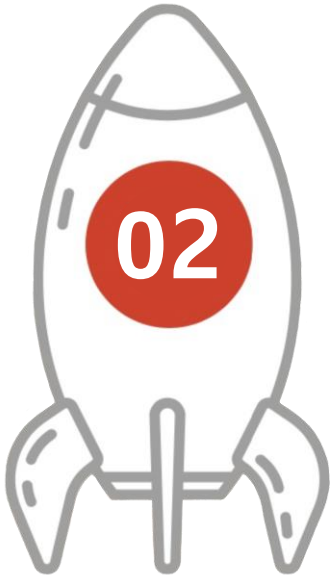
파이썬 웹 개발부터 배포까지!

점프 투 장고

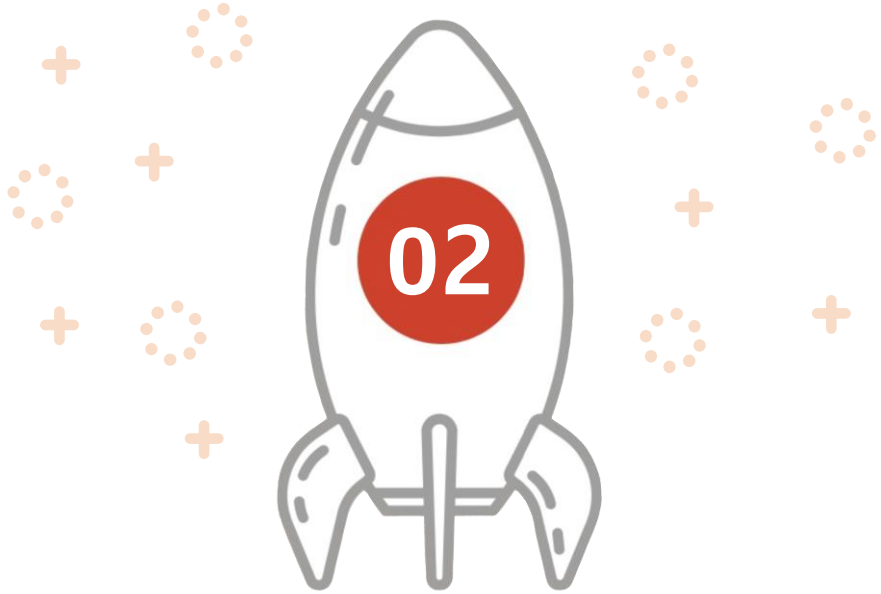


장고의 기본 요소 익히기!

INDEX

장고의 기본 요소 익히기!

INDEX



장고의 기본 요소 익히기!

이장의
목표

- urls.py 파일을 이용해 URL과 매핑되는 뷰 함수를 관리한다.
- 장고 ORM을 이용해 데이터베이스를 제어한다.
- 파이보 게시판에 질문 목록과 질문 상세 기능을 만든다.

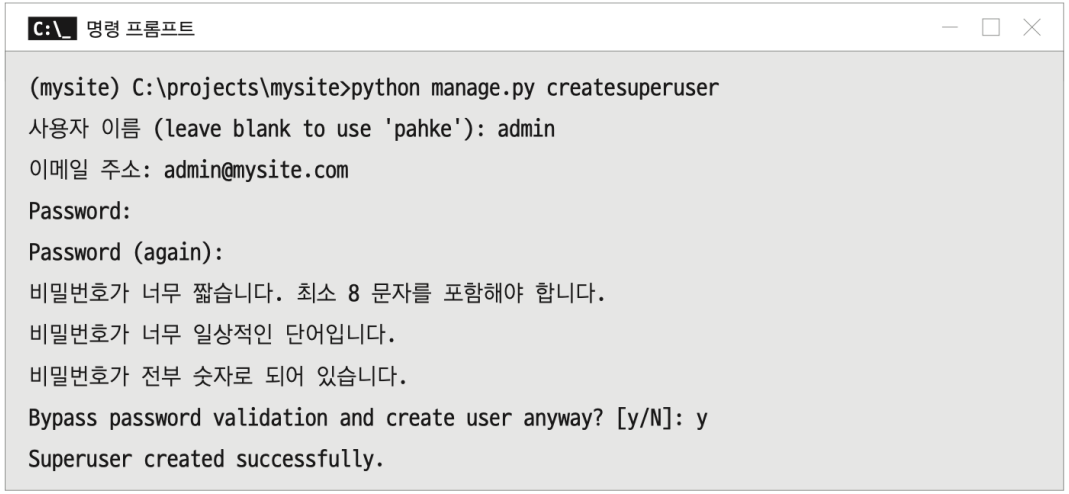
02-3 개발 편의를 제공하는 장고 Admin

• 완성 소스 github.com/pahkey/djangobook/tree/2-03

Do it! 실습

장고 Admin 사용하기

01단계 슈퍼 유저 생성하기



▸ python manage.py createsuperuser 명령을 실행하여 슈퍼 유저를 생성하자.

