웹 페이지 작성하기2

DB&AI Lab.

한양대학교 인공지능학과

목차

1. Oracle DB 연결하기
2. 폼 데이터 받아오기
3. Oracle DB에 데이터 저장하기
4. 다양한 입력 데이터 양식
5. 폼 데이터 받아오기(심화)
6. Oracle DB에 정보 추가하기

Oracle DB 연결하기

- Oracle DB의 여러 테이블 연결하기

```
CREATE TABLE laptop (
   model NUMBER(19) PRIMARY KEY,
   speed NUMBER(19),
   ram NUMBER(19),
   hd NUMBER,
   screen NUMBER,
   price NUMBER(19)
);


CREATE TABLE printer (
   model NUMBER(19) PRIMARY KEY,
   color VARCHAR2(20),
   type VARCHAR2(20),
   price NUMBER(19)
);
```

Oracle DB 연결하기

- Oracle DB의 여러 테이블 연결하기
  - 테이블 이름 간에 띄어쓰기를 이용하여 나열

기존

```
(django_project) (venv) C:\Users\DBLAB\Desktop\django_project>python manage.py inspectdb pc>store/models.py
```

여러 테이블 연결하기

```
(django_project) C:\Users\DBLAB\Desktop\django_project>python manage.py inspectdb pc laptop printer > store/models.py
```

Oracle DB 연결하기(cont'd)

- 오류 코드
  - Import error: admin.py에서 PC table을 import하는 과정에서 생긴 오류
  - admin.py 주석 처리 후 다시 inspectdb 명령어 입력

```
  File "<frozen importlib._bootstrap>", line 1007, in _find_and_load
  File "<frozen importlib._bootstrap>", line 986, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 680, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 790, in exec_module
  File "<frozen importlib._bootstrap>", line 228, in _call_with_frames_removed
  File "C:\Users\DBLAB\Desktop\django_project\store\admin.py", line 2, in <module>
    from .models import Pc
ImportError: cannot import name 'Pc' from 'store.models' (C:\Users\DBLAB\Desktop\django_project\store\models.py)
```

admin.py

```
1  # from django.contrib import admin
2  # from .models import Pc
3  #
4  #
5  # # Register your models here.
6  # class PcAdmin(admin.ModelAdmin):
7  #     search_fields = ['model']
8  #
9  #
10 # admin.site.register(Pc, PcAdmin)
11 #
12 #
13
```

Oracle DB 연결하기(cont'd)

- 오류 코드
  - Import error: admin.py에서 PC table을 import하는 과정에서 생긴 오류
  - admin.py 주석 처리 후 다시 inspectdb 명령어 입력

admin.py

```python
# This is an auto-generated Django model module.
# You'll have to do the following manually to clean this up:
#   * Rearrange models' order
#   * Make sure each model has one field with primary_key=True
#   * Make sure each ForeignKey and OneToOneField has `on_delete` set to the desired
#   * Remove `managed = False` lines if you wish to allow Django to create, modify,
# Feel free to rename the models, but don't rename db_table values or field names.
from django.db import models


class Pc(models.Model):  3 usages
    model = models.BigIntegerField(primary_key=True)
    speed = models.BigIntegerField()
    ram = models.BigIntegerField()
    hd = models.FloatField()
    cd = models.CharField(max_length=20)
    price = models.BigIntegerField()

    class Meta:
        managed = False
        db_table = 'pc'
```

models.py

```python
class Laptop(models.Model):
    model = models.BigIntegerField(primary_key=True)
    speed = models.BigIntegerField()
    ram = models.BigIntegerField()
    hd = models.FloatField()
    screen = models.FloatField()
    price = models.BigIntegerField()

    class Meta:
        managed = False
        db_table = 'laptop'


class Printer(models.Model):
    model = models.BigIntegerField(primary_key=True)
    color = models.CharField(max_length=20)
    type = models.CharField(max_length=20)
    price = models.BigIntegerField()

    class Meta:
        managed = False
        db_table = 'printer'
```

폼 데이터 받아오기

- 폼에 입력한 데이터 받아오기
  - 폼 속성 알아보기
    - action: 서버의 어떤 주소로 전송할지 지정
    - method: 사용자가 입력한 내용을 서버로 어떤 방식을 활용해서 넘겨줄지 지정
    - csrf_token: 사이트 간 요청 위조 방지 토큰

```
</body>
</html>
<form action="{% url 'store:purchase' }" method="post">
    {% csrf_token %}
    <fieldset>
        <legend>상품 선택</legend>
        <li>
                <label for="item">상품 종류 선택</label>
```

    - action= "{% url 'store:purchase' %} → store 앱 내의 purchase 별칭을 가진 url 실행
    - method="post" → 입력한 내용을 POST 방식으로 보냄

폼 데이터 받아오기(cont'd)

