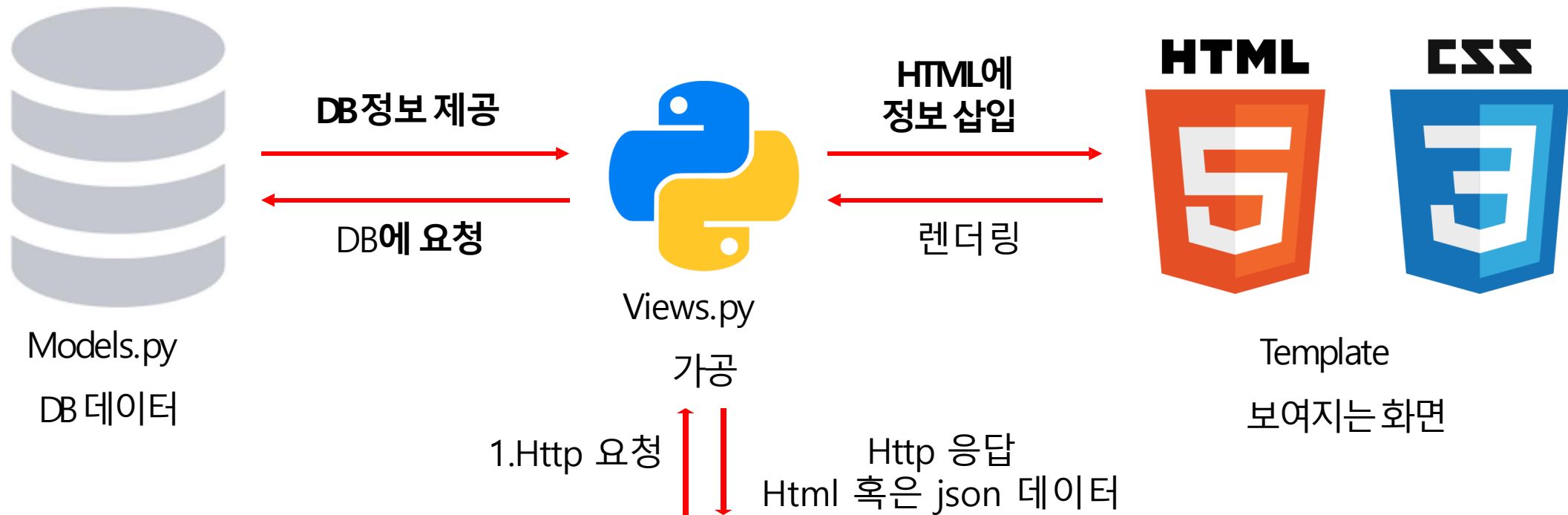


HICC 2학기 백엔드 세미나

2강 데이터베이스 ORM

MTV 구조

- Model Template View 구조



Django 앱 migrate

Django에는 미리 구현된 여러 기능들이 있습니다.
이를 서버에 바로 적용시키려면 migrate 명령어를 사용해야 합니다.
이번 세미나에선 Database 기능을 사용하기 위해 migrate 명령어를 사용하겠습니다.

```
(mysite) PS C:\hiccseminar\projects\mysite> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
```

터미널에 python manage.py migrate 명령어를 입력하고 위와 같이 뜨면 성공

Django 앱 migrate

```
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

여담으로 settings.py 파일의 DATABASES 부분을 보면, 어떤 DB 프로그램을 바탕으로 작동하는지 명시되어 있습니다.
Django에서는 매우 가벼운 DB 프로그램인 sqlite3를 사용하는 것이 기본 설정입니다.

Model 생성하기

지난 주에 생성한 질문 페이지, 질문 생성 페이지를 구현하기 위해 데이터들을 저장할 DB를 생성합니다.
hiccproject 폴더의 models.py 파일에 해당 코드를 입력합니다.

