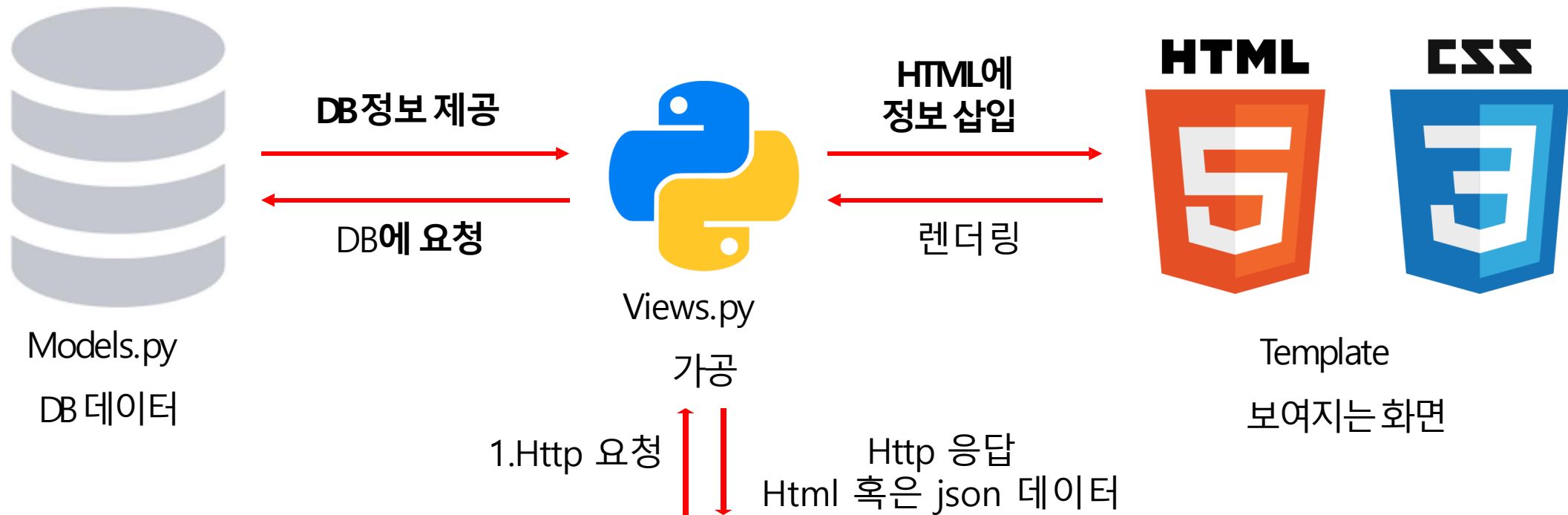


HICC 2학기 백엔드 세미나

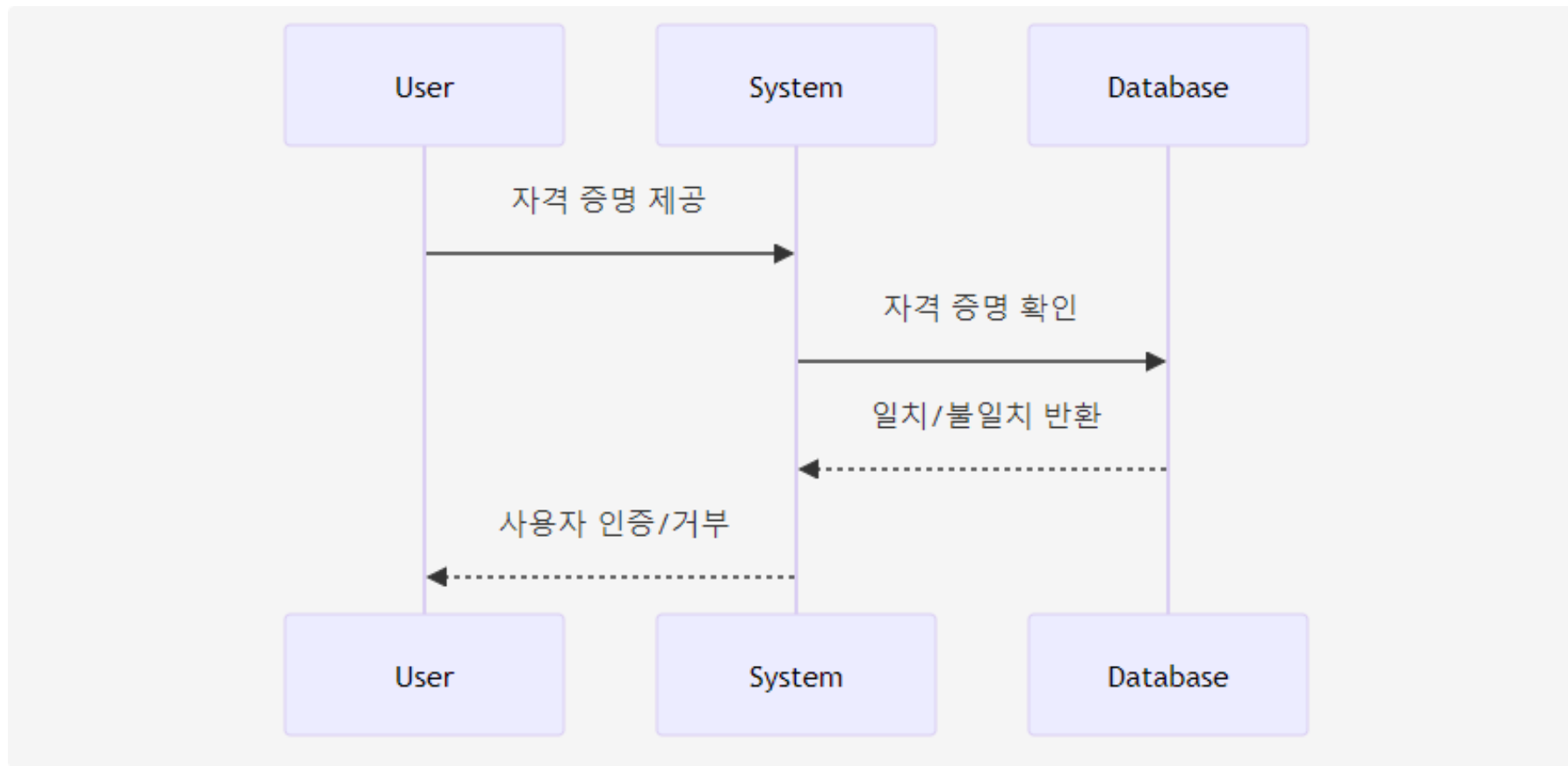
# 3강 회원가입, 로그인 기능 구현

# MTV 구조

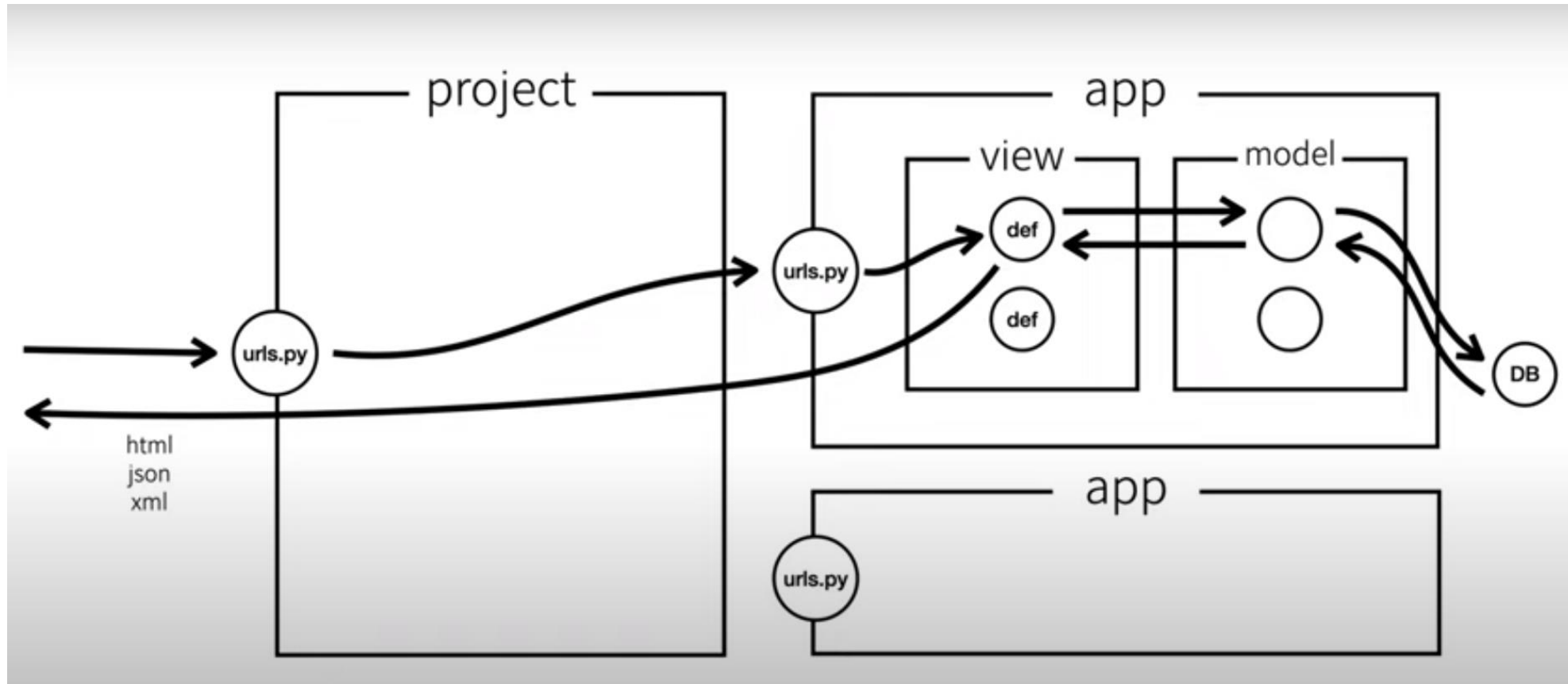
- Model Template View 구조



# Django 인증 구조

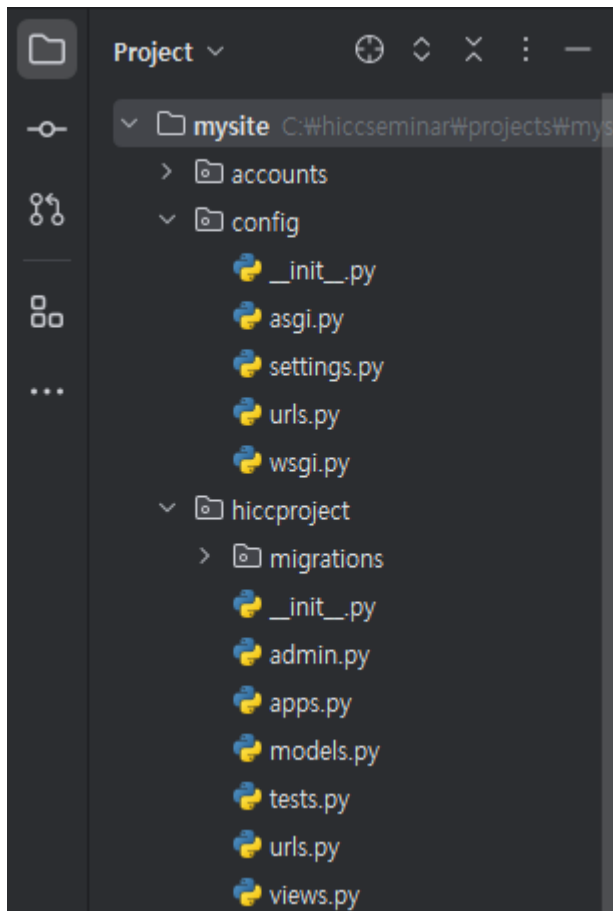


# Django 프로젝트 구조



# accounts 앱 생성하기

파이참 터미널에서 `python manage.py startapp accounts` 입력



이처럼 나오면 성공

계정과 관련된 기능은 **accounts** 앱에서 구현

# accounts 앱 생성하기



Django에서는 회원 정보를 저장하는 기능을 처음부터  
제공해줌

-> 따로 model을 만들지 않아도 됨

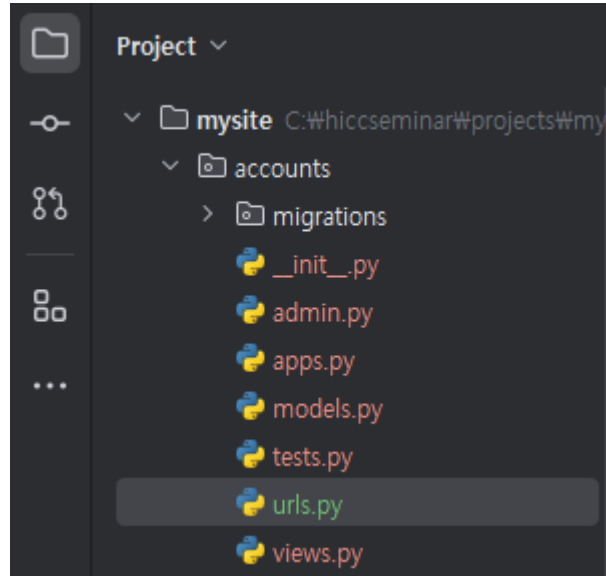