- 폼에 입력한 데이터 받아오기
  - learn_html.html에 name 추가
  - form 내부의 입력 요소들의 name 속성의 값들을 이용해서 특정 데이터만 불러올 수 있음

```
</body>
</html>
<form action="{% url 'store:purchase' %}" method="post">
    {% csrf_token %}
    <fieldset>
        <legend>상품 선택</legend>
        <li>
                <label for="item">상품 종류 선택</label>
                <select id="item" name="product_type">
                        <option value="pc">PC</option>
                        <option value="laptop">Laptop</option>
                </select>
        </li>
        <li>
                <label for="item-id">상품 번호(4자리)</label>
                <input type="search" id="item-model" name="item_model">
        </li>
        <li>
                <label for="item-quantity">상품 수량</label>
                <input type="number" min="1" max="100" id="item-quantity" name="item_quantity">
        </li>
        <li>
                <label for="buy-date">구매일</label>
                <input type="date" id="buy-date" name="buy_date">
        </li>
```

폼 데이터 받아오기(cont'd)

- 폼에 입력한 데이터 받아오기
  - store/urls.py에 purchase url 추가

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

app_name = 'store'

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:pc_model>/', views.detail, name='detail'),
    path('learn_html/', views.learn_html, name='learn_html'),

    # 새로 추가: 구매 URL
    path('purchase/', views.purchase, name = 'purchase'),
]
```

폼 데이터 받아오기(cont'd)

- 폼에 입력한 데이터 받아오기
  - store/views.py에 재고 확인 및 문구 출력용 함수 구현

```
def purchase(request):
    pc_stock_list = Pc.objects.order_by('model')

    context = {
        'pc_stock_list': pc_stock_list,
        'message': None,
    }

    if request.method == 'POST':
        input_model = request.POST.get('item_model')

        if input_model:
            if Pc.objects.filter(model=input_model).exists():
                context['message'] = f"모델 번호 {input_model}: 재고가 있습니다. 구매가 완료되었습니다!"

            else: context['message'] = f"모델 번호 {input_model}: 재고가 없습니다."
        else: context['message'] = "모델 번호를 입력해주세요."

    return render(request, 'store/purchase.html', context)
```

PC 목록 호출 및 저장

결과 메시지 보관할 곳

폼이 POST 메소드로 제출됐을 경우

name = item_model인 값 가져옴

결과에 따른 메시지 설정

폼 데이터 받아오기(cont'd)

- 폼에 입력한 데이터 받아오기
  - Purchase.html 생성

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>PC 구매</title>
</head>
<body>
    <h1>PC 구매 및 재고 확인</h1>

    <form action="{% url 'store:purchase' %}" method="post">
        {% csrf_token %}

        <fieldset>
            <legend>상품 선택</legend>
            <div>
                <label for="model-input">구매할 모델 번호:</label>
                <input type="text" id="model-input" name="item_model" placeholder="예 : 1001" required>
                <input type="submit" value="확인 및 구매">
            </div>
        </fieldset>
    </form>

    <hr>

    {% if message %}
        <h2 style="color: blue;">{{ message }}</h2>
    {% endif %}

    <hr>
```

구매 입력 폼, 마찬가지로 item_model을 이용

Purchase 함수의 Message 출력

폼 데이터 받아오기(cont'd)

- 폼에 입력한 데이터 받아오기
  - Purchase.html 이어서 작성

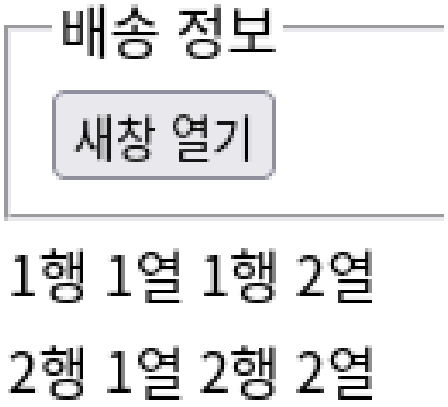
```
    <h3>현재 PC 재고 목록</h3>
    {% if pc_stock_list %}
        <ul>
        {% for pc_stock in pc_stock_list %}
            <li>
                <a href="{% url 'store:detail' pc_stock.model %}">
                    모델명 : {{ pc_stock.model }} (가격 : {{ pc_stock.price }})
                </a>
            </li>
        {% endfor %}
        </ul>
    {% else %}
        <p> 재고가 없습니다. </p>
    {% endif %}

    <a href="{% url 'store:learn_html' %}">
        <button type="button">이전으로</button>
    </a>

</body>
</html>
```

<a> 태그에 url 경로 별칭을 적어 이동

다양한 입력 데이터 양식

- 표로 영역을 지정하기
  - <table></table>: 표의 시작과 끝을 나타냄
  - <tr></tr>: 행을 나타냄
  - <td></td>: 열을 나타냄

```
<table>
    <tr>
        <td>1행  1열 </td>
        <td>1행  2열 </td>
    </tr>
    <tr>
        <td>2행  1열 </td>
        <td>2행  2열 </td>
    </tr>
</table>
```



