**장고(Django)로 웹 사이트와 파이썬을 연결하여 웹 페이지에서 입출력하기**

사용한 IDE: PyCharm Professional (학교 이메일 있으면 무료로 사용 가능. 그냥 사면 1년에 $89)

[Download PyCharm: Python IDE for Professional Developers by JetBrains](https://www.jetbrains.com/pycharm/download/?source=google&medium=cpc&campaign=14124132441&term=pycharm&gclid=Cj0KCQjwyYKUBhDJARIsAMj9lkEg9qHvC7c8O1G_uFQChKGGCHtcrvZU0bRmj9MCBLLXTwWWx28O3CcaApKSEALw_wcB#section=windows)

사용한 운영체제: 윈도우10 21H2 64비트 19044.1706

사용한 파이썬 버전: 64비트 3.10.4

장고 버전: 4.0.4

파이참을 처음 실행하시면 이런 화면이 나옵니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

새 프로젝트를 클릭하고 왼쪽에서 Django 탭을 선택한다. 이게 별것 아닌 것처럼 보이겠지만 파이참의 큰 장점이다. 다른 개발 도구는 장고를 쓰려면 장고 관련된 걸 하나씩 수동으로 설치해야 하지만 파이참은 이걸 선택해주기만 하면 파이썬 가상환경 설정과 pip를 통한 장고 패키지 설치, 장고와 관련된 기본 설정 등 어지간한 장고 관련 밑작업은 다 해줘서 편리하다.

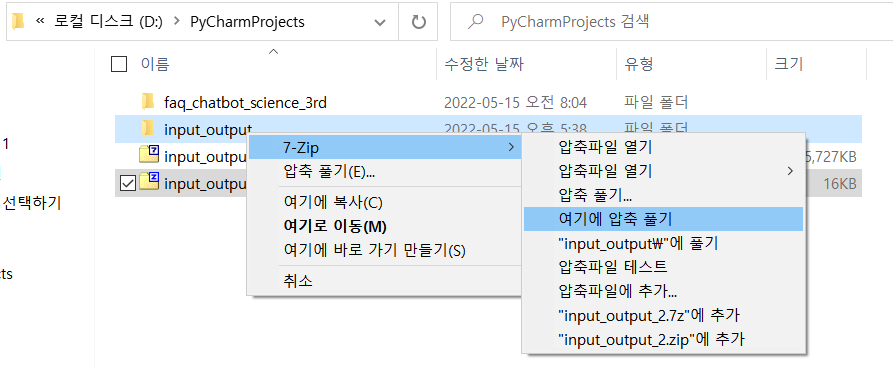
위치에서 마지막 경로의 기본값인 djangoProject를 지우고 input\_output으로 수정한 다음 생성을 누른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

앞으로 이 프로젝트의 이름은 input\_output이다. 자기 혼자 뭘 막 설치하고 설정하느라 시간이 좀 걸린다.

다 되면 프로젝트가 만들어진 폴더에 가서 input\_output.zip 을 내려 받아서 압축 파일을 풀고 덮어쓰기 한다.



코드를 새로 쓰느니 덮어쓰기 한 파일을 수정하여 파이썬으로 만들었놨던 코드를 장고에 붙여서웹 페이지에서 입력-출력할 수 있게 하면 원하는 기능을 쉽게 구현할 수 있을 것이다.

웹서버 구동은 파이참 아래쪽에 있는 탭에서 터미널을 눌러서 열고

텍스트, 스크린샷, 모니터, 화면이(가) 표시된 사진

자동 생성된 설명

python manage.py runserver 을 실행하면 된다.

이제 <http://127.0.0.1:8000> 으로 접속하면 웹페이지가 나온다.

**주의사항 및 기본 설정**

프로젝트 이름은 input\_output (그냥 이렇게 정했음) 인데 바꾸면 오류 난다. /input\_output/settings.py와 가상 환경에서 모두 input\_output을 프로젝트 이름으로 쓰니까 바꾸면 안 된다.

장고에서 돌리는 앱 이름은 inputoutput (그냥 이렇게 정했음) 인데 얘도 바꾸면 오류 난다. 기본적인 입력-출력 시 참조하는 앱과 그 앱과 관련된 폴더 이름을 모두 inputoutput으로 설정해놔서 바꾸면 안 된다.