```
from Django.db import models
```

```
class Question(models.Model):
```

```
    subject = models.CharField(max_length=200)
```

```
    content = models.TextField()
```

```
    create_date = models.DateTimeField()
```

```
class Answer(models.Model):
```

```
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
```

```
    content = models.TextField()
```

```
    create_date = models.DateTimeField()
```

관계 데이터 모델

관계 데이터 모델의 기본 용어

- 릴레이션 – 하나의 개체에 관한 데이터를 2차원 테이블의 구조로 정의한 것, 간단하게 테이블이라고도 함
- 속성 – 릴레이션의 열
- 튜플 – 릴레이션의 행
- 도메인 – 하나의 속성이 가질 수 있는 모든 값의 집합, 쉽게 말해 **데이터 타입**

열(속성, 애트리뷰트)

고객아이디	고객이름	나이	등급	직업	적립금
CHAR(20)	CHAR(20)	INT	CHAR(10)	CHAR(10)	INT
apple	김현준	20	gold	학생	1000
banana	정소화	25	vip	간호사	2500
carrot	원유선	28	gold	교사	4500
orange	정지영	22	silver	학생	0

행(튜플)

도메인

관계 데이터 모델

릴레이션의 특성

- 튜플의 유일성 – 하나의 릴레이션에는 동일한 튜플이 존재할 수 없다.
- 튜플의 무순서 – 하나의 릴레이션에서 튜플 사이의 순서는 무의미하다.
- 속성의 무순서 – 하나의 릴레이션에서 속성 사이의 순서는 무의미하다.
- 속성의 원자성 – 속성 값으로 원자 값만 사용할 수 있다.
 - > 한 튜플에 다중 값(여러 개 나눌 수 있는 값), 복합 값(여러 개로 나눌 수 있는 값)은 존재하지 않음

관계 데이터 모델

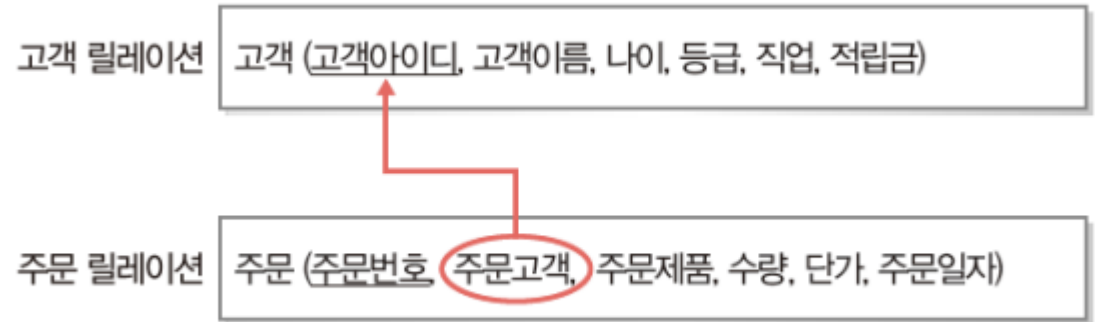
키의 종류(핵심만)

기본키(primary key)

- 기본적으로 사용하기 위한 키
- 유일성을 만족
- ex) 고객 릴레이션의 기본키 : 고객아이디

외래키(foreign key)

- 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합
 - 릴레이션들 간의 관계를 표현
- ex) 주문 릴레이션의 외래키 : 주문고객 (고객아이디를 참조하기 때문에 고객 릴레이션을 참조함)



기본키, 외래키 구분법

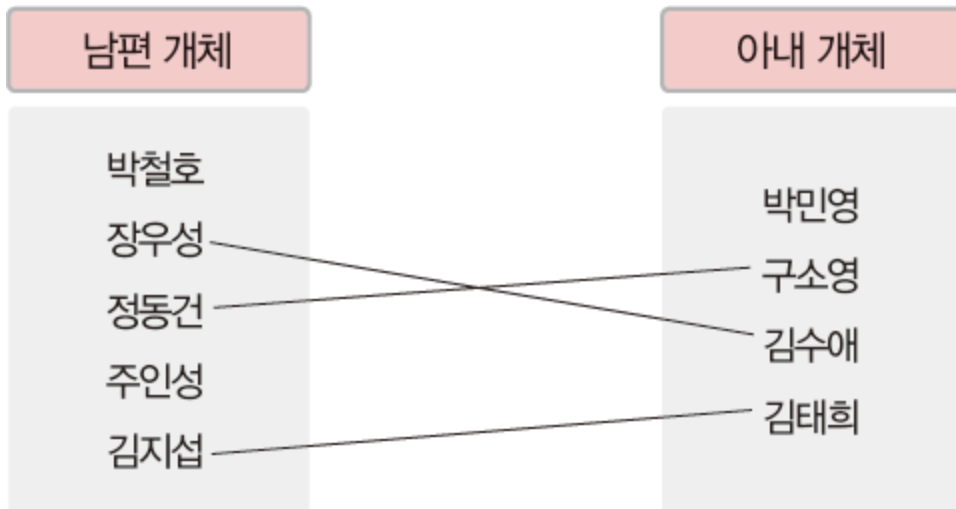
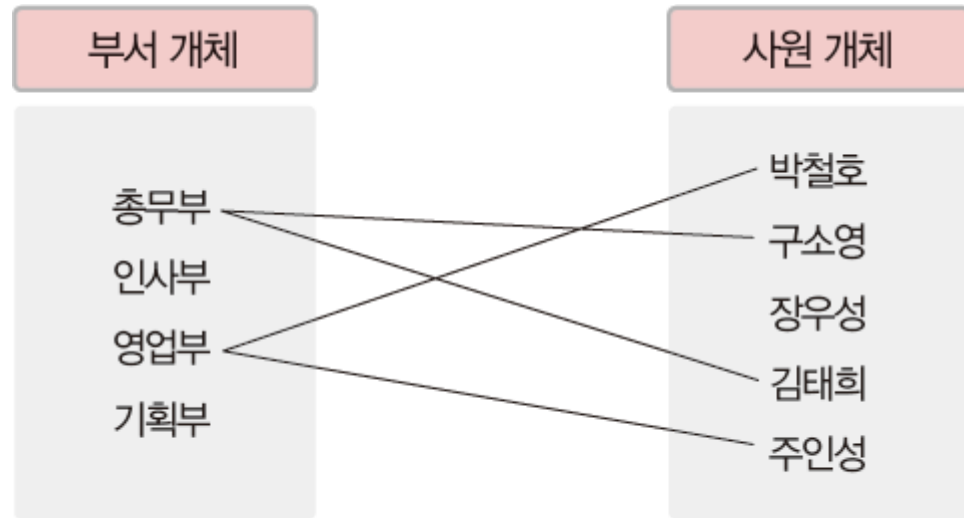


그림 4-14 일대일 관계의 예 : 남편과 아내 개체의 혼인 관계

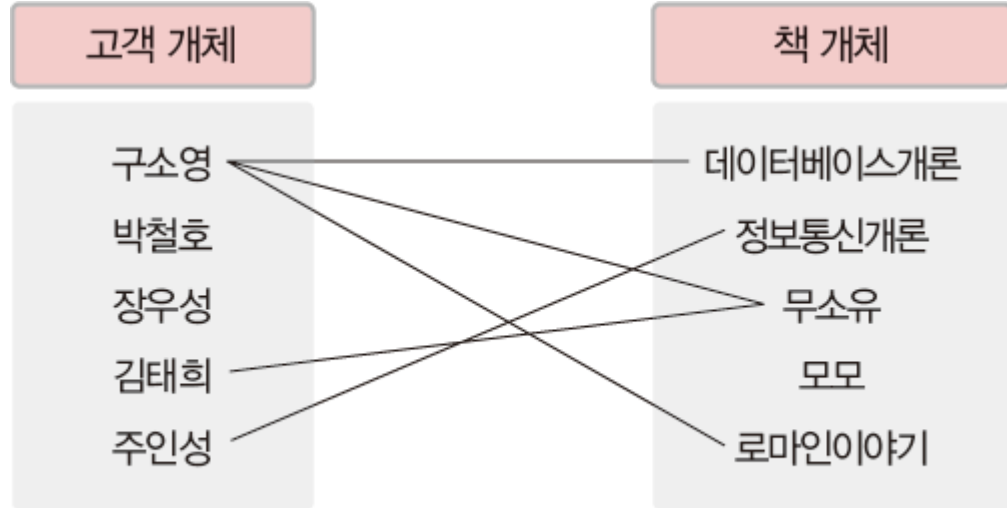
일대일 관계에서는 아내가 남편의 정보를 외래키로 가져도 되고, 남편이 아내의 정보를 외래키로 가져도 됨

기본키, 외래키 구분법



일대다 관계에서는 사원이 부서의 정보를 외래키로 가져야만 함

기본키, 외래키 구분법



다대다 관계에서는 '구매'라는 관계 릴레이션을 하나 생성하고, 그 릴레이션에서 고객과 책의 데이터를 외래키로 가져야만 함

Model 코드 다시보기

```
from Django.db import models
```

```
class Question(models.Model):
```

```
    subject = models.CharField(max_length=200) # CharField는 데이터 타입, 글자 타입을 의미하고 최대 길이는 200  
    content = models.TextField() # TextField는 데이터 타입, 글자 타입이지만 최대 길이가 정해져 있지 않으면 TextField  
    create_date = models.DateTimeField() # DateTimeField는 데이터 타입, 날짜 데이터를 저장하는 속성
```