항목	값
사용자명	admin
이메일 주소	admin@mysite.com
비밀번호	1111

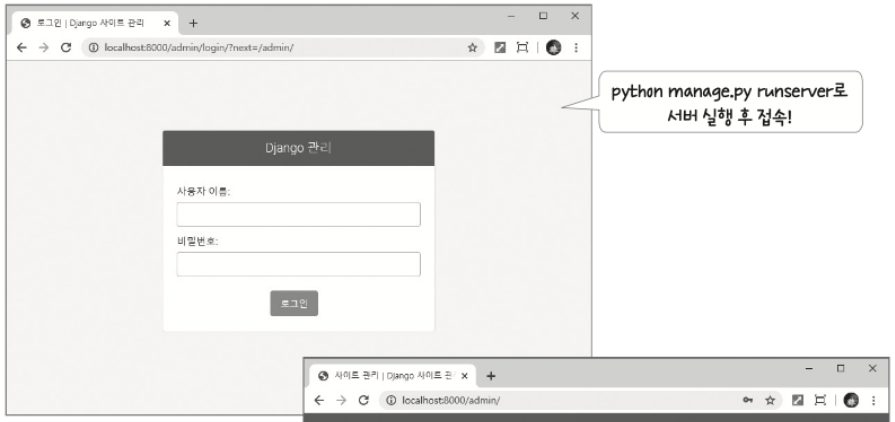
02-3 개발 편의를 제공하는 장고 Admin

• 완성 소스 github.com/pahkey/djangobook/tree/2-03

Do it! 실습

장고 Admin 사용하기

02단계 장고 Admin에 접속해 로그인하기



장고 Admin 로그인 화면

장고 Admin 로그인 후 볼 수 있는 화면

- 장고 개발 서버를 구동한 후 localhost:8000/admin에 접속해 보자.
- 장고 Admin에서는 현재 등록된 그룹 및 사용자에 대한 정보 확인과 수정을 할 수 있다.

02-3 개발 편의를 제공하는 장고 Admin

• 완성 소스 github.com/pahkey/djangobook/tree/2-03

Do it! 실습

장고 Admin 사용하기

03단계 장고 Admin에서 모델 관리하기

파일 이름 C:/projects/mysite/pybo/admin.py

```
from django.contrib import admin
from .models import Question

admin.site.register(Question)
```

- Question, Answer 모델을 장고 Admin에 등록하면 손쉽게 관리할 수 있다.

04단계 장고 Admin 새로고침 하기

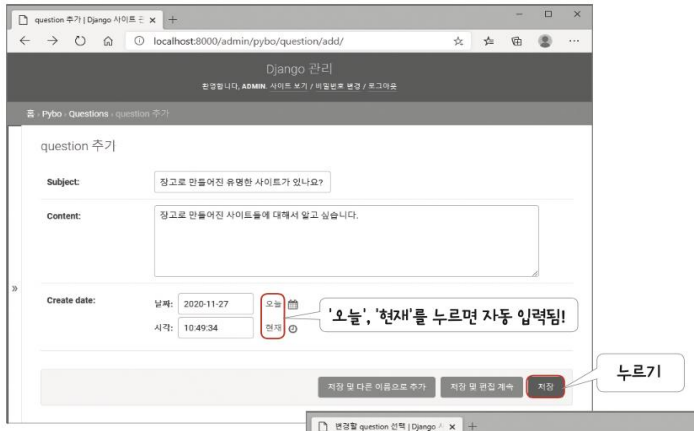
02-3 개발 편의를 제공하는 장고 Admin

• 완성 소스 github.com/pahkey/djangobook/tree/2-03

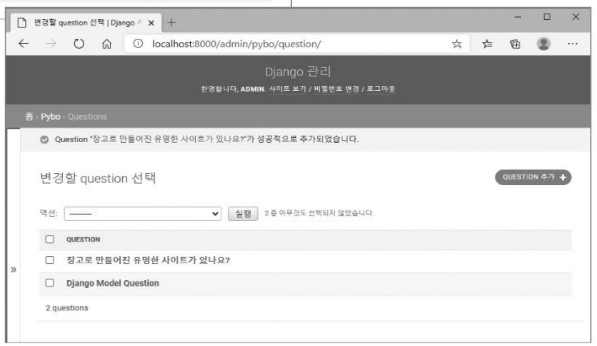
Do it! 실습

장고 Admin 사용하기

05단계 Question 모델 데이터 추가하기



모델 데이터 등록 화면



모델 데이터 등록 후 볼 수 있는 화면

- 화면에서 Question 모델의 <+추가> 버튼을 누르자.
- Question 모델의 속성에 맞는 값을 입력하고 <저장> 버튼을 누르자.
- 저장 후 이전에 연습한 shell에서 확인하면, 정상적으로 등록이 되었다.

```
(venv) C:\projets\mysite>python manage.py shell
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from pybo.models import Question, Answer
>>> Question.objects.filter()
<QuerySet [<Question: 제목 테스트 1>]>
```

