 로그인 정보를 삭제하기 위해 로그아웃을 한다.

session, cookie

<https://ssungkang.tistory.com/entry/Django%EB%A1%9C%EA%B7%B8%EC%9D%B8-%EC%9C%A0%EC%A7%80%ED%95%98%EA%B8%B0-%EC%BF%A0%ED%82%A4%EC%99%80-%EC%84%B8%EC%85%98>

20.10 답변형 자료실 만들기

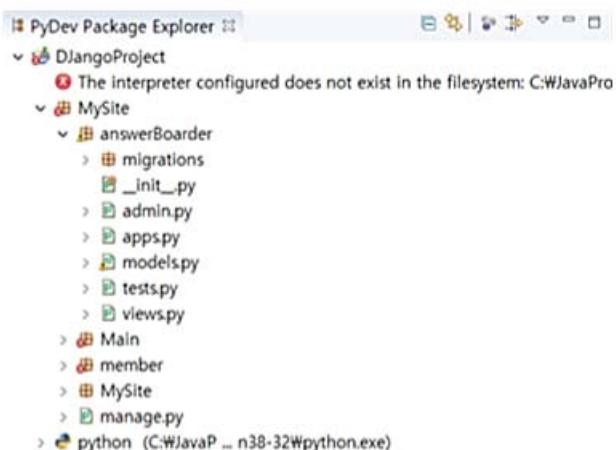
20.10.1 답변형 자료실 애플리케이션 생성

```
PS D:\Hk004\workspace\DJangoProject\MySite> python manage.py startapp answerBoarder
```

```
PS D:\Hk004\workspace\DJangoProject\MySite> ls
```

디렉터리: D:\Hk004\workspace4\DJangoProject\MySite

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	2019-12-23	오후 1:59	answerBoarder
d----	2019-12-09	오후 2:45	Main
d----	2019-12-10	오전 9:35	member
d----	2019-12-05	오후 2:14	MySite
d----	2019-12-10	오전 9:34	__pycache__
-a----	2019-12-05	오전 11:45	647 manage.py



(1) 답변형 자료실에 해당하는 Model Class를 만든다. 오라클 데이터베이스에서 테이블을 만들 필요는 없다.

데이터베이스 구조

```
create table ANSWERBOARDER(
    BOARD_NUM NUMBER(6) not null,
    USER_ID NVARCHAR2(20) not null,
    BOARD_NAME NVARCHAR2(20) not null,
    BOARD_PASS NVARCHAR2(200) not null,
    BOARD SUBJECT NVARCHAR2(50) not null,
    BOARD_CONTENT NVARCHAR2(2000) null,
    BOARD RE REF NUMBER(6) not null,
```

```

        BOARD_RE_LEVEL NUMBER(6) not null,
        BOARD_RE_SEQ NUMBER(6) not null,
        BOARD_READCOUNT NUMBER(6) not null,
        BOARD_DATE TIMESTAMP not null,
        BOARD_STORE_FILENAME NVARCHAR2(200) null,
        BOARD_ORIGINAL_FILENAME NVARCHAR2(200) null,
        BOARD_FILE_SIZE NUMBER(6) null
);

alter table ANSWERBOARDER
add constraint ANSWERBOARDER_BOARD_NUM_PK primary key (BOARD_NUM);

alter table ANSWERBOARDER
add constraint ANSWERBOARDER_USER_ID_FK foreign key (USER_ID)
references member(USER_ID) on delete cascade;

```

위 데이터 베이스 구조에 맞게 모델 클래스를 만든다.

answerBoarder/models.py	
1	from django.db import models
2	
3	# Create your models here.
4	from member.models import Member
5	# Create your models here.
6	from django.core.validators import MinLengthValidator
7	
8	from datetime import datetime
9	from django.conf import settings
10	
11	import os
12	# Create your models here.
13	
14	class AnswerBoarder(models.Model):
15	def __str__(self):
16	return self.BOARD_NUM
17	pw_validator = MinLengthValidator(8, "8자 이상이어야 합니다.")
18	BOARD_NUM = models.BigAutoField(blank=False, null=False, primary_key=True)
19	USER_ID = models.ForeignKey(Member, on_delete=models.CASCADE)
20	BOARD_NAME = models.CharField(max_length=20, blank=False, null=False)
21	BOARD_PASS = models.CharField(max_length=200, blank=False, null=False, validators=[pw_validator])
22	BOARD SUBJECT = models.CharField(max_length=50, blank=False, null=False)

```

23     BOARD_CONTENT = models.CharField(max_length=2000, blank=True, null=True)
24     BOARD_RE_REF = models.DecimalField(max_digits = 6, decimal_places = 0, blank=False,
25                                         null=False, default = 0)
26     BOARD_RE_LEV = models.DecimalField(max_digits = 6, decimal_places = 0, blank=False,
27                                         null=False, default = 0)
28     BOARD_RE_SEQ = models.DecimalField(max_digits = 6, decimal_places = 0, blank=False,
29                                         null=False, default = 0)
30     BOARD_READCOUNT = models.DecimalField(max_digits = 6, decimal_places = 0, blank=False,
31                                         null=False, default = 0)
32     BOARD_DATE = models.DateTimeField(blank=False, null=False, default=datetime.now)
33     BOARD_ORIGINAL_FILENAME = models.CharField(max_length=500, blank=True, null=True)
34     BOARD_STORE_FILENAME = models.CharField(max_length=500, blank=True, null=True)
35     BOARD_FILE_SIZE = models.CharField(max_length=200, blank=True, null=True)
36
37     def delete(self, *args, **kargs):
38         fileNames = self.BOARD_STORE_FILENAME.split("\\")
39         for index, value in enumerate(fileNames):
40             print(value)
41             if value is not "":
42                 os.remove(os.path.join(settings.MEDIA_ROOT+"\\"uploads\\\",value))
43         super(AnswerBoarder, self).delete(*args, **kargs)
44
45     class Meta:
46         db_table = "ANSWERBOARDER"

```

33행은 Trash 파일을 안남기기 위해 객체와 함께 파일 삭제하려면, 현재 AnswerBoarder 모델에 delete 함수를 오버라이딩(재정의) 한다.

모델 클래스를 만든 후 makemigrations을 해 준다.