# accounts 경로 지정하기

config 폴더의 `urls.py`에 들어가서 다음과 같이 코드 수정

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('home/', include('hiccproject.urls')),
    path('accounts/', include('accounts.urls')),
]
```

# accounts 앱 urls.py 만들어주기



기존의 app 과 마찬가지로, accounts 앱에도 urls.py 파일을 생성

```
from django.urls import path
```

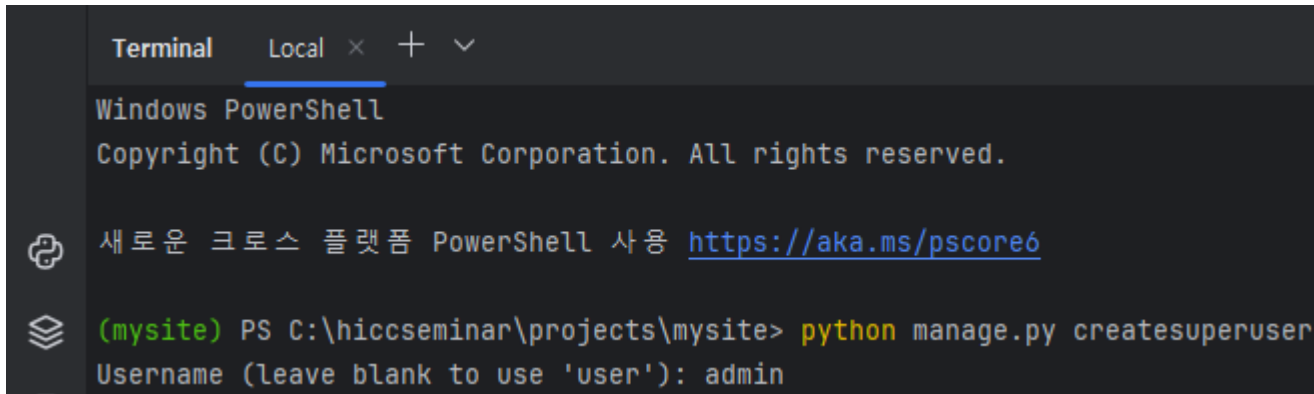
```
from . import views # .은 해당 폴더 (accounts) 의미
```

```
urlpatterns = [  
  
]
```

이와 같은 코드 작성



# Django 관리자 아이디 생성



```
Terminal Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