```
class Answer(models.Model):
```

```
    question = models.ForeignKey(Question, on_delete=models.CASCADE)  
    #ForeignKey는 외래키를 지정하는 함수, Question 클래스를 참조함, on_delete=models.CASCADE는 Question 클래스의 데이터가  
    삭제됐을 때, Answer 데이터도 같이 삭제됨을 의미하는 옵션  
    content = models.TextField()  
    create_date = models.DateTimeField()
```

해당 코드와 같이 클래스에서 primary key를 따로 지정하지 않으면, **id**라는 **primary key** 속성이 자동으로 생성됨.

더 많은 필드와 옵션은

<https://github.com/dkyou7/TIL/blob/master/%ED%8C%8C%EC%9D%B4%EC%8D%AC/Django/5.%20%5BDjango%5D%20Model%20%E D%95%84%EB%93%9C%ED%83%80%EC%9E%85%20%EC%A0%95%EB%A6%AC.md> 참고

DB에 테이블 생성하기

models.py 파일을 작성했으니, 이제 sqlite에 테이블을 실제로 생성합니다
그러기 위해 settings.py 파일에서 INSTALLED_APPS에 해당 부분을 추가해줍니다
HiccpjectConfig 클래스는 apps.py 파일에 처음부터 구현되어 있습니다

```
# Application definition
```

```
INSTALLED_APPS = [
```

```
| 'hiccproject.apps.HiccpjectConfig',
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
```

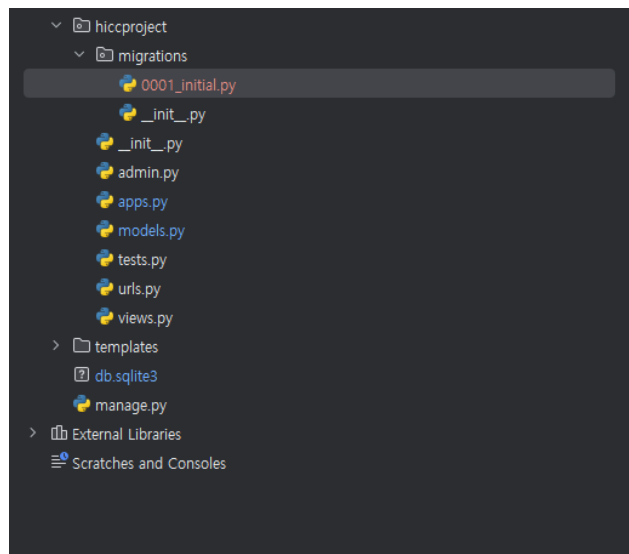
```
]
```

DB에 테이블 생성하기

실제로 DB에 테이블을 생성하기 위해 터미널에 `python manage.py makemigrations` 명령어를 입력합니다.

```
(mysite) PS C:\hiccseminar\projects\mysite> python manage.py makemigrations
Migrations for 'hiccproject':
  hiccproject\migrations\0001_initial.py
    - Create model Question
    - Create model Answer
```

위와 같은 상태가 나오고, 앱 폴더의 `migrations` 폴더에 아래와 같은 파일이 생성되면 성공이다.
우리는 DB에 실제로 `models.py` 파일의 테이블을 구현할 준비를 마쳤다.

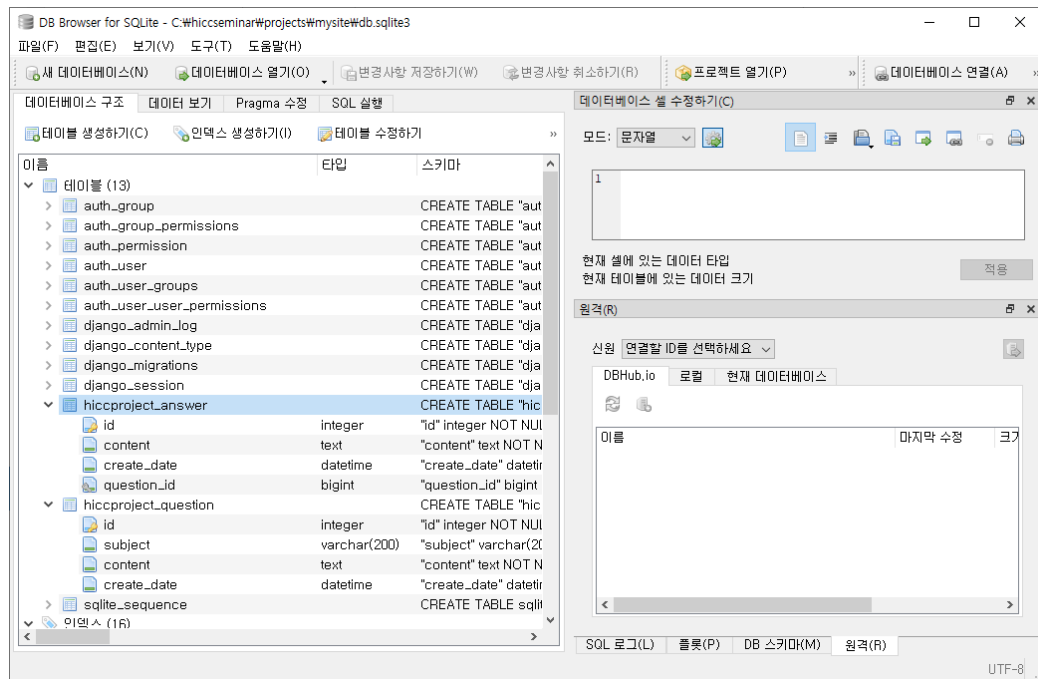


DB에 테이블 생성하기

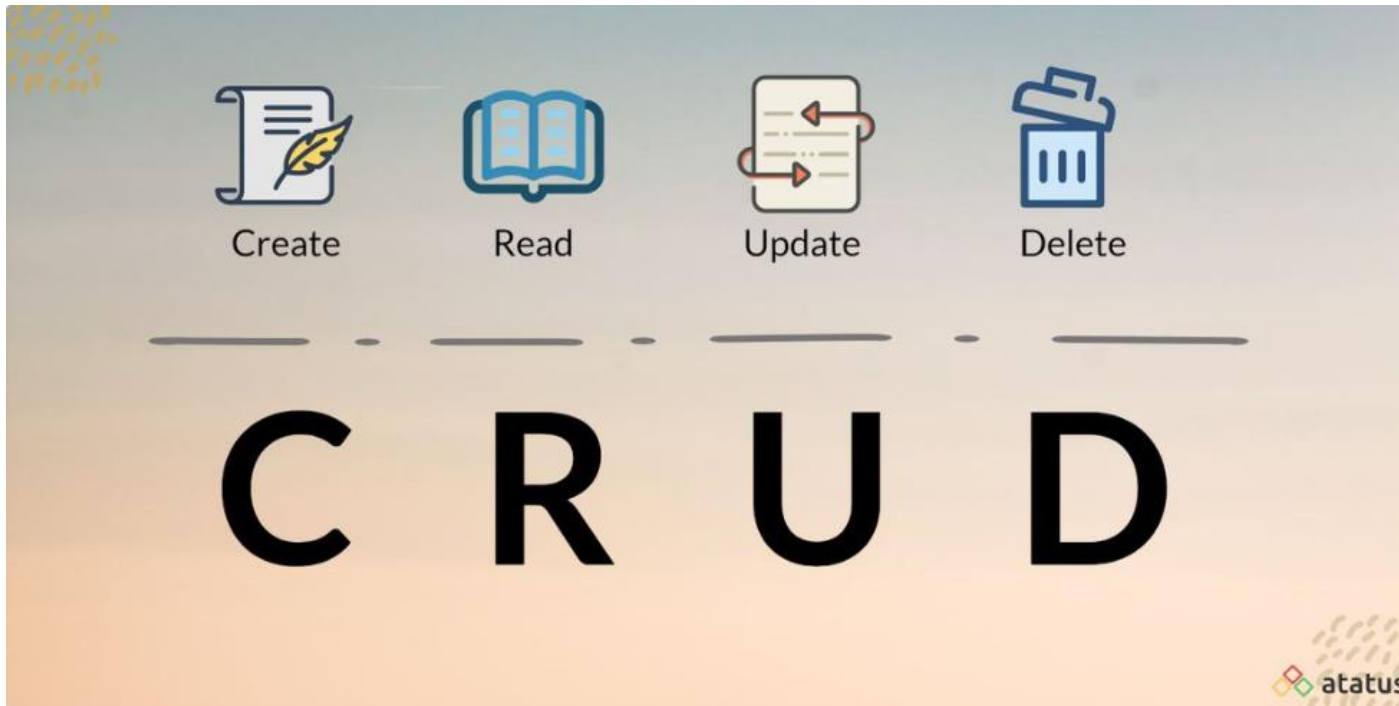
다시 터미널에서 `python manage.py migrate` 명령어를 실행시키면, 실제로 DB에 우리가 작성한 테이블이 적용된다.