> python manage.py makemigrations

PS D:\Hk004\workspace4\DJangoProject\MySite> python manage.py makemigrations

Migrations for 'member':

member\migrations\0002_auto_20191223_1549.py

- Alter field USER_PH1 on member
- Alter field USER_PWD on member

Migrations for 'answerBoarder':

answerBoarder\migrations\0001_initial.py

- Create model AnswerBoarder

PyDev Package Explorer

```

DjangoProject
  -> The interpreter configured does not exist in the filesystem: C:\JavaPro
  -> MySite
    -> answerBoarder
      -> migrations
        -> __init__.py
        -> 0001_initial.py
      -> admin.py
      -> apps.py
      -> models.py
      -> tests.py
      -> views.py
    -> Main
    -> member
    -> MySite
    -> management
  -> python (C:\JavaPro\... n38-32\python.exe)

```

```

0001_initial.py
1 | Generated by Django 3.0 on 2019-12-23 06:49
2 |
3 > import django.core.validators
4 from django.db import migrations, models
5 import django.db.models.deletion
6
7
8 class Migration(migrations.Migration):
9     initial = True
10
11     dependencies = [
12         ('member', '0002_auto_20191223_1549'),
13     ]
14
15     operations = [
16         migrations.CreateModel(
17             name='AnswerBoarder',
18             fields=[
19                 ('BOARD_NUM', models.DecimalField(decimal_places=0, max_digits=6, primary_key=True, serialize=False)),
20                 ('BOARD_NAME', models.CharField(max_length=20)),
21                 ('BOARD_PASS', models.CharField(max_length=200, validators=[django.core.validators.MinLengthValidator(8, '8자 이상이어야 합니다.')])),
22                 ('BOARD_SUBJECT', models.CharField(max_length=50)),
23                 ('BOARD_CONTENT', models.CharField(blank=True, max_length=2000, null=True)),
24                 ('BOARD_RE_REF', models.DecimalField(decimal_places=0, max_digits=6)),
25                 ('BOARD_RE_LEVEL', models.DecimalField(decimal_places=0, max_digits=6)),
26                 ('BOARD_RE_SEQ', models.DecimalField(decimal_places=0, max_digits=6)),
27                 ('BOARD_READCOUNT', models.DecimalField(decimal_places=0, max_digits=6)),
28                 ('BOARD_DATE', models.DateTimeField()),
29                 ('BOARD_ORIGINAL_FILENAME', models.CharField(blank=True, max_length=200, null=True)),
30                 ('BOARD_FILE_SIZE', models.DecimalField(blank=True, decimal_places=0, max_digits=6, null=True)),
31                 ('USER_ID', models.ForeignKey(on_delete=models.deletion.CASCADE, to='member.Member')),
32             ],
33             options={
34                 'db_table': 'ANSWERBOARDER',
35             },
36         ),
37     ],
38 )

```

> python manage.py migrate

Operations to perform:

Apply all migrations: admin, answerBoarder, auth, contenttypes, member, sessions

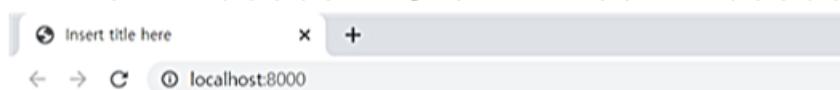
Running migrations:

Applying answerBoarder.0001_initial... OK

여기까지 데이터베이스를 django를 이용하여 데이터베이스를 만들었다.

이름	형식
BOARD_NUM	NOT NULL NUMBER(19)
BOARD_NAME	NVARCHAR2(20)
BOARD_PASS	NVARCHAR2(200)
BOARD_SUBJECT	NVARCHAR2(50)
BOARD_CONTENT	NVARCHAR2(2000)
BOARD_RE_REF	NOT NULL NUMBER(6)
BOARD_RE_LEVEL	NOT NULL NUMBER(6)
BOARD_RE_SEQ	NOT NULL NUMBER(6)
BOARD_READCOUNT	NOT NULL NUMBER(6)
BOARD_DATE	NOT NULL TIMESTAMP(6)
BOARD_ORIGINAL_FILENAME	NVARCHAR2(100)
BOARD_FILE_SIZE	NUMBER(6)
USER_ID	NOT NULL NVARCHAR2(20)

로그인 후 main페이지에서 답변형 자료실을 클릭하면 list페이지에서 등록페이지를 열수 있도록 한다.



내 정보 로그아웃 회원리스트 댓글 게시판 **답변형 자료실** 상품목록 단체 메일보내기 설문지

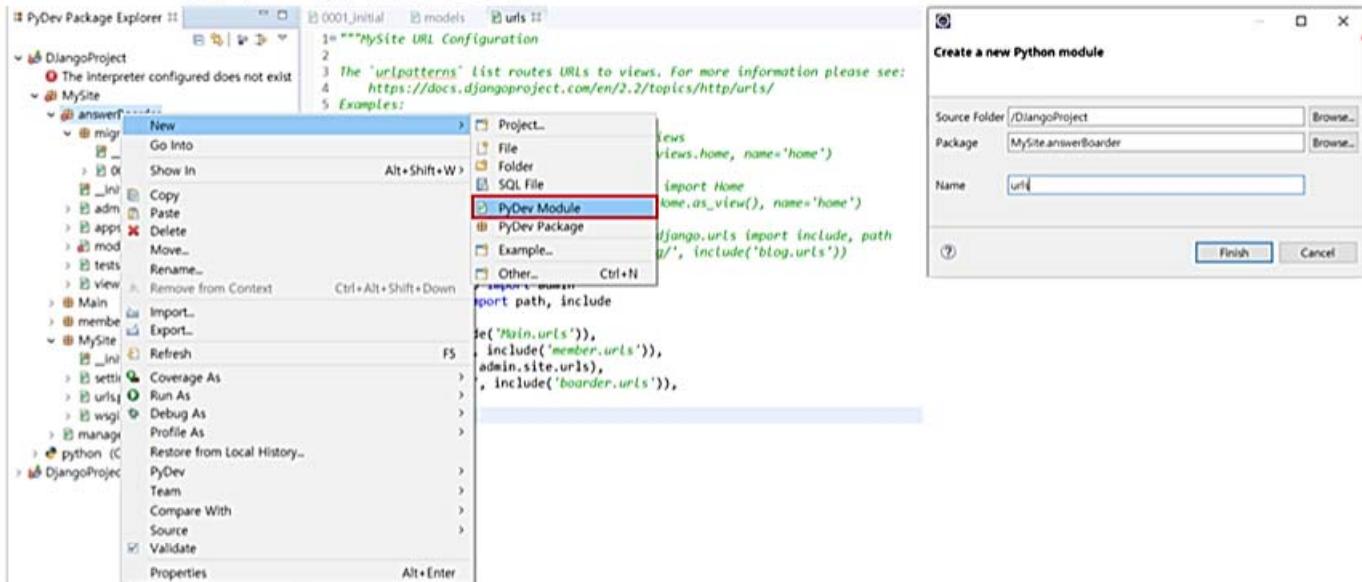
"<http://localhost:8000/board/answerBoardList>" 주소로 이동하도록 한다.