02-3 개발 편의를 제공하는 장고 Admin

• 완성 소스 github.com/pahkey/djangobook/tree/2-03

Do it! 실습

장고 Admin 사용하기

06단계 장고 Admin에 데이터 검색 기능 추가하기

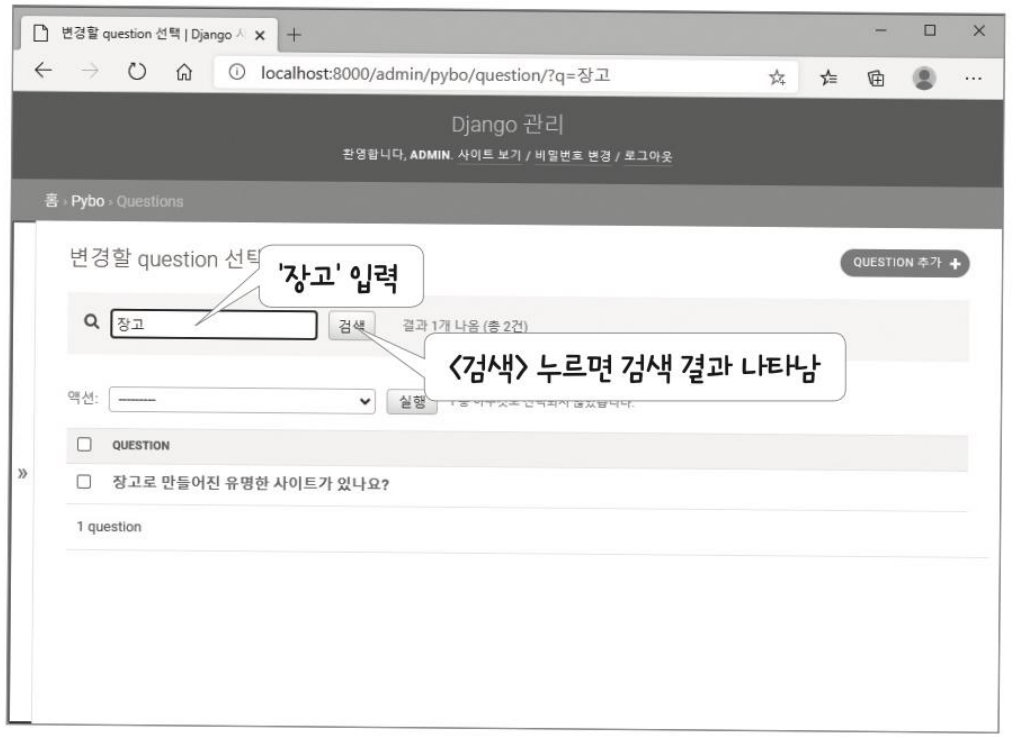
파일 이름 C:/projects/mysite/pybo/admin.py

```
from django.contrib import admin
from .models import Question
class QuestionAdmin(admin.ModelAdmin):
    search_fields = ['subject']

admin.site.register(Question, QuestionAdmin)
```

- pybo/admin.py 파일에 QuestionAdmin 클래스를 추가하고 search_fields에 'subject'를 추가하자.

07단계 장고 Admin에서 데이터 검색해 보기



02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

질문 목록 조회 구현하기

01단계 Question 모델 데이터 작성일시 역순으로 조회하기

파일 이름 C:/projects/mysite/pybo/views.py

```
from django.http import HttpResponse
from .models import Question

def index(request):
    """
    pybo 목록 출력
    """
    question_list = Question.objects.order_by('-create_date')
    context = {'question_list': question_list}
    return HttpResponse("안녕하세요 pybo에 오신것을 환영합니다.")
```

- order_by 함수는 조회한 데이터를 특정 속성으로 정렬하며, '-create_date'는 작성일시의 역순을 의미한다.

02단계 render로 화면 출력하기

파일 이름 C:/projects/mysite/pybo/views.py

```
from django.shortcuts import render
from .models import Question

def index(request):
    """
    pybo 목록 출력
    """
    question_list = Question.objects.order_by('-create_date')
    context = {'question_list': question_list}
    return render(request, 'pybo/question_list.html', context)
```

- render 함수는 context에 있는 Question 모델 데이터 question_list를 pybo/question_list.html 파일에 적용하여 HTML 코드로 변환한다.

02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

질문 목록 조회 구현하기

03단계 템플릿을 모아 저장할 디렉터리 만들기

C:_ 명령 프롬프트

```
(mysite) C:\projects\mysite>mkdir templates
```

- 템플릿을 저장할 디렉터리를 루트 디렉터리 바로 밑에 만들자.

04단계 템플릿 디렉터리 위치 config/settings.py에 등록하기

파일 이름 C:/projects/mysite/config/settings.py

```
TEMPLATES = [
    {
        (... 생략 ...)

        'DIRS': [BASE_DIR / 'templates'],

        (... 생략 ...)
    },
]
```

- DIRS에는 템플릿 디렉터리를 여러 개 등록할 수 있다.
- 현재 우리가 개발하는 파이보는 1개의 템플릿 디렉터리를 쓸 것이므로 1개의 디렉터리만 등록하자.