```
(mysite) PS C:\hiccseminar\projects\mysite> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, hiccproject, sessions
Running migrations:
  Applying hiccproject.0001_initial... OK
```

다운 받아왔던 `sqlite`로 프로젝트 폴더의 `db.sqlite3` 파일을 열어보면 아래와 같이 실제 테이블들이 db에 구현되었음을 볼 수 있다.



기본적인 데이터 처리 기능



DB의 데이터를 만들고(CREATE), 읽고(READ), 변경하고(UPDATE), 삭제하고(DELETE)

Django에서는 아쉽게도 Update, Delete http method인 PATCH, DELETE는 지원하지 않는다. 해당 기능들은 대신 다른 방식으로 구현하게 되는데, 실습을 하면서 알아보자.

Read 기능 구현하기

먼저 비어있는 DB를 채워주기 위해 python shell을 통해서 임의의 데이터를 생성해주자

```
(mysite) PS C:\hiccseminar\projects\mysite> python manage.py shell
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> 
```

파이참의 터미널에서 python manage.py shell 을 입력하여 python shell로 접속
➔ 터미널에서 직접 데이터베이스에 접근할 수 있다

터미널에서

```
>>> from hiccproject.models import Question, Answer
```

```
>>> from django.utils import timezone
```

```
>>> q = Question(subject='hicc동아리방은 몇호실인가요?', content='동아리방 위치를 알고 싶습니다.',
create_date=timezone.now())
```

```
>>> q.save()
```

해당 코드를 한줄 씩 입력

Read 기능 구현하기

```
>>> from hiccproject.models import Question, Answer
```

hiccproject 폴더의 models 파일에 있는 Question, Answer 클래스를 사용하기 위해 import

```
>>> from Django.utils import timezone
```

현재 시간을 알려주는 timezone.now() 함수를 사용하기 위해 import 함

```
>>> q = Question(subject='hicc동아리방은 몇호실인가요?', content='동아리방 위치를 알고 싶습니다.',  
create_date=timezone.now())
```

Question은 models의 클래스, 해당 클래스를 통해 q라는 객체에 하나의 튜플을 대입, Question이라는 테이블의 속성 subject, content, create_date의 값도 넣을 수 있음.

```
>>> q.save()
```

q.save()를 선언해야 방금 선언한 튜플들을 DB에 저장할 수 있음.

Read 기능 구현하기

```
>>> q.id  
1
```

터미널에 q.id 입력하면 현재 한 개의 튜플이 생성되었음을 확인 가능

추가로 해당 코드를 입력하여 DB에 튜플 하나 더 생성

```
>>> q = Question(subject='백엔드 세미나는 언제 열리나요?', content='날짜와 장소를 알고 싶습니다.',  
create_date=timezone.now())  
>>> q.save()
```

q.id를 입력하면 2가 출력됨

```
>>> Question.objects.all()
```

를 입력해서 현재 모든 Question 데이터를 조회할 수 있음.

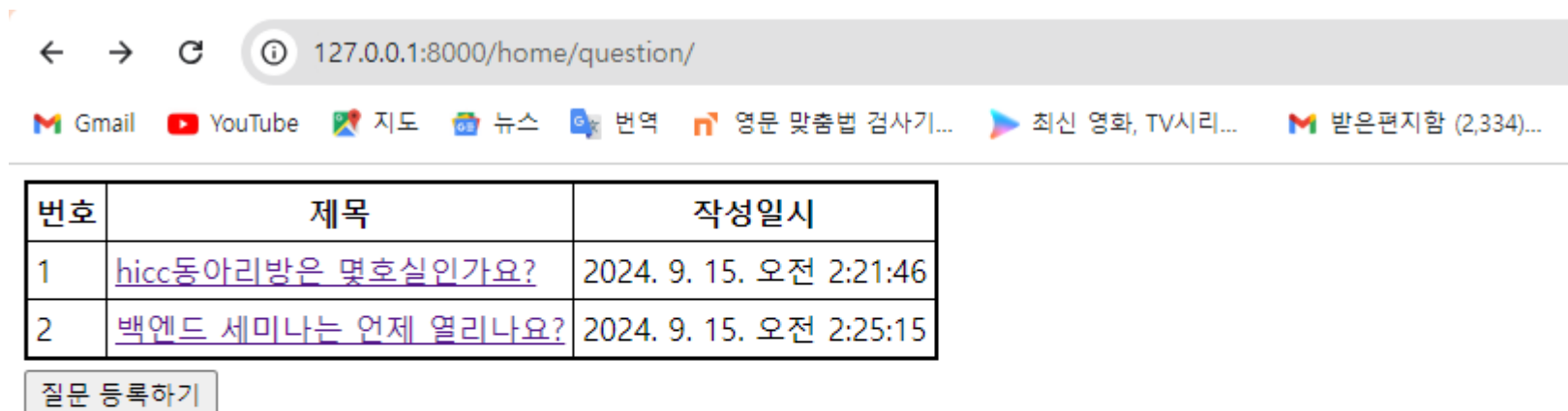
Question.objects는 해당 모델에 저장된 데이터를 조회할 수 있는 함수.