**장고와 HTML 파일의 차이점**

장고는 중, 고등학교 때 다뤘던 HTML과는 성격이 다르다. HTML은 정적인 웹 페이지를 구현하기에는 좋다. 전달할 정보가 있으면 HTML에 실어서 서버에 올리고, 사용자는 서버에 올라간 파일을 받아서 보기만 하면 되는 용도로 적합하다. 장고는 파이썬으로 동적인 웹 페이지를 구현하는 데에 적합하다. 사용자가 웹 사이트에 접속해서 무언가 입력하고, 입력한 값에 대하여 원하는 출력 값을 얻도록 하는 데에 좋다.

그래서 당연히(?) 구조가 복잡하다. HTML로 만든 정보 전달 위주의 페이지는 한 페이지를 구현하려면 파일 개수가 하나여도 되고, 복잡하게 만들어도 몇 개 정도면 충분하다. 장고는 한 페이지를 구현하려면 기본적으로 설정할 파일 개수가 최소 5개는 된다.

**중요한 파일 다섯 가지의 간략한 소개**

파일 5개는 settings.py, urls.py, views.py, models.py, 화면에 뿌릴 HTML 파일이고, 이들 파일은 한 세트로 움직여서 다섯 중 하나만 오류가 있으면 전체가 구동이 안 된다. 다섯 개 파일 간 변수명은 반드시 통일시켜야지 파일 각각 막 쓰면 오류 나기 십상이다.

settings.py: 기본 설정을 담은 파일

urls.py. 어느 주소로 어떤 기능을 넘겨줄지 설정하는 파일

vews.py: urls.py에서 요청을 주면 화면에 그린 HTML 파일에서 변수를 받아서 파이썬에서 어느 정도 처리하고 다시 HTML 파일로 넘겨주는 중간 다리 역할을 하는 파일

models.py: 원래는 데이터베이스 연동 때 쓰지만 이번엔 내가 원하는 기능을 구현한 함수를 담은 파일

화면에 뿌릴 HTML 파일: 화면에 구현할 인터페이스와 views.py에서 받은 변수를 어디다 넣을지 정하고 views.py로 넘겨줄 변수를 담은 파일.

**앞으로 살펴볼 파일**

**/input\_output/settings.py**

이 파일은 이미 다 수정했으므로 손댈 게 없다. 그래도 손대 부분을 살펴보면…

INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'inputoutput'  
]

라고 되어 있다. 마지막에 inputoutput 앱을 쓰겠다고 알려주는 부분이 추가되어 있다.

ROOT\_URLCONF = 'input\_output.urls'

이 부분도 최상위 디렉토리를 설정하여 수정되어 있다.

WSGI\_APPLICATION = 'input\_output.wsgi.application'

이 부분도 마찬가지다.

LANGUAGE\_CODE = 'ko-kr'  
  
TIME\_ZONE = 'Asia/Seoul'

이 부분은 우리나라 환경에 맞게 수정한 부분이다.

**/input\_output/urls.py**

이 파일은 장고 전체에서 웹 주소를 담당한다. 우리 프로젝트에 맞게 inputoutput 앱을 쓰려고 수정했다. 마찬가지로 더 이상 손댈 것은 없다.

from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('inputoutput.urls')),  
]

우리는 외부 웹서버가 아니라 내 컴퓨터에서 내부적으로 웹서버를 돌릴 거라서 장고 웹 서버에 <http://127.0.0.1:8000> 으로 접속하여 테스트를 해야 한다.

이 주소 뒤에 아무 것도 붙지 않으면 path('',, /inputoutput 폴더 속의 urls.py 파일을 참조하라는 뜻이다.

**/inputoutput/urls.py**

이 파일은 inputoutput 앱에서 웹 주소를 담당한다.

from django.urls import path  
from . import views  
  
'''  
/input\_output/urls.py는 프로젝트 이름이 input\_output이라서 완전 기본값으로 준 urls.py를 먼저 참조하는데  
거기에서 path('', include('inputoutput.urls')),라고 써놨으므로 이 파일(/inputoutput/urls.py) 내용을 참조하러 온다  
이걸 보면 path를 ''로 둬서 127.0.0.1로 그냥 접속하면 views.py 에 있는 inputoutput을 찾아가도록 하고  
path를 'result/'로 둬서 127.0.0.1/result로 접속하면 views.py에 있는 result를 찾아가도록 한다  
'''  
  
urlpatterns = [  
 path('', views.inputoutput, name='inputoutput'),  
 path('result/', views.result, name='result'),  
 path('inputoutput\_samepage/', views.inputoutput\_samepage, name='inputoutput\_samepage'),  
]

http://127.0.0.1:8000 뒤에 아무 것도 붙지 않으면 path('',, 같은 폴더(inputoutput 폴더)에 있던 views.py를 열어서 그 안에 있는 inputoutput 함수에 해당하는 작업을 하도록 한다.