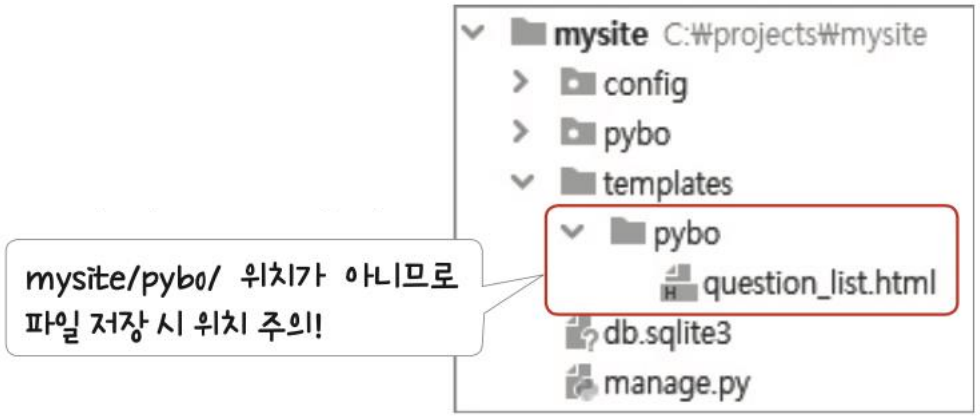
02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

질문 목록 조회 구현하기

05단계 템플릿 파일 만들기



- question_list.html 템플릿 파일을 mysite/templates/pybo/ 디렉터리에 생성하자.

파일 이름 C:/projects/mysite/templates/pybo/question_list.html

```
{% if question_list %}
    <ul>
    {% for question in question_list %}
        <li><a href="/pybo/{{ question.id }}/">{{ question.subject }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>질문이 없습니다.</p>
{% endif %}
```

- 템플릿 태그는 {% 와 %}로 둘러싸인 문장을 말한다.

템플릿 태그	의미
{% if question_list %}	question_list가 있다면
{% for question in question_list %}	question_list를 반복하며 순차적으로 question에 대입
{{ question.id }}	for 문에 의해 대입된 question 객체의 id 출력
{{ question.subject }}	for 문에 의해 대입된 question 객체의 subject 출력

02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

질문 목록 조회 구현하기

06단계 질문 목록이 잘 출력되는지 확인해 보기

- 장고 개발 서버를 다시 시작하지 않으면 장고가 템플릿 디렉터리를 인식하지 못해 오류가 발생한다.

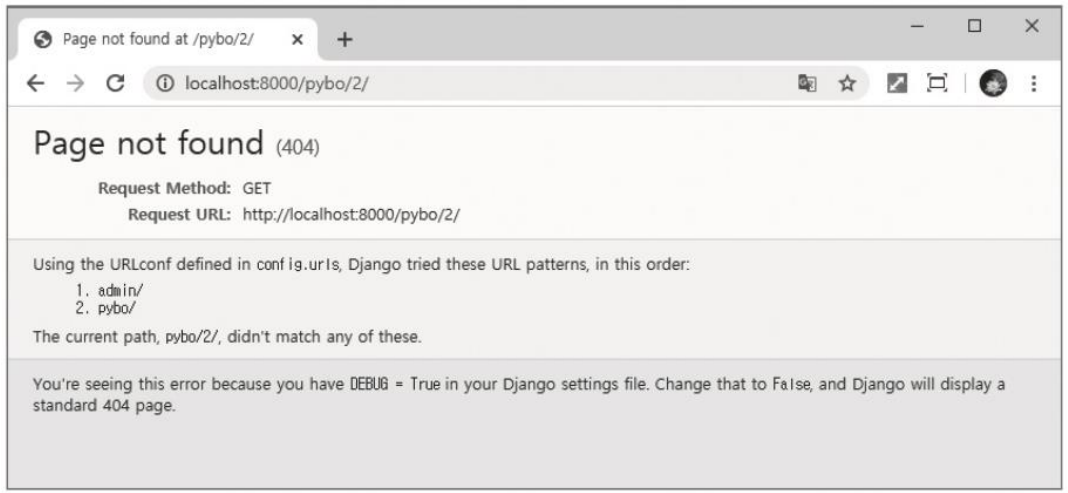
02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

질문 상세 기능 구현하기

01단계 질문 목록에서 아무 질문이나 눌러 보기



- 오류 화면이 나타난 이유는 /pybo/2/ 에 대한 URL 매핑을 추가하지 않았기 때문이다.

02단계 pybo/urls.py 열어 URL 매핑 추가하기

파일 이름 C:/projects/mysite/pybo/urls.py

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

urlpatterns = [
    path('', views.index),
    path('<int:question_id>/', views.detail),
]
```

- /pybo/2/가 요청되면 이 매핑 규칙에 의해 /pybo/<int:question_id>/가 적용되어 `question_id`에 2라는 값이 저장되고 `views.detail` 함수가 실행된다.

02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

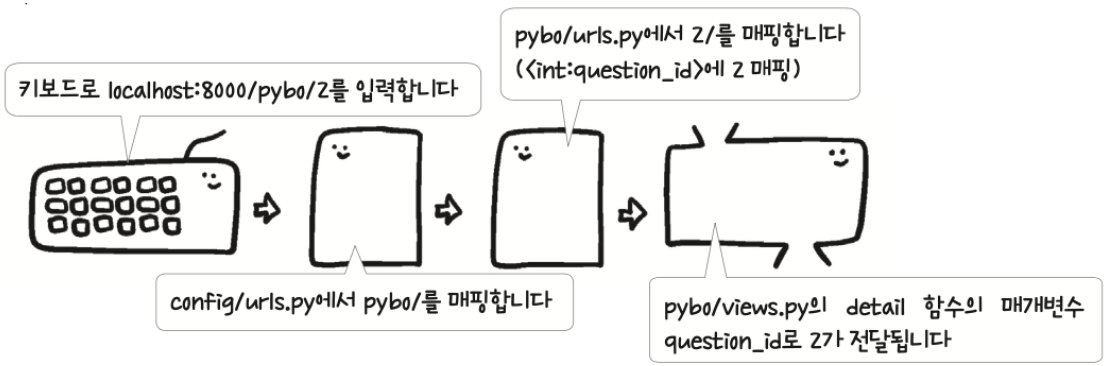
질문 상세 기능 구현하기

03단계 pybo/views.py 열어 화면 추가하기

파일 이름 C:/projects/mysite/pybo/views.py