(2) "board"로 시작하는 주소가 이동시키기 위해 "MySite/urls.py"에 먼저 설정을 해준다.

```
# MySite/urls.py
1 from django.contrib import admin
2 from django.urls import path, include
3 from django.conf.urls import url
4 from core import views
5
6 from django.conf import settings
7 from django.conf.urls.static import static
8 urlpatterns = [
9     path('', include('Main.urls')),
10    path('member/', include('member.urls')),
11    path('admin/', admin.site.urls),
12    path('board/', include('answerBoarder.urls')),
13 ]
```

12행을 추가하여 boarder App에 있는 urls.py파일에 있는 내용을 include시킨다.

- "boarder" App에 urls.py파일을 만든다.



```
#answerBoarder/urls.py
1 from django.conf.urls import url
2 from . import views
3 urlpatterns = [
4     url('answerBoardList', views.answerBoardList),
5 ]
```

위 표와 같이 내용을 작성한다.

4행은 “/board/answerBoardList”로 주소로 왔을 때 “answerBoardList”인 경우 views.py에서 “answerBoardList”메소드를 실행시킨다.

(3) html 페이지를 열기 위해서는 “answerBoarder” App안에 “templates”폴더를 만들고 폴더 안에 “answerBoarder” 폴더를 만든다. 그리고 answerBoardList.html파일을 만든다.

answerBoarder/views.py

```
1 from django.shortcuts import render
2
3 # Create your views here.
4 def answerBoardList(request):
5     return render(request, 'answerBoarder/answerBoardList.html')
```

4행에서 “answerBoardList()”메서드를 선언한다.

5행에서 “answerBoardList.html”페이지를 전송한다.

answerBoarder/answerBoardList.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8     <a HREF="answerBoard"> 글쓰기 </a>
9 </body>
10 </html>
```

(4) “글쓰기”를 클릭하면 “http://localhost:8000/board/answerBoard”로 이동할 수 있게 “urls.py”에 코드를 추가한다.

answerBoarder/urls.py

```
1 from django.conf.urls import url
2 from . import views
3 urlpatterns = [
4     url('answerBoardList', views.answerBoardList),
5     url('answerBoard', views.answerBoard),
6 ]
```

6행을 추가하여 “http://localhost:8000/board/answerBoard”로 요청하면 “views.py”에서 “answerBoard”메소드가 실행되도록한다.

answerBoarder/views.py

```
1 from django.shortcuts import render
2 from .forms import BoardWriteForm
3
```

```

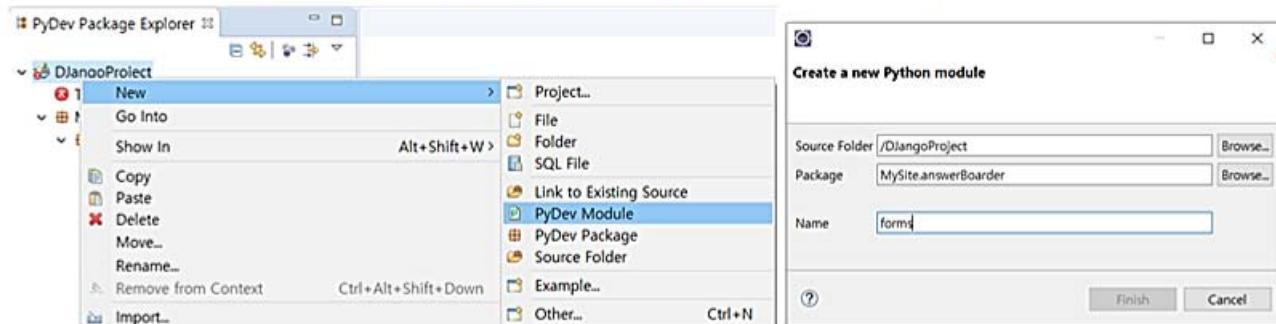
4 def answerBoard(request):
5     return render(request, 'answerBoarder/Board_Write.html',{'f':BoardWriteForm()})

```

2행은 추가하여 “Board_Write.html”페이지에 출력될 내용을 가져온다.

5행에서 “Board_Write.html”페이지를 전송하면서 “BoardWriteForm()”메소드를 실행하여 Form내용을 받아온다.

(5) forms.py파일을 만들어서 html페이지에 출력될 내용을 Model Class로부터 받아온다.



answerBoarder/forms.py

```

1 from django.forms import ModelForm
2 from .models import AnswerBoarder
3 from django import forms
4 from django.utils.translation import gettext as _
5
6 class BoardWriteForm(ModelForm): #회원가입을 제공하는 class이다.
7     BOARD_STORE_FILENAME = forms.FileField(label = '등록 할 파일',
8 required=False,widget=forms.ClearableFileInput(attrs={'multiple': True}))
9     class Meta:
10         model=AnswerBoarder
11         widgets = {'BOARD_PASS':forms.PasswordInput,'BOARD_CONTENT':forms.Textarea}
12         fields = ['BOARD_NAME','BOARD_PASS','BOARD SUBJECT','BOARD_CONTENT','BOARD_STORE_FILENAME']
13         labels = {
14             'BOARD_NAME': _('글쓴이'),
15             'BOARD_PASS': _('비밀번호'),
16             'BOARD SUBJECT': _('글 제목'),
17             'BOARD_CONTENT': _('글 내용'),
18         }

```

7행에서 BOARD_ORIGINAL_FILENAME는 “input type”이 “file”이어야 하므로 “forms.FileField()”을 사용했으며 file속성에 “multiple”을 가지도록 “widget=forms.ClearableFileInput(attrs={'multiple': True})”을 추가하였다.

9행은 “Board_Write.html”에 표시할 자료항목을 AnswerBoarder Model로부터 받아온다.

10행은 “BOARD_PASS”와 “BOARD_CONTENT” 위젯을 각각 “password”와 “Textarea”로 지정해 주었다.

11행은 “Board_Write.html” 페이지에 출력할 자료항목들을 AnswerBoarder Model Class로부터 가져와 나열

한다.

12행은 각 자료항목들에 대해서 출력될 내용을 정의한다.

(6) "Board_Write.html" 문서를 "templates/answerBoarder/" 폴더에 만든다.