```
<QuerySet [<Question: Question object (1)>, <Question: Question object (2)>]>
```

라는 값이 출력됨을 알 수 있는데, 두개의 객체가 QuerySet의 형태로 저장되어있음을 나타냄.

(QuerySet은 list 자료형이 아닌 ORM에서만 사용하는 자료형)

Read 기능 구현하기



번호	제목	작성일시
1	hicc동아리방은 몇호실인가요?	2024. 9. 15. 오전 2:21:46
2	백엔드 세미나는 언제 열리나요?	2024. 9. 15. 오전 2:25:15

질문 등록하기

<http://127.0.0.1:8000/home/question/> 해당 링크에 위 사진같은 페이지를 구현하고자 한다. (Read 기능 구현)

프론트엔드는 백엔드로 부터 데이터를 받아올 js 코드를 짜야하고,
백엔드는 프론트엔드에게 DB의 데이터를 전송할 코드를 짜야한다.
(거의 모든 정보는 Json 형식으로 전달됨) ex) { 'price' : 2000 }

그것을 위해 백엔드는 어떠한 경로로 접속했을 때, 어떠한 정보를 전달해주겠다는 일종의 메뉴판을 미리 작성한다.
웹 프로그램에서 이 메뉴판의 역할을 API라고 부른다.

실습의 편의성을 위하여 api 경로는 method에 맞게 뒤에 CRUD를 직접 붙이는 것으로 통일함 (Create, Read, Update, Delete)

(자세한 백엔드 api 관련 내용은 4주차에 다룹니다.)

Read 기능 구현하기

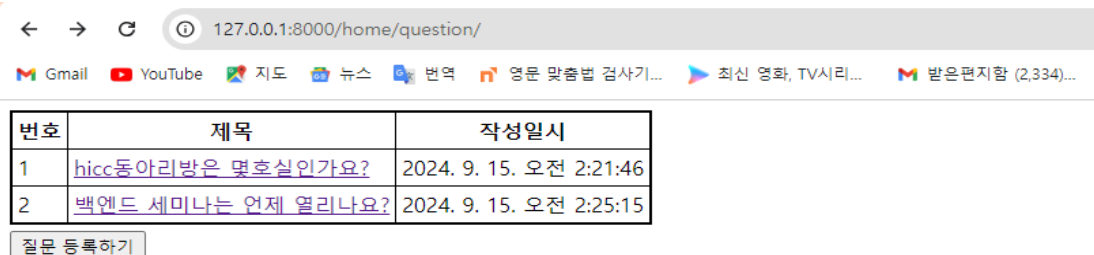
hicproject 폴더의 urls.py 파일을 수정해보자

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

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

```
urlpatterns = [  
    path("", views.index),  
    path('question/', views.question),  
    path('question/read/', views.question_read),  
    path('question/create/', views.question_create),  
]
```

Read 기능 구현하기



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/home/question/'. Below the address bar, there are several icons for Gmail, YouTube, 지도 (Map), 뉴스 (News), 번역 (Translate), 영문 맞춤법 검사기 (English Grammar Checker), 최신 영화, TV시리... (Latest Movies, TV Series...), and 받은편지함 (2,334) (Inbox). The main content area displays a table with three columns: 번호 (Number), 제목 (Subject), and 작성일시 (Created Date). The table contains two rows of data. Below the table, there is a button labeled '질문 등록하기' (Register Question).

번호	제목	작성일시
1	hicc동아리방은 몇호실인가요?	2024. 9. 15. 오전 2:21:46
2	백엔드 세미나는 언제 열리나요?	2024. 9. 15. 오전 2:25:15

질문 등록하기

해당 페이지를 구현하려면, 모든 Question 모델의 데이터들을 가져와야 한다.

hiccproject 폴더의 views.py 파일에도 해당 함수를 추가하자

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

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

```
    #Question 모델의 모든 데이터를 가져오는데, id 값은 'id'로, subject 값은 'subject'로, content 값은 'content'로 create_date 값은 'create_date'라는 key의 값으로 저장한다.
```

```
    데이터들은 Question 모델 객체의 변수로 저장되어 있는데, 해당 변수들을 json 형식으로 변환
```

```
    return JsonResponse({ ' questions ' : list(questions)})
```

```
    # questions는 QuerySet 형식이니 list 형식으로 강제 형변환 후 response
```

Read 기능 구현하기

<http://127.0.0.1:8000/home/question/read/>

서버를 가동 후 해당 링크에 접속하면 아래 사진처럼 데이터가 전송되는걸 확인할 수 있다.

pretty print 적용 ☒

```
{
  "questions": [
    {
      "id": 1,
      "subject": "hicc동마리방은 몇호실인가요? ",
      "content": "동마리방 위치를 알고 싶습니다.",
      "create_date": "2024-09-14T17:21:46.365Z"
    },
    {
      "id": 2,
      "subject": "백엔드 세미나는 언제 열리나요? ",
      "content": "날짜와 장소를 알고 싶습니다.",
      "create_date": "2024-09-14T17:25:15.336Z"
    }
  ]
}
```

프론트엔드는 해당 <http://127.0.0.1:8000/home/question/read/> url로 부터 정보를 받아와서 <http://127.0.0.1:8000/home/question/> 의 html 페이지에 정보들을 출력시킨다.

```
<script>
const fetchUrl = 'http://127.0.0.1:8000/home/question/read/';

fetch(fetchUrl)
  .then((response) => {
    if (!response.ok) {
      throw new Error('Network response was not ok.');
```

프론트엔드에서 미리 작성한 question.html의 js 코드
fetchUrl을 보면 <http://127.0.0.1:8000/home/question/read/> 를
가져온다

Read 기능 구현하기

이제 질문들이 모여있는 페이지는 구현했으니, 질문을 눌렀을 때 내용과 답변까지 볼 수 있는 페이지도 구현해보자.

먼저 Question처럼 Python Shell에서 Answer 테이블에 데이터를 넣어주자

python manage.py shell을 입력하고

```
>>> from hiccproject.models import Question, Answer
>>> from django.utils import timezone
>>> q = Question.objects.get(id=2)
>>> a = Answer(question=q, content='매주 목요일 6시에 진행됩니다.', create_date=timezone.now())
>>> a.save()
>>> a = Answer(question=q, content='강의실은 c710입니다.', create_date=timezone.now())
>>> a.save()
```

를 입력해주자

Read 기능 구현하기

```
>>> from hiccproject.models import Question, Answer
```

```
>>> from django.utils import timezone
```

```
>>> q = Question.objects.get(id=2)
```

objects.all과 달리 objects.get은 인자에 속성과 값을 넣어 조건에 맞는 데이터만 가져올 수 있다.

```
>>> a = Answer(question=q, content= ' 매주 목요일 6시에 진행됩니다. ', create_date=timezone.now())
```

question 속성은 외래키이고, 아까 데이터를 가져온 q를 넣어서 참조시킨다.

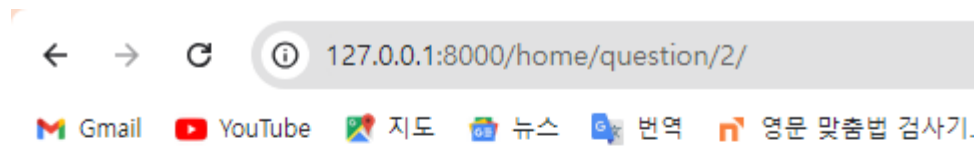
```
>>> a.save()
```

```
>>> a = Answer(question=q, content= ' 강의실은 C710입니다. ', create_date=timezone.now())
```

하나의 질문에 답변은 여러 개 넣을 수 있기 때문에 shell에서 두번째 질문에 두개의 답변을 만들어주자.