```
(... 생략 ...)

def detail(request, question_id):
    """
    pybo 내용 출력
    """
    question = Question.objects.get(id=question_id)
    context = {'question': question}
    return render(request, 'pybo/question_detail.html', context)
```



/pybo/2/ 페이지 호출 시 장고에서 벌어지는 일

- /pybo/2/ 페이지가 호출되면 최종으로 `detail` 함수의 매개변수 `question_id`에 2가 전달된다.

02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

질문 상세 기능 구현하기

04단계 pybo/question_detail.html 작성하기

파일 이름 C:/projects/mysite/templates/pybo/question_detail.html

```
<h1>{{ question.subject }}</h1>

<div>
    {{ question.content }}
</div>
```

- {{ question.subject }}, {{ question.content }}의 question 객체는 detail 함수에서 render 함수에 전달한 context에 저장한 데이터이다.

05단계 질문 상세 페이지에 접속해 보기

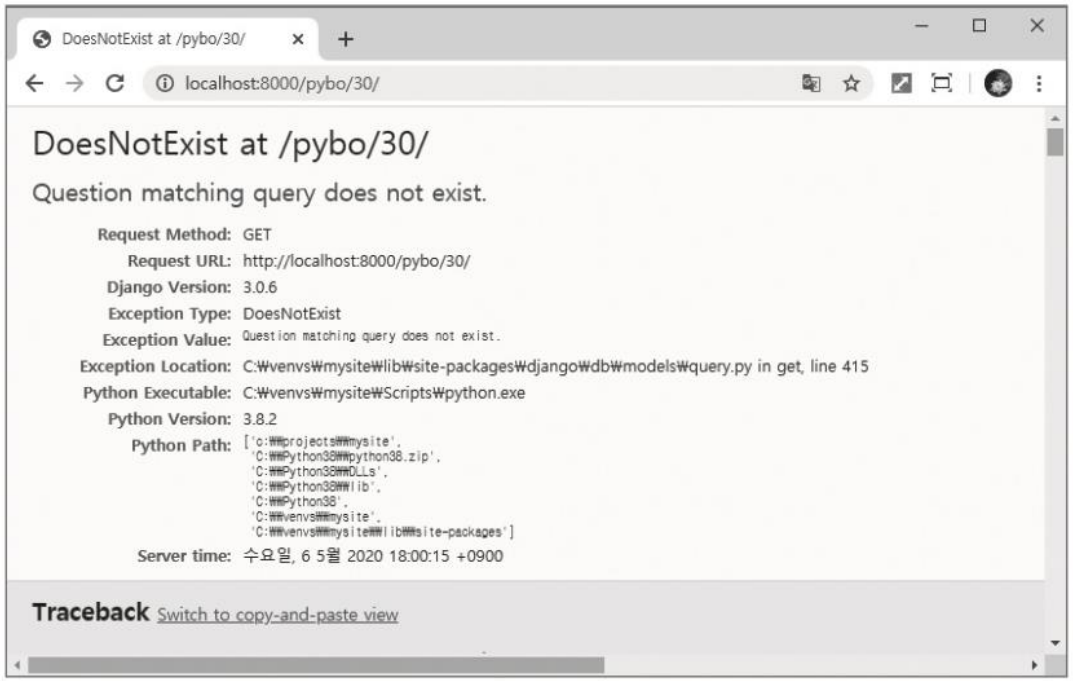
02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

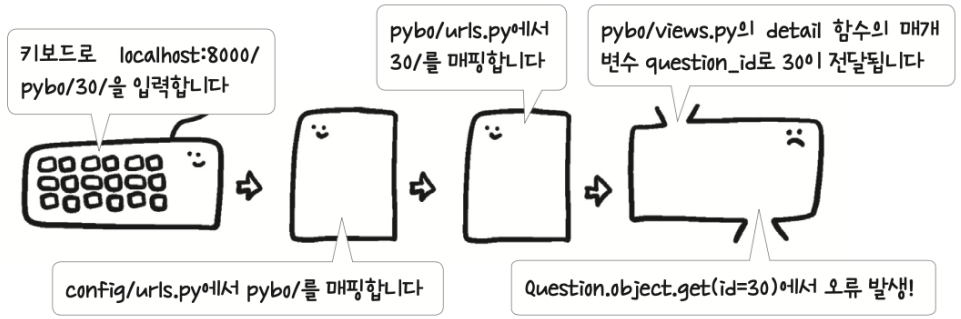
Do it! 실습

오류 화면 구현하기

01단계 잘못된 주소에 접속해 보기



- question_id가 30인 데이터를 조회하는 Question.object.get(id=30)에서 오류가 발생했기 때문이다.



/pybo/30/ 페이지 호출 시 장고에서 벌어지는 일

02-4 질문 목록과 질문 상세 기능 구현하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-04

Do it! 실습

오류 화면 구현하기

02단계 페이지가 존재하지 않음(404페이지) 출력하기

파일 이름 C:/projects/mysite/pybo/views.py

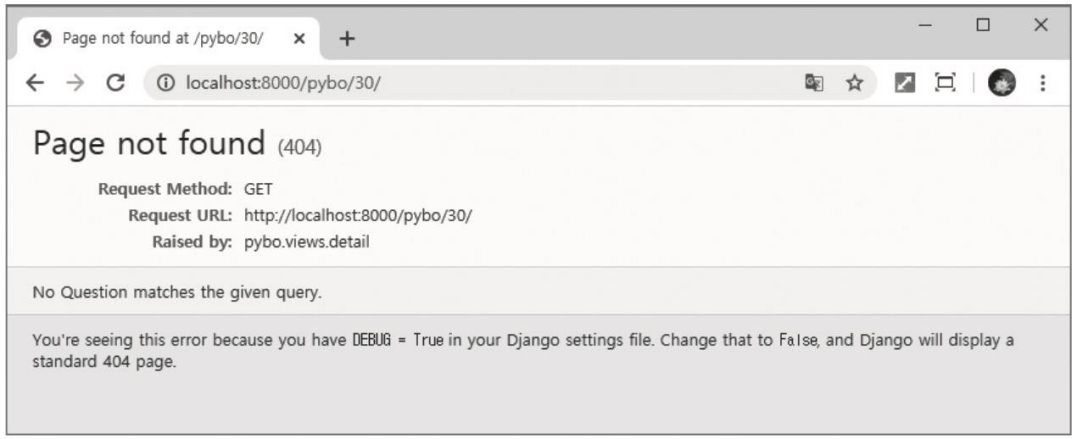
```
from django.shortcuts import render, get_object_or_404