```
answerBoarder/Board_Write.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <form name = "frm" id="frm" method = "post" action="answerBoardWritePro"
9 enctype="multipart/form-data">{{ csrf_token }}
10 <table width = 600 align = "center" border = 1 >
11     <tr align="center" valign="middle">
12         <td colspan="2">답변형 자료실</td>
13     </tr>
14     {{ f.as_table }}
15     {% if error %}
16     <tr>
17         <td colspan = 2>
18             <h3 style="color:red">{{error}}</h3>
19         </td>
20     {% endif %}
21     </tr>
22     <tr>
23         <td colspan = 2>
24             <input type= "submit" value="등록" />
25             <input type="button" value="뒤로"
26             onclick="javascript:history.back()"/>
27         </td>
28     </tr>
29 </table>
</form>
</body>
```

8행은 작성된 내용을 "answerBoardWritePro"로 전송한다.

13행은 "answerBoarder/forms.py"로 부터 받아온 내용을 table로 출력한다.

14행부터 19행은 전송오류가 발생한 경우 오류 메시지를 받아온다.

(7) "answerBoardList.html"에서 작성된 내용을 "answerBoardWritePro"주소에서 받을 수 있게 설정한다.

- 파일을 전송 받기 위해 "settings.py"파일을 변경한다.

MySite/settings.py

```
1 ... 중략  
2 STATIC_URL = '/static/'  
3 STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')  
4 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')  
5 MEDIA_URL = '/media/'  
6 MAX_UPLOAD_SIZE = "5242880"
```

2행은 “STATIC_URL”을 “/static/”경로로 잡아 준다.

3행은 이미지 파일이나 업로드한 파일의 경로를 “static” 폴더로 지정한다.

4행은 “BASE_DIR”디렉토리에 ‘media’폴더가 만들어진다.(파일을 업로드할 장소를 뜻한다)

5행은 MEDIA_ROOT를 사용하기 위한 경로를 지정한다.(MEDIA_ROOT에 접근할 URL을 뜻한다)

6행은 전송된 파일의 최대크기를 지정한다. (5M로 지정)

-- “MySite”에 있는 “urls.py”에 “MEDIA_URL”과 “MEDIA_ROOT”를 지정해 준다.

MySite/urls.py

```
1 from django.contrib import admin  
2 from django.urls import path, include  
3 from django.conf.urls import url  
4 from core import views  
5  
6 from django.conf import settings  
7 from django.conf.urls.static import static  
8 urlpatterns = [  
9     path('', include('Main.urls')),  
10    path('member/', include('member.urls')),  
11    path('admin/', admin.site.urls),  
12    path('board/', include('answerBoarder.urls')),  
13 ]  
14 urlpatterns += static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
```

14행을 추가하여 settings.py에 지정한 “static”경로와 “MEDIA_URL”경로를 지정한다.

MEDIA_ROOT에 저장이 되고, 요청할 때는 MEDIA_URL을 이용한다.

MEDIA_URL을 활성화 시켜줬다면 “<http://localhost:8000/media/파일명>”과 같은 URL로 파일을 불러올 수 있다

- “answerBoardWritePro”주소를 받을 수 있도록 “urls.py”파일을 수정한다.

answerBoarder/urls.py

```
1 from django.conf.urls import url  
2 from . import views  
3  
4 app_name = 'answerBoarder'  
5 urlpatterns = [  
6     url('answerBoardWritePro', views.answerBoardWritePro),
```

```

7     url('answerBoardList', views.answerBoardList),
8     url('answerBoard', views.answerBoard),
9 ]

```

6행을 추가하여 “answerBoardWritePro”주소를 요청하면 “views.py”에 “answerBoardWritePro”메서드를 실행한다.

answerBoarder/views.py

```

1 ... 중략
2 import os
3 from member.models import Member
4 from member.views import pwEncrypt
5 from django.core.files.storage import FileSystemStorage
6 from django.db.models import Max
7
8 import uuid
9 @csrf_exempt
10 def answerBoardWritePro(request):
11     member = Member.objects.get(USER_ID=request.session['member']['USER_ID'])
12     boarderNum = AnswerBoarder.objects.aggregate(BOARD_NUM=Max('BOARD_NUM'))
13     if boarderNum['BOARD_NUM'] is None:
14         boarderNum['BOARD_NUM'] = 1
15     else :
16         boarderNum['BOARD_NUM'] += 1
17     print(boarderNum['BOARD_NUM'])
18     form = BoardWriteForm(request.POST, request.FILES)
19     if request.method == 'POST':
20         if form.is_valid():
21             answerBoarder = AnswerBoarder()
22             fs = FileSystemStorage(location='..//static/media/uploads/')
23             myfiles = request.FILES.getlist('BOARD_ORIGINAL_FILENAME')
24             fileNames = ""
25             fileSize = ""
26             originalName=""
27             for myfile in myfiles:
28                 file = fs.save(myfile.name, myfile)
29                 fileSize += str(os.path.getsize("..//static/media/uploads/" + file)) + " "
30                 extension = file.rsplit('.', 1)[1]
31                 fileName = uuid.uuid4().__str__().replace('-', ' ')
32                 os.rename('..//static/media/uploads/' + file , '..//static/media/uploads/' +
33                 fileName + "." + extension)
34                 originalName += file + " "

```