http://127.0.0.1:8000 뒤에 result를 붙여서 <http://127.0.0.1:8000/result> 로 접속하면 같은 폴더에 있던 views.py를 열어서 result 함수에 해당하는 작업을 하도록 한다.

http://127.0.0.1:8000 뒤에 inputoutput\_samepage를 붙여서 <http://127.0.0.1:8000/inputoutput_samepage> 로 접속하면 같은 폴더에 있던 views.py를 열어서 inputoutput\_samepage 함수에 해당하는 작업을 하도록 한다.

이번 예제에서는 127.0.0.1로 그냥 접속해서 나온 inputoutput.html 웹페이지에서 사용자가 내용 두 가지를 입력하면 무언가 하게끔 만든 함수를 통과한 결과와 함께 result.html에서 출력하는 방법을 다룬다. 같은 기능을 한 웹 페이지 안에서 입력과 출력을 모두 할 수 있도록 만든 것이 inputoutput\_samepage.html 이다. 입력 받는 화면과 출력할 화면이 다르면

path('', views.inputoutput, name='inputoutput'),  
 path('result/', views.result, name='result'),  
만 남겨서 쓰면 되고, 입력과 출력을 한 화면에서 하려면

path('inputoutput\_samepage/', views.inputoutput\_samepage, name='inputoutput\_samepage'),  
만 남겨서 쓰면 된다.

**/inputoutput/views.py**

views.py는 /templates/inputoutput 폴더 속에 있는 html 파일과 /inputoutput/models.py에 파이썬으로 만들어 놓은 기능 구현 함수를 연결해준다. 이 파일은 구현하려는 기능을 추가할 수 있는 부분이므로 여기를 수정해야 원하는 웹 페이지를 만들 수 있다.

기본적인 구조는 같은 폴더 속의 models.py 에서 내가 만들어 놓은 myfunction1 함수를 불러오고, 파이썬으로 이것저것 조작한 다음 웹페이지를 render 해서 HTML로 바꿔주게끔 되어 있다.

from django.shortcuts import render  
from .models import myfunction1 # model.py에 있는 myfunction함수 가져오기  
from django import forms  
from django.http import HttpResponseRedirect  
  
# Create your views here.  
  
'''  
/inputoutput/urls.py에서 result로 가라고 하면 /templates/inputoutput/result.html을 돌려준다  
inputoutput.html에서 content1과 content2를 입력 받으면  
models.py에서 실제 기능을 구현할 함수 myfunction1을 불러와서 content1과 content2를 집어 넣어서  
myfunction1\_result 함수에서 나온 값을 받아 myfunction1\_result에 넣는다.  
data 속에 딕셔너리 형태로 content1, content2, myfunction1\_result를 넣고  
result.html로 content1, content2, myfunction1\_result를 보내준다.  
'''  
def result(request):  
 # content1과 content2를 입력받아서 result.html로 POST 방법을 써서 보냄  
 if request.method == 'POST':  
 content1 = request.POST['content1']  
 content2 = request.POST['content2']  
 myfunction1\_result = myfunction1(content1, content2)  
 data = { 'content1': content1, 'content2': content2, 'myfunction1\_result':myfunction1\_result }  
 return render(request, 'inputoutput/result.html', data)  
  
# /inputoutput/urls.py에서 inputoutput으로 가라고 하면 /templates/inputoutput/inputoutput.html을 돌려준다  
def inputoutput(request):  
 return render(request, 'inputoutput/inputoutput.html')  
  
  
def inputoutput\_samepage(request):  
 if request.method == 'POST':  
 content1 = request.POST['content1']  
 content2 = request.POST['content2']  
 myfunction1\_result = myfunction1(content1, content2)  
 data = { 'content1': content1, 'content2': content2, 'myfunction1\_result':myfunction1\_result }  
 else:  
 myfunction1\_result = "입력 없음"  
 data = {'myfunction1\_result': myfunction1\_result}  
 return render(request, 'inputoutput/inputoutput\_samepage.html', data)

가장 간단한 웹 페이지부터 작동 원리를 살펴보자. urls.py에서 inputout으로 가라는 요청(request)이 들어오면 /inputoutput/inputouput.html 을 돌려주어 화면에 뿌려주는 부분이 이 부분이다.