(... 생략 ...)

def detail(request, question_id):
    """
    pybo 내용 출력
    """
    question = get_object_or_404(Question, pk=question_id)
    context = {'question': question}
    return render(request, 'pybo/question_detail.html', context)
```

- get_object_or_404 함수는 모델의 기본키를 이용하여 모델 객체 한 건을 반환한다.

03단계 404 페이지 출력 확인하기



- /pybo/30/ 에 접속하면 404 페이지가 출력된다.

02-5 URL 더 똑똑하게 사용하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-05

URL 하드 코딩이란 무엇일까?

question_list.html 템플릿의 href값

```
<li><a href="/pybo/{{ question.id }}/">{{ question.subject }}</a></li>
```

URL 규칙

- **URL 하드 코딩의 한계**

  : URL 규칙이 자주 변경된다면 템플릿에 사용된 모든 href값들을 일일이 찾아 수정해야 한다.

  : ex) '/pybo/question/2/', '/pybo/2/question/' 같은걸로 수정 한다면 ?

- **해결 방법**

  : 해당 URL에 대한 실제 주소가 아닌 주소가 매핑된 URL 별칭을 사용해야 한다.

02-5 URL 더 똑똑하게 사용하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-05

Do it! 실습

URL 별칭으로 URL 하드 코딩 문제 해결하기

01단계 pybo/urls.py 수정하여 URL 별칭 사용하기

파일 이름 C:/projects/mysite/pybo/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>/', views.detail, name='detail'),
]
```

- path 함수에 있는 URL 매핑에 name 속성을 부여하자.

02단계 pybo/question_list.html 템플릿에서 URL별칭 사용하기

파일 이름 C:/projects/mysite/templates/pybo/question_list.html

```
(... 생략 ...)
    {% for question in question_list %}
        <li><a href="{% url 'detail' question.id %}">{{ question.subject }}</a></li>
    {% endfor %}
(... 생략 ...)
```

- /pybo/{{ question.id }}를 {% url 'detail' question.id %} 로 변경하자.

02-5 URL 더 똑똑하게 사용하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-05

Do it! 실습

URL 네임스페이스 알아보기

문제점 서로 다른 앱에서 같은 URL 별칭을 사용하면 중복 문제가 생긴다.

해결방법 pybo/urls.py 파일에 네임스페이스 라는 개념을 도입해야 한다.