(mysite) PS C:\hiccseminar\projects\mysite> python manage.py createsuperuser
Username (leave blank to use 'user'): admin
```

로그인 기능 구현 전에, 먼저 관리자 아이디를 생성해주자.

터미널에서 `python manage.py createsuperuser`를 입력 후  
Username(admin 권장)과 email, password를 입력하면

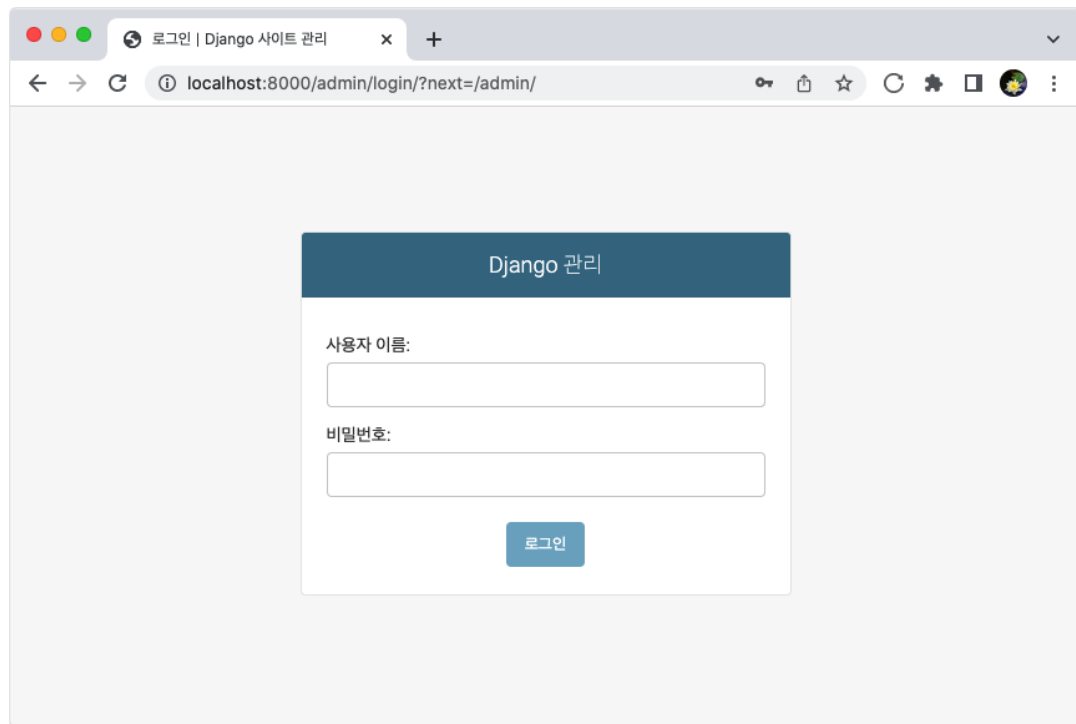
관리자 아이디가 생성된다.

# Django 관리자 아이디 생성

터미널에서 `python manage.py runserver` 입력 후

<http://127.0.0.1:8000/admin/>

해당 링크로 들어가면, 관리자 페이지가 뜨는데  
방금 입력한 username과 password를 입력 후 로그인 할 수 있다.



# 로그인 기능 구현

accounts 앱의 urls.py로 돌아가서 해당 코드를 한 줄 추가해주자