#/inputoutput/urls.py에서 inputoutput으로 가라고 하면 /templates/inputoutput/inputoutput.html을 돌려준다  
def inputoutput(request):  
 return render(request, 'inputoutput/inputoutput.html')

<http://127.0.0.1:8000> 으로 접속하면 아래처럼 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그럼 이 페이지는 어디서 왔냐면 /inputoutput/inputouput.html 에서 왔다.

이 페이지를 살펴봐야겠다.

**/templates/inputoutput/inputoutput.html**

<!DOCTYPE html>  
<html lang="ko">  
  
<head>  
 <meta charset="UTF-8">  
 <title>inputoutput test html</title>  
 <!-- 저쪽 아래에 있는 form 속의 <button type="button" onclick="send();">의 send() 함수에 관한 내용이다.  
 테스트를 위하여 send 됐다는 알림창을 띄우고(이건 함수 작동 확인용으로 넣었을 뿐이라 지워도 됨)  
 form id="contents" 이므로 contents 이름의 폼 속에 있는 content1과 content2를 보낸 다음 폼 내용을 초기화한다. -->  
 <script> function send() {  
 alert("send");  
 ***document***.getElementById("contents").submit();  
 ***document***.getElementById("contents").reset();  
 }  
 </script>  
</head>  
  
<body>  
 <p>입출력 테스트</p>  
 <form id="contents" name="contents" method="post" action="{% url 'result' %}">  
 <!-- 아래에 있는 content1, content2를 입력받아서 javascript 함수 "send();"를 실행시켜 POST 방법으로 result.html에 전송  
 숫자를 입력 받으려면 type="number"> 로 설정한다.  
 -->  
 {% csrf\_token %}  
 content1: <input name="content1" type="text">  
 content2: <input name="content2" type="text">  
 <button type="button" onclick="send();">전송</button>  
 </form>  
</body>  
</html>

파일의 확장자는 HTML인데 이 파일 속에는 HTML도 있고 자바스크립트도 있고 장고 문법까지 다 섞여 있다.

<body>부터 보면 입출력 테스트라는 내용으로 한 문단(paragraph)이 <p> 태그로 싸여 있다.

그 다음 줄에는 form이 있다. Form은 무언가 입력 받을 때 쓰는 기능이다. 예시에서는 이 폼의 id와 이름을 contents로, 이 폼에 작성한 내용을 어디론가 전송할 때는 POST라는 방법으로, 그리고 보낼 내용은 result라는 URL로 보내줄 것이다.

입력하는 칸은 content1, content2로 총 두 개다. 하나만 입력 받아도 되면 하나 지우면 된다. . 이 부분을 적절히 잘 바꾸면 파이썬에서 입력 받아야 할 값들을 얼마든지 받을 수 있고, 값이 많으면 한 줄씩 비슷하게 추가하면 된다.

버튼을 누르면 send();라는 자바스크립트 함수가 실행된다. 이 함수는 저 위에 있는 <head> 속에 <script> 태그에 감싸여 들어 있다. 버튼을 누르면 알림 메시지가 하나 나오고, 그 다음 폼 중에 ID가 contents인 걸 찾아서 전송하는 기능이 들어 있다.

이 파일이 하는 기능은 여기까지이고, 파이썬 파일이 아니기 때문에 더 많은 기능은 하기 어렵다. 그럼 자연스레 보낼 내용을 전달할 result 함수를 살펴보기 위해 다시 views.py를 돌아봐야 한다.