```

34         fileNames += fileName + "." + extension+ ""
35
36     answerBoarder.BOARD_NUM = boarderNum['BOARD_NUM']
37     answerBoarder.BOARD_ORIGINAL_FILENAME = originalName
38     answerBoarder.BOARD_STORE_FILENAME = fileNames
39     answerBoarder.BOARD_FILE_SIZE = fileSize
40     answerBoarder.BOARD_NAME = request.POST['BOARD_NAME']
41     answerBoarder.BOARD_PASS = pwEncrypt(request.POST['BOARD_PASS'])
42     answerBoarder.BOARD_SUBJECT = request.POST['BOARD SUBJECT']
43     answerBoarder.BOARD_CONTENT = request.POST['BOARD_CONTENT']
44     answerBoarder.BOARD_RE_REF = boarderNum['BOARD_NUM']
45     answerBoarder.USER_ID = member # 외래키로 지정된 경우 객체를 할당해야한다.
46     answerBoarder.save()
47     return HttpResponseRedirect("answerBoardList")
48 else :
49     return render(request, 'answerBoarder/Board_Write.html',{'f':form,'error':'비밀번호
50 가 8자 이상이어야 합니다.'})
51 else:
52     return render(request, 'answerBoarder/Board_Write.html',{'f':form,'error':'저장되지 않
53 았습니다.'})

```

11행은 session이 USER_ID인 Member데이터를 가져온다.

12행은 데이블에서 Max값을 가져온다. boarderNum에 'BOARD_NUM'컬럼이 딕셔너리로 저장된다.

13행은 "boarderNum"에 있는 "BOARD_NUM"의 값이 null인지 물어본다.

14행은 12행이 null이면 "boarderNum"에 있는 "BOARD_NUM"의 값으로 1을 저장

16행은 "BOARD_NUM"의 값이 있는 경우 1을 증가 시킨다.

18행은 "forms.py"에서 "Board_Write.html"에 출력할 내용을 가져온다.

21행은 자료실 데이터를 저장하기 위한 AnswerBoarder객체를 만든다.

22행은 "../static/media/uploads/"디렉터리에 파일을 저장하기 위해 FileSystemStorage객체를 생성한다.

23행은 "answerBoardList.html"페이지로부터 넘어온 파일 데이터들을 "myfiles"변수에 저장한다.

27행은 "myfiles"에 있는 파일을 하나씩 "myfile"변수에 저장한다.

28행은 "myfile"변수에 있는 파일을 media/uploads/"디렉터리에 저장하고 file변수에 파일이름을 저장한다.

29행은 구분자 "\\"를 이용하여 파일들의 크기를 저장한다.

33행은 "file"에 저장된 파일명을 "fileNames" 변수에 구분자 "\\"를 이용하여 저장한다.

36행은 "boarderNum"에 있는 "BOARD_NUM"의 값을 저장한다.

37행부터 44행까지는 "answerBoardList.html"페이지로부터 넘어온 데이터를 각각 저장한다.

44행은 "BOARD_RE_REF"값을 "BOARD_NUM"와 같은 값을 저장하므로 부모글임을 알 수 있게 한다.

45행은 "USER_ID"는 외래키로 지정되어 있으므로 객체를 할당해주어야 한다. 8행에서 만든 member객체를 대입한다.

46행에서 디비에 저장한다.

- “answerBoardList”주소인 경우 자료실 리스트가 출력되게 한다.

answerBoarder/views.py

```

1 ... 중략
2 from django.core.paginator import Paginator
3 import math
4 def answerBoardList(request):
5     answerBoarder = AnswerBoarder.objects.all().order_by('-BOARD_RE_REF','BOARD_RE_SEQ')
6     page = request.GET.get('page',1)
7     limit = 10
8     page_range = 10
9     paginator = Paginator(answerBoarder, limit)
10    contacts = paginator.get_page(page)
11
12    current_block = math.ceil(int(page)/page_range)
13    start_block = (current_block-1) * page_range
14    end_block = start_block + page_range
15    p_range = paginator.page_range[start_block:end_block]
16
17    context = {'contacts': contacts,'p_range' : p_range,}
18    return render(request, 'answerBoarder/Board_List.html',context)

```

5행에서 자료실 테이블에 있는 데이터를 모두 가져온다.

6행은 현재 페이지에 대한 값이 없는 경우 현 페이지를 1로 지정한다. (page는 1)

7행은 “Board_List.html”에 출력될 리스트의 수를 지정한다. (한페이지에 출력될 개수는 10개)

8행은 페이지된 번호가 몇 개 출력할 것인지 설정한다.(페이지 번호가 10개씩 출력)

9행은 “answerBoarder”에 있는 데이터 중 7행에 설정된 갯수 만큼씩 페이지팅한다.

10행은 9행에서 페이지팅된 데이터들 중 해당 페이지 번호에 해당되는 데이터만 가져온다.

13행은 페이지 된 번호에서 현 페이지에 보여질 첫 번째 페이지 번호를 가져온다.

14행은 페이지 된 번호에서 현 페이지에 보여질 마지막 페이지 번호를 가져온다.

15행은 현 페이지에 보여질 첫 번째 페이지부터 현 페이지에 보여질 마지막 페이지의 번호까지 백터에 저장한다.

17행은 한 페이지에 보여질 데이터인 “contacts”와 페이지 번호를 가진 백터인 “p_range”를 딕셔너리로 저장

18행은 “Board_List.html”로 “context” 딕셔너리 전송

answerBoarder/answerBoardList.html

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>

```

```

8 <table>
9 <tr><th>번호</th><th>제목</th><th>이름</th><th>등록일</th></tr>
10 {% for answerBoarder in contacts %}
11 <tr>
12     <th>
13         <a href="AnswerBoarderDetail/{{answerBoarder.BOARD_NUM}}">
14             {{answerBoarder.BOARD_NUM}}</a></th>
15     <th>{{answerBoarder.BOARD SUBJECT}}</th>
16     <th>{{answerBoarder.BOARD_NAME}}</th>
17     <th>{{answerBoarder.BOARD_DATE | date:"Y-m-d"}}</th>
18 </tr>
19 {% endfor %}
20 <tr alight="center" height=20 >
21     <td colspan = 3>
22             {% if not contacts.has_previous  %}
23                 [이전]&nbsp;
24                 {% else  %}
25                     [ <a href="answerBoardList?page={{ contacts.previous_page_number }}">이전</a> ]&nbsp;
26                     {% endif %}
27
28             {% for i in p_range %}
29                 [ <a href="answerBoardList?page={{i}}>{{ i }}</a> ]
30             {% endfor %}
31
32             {% if not contacts.has_next %}>
33                 [이후]&nbsp;
34                 {% else %}>
35                     [ <a href="answerBoardList?page={{contacts.next_page_number }}">이
36                     후</a> ]&nbsp;
37                     {% endif %}
38     </td>
39 <tr>
40 </table>
41 <a HREF="answerBoard"> 글쓰기 </a>
42 </body>
43 </html>

```

10행부터 18행은 딕셔너리 contacts에 있는 값을 반복하여 출력한다.

11행에서는 상세 페이지로 가기 위한 주소 "AnswerBoarderDetail"로 가기 위해 링크를 걸었다.

answerBoarder/urls.py

```
1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'answerBoarder'
5 urlpatterns = [
6     url('answerBoardWritePro', views.answerBoardWritePro),
7     url('answerBoardList', views.answerBoardList),
8     url('answerBoard', views.answerBoard),
9     url('AnswerBoarderDetail/(?P<BOARD_NUM>\w+)/$', views.AnswerBoarderDetail),
10 ]
```

9행은 “answerBoardList.html”에서 링크된 번호를 클릭한 경우 ← → ⌂ localhost:8000/board/answerBoardList?page=1
“AnswerBoarderDetail”주소로 이동한다.