```
from django.urls import path
```

```
from . import views # .은 해당 폴더 (accounts) 의미
```

```
urlpatterns = [  
    path('login', views.login_view, name = "login"),  
]
```

그리고 이제부터 2개의 app을 사용하기 때문에, 경로 이름 구분을 위해서 각 urls.py 파일마다 app\_name을 설정해주자.

# 로그인 기능 구현

그리고 이제부터 2개의 app을 사용하기 때문에, 경로 이름 구분을 위해서 각 urls.py 파일마다 app\_name을 설정해주자.

```
from django.urls import path
```

```
from . import views # .은 해당 폴더 (accounts) 의미
```

```
app_name = 'accounts'
```

```
urlpatterns = [  
    path('login', views.login_view, name = "login"),  
]
```

```
from django.urls import path
```

```
from . import views # .은 해당 폴더 (hiccproject) 의미
```

```
app_name = 'hiccproject'
```

```
urlpatterns = [  
    path("", views.index, name = "home"),  
    ...  
]
```

# 로그인 기능 구현

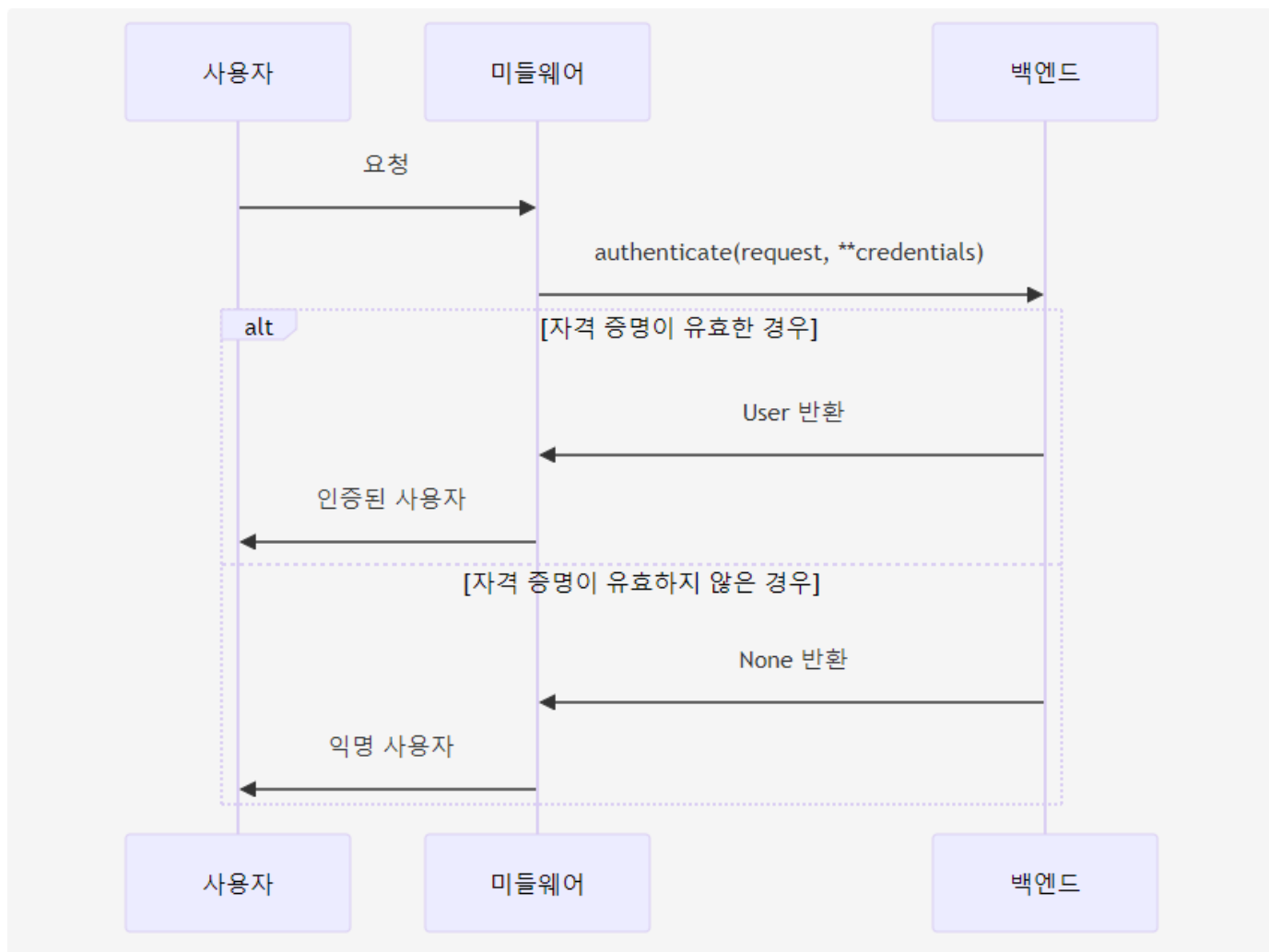
accounts 앱의 views.py로 가서 해당 코드를 추가해주자

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login

# Create your views here.

def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username,
password=password)
        if user is not None:
            login(request, user)
            return redirect('hiccproject:home')
        else:
            return render(request, 'accounts/login_view.html')
    return render(request, 'accounts/login_view.html')
```

# 로그인 기능 구현



# 로그인 기능 구현

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login
#django에서 여러가지 기능을 가져옴
```