**/inputoutput/views.py**

from django.shortcuts import render  
from .models import myfunction1 # model.py에 있는 myfunction함수 가져오기  
from django import forms  
from django.http import HttpResponseRedirect  
  
# Create your views here.  
  
'''  
/inputoutput/urls.py에서 result로 가라고 하면 /templates/inputoutput/result.html을 돌려준다  
inputoutput.html에서 content1과 content2를 입력 받으면  
models.py에서 실제 기능을 구현할 함수 myfunction1을 불러와서 content1과 content2를 집어 넣어서  
myfunction1\_result 함수에서 나온 값을 받아 myfunction1\_result에 넣는다.  
data 속에 딕셔너리 형태로 content1, content2, myfunction1\_result를 넣고  
result.html로 content1, content2, myfunction1\_result를 보내준다.  
'''  
def result(request):  
 # content1과 content2를 입력받아서 result.html로 POST 방법을 써서 보냄  
 if request.method == 'POST':  
 content1 = request.POST['content1']  
 content2 = request.POST['content2']  
 myfunction1\_result = myfunction1(content1, content2)  
 data = { 'content1': content1, 'content2': content2, 'myfunction1\_result':myfunction1\_result }  
 return render(request, 'inputoutput/result.html', data)  
  
# /inputoutput/urls.py에서 inputoutput으로 가라고 하면 /templates/inputoutput/inputoutput.html을 돌려준다  
def inputoutput(request):  
 return render(request, 'inputoutput/inputoutput.html')  
  
  
def inputoutput\_samepage(request):  
 if request.method == 'POST':  
 content1 = request.POST['content1']  
 content2 = request.POST['content2']  
 myfunction1\_result = myfunction1(content1, content2)  
 data = { 'content1': content1, 'content2': content2, 'myfunction1\_result':myfunction1\_result }  
 else:  
 myfunction1\_result = "입력 없음"  
 data = {'myfunction1\_result': myfunction1\_result}  
 return render(request, 'inputoutput/inputoutput\_samepage.html', data)

def result(request): 부분을 보면 뭔가 요청이 왔는데 result 함수를 두드릴 때 어떻게 해야 하는지가 나와 있다. 앞서 inputoutput.html 에서 content1과 content2를 POST 방법으로 전송하여 result 로 가라고 했다. 그래서 이 부분에서 만약 POST로 요청이 들어오면 content1이라는 파이썬 변수에는 POST로 받은 content1의 내용을 집어넣고, content2라는 파이썬 변수에는 POST로 받은 content2의 내용을 집어넣으라고 했다. 이 부분이 파이썬과 HTML 파일이 연결되는 부분이다.

미리 만들어준 파이썬 함수인 myfunction1을 쓰려고 models.py에서 불러왔는데, 여기에서 content1과 content2를 변수로 받아서 결과를 myfunction1\_result 파이썬 변수에 넣었다. myfunction1 부분을 응용하면 원하는 기능을 얼마든지 넣고 뺄 수 있다. 이렇게 얻은 content1, content2, myfunction1\_result 값을 data에 딕셔너리 형태로 만들어서 넣고 /iputoutput/result.html로 보내준다.

그럼 자연스레 models.py 속에 있는 myfunction1 함수가 어떻게 생겼는지 봐야 한다.

**/inputoutput/models.py**

이 파일은 일반적으로 데이터베이스 연동 때 사용하는데 우리 예시에서는 파이썬으로 구현했던 함수를 넣고 운영하는 데에 사용하는 데에 쓴다.

from django.db import models  
  
# Create your models here.  
'''  
파이썬으로 구현한 함수가 들어갈 곳이다.  
여기에 선생님들께서 원하는 기능을 구현하셨던 함수를 넣고 함수의 입력 변수로 content1과 content2를 입력 받는다.  
myfunction1이라는 함수가 돌아간 후 돌려줄 값을 return 뒤에 써준다. 돌려준 값은 views.py에서 받아서 쓸 수 있다.  
여기에 함수를 여러 개 넣어서 추가할 수도 있다.  
'''  
  
# 두 입력값을 합치는 간단한 함수. 숫자끼리면 더하고, 숫자 아닌 것 같아서 오류가 나면 그냥 문자열을 붙여서 돌려주는 함수  
def myfunction1(content1, content2):  
 try:  
 sum = int(content1) + int(content2)  
 return "content1과 content2를 합치면 {0}이다".format(sum)  
 except:  
 sum = content1 + content2  
 return "content1과 content2 문자열을 붙이면 {0}이다".format(sum)

여기에 쓴 예시는 단순하게 변수 두 개를 받아서 합치는 함수에 불과하다. 이 부분이 파이썬으로만든 핵심 코드가 들어갈 자리이다. 이 부분은 그대로 쓸 이유가 없으니 주피터 노트북이나 코랩에서 짠 코드를 여기에 넣고 변수 이름만 적당히 손보면 된다.

**/templates/inputoutput/result.html**

마지막으로 result.html이 어떻게 생겼는지 살펴본다.