```
>>> a.save()
```

Read 기능 구현하기



백엔드 세미나는 언제 열리나요?

날짜와 장소를 알고 싶습니다.

2개의 답변이 있습니다.

- 매주 목요일 6시에 진행됩니다.
- 강의실은 C710입니다.

현재 이와같은 페이지를 구현하고자 한다.

이를 위해 question 페이지 처럼, 해당 html을 렌더링할 url 경로를 지정해주고 db로부터 데이터들을 받아올 api url 경로도 지정해주자.

Read 기능 구현하기

hiccproject 폴더의 urls.py에 들어가서 urlpatterns에

```
path('question/<int:question_id>/', views.question_detail),  
#렌더링 경로 지정
```

```
path('question/<int:question_id>/read/', views.question_detail_read),  
#api 경로 지정
```

해당 코드를 추가해주자

<int:question_id>는 url 경로의 해당 위치에 있는 int 형식의 값을 question_id라는 변수에 저장하여 views 함수로 전달한다는 의미이다.

즉 <http://127.0.0.1:8000/home/question/2/>

해당 링크의 2라는 값을 question_id라는 인자로 views.question_detail 함수에 전달한다는 의미

Read 기능 구현하기

views.py에 해당 함수를 추가해주자

```
def question_detail(request, question_id):  
    return render(request, 'hiccproject/question_detail.html')  
  
def question_detail_read(request, question_id):  
    question = Question.objects.get(id = question_id)  
    # 링크의 question_id와 id가 일치하는 Question만 가져옴  
  
    question_data = {  
        'id': question.id,  
        'subject': question.subject,  
        'content': question.content,  
        'create_date': question.create_date  
    }  
    # question이라는 객체의 변수들을 json 형식으로 변환하는 과정  
  
    return JsonResponse(question_data)
```

Read 기능 구현하기

<http://127.0.0.1:8000/home/question/2/>

해당 링크로 접속하면 질문과 내용을 읽어오는 페이지가 나온다.

하지만 현재 코드에서 문제점이 하나 있다.

<http://127.0.0.1:8000/home/question/3/>

같은 존재하지 않는 질문의 페이지로 이동해도 접근 불가가 뜨지 않고 html 페이지가 렌더링된다.

이를 해결하기 위해 코드를 살짝 수정해보자.

```
from django.shortcuts import render, get\_object\_or\_404
```

```
def question_detail(request, question_id):
```

```
    question = get\_object\_or\_404(Question, id = question_id)
```

```
    #question_id가 존재하지 않는 페이지로 들어가면 404 에러가 뜨도록 설정, question =  
    Question.objects.get(id = question_id)와 동일한 역할이지만 오류 발생시 404가 뜬다는게 차이,
```

```
    return render(request, 'hiccproject/question_detail.html')
```

Read 기능 구현하기

마지막으로 해당 페이지에서 질문의 답변들을 화면에 출력시키는 기능을 구현해보자.
해당 페이지를 구현하려면, 모든 Question 모델의 데이터들을 가져와야 한다.

hicproject 폴더의 urls.py에 들어가서 urlpatterns에

`path('question/<int:question_id>/answer/read/', views.answer_read),`
해당 코드를 추가

views.py에

```
def answer_read(request, question_id):  
    question = Question.objects.get(id = question_id)  
  
    answers = Answer.objects.filter(question = question_id).values('id', 'question_id', 'content',  
        'create_date')  
  
    return JsonResponse({'answers' : list(answers)})
```

해당 코드를 추가

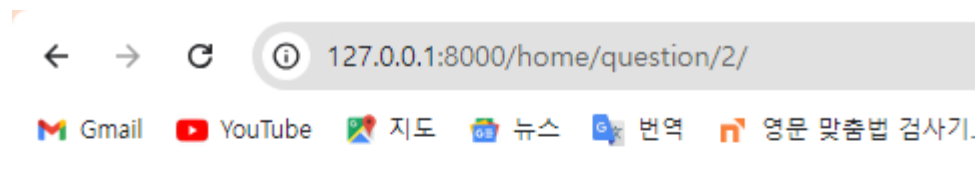
Read 기능 구현하기

```
def answer_read(request, question_id):  
    question = Question.objects.get(id = question_id)  
    #해당 질문을 참조하고 있는 답변을 가져와야 하기 때문에 작성  
  
    answers = Answer.objects.filter(question = question_id).values('id', 'question_id', 'content',  
'create_date')  
    #모든 답변 데이터를 가져올건데, objects.all()이 아닌 objects.filter(question =  
    question_id)는 인자로 받는 question_id와 id가 일치하는 question들의 답변 데이터만  
    가져온다는 의미  
  
    return JsonResponse({ ' answers ' : list(answers)})
```

<http://127.0.0.1:8000/home/question/2/>

해당 링크로 접속하면 우측과 같은 페이지가 구현됨을 확인

우리는 이제 db의 데이터를 받아서 웹페이지에서 읽는
Read 기능을 구현해냈다.



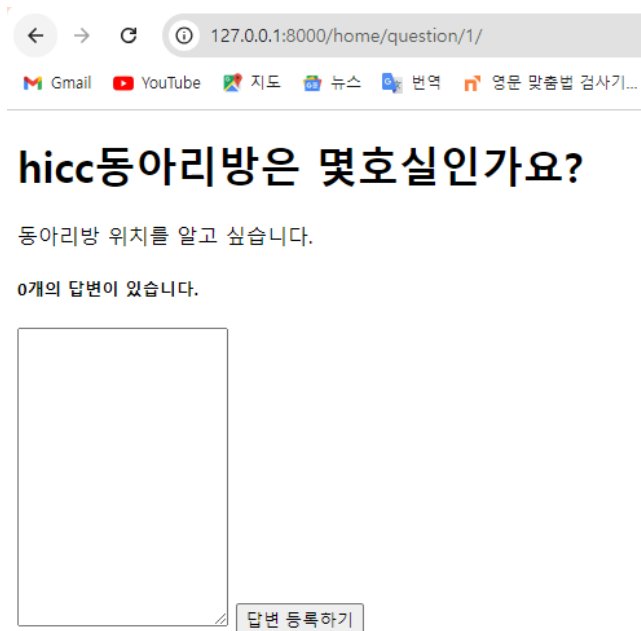
백엔드 세미나는 언제 열리나요?

날짜와 장소를 알고 싶습니다.

2개의 답변이 있습니다.

- 매주 목요일 6시에 진행됩니다.
- 강의실은 C710입니다.

Create 기능 구현하기



질문에 대한 답변을 등록하는 기능을 먼저 구현해보자.

해당 텍스트 상자에 내용을 적고 답변 등록하기를 누르면 답변이 저장되는 기능을 구현하고자 한다.

Create 기능 구현하기

hiccproject 폴더의 urls.py에 들어가서 urlpatterns에