```
# Create your views here.
```

```
def login_view(request): #login은 이미 Django 내에 구현된 함수 이름이라 사용하면 오류남
    if request.method == 'POST': #POST 요청일 때, 즉 로그인 버튼 누를 때 상황
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password) #django의 authenticate 기능을 사용해 로그인 인증 요청 후 user에 저장
        if user is not None:
            login(request, user)
            return redirect('hiccproject:home') #user가 None이 아닐 때, 즉 존재하는 아이디일 때, 로그인해주고, hiccproject 앱의 home으로 redirect 해줌
        else:
            return render(request, 'accounts/login_view.html') #user가 None일 때, 즉 로그인에 실패했을 때 로그인 페이지로 랜더링
    return render(request, 'accounts/login_view.html') #POST 요청이 아닐 때, 즉 GET 요청으로 로그인 페이지에 접근했을 때, 바로 로그인 페이지 랜더링
```

# 로그아웃 기능 구현

accounts 앱의 urls.py로 돌아가서 해당 코드를 한 줄 추가해주자

```
from django.urls import path
```

```
from . import views # .은 해당 폴더 (accounts) 의미
```

```
app_name = 'accounts'
```

```
urlpatterns = [  
    path('login', views.login_view, name = "login"),  
    path('logout', views.logout_view, name = "logout"),  
]
```



# 로그아웃 기능 구현

accounts 앱의 views.py로 가서 해당 코드를 추가해주자

```
from django.shortcuts import render, redirect  
from django.contrib.auth import authenticate, login, logout
```

```
def logout_view(request): #logout은 이미 있는 함수라서 다른 이름 사용  
    logout(request)  
    return redirect('hiccproject:home')
```

# 회원가입 기능 구현

accounts 앱의 urls.py로 가서 해당 코드를 추가해주자

```
from django.urls import path
```

```
from . import views # .은 해당 폴더 (accounts) 의미
```

```
app_name = 'accounts'
```

```
urlpatterns = [  
    path('login', views.login_view, name = "login"),  
    path('logout', views.logout_view, name = "logout"),  
    path('register', views.register, name = "register"),  
]
```

# 회원가입 기능 구현

accounts 앱의 views.py로 가서 해당 코드를 추가해주자

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.models import User
```

```
def register(request):
    if request.method == 'POST':
        if request.POST['password1'] == request.POST['password2']:
            User.objects.create_user(
                username=request.POST['username'],
                password=request.POST['password1'],
                email=request.POST['email'], )
            return redirect('accounts:login')
        return render(request, 'accounts/login_view.html')
    return render(request, 'accounts/login_view.html')
```

# 회원가입 기능 구현

accounts 앱의 views.py로 가서 해당 코드를 추가해주자

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.models import User #Django의 auth 기능에서 지원하고 있는 User 모델을 가져옴

def register(request):
    if request.method == 'POST': #login_view 함수와 동일, '회원가입 하기' 버튼을 누를 때 아래 부분이 실행됨
        if request.POST['password1'] == request.POST['password2']: #비밀번호와 비밀번호 확인 텍스트 상자가 일치할 때만
            User.objects.create_user(#User 테이블에 데이터를 생성하는 함수, user 아래 값들을 변수에 저장
                                    username=request.POST['username'],
                                    password=request.POST['password1'],
                                    email=request.POST['email'], )
            return redirect('accounts:login') #User 테이블에 데이터 저장되었으니, 로그인 하는 페이지로 리다이렉트
        return render(request, 'accounts/register.html') #만약 비밀번호와 비밀번호 확인 텍스트 상자가 불일치하면, 회원가입 페이지 다시 렌더링
    return render(request, 'accounts/register.html') #POST 요청이 아닌 GET 요청일 시, 회원가입 페이지 렌더링
```

# 회원가입 기능 구현

accounts 앱의 views.py로 가서 해당 코드를 추가해주자

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.models import User #Django의 auth 기능에서 지원하고 있는 User 모델을 가져옴

def register(request):
    if request.method == 'POST': #login_view 함수와 동일, '회원가입 하기' 버튼을 누를 때 아래 부분이 실행됨
        if request.POST['password1'] == request.POST['password2']: #비밀번호와 비밀번호 확인 텍스트 상자가 일치할 때만
            User.objects.create_user(#User 테이블에 데이터를 생성하는 함수, user 아래 값들을 변수에 저장
                                    username=request.POST['username'],
                                    password=request.POST['password1'],
                                    email=request.POST['email'], )
            return redirect('accounts:login') #User 테이블에 데이터 저장되었으니, 로그인 하는 페이지로 리다이렉트
        return render(request, 'accounts/register.html') #만약 비밀번호와 비밀번호 확인 텍스트 상자가 불일치하면, 회원가입 페이지 다시 렌더링
    return render(request, 'accounts/register.html') #POST 요청이 아닌 GET 요청일 시, 회원가입 페이지 렌더링
```

# 회원가입 기능 구현

db.sqlite3 파일을 열어서 확인해보면 실제로 회원가입 기능이 구현되었음을 확인할 수 있다.

DB Browser for SQLite - C:\hiccseminar\projects\mysite\db.sqlite3

파일(F) 편집(E) 보기(V) 도구(T) 도움말(H)

새 데이터베이스(N) 데이터베이스 열기(O) 변경사항 저장하기(W) 변경사항 취소하기(R) 프로젝트 열기(P) 프로젝트 저장하기(V) 데이터베이스 연결(A) 데이터베이스 닫기(C)

데이터베이스 구조 데이터 보기 Pragma 수정 SQL 실행

테이블(T): auth\_user

모든 열에서 필터링

	id	password	last_login	is_superuser	username	last_name	email	is_staff	is_active	date_joined	first_name
1	1	pbkdf2_sha256\$320000\$IPhkpTZWKqxXAYPX...	2024-10-09 17:31:25.452702	1	admin		gusgkr785@gmail.com	1	1	2024-09-16 06:34:36.469208	
2	2	pbkdf2_sha256\$320000\$IphthfNmV0UjyXvHzZ...	2024-10-09 17:05:06.912886	0	1234		1234	0	1	2024-09-16 06:41:56.242724	

# hiccproject 모델 수정

회원가입, 로그인 기능을 구현했으니 이에 맞게 기존의 Question, Answer 모델들을 수정해보자.

Question과 Answer 속성에 질문 및 답변의 작성자를 의미하는 author 속성을 추가해주면 될 것이다.