<!DOCTYPE html>  
<html lang="ko">  
  
<head>  
 <meta charset="UTF-8">  
 <title>result</title>  
</head>  
  
<body>  
 {{ data }} <br/>  
 content1 = {{ content1 }} <br/>  
 content2 = {{ content2 }} <br/>  
 {{ myfunction1\_result }} <br/>  
 <input type="button" value="이전 화면으로" onClick="history.go(-1)">  
</body>  
</html>

views.py에서 content1과 content2, myfunction1\_result를 result.html로 넘겨준다고 했다. 이걸 넘겨 받아서 HTML에 뿌려줘야(render) 하는데, HTML 파일 안에서 장고의 변수를 넘겨 받아 쓸 때에는 {{ }} 로 변수 이름을 감싼다.

<input type=”button” value=”이전 화면으로” onClick=”history.go(-1)”> 은 이전 웹페이지로 돌아가는 버튼이다. 뒤로 버튼 누르기 귀찮아서 만들었을 뿐 필수적인 기능은 아니라 없어도 되는 줄이다.

이 기능이 구현되면 아래처럼 결과가 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**한 페이지 안에서 입력과 출력 모두 하기**

앞서 살펴본 예시는 입력하는 화면과 출력하는 화면이 다를 때에 쓸 수 있는 예시이다. 이번에 살펴볼 예시는 입력과 출력을 한 화면에서 할 때 쓸 수 있는 예시이다. 기본적인 원리는 앞서 살펴본 것과 같고, 출력하는 부분에서 파일을 같은 파일로 불러오게 하는 부분과 입출력을 한 파일에 모두 넣는다는 점이 다르다.

**앞으로 살펴볼 파일**

**/inputoutput/urls.py**

이 파일은 inputoutput 앱에서 웹 주소를 담당한다.

from django.urls import path  
from . import views  
  
'''  
/input\_output/urls.py는 프로젝트 이름이 input\_output이라서 완전 기본값으로 준 urls.py를 먼저 참조하는데  
거기에서 path('', include('inputoutput.urls')),라고 써놨으므로 이 파일(/inputoutput/urls.py) 내용을 참조하러 온다  
이걸 보면 path를 ''로 둬서 127.0.0.1로 그냥 접속하면 views.py 에 있는 inputoutput을 찾아가도록 하고  
path를 'result/'로 둬서 127.0.0.1/result로 접속하면 views.py에 있는 result를 찾아가도록 한다  
'''  
  
urlpatterns = [  
 path('', views.inputoutput, name='inputoutput'),  
 path('result/', views.result, name='result'),  
 path('inputoutput\_samepage/', views.inputoutput\_samepage, name='inputoutput\_samepage'),  
]

http://127.0.0.1:8000 뒤에 inputoutput\_samepage를 붙여서 <http://127.0.0.1:8000/inputoutput_samepage> 로 접속하면 같은 폴더에 있던 views.py를 열어서 inputoutput\_samepage 함수에 해당하는 작업을 하도록 한다.

이번 예제에서는 <http://127.0.0.1:8000/inputoutput_samepage> 로 접속해서 나온 inputoutput\_samepage.html 웹페이지에서 앞서 살펴본 것과 같은 기능을 하게끔 만든다.

이 기능이 필요하면 path('inputoutput\_samepage/', views.inputoutput\_samepage,

만 남겨서 쓰면 된다.

**/inputoutput/views.py**

마지막에 있는 def inputoutput\_samepage(request): 이 입력과 출력을 한 함수로 합친 부분이다.

from django.shortcuts import render  
from .models import myfunction1 # model.py에 있는 myfunction함수 가져오기  
from django import forms  
from django.http import HttpResponseRedirect  
  
# Create your views here.  
  
'''  
/inputoutput/urls.py에서 result로 가라고 하면 /templates/inputoutput/result.html을 돌려준다  
inputoutput.html에서 content1과 content2를 입력 받으면  
models.py에서 실제 기능을 구현할 함수 myfunction1을 불러와서 content1과 content2를 집어 넣어서  
myfunction1\_result 함수에서 나온 값을 받아 myfunction1\_result에 넣는다.  
data 속에 딕셔너리 형태로 content1, content2, myfunction1\_result를 넣고  
result.html로 content1, content2, myfunction1\_result를 보내준다.  
'''  
def result(request):  
 # content1과 content2를 입력받아서 result.html로 POST 방법을 써서 보냄  
 if request.method == 'POST':  
 content1 = request.POST['content1']  
 content2 = request.POST['content2']  
 myfunction1\_result = myfunction1(content1, content2)  
 data = { 'content1': content1, 'content2': content2, 'myfunction1\_result':myfunction1\_result }  
 return render(request, 'inputoutput/result.html', data)  
  