```
path('question/<int:question_id>/answer/create/', views.answer_create),
```

해당 코드를 추가해주자

views.py에 들어가서

```
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
```

```
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())
    answer.save()
    return redirect('question_detail', question_id=question.id)
```

해당 코드를 추가해주자

Create 기능 구현하기

추가로 hiccproject의 urls.py에 다시 돌아와서

```
path('question/<int:question_id>/', views.question_detail, name = "question_detail"),
```

해당 코드를 추가해주자.

하는 김에 다른 url 링크에도 함수 이름과 동일한 name을 설정해주자.

Create 기능 구현하기

`path('question/<int:question_id>/answer/create/', views.answer_create),`
해당 코드를 통해 db에 데이터를 Create하고 저장한다.

```
<form action = "answer/create/" method = "post">
  <textarea id="answer_content" name = "answer_content" rows="15"></textarea>
  <input type="submit" value="답변 등록하기">
</form>
```

해당 html 코드에서, 답변 등록하기 버튼을 누르면

http://127.0.0.1:8000/home/question/<int:question_id>/answer/create/

해당 url 경로로 이동하고, post 방식으로 데이터가 전송되도록 미리 작성해두었고

path 함수로 인해 답변 등록하기 버튼을 누르면 `views.answer_create` 함수가 바로 실행된다.

Create 기능 구현하기

```
def answer_create(request, question_id):
```

```
    question = get_object_or_404(Question, id = question_id)
```

#question_id 값의 페이지가 존재하지 않으면 404 뜨도록 설정

```
    answer = Answer(question=question, content=request.POST.get('answer_content'),
```

```
    create_date=timezone.now())
```

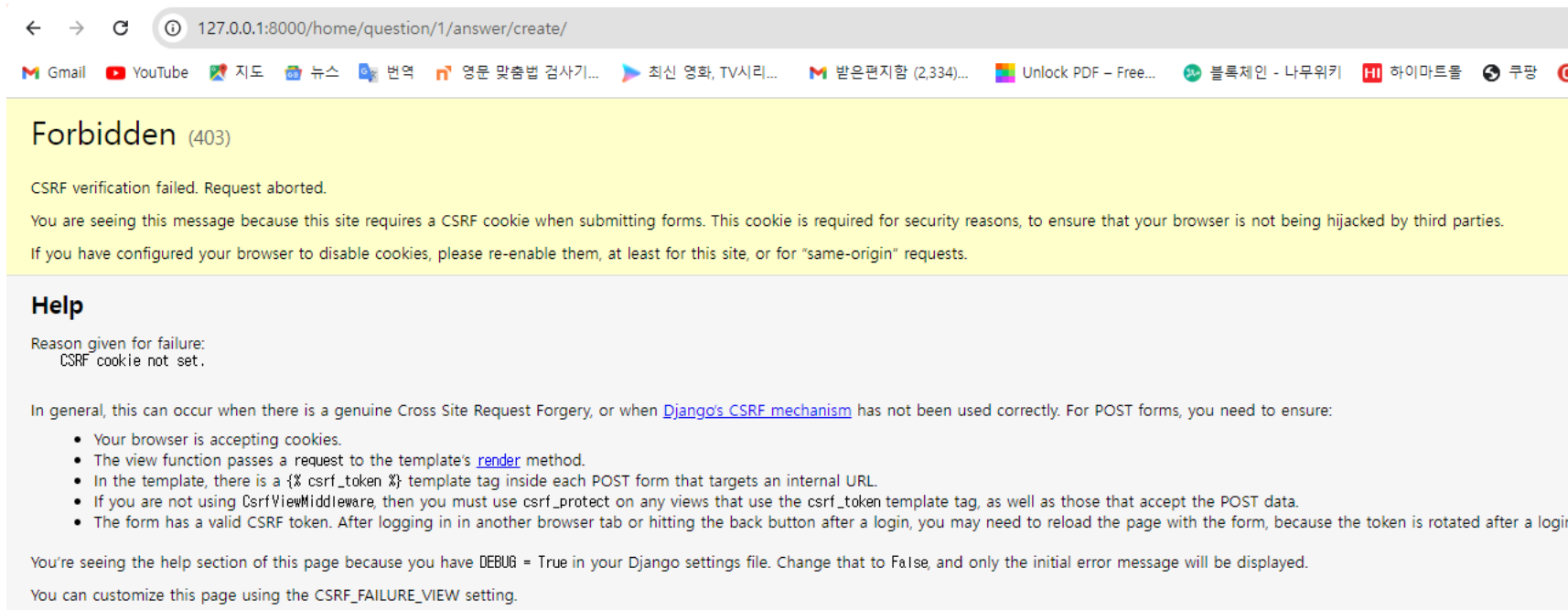
#content=request.POST.get('answer_content')는 POST 요청일 때 name이 answer_content인 영역의 값을 content라는 변수에 저장한다는 의미