먼저 Question 모델부터 수정해주자. hiccproject 앱의 models.py에 들어가서 해당 코드를 추가해주자

```
from django.db import models
from django.contrib.auth.models import User #User 테이블을 가져옴

class Question(models.Model):
    subject = models.CharField(max_length=200)
    content = models.TextField()
    create_date = models.DateTimeField()
    author = models.ForeignKey(User, on_delete=models.CASCADE) #User 테이블을 참조하는 외래키 author 속성을 추가, CASCADE는 아이디
삭제시 질문도 날라간다는 의미
```

# hiccproject 모델 수정

```
(mysite) PS C:\hiccseminar\projects\mysite> python manage.py makemigrations
It is impossible to add a non-nullable field 'author' to answer without specifying a default. This is because the database needs something to populate existing rows.
Please select a fix:
  1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
  2) Quit and manually define a default value in models.py.
Select an option: 1
```

터미널에서 모델의 수정사항을 migrate 해주자.

python manage.py makemigrations 명령어를 입력하면 위와 같이 뜰건데,  
지금까지 작성했던 Question 데이터의 author 값은 어떻게 할것인가? 라고 물어보는 것이다.

1번은 지금까지 Question 데이터의 모든 author 값을 터미널에서 강제로 계정 정보를 추가하는 것이고,  
2번은 모든 author값을 null (값 없음)으로 설정하는 것이다.

터미널에서 1을 입력해주자.



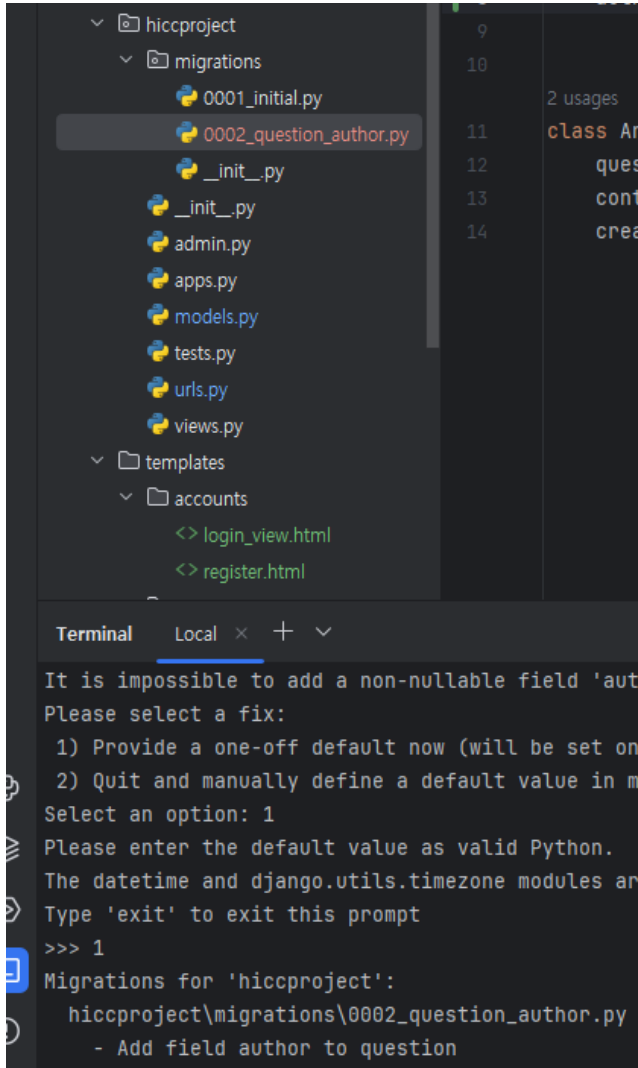
# hiccproject 모델 수정

```
Select an option: 1
Please enter the default value as valid Python.
The datetime and django.utils.timezone modules are available, so it is possible to provide e.g. timezone.now as a value.
Type 'exit' to exit this prompt
>>> █
```

그러면 위처럼 뜰건데, 여기에서 우리가 넣고 싶은 author 속성의 id 값을 입력하면 된다.  
(User 테이블의 id 속성과 username 속성은 다르다! 우리가 입력한 아이디는 username이고, User 테이블의 id 속성은 자동으로 생성된다! 주의!  
앞으로 id는 User 테이블에서 자동으로 생성되는 속성 id를 지칭하여 언급할 것임)

우리는 해당 터미널에서 admin의 id인 1을 입력하자.

# hiccproject 모델 수정



이처럼 뜨면서, 해당 파일이 생성되면 성공이다.

이어서 터미널에서 `python manage.py migrate`도 입력해주자.

# hiccproject 모델 수정

Question과 똑같이 Answer 모델도 수정해주자.

```
class Answer(models.Model):  
    question = models.ForeignKey(Question, on_delete=models.CASCADE)  
    content = models.TextField()  
    create_date = models.DateTimeField()  
    author = models.ForeignKey(User, on_delete=models.CASCADE)
```

이후 makemigrations 부터 migrate 까지 아까와 똑같이 진행해주자.

# hiccproject 모델 수정

db.sqlite3 파일을 열어서 확인해보면 실제로 author 속성이 추가되었음을 확인할 수 있다.

DB Browser for SQLite - C:\hiccseminar\projects\mysite\db.sqlite3

파일(F) 편집(E) 보기(V) 도구(T) 도움말(H)

새 데이터베이스(N) 데이터베이스 열기(O) 변경사항 저장하기(W) 변경사항 취소하기(R) 프로젝트 열기

데이터베이스 구조 데이터 보기 Pragma 수정 SQL 실행

테이블(T): hiccproject\_question

	id	subject	content	create_date	author_id
...	...	필터	필터	필터	필터
1	1	hicc동아리방은 몇호실인가요?	동아리방 위치를 알고 싶습니다.	2024-09-14 17:21:46.365164	1
2	2	백엔드 세미나는 언제 열리나요?	날짜와 장소를 알고 싶습니다.	2024-09-14 17:25:15.336585	1
3	3	학교 축제는 언제인가요?	축제 기간이 궁금합니다.	2024-09-15 10:51:48.347173	1
4	4	HICC는 무슨 뜻인가요?	Hong Ik Computer Club이라는 의미입니다.	2024-09-15 12:06:25.469825	1

# hiccproject views.py 수정

이제 달라진 model의 속성에 맞게 views.py 함수들도 수정해주자.