# /inputoutput/urls.py에서 inputoutput으로 가라고 하면 /templates/inputoutput/inputoutput.html을 돌려준다  
def inputoutput(request):  
 return render(request, 'inputoutput/inputoutput.html')  
  
  
def inputoutput\_samepage(request):  
 if request.method == 'POST':  
 content1 = request.POST['content1']  
 content2 = request.POST['content2']  
 myfunction1\_result = myfunction1(content1, content2)  
 data = { 'content1': content1, 'content2': content2, 'myfunction1\_result':myfunction1\_result }  
 else:  
 myfunction1\_result = "입력 없음"  
 data = {'myfunction1\_result': myfunction1\_result}  
 return render(request, 'inputoutput/inputoutput\_samepage.html', data)

작동 원리는 POST 방법으로 요청이 들어오면 값을 입력 받아서 다시 inputoutput\_samepage.html로 넘기고, 그냥 요청이 들어오면 입력이 없다고 한 다음 다시 inputoutput\_samepage.html로 넘기는 식이다.

**/inputoutput/inputoutput\_samepage.html**

마찬가지로 다 똑같은데 입력과 출력을 한 페이지에 안에서 할 수 있게끔 조합만 한 파일이다. 앞서 소개한 inputoutput.html과 result.html을 합친 형태이다.

주의할 게 하나 있는데 form의 action에서 어느 URL에 입력 값을 넘길지 알려주는데, 이 때 URL을 이미 입력 받고 있는 파일인 inputoutput\_samepage.html로 설정하는 부분만 생각해주면 한 페이지에서도 얼마든지 입력과 출력을 같이 할 수 있다.

<!DOCTYPE html>  
<html lang="ko">  
  
<head>  
 <meta charset="UTF-8">  
 <title>한 화면에서 입출력 테스트</title>  
 <script>  
 function send() {  
 alert("send");  
 ***document***.getElementById("contents").submit();  
 ***document***.getElementById("contents").reset();  
 }  
 </script>  
</head>  
  
<body>  
 <p>입출력 테스트</p>  
 <form id="contents" name="contents" method="post" action="{% url 'inputoutput\_samepage' %}">  
 <!-- 아래에 있는 content1, content2를 입력받아서 javascript 함수 "send();"를 실행시켜 POST 방법으로 inputoutput\_samepage.html에 전송  
 숫자를 입력 받으려면 type="number"> 로 설정한다.  
 -->  
 {% csrf\_token %}  
 content1: <input name="content1" type="text">  
 content2: <input name="content2" type="text">  
 <button type="button" onclick="send();">전송</button>  
 </form>  
  
 {{ data }} <br/>  
 content1 = {{ content1 }} <br/>  
 content2 = {{ content2 }} <br/>  
 {{ myfunction1\_result }} <br/>  
</body>  
</html>

나머지 models.py는 손대지 않았으므로 생략한다.

사용자는 아래처럼 이 기능을 사용하게 된다. 일단 웹서버를 돌리기 위해 터미널을 열고 python manage.py runserver 실행 후 <http://127.0.0.1/inputoutput_samepage> 에 접속하여 입력값에 1234와 12345를 누르고 전송을 누르면 아래처럼 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

기본적인 입출력은 끝났다. 나머지는 사용자가 쓰기 편하게 꾸미는 일만 남았다. 사용자가 쓰기 좋게 겉모습을 꾸미려면 CSS를 검색해서 기능을 덧붙이면 모양과 색을 예쁘게 꾸밀 수 있다. 자료 입출력은 처음에는 파이썬의 openpyxl 모듈과 엑셀을 이용하여 엑셀로 입력과 출력을 하게 하여 충분히 테스트를 하고, 데이터가 충분히 많아져서 동시에 입출력을 해야 할 때가 됐을 때 실제 서비스를 위해 파이썬의 MariaDB (MySQL의 완전 무료 버전) 연동 모듈과 파이썬을 연계해서 쓰면 가장 빠르고 편리하게 기능을 구현할 수 있을 것이다.