^ : ^ 문자 뒤에 나열된 문자열로 시작
[0-9] : 0부터 9까지 범위에 속하는 문자
+ : 앞에 지정한 문자열 패턴이 한 번 이상 반복
() : 패턴 부분을 묶어냄(grouping)
?P<pk> : 묶어낸 패턴 부분에 이름을 pk로 붙임.
\$: \$ 문자 앞에 나열된 문자열로 끝

번호	제목	이름	등록일
3	제목	2019-12-26	
2	제목	2019-12-26	
1	제목	2019-12-26	

[이전] [1] [이후]
[글쓰기](#)

answerBoarder/views.py

```
1 ... 중략
2 def AnswerBoarderDetail(request, BOARD_NUM):
3     answerBoarder = AnswerBoarder.objects.get(BOARD_NUM=BOARD_NUM);
4     answerBoarder.BOARD_READCOUNT += 1
5     answerBoarder.save()
6     fileNames = answerBoarder.BOARD_ORIGINAL_FILENAME.split("\\")
7     StorefileNames = answerBoarder.BOARD_STORE_FILENAME.split("\\")
8     file_results = []
9     for index, value in enumerate(fileNames):
10         result = {}
11         result['fileNames'] = value
12         result['StorefileNames'] = StorefileNames[index]
13         file_results.append(result)
14     context = {'answerBoarder': answerBoarder, 'file_results' : file_results,
15 'BOARD_READCOUNT': answerBoarder.BOARD_READCOUNT}
16
17     return render(request, 'answerBoarder/boarder_Detail.html', context)
```

2행에서 “BOARD_NUM”은 “urls.py”的 9행의 “BOARD_NUM”을 받아온다.

3행은 “BOARD_NUM”에 해당하는 데이터를 가져온다.

4행은 “BOARD_READCOUNT”的 1을 증가시킨다.

5행은 1 증가시킨 “BOARD_READCOUNT”를 저장한다.

6행은 “BOARD_ORIGINAL_FILENAME”에 있는 데이터를 “`”를 구분자로 split()을 하여 벡터로 저장한다..
 7행은 “BOARD_STORE_FILENAME”에 있는 데이터를 “`”를 구분자로 split()을 하여 벡터로 저장한다..
 8행은 “file_results”을 리스트를 선언한다.
 9행에서 6행의 벡터를 index, value를 각각 가져온다.
 10행은 “result” 딕셔너리를 선언한다.
 11행은 “result” 딕셔너리에 “fileNames” 딕셔너리를 만들어 9행의 “fileNames” 벡터에 있는 값을 저장한다.
 12행은 “result” 딕셔너리에 “StorefileNames” 딕셔너리를 만들어 7행에서 split한 벡터 데이터를 index로 가져온다.
 13행은 8행에 선언한 리스트에 “result” 딕셔너리를 저장한다.
 14행은 “context” 딕셔너리에 “answerBoarder”와 “file_results” 그리고 ‘BOARD_READCOUNT’를 저장한다.
 16행은 “context” 딕셔너리를 “boarder_Detail.html”에 전송한다.

answerBoarder/boarder_Detail.html	
1	<!DOCTYPE html>
2	{% load static %}
3	<html>
4	<head>
5	<meta charset="UTF-8">
6	<title>Insert title here</title>
7	<script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>
8	<script type="text/javascript">
9	\$function(){
10	\$("#modify").click(function(){
11	location.href="../../BoardModify?num={{answerBoarder.BOARD_NUM}}";
12);
13	});
14	</script>
15	</head>
16	<body>
17	번호 : {{ answerBoarder.BOARD_NUM }}
18	글쓴이 : {{ answerBoarder.BOARD_NAME }}
19	제목 : {{ answerBoarder.BOARD SUBJECT }}
20	내용 : {{ answerBoarder.BOARD_CONTENT }}
21	등록일 : {{answerBoarder.BOARD_DATE date:"Y-m-d"}}
22	방문자 수 : {{ BOARD_READCOUNT }}
23	등록된 파일 :
24	{% for file in file_results %}
25	
26	{{file.fileNames}}
27	{% endfor %}
	<input type="button" name="BoardReply" id = "BoardReply" value="답변">

```

28 <input type="button" name="modify" id = "modify" value="수정">
29 <input type="button" value="취소" onclick="javascript:history.back();">
30 </body>
31 </html>

```

10행에서 12행은 27행에서 클릭을 하면 “BoardModify”주소로 이동한다.

현재 주소가 “<http://localhost:8000/board/AnswerBoarderDetail/번호/>” 이므로 “[../../](#)”를 해주지 않으면 “번호”가 있는 자리가 “BoardModify”로 변경된다.

즉 주소는 “<http://localhost:8000/board/AnswerBoarderDetail/BoardModify/>”되므로 원하는 주소인 “<http://localhost:8000/board/BoardModify/>”를 하기 위해서는 “[../../](#)”을 해주어야 한다.

17행부터 20행은 views.py파일에서 넘어온 딕셔너리에 있는 데이터를 출력한다.

22행은 views.py파일에서 넘어온 딕셔너리에 있는 BOARD_READCOUNT 값을 출력한다.

24행부터 26행은 등록된 파일이 여러개인 경우 반복문을 사용하여 출력한다.

25행은 저장된 파일을 출력하기 위해 “MEDIA_URL”경로를 지정해 준다.

answerBoarder/urls.py

```

1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'answerBoarder'
5 urlpatterns = [
6     url('answerBoardWritePro', views.answerBoardWritePro),
7     url('answerBoardList', views.answerBoardList),
8     url('answerBoard', views.answerBoard),
9     url('AnswerBoarderDetail/(?P<BOARD_NUM>\w+)/$', views.AnswerBoarderDetail),
10    url('BoardModify', views.BoardModify),
11 ]

```

10행 추가하여 views.py파일에 있는 BoardModify()메소드를 실행한다.

answerBoarder/views.py

```

1 ... 중략
2 def BoardModify(request):
3     BOARD_NUM = request.GET['BOARD_NUM']
4     answerBoarder = AnswerBoarder.objects.get(BOARD_NUM=BOARD_NUM);
5     answerBoarder.BOARD_READCOUNT += 1
6     answerBoarder.save()
7     fileNames = answerBoarder.BOARD_ORIGINAL_FILENAME.split("\\")
8     StorefileNames = answerBoarder.BOARD_STORE_FILENAME.split("\\")
9     file_results = []
10    for index, value in enumerate(fileNames):
11        result = {}
12        result['fileNames'] = value
13        result['StorefileNames'] = StorefileNames[index]