폼 데이터 받아오기(심화)

- 여러 테이블의 데이터 참고
  - store./views.py 수정

```
from .models import Pc, Laptop, Printer
```

나머지 모델도 import

```
def purchase(request):
    pc_stock_list = Pc.objects.order_by('model')
    laptop_stock_list = Laptop.objects.order_by('model')
    printer_stock_list = Printer.objects.order_by('model')

    context = {
        'pc_stock_list': pc_stock_list,
        'laptop_stock_list': laptop_stock_list,
        'printer_stock_list': printer_stock_list,
        'message': '구매하시려는 모델을 선택해주세요', # 결과 메시지 초기화
    }
```

출력을 위해
다른 모델 리스트도 생성

```
    if request.method == 'POST':
        product_type = request.POST.get('product_type')
        input_model = request.POST.get('item_model')

        if not product_type or not input_model:
            context['message'] = '상품 종류와 모델 번호를 모두 입력해주세요.'
            return render(request, 'store/purchase.html', context)
```

제품 종류 수집

폼 데이터 받아오기(심화)(cont'd)

- 여러 테이블의 데이터 참고
  - store./views.py 수정

```
    if not product_type or not input_model:
        context['message'] = '상품 종류와 모델 번호를 모두 입력해주세요.'
        return render(request, 'store/purchase.html', context)

    if product_type == 'pc':
        ModelClass = Pc
        item_type = 'PC'
    elif product_type == 'laptop':
        ModelClass = Laptop
        item_type = '노트북'
    elif product_type == 'printer':
        ModelClass = Printer
        item_type = '프린터'
    else:
        context['message'] = '잘못된 상품 종류입니다.'
        return render(request, 'store/purchase.html', context)

    item_to_purchase = ModelClass.objects.filter(model=input_model).first()

    if item_to_purchase:context['message'] = f"[{item_type}] 모델 번호 {input_model}의 구매가 완료되었습니다."

    else:
        context['message'] = f"[{item_type}] 모델 번호 {input_model}의 재고가 없습니다."

return render(request, 'store/purchase.html', context)
```

종류에 맞춰 정보 입력

폼 데이터 받아오기(심화)(cont'd)

- 여러 테이블의 데이터 참고
  - Templates/store/learn_html.html 수정

```
<label for="item">상품 종류 선택</label>
<select id="item" name="product_type">
        <option value="pc">PC</option>
        <option value="laptop">Laptop</option>
        <option value="printer">Printer</option>
</select>
```

Printer 선택지 추가

폼 데이터 받아오기(심화)(cont'd)

- 여러 테이블의 데이터 참고
  - Templates/store/learn_html.html 수정

```
{% else %}
    <p> 재고가 없습니다. </p>
{% endif %}

<hr>

<h3>현재 Laptop 재고 목록</h3>
{% if laptop_stock_list %}
    <ul>
    {% for laptop_stock in laptop_stock_list %}
        <li>
            모델명: {{ laptop_stock.model }} (가격: {{ laptop_stock.price }})
        </li>
    {% endfor %}
    </ul>
{% else %}
    <p>Laptop 재고가 없습니다.</p>
{% endif %}

<hr>

<h3>현재 Printer 재고 목록</h3>
{% if printer_stock_list %}
    <ul>
    {% for printer_stock in printer_stock_list %}
        <li>
            모델명: {{ printer_stock.model }} (가격: {{ printer_stock.price }})
        </li>
    {% endfor %}
    </ul>
{% else %}
    <p>Printer 재고가 없습니다.</p>
{% endif %}

<hr>
<a href="{% url 'store:learn_html' %}">
    <button type="button">이전으로</button>
</a>
```

Pc와 마찬가지로
laptop과 printer도 추가

Oracle DB에 정보 추가하기

- 데이터 추가(복습)
  - Shell을 실행하여 두 방법 중 한 방법으로 정보를 추가

```
(Django_project) [oracle@localhost django_project]$ python manage.py shell
Python 3.6.8 (default, Jul  1 2025, 16:07:56)
[GCC 8.5.0 20210514 (Red Hat 8.5.0-26.0.1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from store.models import Laptop, Printer
>>> data = Laptop(model=1002, speed=1, ram=2, hd=0.3, screen=15.0, price=100)
>>> data.save()
>>> Printer.objects.create(model=1003, color='red', type='color', price=120)
<Printer: Printer object (1003)>
>>> Laptop.objects.all()
<QuerySet [<Laptop: Laptop object (1002)>]>
>>> Printer.objects.all()
<QuerySet [<Printer: Printer object (1003)>]>
```

Q&A

THANK YOU