*네임스페이스(namespace) : 각각의 앱이 관리하는 독립된 이름 공간

서로 다른 앱에서 같은 URL 별칭을 사용하면 벌어지는 일

02-5 URL 더 똑똑하게 사용하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-05

Do it! 실습

URL 네임스페이스 알아보기

01단계 pybo/urls.py에 네임스페이스 추가하기

파일 이름 C:/projects/mysite/pybo/urls.py

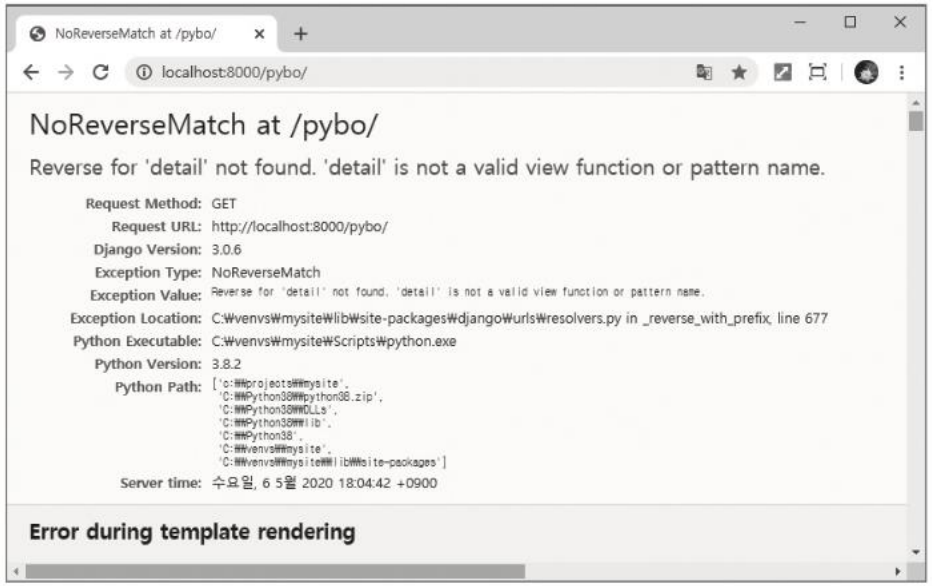
```
from django.urls import path
from . import views

app_name = 'pybo'

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>/', views.detail, name='detail'),
]
```

- pybo/urls.py 파일에 네임스페이스를 추가하려면 간단히 app_name 변수에 네임스페이스 이름을 저장하면 된다.

02단계 네임스페이스 테스트하기 - 오류 발생!



02-5 URL 더 똑똑하게 사용하기

• 완성 소스 github.com/pahkey/djangobook/tree/2-05

Do it! 실습

URL 네임스페이스 알아보기

03단계 pybo/question_list.html 수정하기

파일 이름 C:/projects/mysite/templates/pybo/question_list.html

```
(... 생략 ...)
    {% for question in question_list %}
        <li><a href="{% url 'pybo:detail' question.id %}">{{ question.subject }}</a></li>
    {% endfor %}
(... 생략 ...)
```

- 오류가 발생한 이유는 템플릿에서 아직 네임스페이스를 사용하고 있지 않기 때문이다.

- {% url ‘detail’ question.id %}을 {% url ‘pybo:detail’ question.id %}으로 바꾸자.

02-6 답변 등록 기능 만들기

• 완성 소스 github.com/pahkey/djangobook/tree/2-06

Do it! 실습

답변 저장하고 표시하기

01단계 질문 상세 템플릿에 답변 등록 버튼 만들기

파일 이름 C:/projects/mysite/templates/pybo/question_detail.html

```
<h1>{{ question.subject }}</h1>

<div>
    {{ question.content }}
</div>

<form action="{% url 'pybo:answer_create' question.id %}" method="post">
{% csrf_token %}
<textarea name="content" id="content" rows="15"></textarea>
<input type="submit" value="답변 등록">
</form>
```

- form 엘리먼트 안에 textarea 엘리먼트와 input 엘리먼트를 포함시켜 답변 내용, 답변 등록 버튼을 추가하자.

- {% csrf_token %}는 form 엘리먼트를 통해 전송된 데이터가 실제로 웹 브라우저에서 작성된 데이터인지 판단하는 검사기 역할을 한다.

csrf_token은 장고의 기본 기능이다

csrf_token을 사용하려면 장고에 CsrfViewMiddleware라는 미들웨어를 추가해야 한다. 하지만 이 미들웨어는 장고 프로젝트 생성 시 자동으로 config/settings.py 파일의 MIDDLEWARE라는 항목에 추가되므로 여러분이 직접 입력할 필요는 없다.

파일 이름 C:/projects/mysite/config/settings.py

```
(... 생략 ...)
MIDDLEWARE = [
    (... 생략 ...)
    'django.middleware.csrf.CsrfViewMiddleware',
    (... 생략 ...)
]
(... 생략 ...)
```

혹시라도 csrf_token을 사용하고 싶지 않다면 config/settings.py 파일의 MIDDLEWARE 항목에서 해당 코드를 주석 처리하면 되겠지만, csrf_token 기능을 굳이 제외할 필요는 없다.