먼저 question과 answer의 create 함수들을 수정해주자.

```
def question_create(request):
    if request.method == 'POST':
        question = Question(subject = request.POST.get('subject'), content = request.POST.get('content'),
create_date=timezone.now(), author = request.user) # request.user는 현재 로그인된 유저의 id를 불러옴
        question.save()
        return redirect('hiccproject:question') # app_name으로 구별해봤으니 맞게 수정해줍시다
    else:
        return render(request, 'hiccproject/question_create.html')

def answer_create(request, question_id):
    question = get_object_or_404(Question, id = question_id)
    answer = Answer(question=question, content=request.POST.get('answer_content'), create_date=timezone.now(), author =
request.user)
    answer.save()
    return redirect('hiccproject:question_detail', question_id=question.id)
```

# hiccproject views.py 수정

관리자 계정이 아닌 회원가입을 통해 만든 계정 (자동 생성된 id가 2겠죠?) 으로 로그인하여 질문을 작성해보자.

DB Browser for SQLite - C:\hiccseminar\projects\mysite\db.sqlite3

파일(F) 편집(E) 보기(V) 도구(T) 도움말(H)

새 데이터베이스(N) 데이터베이스 열기(O) 변경사항 저장하기(W) 변경사항 취소하기(R) 프로젝트 열기(P)

데이터베이스 구조 데이터 보기 Pragma 수정 SQL 실행

테이블(T): hiccproject\_question

	id	subject	content	create_date	author_id
1	1	hicc동아리방은 몇호실인가요?	동아리방 위치를 알고 싶습니다.	2024-09-14 17:21:46.365164	1
2	2	백엔드 세미나는 언제 열리나요?	날짜와 장소를 알고 싶습니다.	2024-09-14 17:25:15.336585	1
3	3	학교 축제는 언제인가요?	축제 기간이 궁금합니다.	2024-09-15 10:51:48.347173	1
4	4	HICC는 무슨 뜻인가요?	Hong Ik Computer Club이라는 의미입니다.	2024-09-15 12:06:25.469825	1
5	14	ww	dd	2024-10-09 17:43:58.506301	2

sqlite를 통해서 잘 구현되었음을 확인할 수 있다.

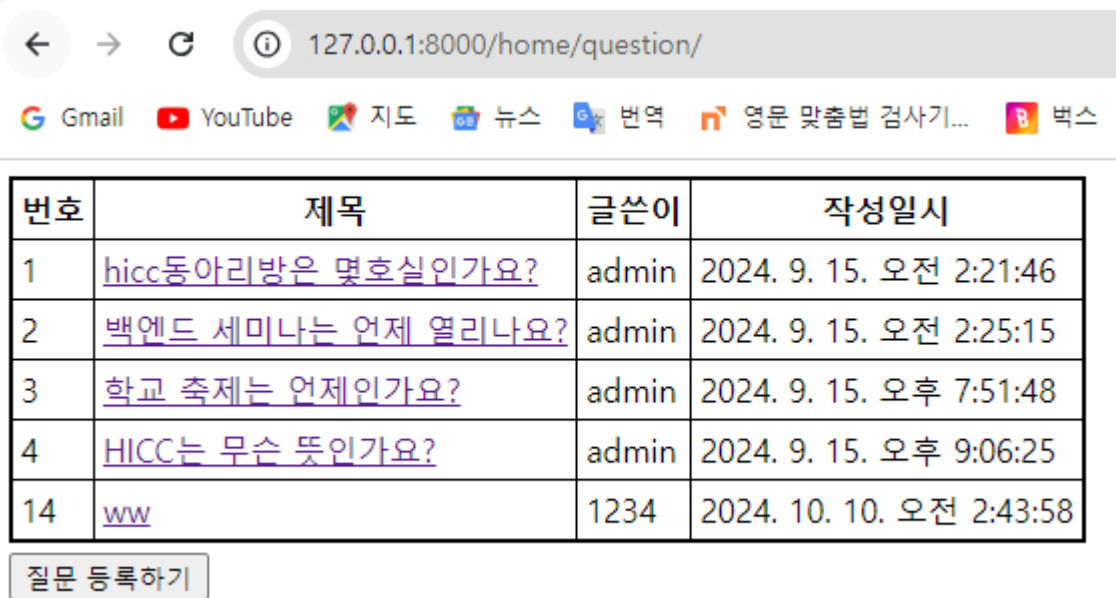
# hiccproject views.py 수정

추가로 같은 로직의 update 함수도 수정해주자.

```
def question_update(request, question_id):
    if request.method == 'POST':
        question = Question(id = question_id, subject = request.POST.get('subject'), content =
request.POST.get('content'), create_date=timezone.now(), author = request.user)
        question.save()
        return redirect('hiccproject:question')
    else:
        return render(request, 'hiccproject/question_update.html')
```

# hiccproject views.py 수정

아래 사진처럼 각 질문마다 그 질문을 작성한 계정의 username도 볼 수 있도록 구현하려고 한다.  
question\_read 함수를 살짝 수정해주자.



번호	제목	글쓴이	작성일시
1	<a href="#">hiccdongaribang은 몇호실인가요?</a>	admin	2024. 9. 15. 오전 2:21:46
2	<a href="#">백엔드 세미나는 언제 열리나요?</a>	admin	2024. 9. 15. 오전 2:25:15
3	<a href="#">학교 축제는 언제인가요?</a>	admin	2024. 9. 15. 오후 7:51:48
4	<a href="#">HICC는 무슨 뜻인가요?</a>	admin	2024. 9. 15. 오후 9:06:25
14	<a href="#">ww</a>	1234	2024. 10. 10. 오전 2:43:58

질문 등록하기

```
def question_read(request):
    questions = Question.objects.all().values('id', 'subject', 'content', 'author_username', 'create_date')

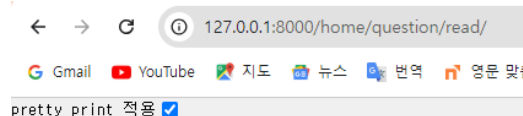
    return JsonResponse({'questions' : list(questions)}) # json 형식으로 설정 후 response
```



# hiccproject views.py 수정

지난 강의에서, Question.objects.all().values 함수를 통해 Question 데이터의 모든 값들을 속성 이름이 키인 딕셔너리로 변환한다고 설명했다.