```
    answer.save()
```

```
    return redirect('question_detail', question_id=question.id)
```

#name = 'question_detail'이라는 url 경로로 리다이렉트 한다, 해당 경로가 'question/<int:question_id>/'이므로 두번째 인자의 question_id=question.id를 통해 대입하여 url 값을 채워준다

Create 기능 구현하기

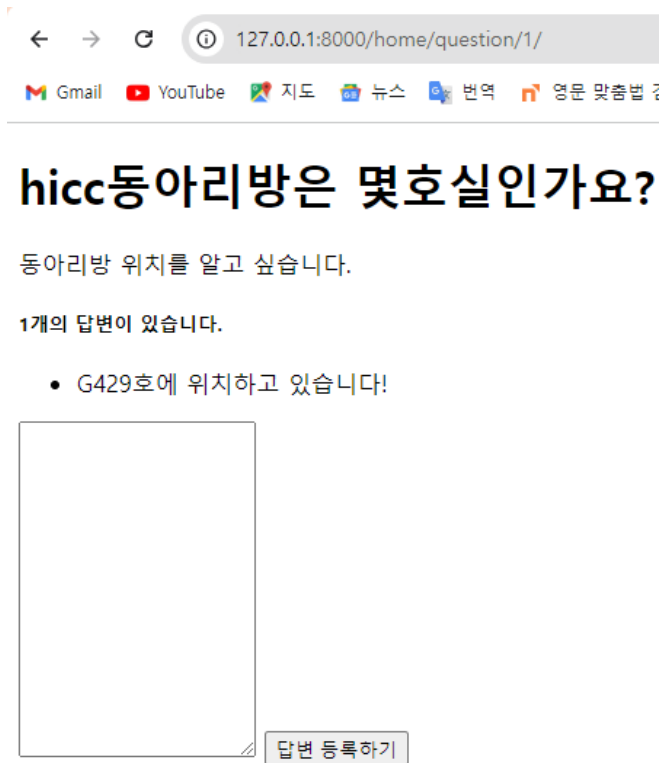


Django의 기본적인 보안 기능인 csrf_token 때문에 답변 등록 기능이 막혔다. 해당 내용은 3강에서 자세히 다루도록 하고, Create 기능 구현에만 집중해보자.

settings.py에 들어가서 드래그 된 해당 부분을 지워서 csrf_token 보안을 해제해보자

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

Create 기능 구현하기



The screenshot shows a web browser interface. The address bar displays the URL '127.0.0.1:8000/home/question/1/'. Below the address bar, there are several icons for services like Gmail, YouTube, and a map. The main content area features a question titled 'hiccdongaribang은 몇호실인가요?' (Which room number is hiccdongaribang?). Below the question, it says '동아리방 위치를 알고 싶습니다.' (I want to know the location of the club room). It also indicates '1개의 답변이 있습니다.' (There is 1 answer). A single bullet point lists 'G429호에 위치하고 있습니다!' (It is located in room G429!). Below this, there is a large empty text input box and a button labeled '답변 등록하기' (Register answer).

← → ↻ ⓘ 127.0.0.1:8000/home/question/1/

Gmail YouTube 지도 뉴스 번역 영문 맞춤법

hiccdongaribang은 몇호실인가요?

동아리방 위치를 알고 싶습니다.

1개의 답변이 있습니다.

- G429호에 위치하고 있습니다!

답변 등록하기

해당 코드를 그대로 적으면, 실제로 텍스트 상자에 넣은 답변이 등록하기 버튼을 통해 등록이 된다!

Create 기능 구현하기

답변 등록하기 기능을 구현했으니, 이번엔 질문 등록하기 기능을 구현해보자.

사실 우리는 질문을 만드는 url 경로는 이미 구현을 해놓았다.

urls.py 파일을 열면 `path('question/create/', views.question_create)`, 코드가 이미 존재하는 것을 볼 수 있다.

하지만 views.py를 열어서 `question_create` 함수를 살펴보면 렌더링은 구현이 됐지만 데이터 저장은 구현이 안되었음을 알 수 있다.

코드를 수정해서 상황에 따라 렌더링을 하거나 데이터를 저장할 수 있도록 구현해보자.

Create 기능 구현하기

views.py 에서 코드를 수정해보자.

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


Create 기능 구현하기

질문 저장하기

```
if request.method == 'POST':
```

#해당 버튼을 클릭하면 <http://127.0.0.1:8000/home/question/create/> 에 POST로 요청이 가도록
html 코드를 미리 작성해두었음

```
    question = Question(subject = request.POST.get('subject'), content =  
request.POST.get('content'), create_date=timezone.now())  
    #html의 name = "subject", name = "content"인 영역의 값을 변수에 저장
```

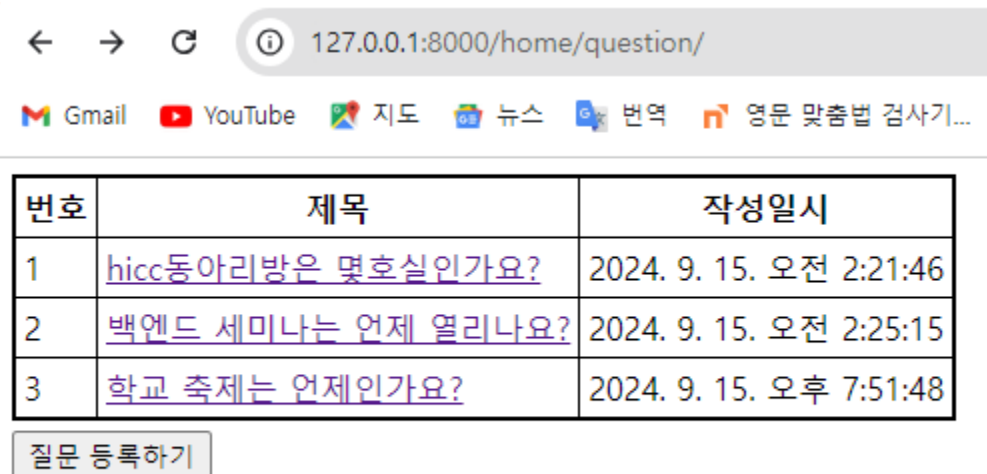
```
    question.save()  
    return redirect('question') #name = "question"인 url로 리다이렉트, 즉  
http://127.0.0.1:8000/home/question/ 로 리다이렉트
```

```
else:
```

#POST 요청이 아니라면, 즉 <http://127.0.0.1:8000/home/question/create/> 링크로 그냥 접속했을
때 (GET 요청일 때)

```
    return render(request, 'hiccproject/question_create.html')  
    #그냥 해당 질문 작성 페이지를 렌더링
```

Create 기능 구현하기



The screenshot shows a web browser interface. The address bar displays the URL '127.0.0.1:8000/home/question/'. Below the address bar, there are several utility links: Gmail, YouTube, 지도 (Map), 뉴스 (News), 번역 (Translate), and 영문 맞춤법 검사기... (English Grammar Checker...). The main content area features a table with three columns: '번호' (Number), '제목' (Title), and '작성일시' (Created Date/Time). The table contains three rows of generated questions. Below the table, there is a button labeled '질문 등록하기' (Register Question).

번호	제목	작성일시
1	hicc동아리방은 몇호실인가요?	2024. 9. 15. 오전 2:21:46
2	백엔드 세미나는 언제 열리나요?	2024. 9. 15. 오전 2:25:15
3	학교 축제는 언제인가요?	2024. 9. 15. 오후 7:51:48

질문 등록하기

질문을 생성하는 기능이 실제로 구현되었음을 확인할 수 있다.

CRUD 기능 구현 실습

학교 축제는 언제인가요?

축제 기간이 궁금합니다.

1개의 답변이 있습니다.

- 9/25 ~ 9/27입니다.

question_detail 페이지를 보면, 질문을 수정하고 삭제할 수 있는 버튼들이 프론트에서 이미 구현이 되어있다. (question_update.html 파일도 백엔드 repo에 있습니다.)

지금까지 배워온 내용을 바탕으로, view.py와 urls.py만 수정하여 update 기능과 delete 기능을 구현해보자.

update 경로는 question/<int:question_id>/update/ 이고,
delete 경로는 question/<int:question_id>/delete/이다.

수정에 성공하면 question_detail 페이지로 리다이렉트되고,
삭제에 성공하면 question 페이지로 리다이렉트된다.

힌트 :

update는 POST method로 구현한다는 점에서 question_create 함수와 유사합니다.

delete는 model 함수 중 객체.delete() 라는 함수를 사용합니다. 해당 함수는 save() 함수를 사용하지 않아도 적용이 됩니다.