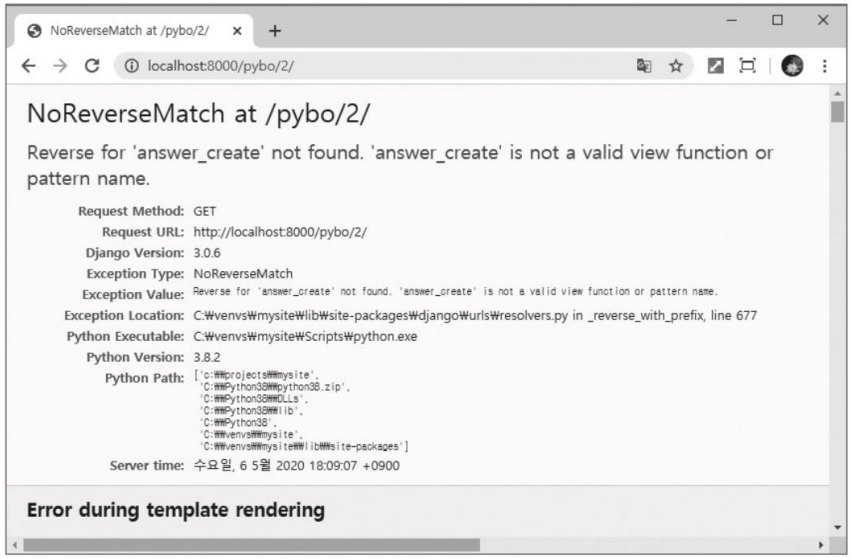
02-6 답변 등록 기능 만들기

• 완성 소스 github.com/pahkey/djangobook/tree/2-06

Do it! 실습

답변 저장하고 표시하기

02단계 질문 상세 페이지에 접속해 보기



- 오류 발생 이유는 form 엘리먼트의 action 속성에 있는 {% url 'pybo:answer_create' question.id %}에 해당하는 URL 매핑이 없기 때문이다.

03단계 답변 등록을 위한 URL 매핑 등록하기

파일 이름 C:/projects/mysite/pybo/urls.py

```
(... 생략 ...)
urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>/', views.detail, name='detail'),
    path('answer/create/<int:question_id>/', views.answer_create, name='answer_create'),
]
(... 생략 ...)
```

- <질문답변> 버튼을 눌렀을 때 작동할 form 엘리먼트의 /pybo/answer/create/2/에 대한 URL 매핑을 추가한 것이다.

02-6 답변 등록 기능 만들기

• 완성 소스 github.com/pahkey/djangobook/tree/2-06

Do it! 실습

답변 저장하고 표시하기

04단계 answer_create 함수 추가하기

파일 이름 C:/projects/mysite/pybo/views.py

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

(... 생략 ...)
def answer_create(request, question_id):
    """
    pybo 답변 등록
    """
    question = get_object_or_404(Question, pk=question_id)
    question.answer_set.create(content=request.POST.get('content'),
                               create_date=timezone.now())
```

- form 엘리먼트에 입력된 값을 받아 데이터베이스에 저장할 수 있도록 answer_create 함수를 pybo/views.py 파일에 추가하자.

05단계 답변 등록 후 상세 화면으로 이동하게 만들기

파일 이름 C:/projects/mysite/pybo/views.py

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

(... 생략 ...)
def answer_create(request, question_id):
    """
    pybo 답변 등록
    """
    question = get_object_or_404(Question, pk=question_id)
    question.answer_set.create(content=request.POST.get('content'),
                               create_date=timezone.now())
    return redirect('pybo:detail', question_id=question.id)
```

- redirect 함수는 함수에 전달된 값을 참고하여 페이지 이동을 수행한다.

02-6 답변 등록 기능 만들기

• 완성 소스 github.com/pahkey/djangobook/tree/2-06

Do it! 실습

답변 저장하고 표시하기

06단계 등록된 답변 표시하기

파일 이름 C:/projects/mysite/templates/pybo/question_detail.html

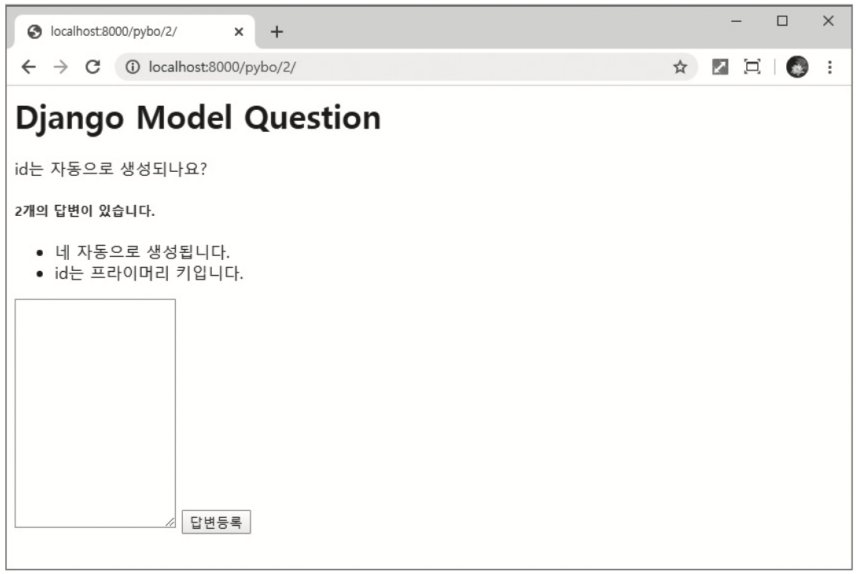
```
<h1>{{ question.subject }}</h1>

<div>
    {{ question.content }}
</div>

<h5>{{ question.answer_set.count }}개의 답변이 있습니다.</h5>
<div>
    <ul>
    {% for answer in question.answer_set.all %}
        <li>{{ answer.content }}</li>
    {% endfor %}
    </ul>
</div>

<form action="{% url 'pybo:answer_create' question.id %}" method="post">
{% csrf_token %}
<textarea name="content" id="content" rows="15"></textarea>
<input type="submit" value="답변 등록">
</form>
```

- question.answer_set.count는 답변 개수를 의미한다.



답변 등록 후 볼 수 있는 답변

Thank You