```

```

14     file_results.append(result)
15     context = {'answerBoarder': answerBoarder, 'file_results' : file_results,
16     'BOARD_READCOUNT': answerBoarder.BOARD_READCOUNT}
17
18     return render(request, 'answerBoarder/boarder_modify.html', context)

```

answerBoarder/boarder_modify.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8">
5  <title>Insert title here</title>
6  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>
7      <script type="text/javascript">
8          function modifyboard(){
9              $("#num").val("1");
10             modifyform.submit();
11         }
12         function deleteboard(){
13             $("#num").val("2");
14             modifyform.submit();
15         }
16     </script>
17 </head>
18 <body>
19 <form action="BoardModifyAction" method="post" name="modifyform">% csrf_token %
20 <input type="hidden" name="BOARD_NUM" value="{{answerBoarder.BOARD_NUM}}>
21 <input type="hidden" name="num" id ="num">
22 <table cellpadding="0" cellspacing="0">
23     <tr align="center" valign="middle">
24         <td colspan="5">MVC 게시판</td>
25     </tr>
26     <tr>
27         <td height="16" style="font-family:돋음; font-size:12">
28             <div align="center">글쓴이</div>
29         </td>
30         <td>
31             <input name="BOARD_NAME" size="50" maxlength="100"
32                 value="{{ answerBoarder.BOARD_NAME }}>
33         </td>

```

```
34    </tr>
35
36    <tr>
37        <td height="16" style="font-family:돋음; font-size:12">
38            <div align="center">제 목</div>
39        </td>
40        <td>
41            <input name="BOARD SUBJECT" size="50" maxlength="100"
42                value="{{ answerBoarder.BOARD SUBJECT }}>
43        </td>
44
45    <tr>
46        <td style="font-family:돋음; font-size:12">
47            <div align="center">내 용</div>
48        </td>
49        <td>
50            <textarea      name="BOARD CONTENT"      cols="67"      rows="15">{{
51 answerBoarder.BOARD CONTENT }}</textarea>
52        </td>
53
54    <tr>
55        <td height="16" style="font-family:돋음; font-size:12">
56            <div align="center">비밀번호</div>
57        </td>
58        <td>
59            <input name="BOARD PASS" type="password">
60        </td>
61
62    <tr>
63        <td height="16" style="font-family:돋음; font-size:12">
64            <div align="center">등록일</div>
65        </td>
66        <td>
67            {{answerBoarder.BOARD DATE | date:"Y-m-d"}}
68        </td>
69
70    <tr>
71        <td style="font-family:돋음; font-size:12">
72            <div align="center">파일 첨부</div>
73        <td>
74            {% for file in file_results %}
```

```

74                                {{file.fileNames}}<br/>
75        {% endfor %}
76        </td>
77    </tr>
78    <tr bgcolor="cccccc">
79        <td colspan="2" style="height:1px;">
80            </td>
81    </tr>
82    <tr><td colspan="2">&ampnbsp</td></tr>
83
84    <tr align="center" valign="middle">
85        <td colspan="5">
86            <font size=2>
87                <a href="javascript:modifyboard()">[수정]</a>&ampnbsp&ampnbsp
88                <a href="javascript:history.go(-1)">[뒤로]</a>&ampnbsp&ampnbsp
89                <a href="javascript:deleteboard()">[삭제]</a>&ampnbsp&ampnbsp
90            </font>
91        </td>
92    </tr>
93 </table>
94 </form>
95 </body>
96 </html>

```

answerBoarder/urls.py

```

1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'answerBoarder'
5 urlpatterns = [
6     url('BoardModifyAction', views.BoardModifyAction),
7     url('answerBoardWritePro', views.answerBoardWritePro),
8     url('answerBoardList', views.answerBoardList),
9     url('answerBoard', views.answerBoard),
10    url('AnswerBoarderDetail/(?P<BOARD_NUM>\w+)/$', views.AnswerBoarderDetail),
11    url('BoardModify', views.BoardModify),
12 ]

```

6행을 추가하여 views.py에 있는 BoardModifyAction()메소드를 실행시킨다.

answerBoarder/views.py

```

1 ... 중략

```

```

2 def BoardModifyAction(request):
3     if request.method == 'POST':
4         BOARD_NUM = request.POST['BOARD_NUM']
5         print("BOARD_NUM : " + request.POST['BOARD_NUM'])
6         answerBoarder = AnswerBoarder.objects.get(BOARD_NUM=BOARD_NUM);
7         print(pwEncrypt(request.POST['BOARD_PASS']))
8         print(answerBoarder.BOARD_PASS)
9         if pwEncrypt(request.POST['BOARD_PASS']) == answerBoarder.BOARD_PASS:
10             print(request.POST['num'])
11             if request.POST['num'] == '2':
12                 print(request.POST['num'])
13                 answerBoarder.delete()
14             elif request.POST['num'] == '1':
15                 answerBoarder.BOARD_NAME = request.POST['BOARD_NAME']
16                 answerBoarder.BOARD SUBJECT = request.POST['BOARD SUBJECT']
17                 answerBoarder.BOARD_CONTENT = request.POST['BOARD_CONTENT']
18                 answerBoarder.save()
19             return HttpResponseRedirect("answerBoardList")
20
21     return render(request, 'answerBoarder/boarder_modify.html', context)

```

13행에선 데이터와 파일을 삭제하기 위해서 models.py에 delete()를 재정의하였다.

-- 답변하기

answerBoarder/boarder_Detail.html	
1	<!DOCTYPE html>
2	{% load static %}
3	<html>
4	<head>
5	<meta charset="UTF-8">
6	<title>Insert title here</title>
7	<script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>
8	<script type="text/javascript">
9	\$function(){
10	\$("#modify").click(function(){
11	location.href=".../BoardModify?num={{answerBoarder.BOARD_NUM}}";
12	});
13	\$("#BoardReply").click(function(){
14	location.href=".../BoardReply?BOARD_NUM={{answerBoarder.BOARD_NUM}}";
15	});
16	});
17	</script>

```

18 ... 중략
19 <input type="button" name="BoardReply" id = "BoardReply" value="답변">
20 <input type="button" name="modify" id = "modify" value="수정">
21 <input type="button" value="취소" onclick="javascript:history.back();">
22 </body>
23 </html>

```

answerBoarder/urls.py