그에 따르면 author라는 속성만 사용 가능할 것 같지만 \_ (언더바 두개)를 활용해 author 라는 외래키가 참조하고 있는 테이블의 속성 값을 가져와서 author\_\_username이라는 키의 값으로 변환하여 딕셔너리 형태로 나타내는 것도 가능하다.



```
{
  "questions": [
    {
      "id": 1,
      "subject": "hicc동마리방은 몇호실인가요? ",
      "content": "동마리방 위치를 알고 싶습니다.",
      "author__username": "admin",
      "create_date": "2024-09-14T17:21:46.365Z"
    },
    {
      "id": 2,
      "subject": "백엔드 세미나는 언제 열리나요? ",
      "content": "날짜와 장소를 알고 싶습니다.",
      "author__username": "admin",
      "create_date": "2024-09-14T17:25:15.336Z"
    },
    {
      "id": 3,
      "subject": "학교 축제는 언제인가요?",
      "content": "축제 기간이 궁금합니다.",
      "author__username": "admin",
      "create_date": "2024-09-15T10:51:48.347Z"
    },
    {
      "id": 4,
      "subject": "HICC는 무슨 뜻인가요?",
      "content": "Hong Ik Computer Club이라는 의미입니다.",
      "author__username": "admin",
      "create_date": "2024-09-15T12:06:25.469Z"
    },
    {
      "id": 14,
      "subject": "ww",
      "content": "dd",
      "author__username": "1234",
      "create_date": "2024-10-09T17:43:58.506Z"
    }
  ]
}
```

```
def question_read(request):
```

```
    questions = Question.objects.all().values('id', 'subject', 'content', 'author__username',
'create_date')
```

```
    return JsonResponse({'questions' : list(questions)}) # json 형식으로 설정 후 response
```

# 접근 제한 설정

로그인 기능을 구현하니, 이제는 로그인을 해야만 특정 기능을 수행할 수 있도록 프로그램에 제한을 걸고 싶다.

해당 조건에 맞도록 프로그램을 수정하려고 한다.

- 1, 비로그인 상태로 질문 등록은 못하도록 하려고 합니다.
- 2, 비로그인 상태로 답변 등록도 못하도록 하려고 합니다.
- 3, 비로그인 상태로 질문 수정은 못하도록 하려고 하고, 본인이 쓴 질문 아니면 수정을 못하게 하려고 합니다.
- 4, 비로그인 상태로 질문 삭제는 못하도록 하려고 하고, 본인이 쓴 질문 아니면 삭제를 못하게 하려고 합니다.

코드를 간단히 추가하여 구현해보자.

# 접근 제한 설정

로그인을 해야만 기능을 작동시킬 수 있도록 설정하는 것은 간단하다.

위에 언급한 4가지 함수(question\_create, answer\_create, question\_update, question\_delete) 에 코드를 살짝 추가해보자.

```
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
from django.contrib.auth.decorators import login_required
```

```
from django.http import JsonResponse
from hiccproject.models import *
```

```
@login_required(login_url='accounts:login')
def question_create(request):
    ...
```

```
@login_required(login_url='accounts:login')
def answer_create(request, question_id):
    ...
```

```
@login_required(login_url='accounts:login')
def question_update(request, question_id):
    ...
```

```
@login_required(login_url='accounts:login')
def question_delete(request, question_id):
    ...
```

# 접근 제한 설정

## @login\_required()

는 장고에서 지원하는 애너테이션 기능으로, 해당 코드가 붙으면 로그인을 해야만 실행 가능한 함수를 의미한다.

서버를 가동 후 home url에서 로그아웃을 하고 질문 작성, 수정, 삭제, 답변 작성 기능을 실행해보면 로그인 페이지로 이동되는 것을 확인할 수 있다.

## @login\_required(login\_url='accounts:login')

에서 login\_url = 'accounts:login'는 로그인 상태가 아닌 채로 해당 함수를 실행하면 리다이렉트되는 페이지를 설정한 것이다. login\_url 변수 안에 url을 넣어주면 된다.

# 접근 제한 설정

3, 비로그인 상태로 질문 수정은 못하도록 하려고 하고, **본인이 쓴 질문 아니면 수정을 못하게 하려고 합니다.**

4, 비로그인 상태로 질문 삭제는 못하도록 하려고 하고, **본인이 쓴 질문 아니면 삭제를 못하게 하려고 합니다.**

해당 기능도 구현하기 위해, 코드를 수정해보자.

```
from django.core.exceptions import PermissionDenied
@login_required(login_url='accounts:login')
def question_update(request, question_id):
    question = get_object_or_404(Question, id=question_id)

    if request.user != question.author:
        raise PermissionDenied # 작성자와 사용자가 다르면 권한 오류 403 발생

    if request.method == 'POST':
        question = Question(id = question_id, subject = request.POST.get('subject'), content = request.POST.get('content'), create_date=timezone.now(),
author = request.user)
        question.save()
        return redirect('hiccproject:question')
    else:
        return render(request, 'hiccproject/question_update.html')

@login_required(login_url='accounts:login')
def question_delete(request, question_id):
    question = get_object_or_404(Question, id=question_id)

    if request.user != question.author:
        raise PermissionDenied # 작성자와 사용자가 다르면 권한 오류 403 발생

    question.delete()
    return redirect('hiccproject:question')
```

# 정리

3주간의 세미나를 통해

- 1, Django의 구조를 배워 프로젝트와 앱을 생성하여 url 경로를 설정하는 방법
- 2, Django의 ORM 기능을 활용해 프로그램에 글을 읽고, 작성하고, 수정하고, 삭제하는 방법
- 3, Django의 auth 기능을 활용해 프로그램에 회원가입하고, 로그인하고, 권한에 맞게 기능을 제한하는 방법을 배웠습니다.

좀 더 심층적이고 실용적인 내용은 개인 프로젝트 혹은 팀 프로젝트를 통해 배워나갑시다

## 다들 수고하셨습니다!