```

1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'answerBoarder'
5 urlpatterns = [
6     url('BoardModifyAction', views.BoardModifyAction),
7     url('answerBoardWritePro', views.answerBoardWritePro),
8     url('answerBoardList', views.answerBoardList),
9     url('answerBoard', views.answerBoard),
10    url('AnswerBoarderDetail/(?P<BOARD_NUM>\w+)/$', views.AnswerBoarderDetail),
11    url('BoardModify', views.BoardModify),
12    url('BoardReply', views.BoardReply),
13 ]

```

13행 추가

answerBoarder/views.py

```

1 ... 중략
2 def BoardReply(request):
3     BOARD_NUM = request.GET['BOARD_NUM']
4     answerBoarder = AnswerBoarder.objects.get(BOARD_NUM=BOARD_NUM)
5     context = {'board': answerBoarder}
6     return render(request, 'answerBoarder/board_reply.html', context)

```

answerBoarder/board_reply.html

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6     <script language="javascript">
7         function replyboard(){
8             boardform.submit();

```

```
9      }
10     </script>
11 </head>
12 <body>
13 <form action="BoardReplyAction" method="post" name="boardform">% csrf_token %
14 <input type="hidden" name="BOARD_NUM" value="{{board.BOARD_NUM }}">
15 <input type="hidden" name="BOARD_RE_REF" value="{{board.BOARD_RE_REF }}">
16 <input type="hidden" name="BOARD_RE_LEV" value="{{board.BOARD_RE_LEV }}">
17 <input type="hidden" name="BOARD_RE_SEQ" value="{{board.BOARD_RE_SEQ }}">
18
19 <table cellpadding="0" cellspacing="0">
20   <tr align="center" valign="middle">
21     <td colspan="5">MVC 게시판</td>
22   </tr>
23   <tr>
24     <td style="font-family:돋음; font-size:12" height="16">
25       <div align="center">글쓰이</div>
26     </td>
27     <td>
28       <input name="BOARD_NAME" type="text"/>
29     </td>
30   </tr>
31   <tr>
32     <td style="font-family:돋음; font-size:12" height="16">
33       <div align="center">제 목</div>
34     </td>
35     <td>
36       <input name="BOARD SUBJECT" type="text" size="50"
37           maxlength="100" value="Re:{{board.BOARD SUBJECT}} "/>
38     </td>
39   </tr>
40   <tr>
41     <td style="font-family:돋음; font-size:12">
42       <div align="center">내 용</div>
43     </td>
44     <td>
45       <textarea name="BOARD_CONTENT" cols="67" rows="15"></textarea>
46     </td>
47   </tr>
48   <tr>
49     <td style="font-family:돋음; font-size:12">
```

```

50          <div align="center">비밀번호</div>
51      </td>
52      <td>
53          <input name="BOARD_PASS" type="password">
54      </td>
55  </tr>
56
57  <tr bgcolor="cccccc">
58      <td colspan="2" style="height:1px;">
59      </td>
60  </tr>
61  <tr><td colspan="2">&nbsp;</td></tr>
62
63  <tr align="center" valign="middle">
64      <td colspan="5">
65          <a href="javascript:replyboard()">[등록]</a>&nbsp;&nbsp;
66          <a href="javascript:history.go(-1)">[뒤로]</a>
67      </td>
68  </tr>
69 </table>
70 </form>
71 </body>
72 </html>

```

answerBoarder/urls.py

```

1 from django.conf.urls import url
2 from . import views
3
4 app_name = 'answerBoarder'
5 urlpatterns = [
6     url('BoardReplyAction', views.BoardReplyAction),
7     url('BoardModifyAction', views.BoardModifyAction),
8     url('answerBoardWritePro', views.answerBoardWritePro),
9     url('answerBoardList', views.answerBoardList),
10    url('answerBoard', views.answerBoard),
11    url('AnswerBoarderDetail/(?P<BOARD_NUM>\w+)/$', views.AnswerBoarderDetail),
12    url('BoardModify', views.BoardModify),
13    url('BoardReply', views.BoardReply),
14 ]

```

6행 추가

answerBoarder/views.py

```
1 ... 중략
2 from django.db import connection
3 def BoardReplyAction(request):
4     if request.method == 'POST':
5         REF = request.POST['BOARD_RE_REF']
6         SEQ = request.POST['BOARD_RE_SEQ']
7         print(type(REF))
8         cursor = connection.cursor()
9         cursor.execute("update ANSWERBOARDER set BOARD_RE_SEQ = BOARD_RE_SEQ + 1
where BOARD_RE_REF = %s and BOARD_RE_SEQ > %s", (REF, SEQ))
10        member = Member.objects.get(USER_ID=request.session['member']['USER_ID'])
11        boarderNum = AnswerBoarder.objects.aggregate(BOARD_NUM=Max('BOARD_NUM'))
12        if boarderNum['BOARD_NUM'] is None:
13            boarderNum['BOARD_NUM'] = 1
14        else :
15            boarderNum['BOARD_NUM'] += 1
16        answerBoarder = AnswerBoarder()
17        answerBoarder.BOARD_NUM = boarderNum['BOARD_NUM']
18        answerBoarder.BOARD_NAME = request.POST['BOARD_NAME']
19        answerBoarder.BOARD_PASS = pwEncrypt(request.POST['BOARD_PASS'])
20        answerBoarder.BOARD SUBJECT = request.POST['BOARD SUBJECT']
21        answerBoarder.BOARD_CONTENT = request.POST['BOARD_CONTENT']
22        answerBoarder.BOARD RE REF = int(REF)
23        answerBoarder.BOARD RE LEV = int(request.POST['BOARD RE LEV']) + 1
24        answerBoarder.BOARD RE SEQ = int(SEQ) + 1
25        answerBoarder.USER_ID = member # 외래키로 지정된 경우 객체를 할당해야한다.
26        answerBoarder.save()
27    return HttpResponseRedirect("answerBoardList")
```

참고 사이트)

<https://cjh5414.github.io/django-file-upload/>

<https://wayhome25.github.io/django/2017/05/10/media-file/>

<https://velog.io/@ground4ekd/django-file-upload>

<https://blog.limhm.com/patentstart/2017/10/18/file-upload/>

<https://www.it-swarm.net/ko/django/%EC%B5%9C%EC%86%8C%ED%95%9C%EC%9D%98-%EC%9E%A5%>

EA%B3%A0-%ED%8C%8C%EC%9D%BC-%EC%97%85%EB%A1%9C%EB%93%9C-%EC%98%88%EC%A0%9C-%ED%95%84%EC%9A%94/973143290/