

Django

1. pyDev 설치
2. 북마크앱
3. 설문조사
4. 게시판

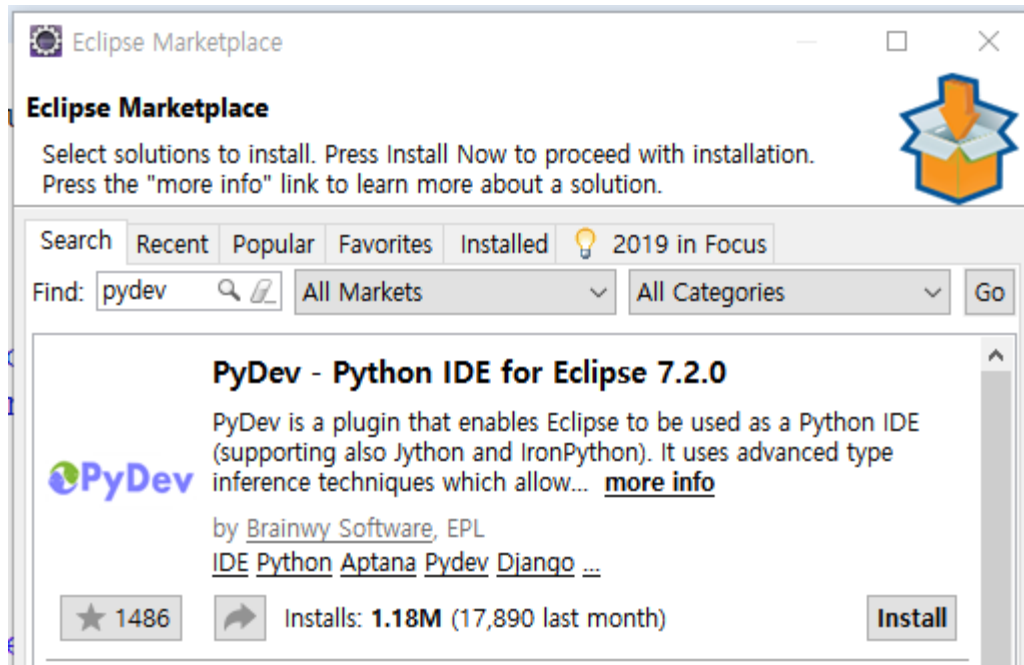
1. pyDev 설치

□ Python 개발 툴

- ▣ ipython notebook-웹브라우저에서 실행
- ▣ pyCham-python전용 개발툴(IntelliJ기반)
- ▣ PyDev- 이클립스 플러그인

□ PyDev 설치

- ▣ eclipse-> Help->Maketplace 메뉴에서 pydev 검색하여 설치



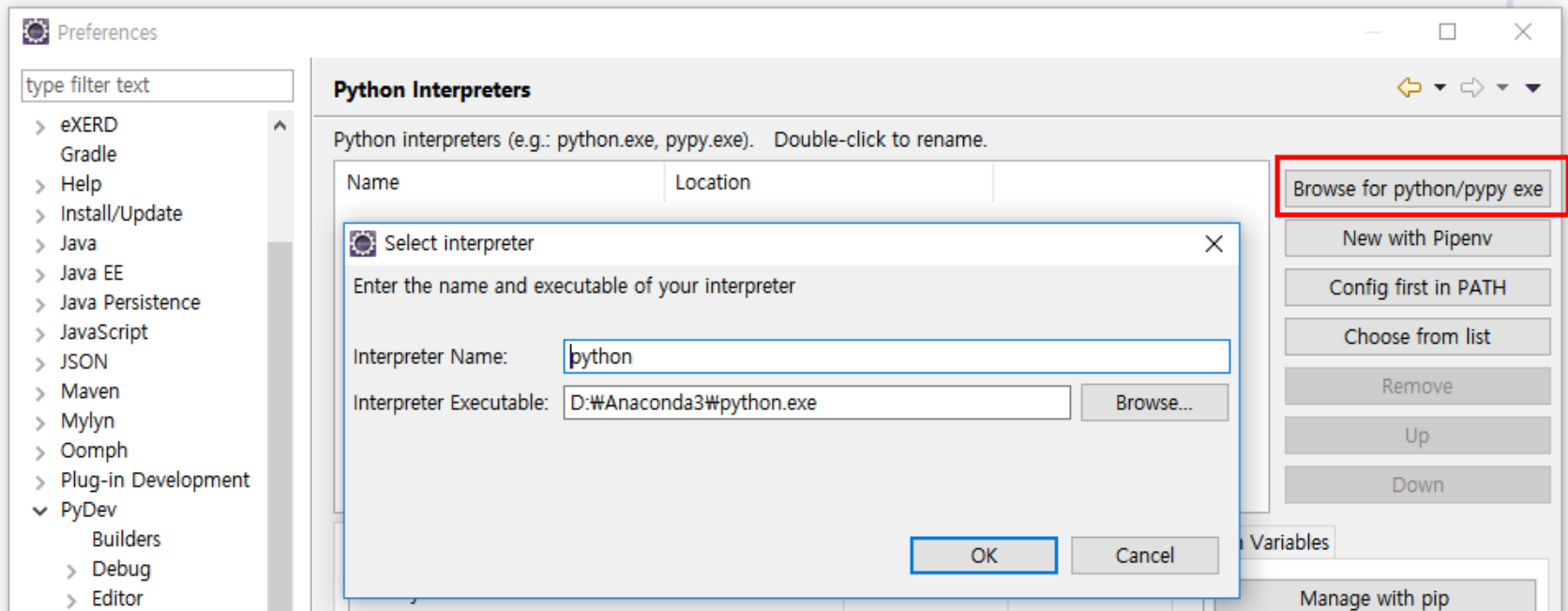
1. pyDev 설치

□ 환경설정

- eclipse 재 실행 후 open perspective 선택->pyDev선택

□ 파이썬 인터프리터 설정

- window->Preference->PyDev-interpreters->python interpreter
->Browse for python/pypy.exe 선택



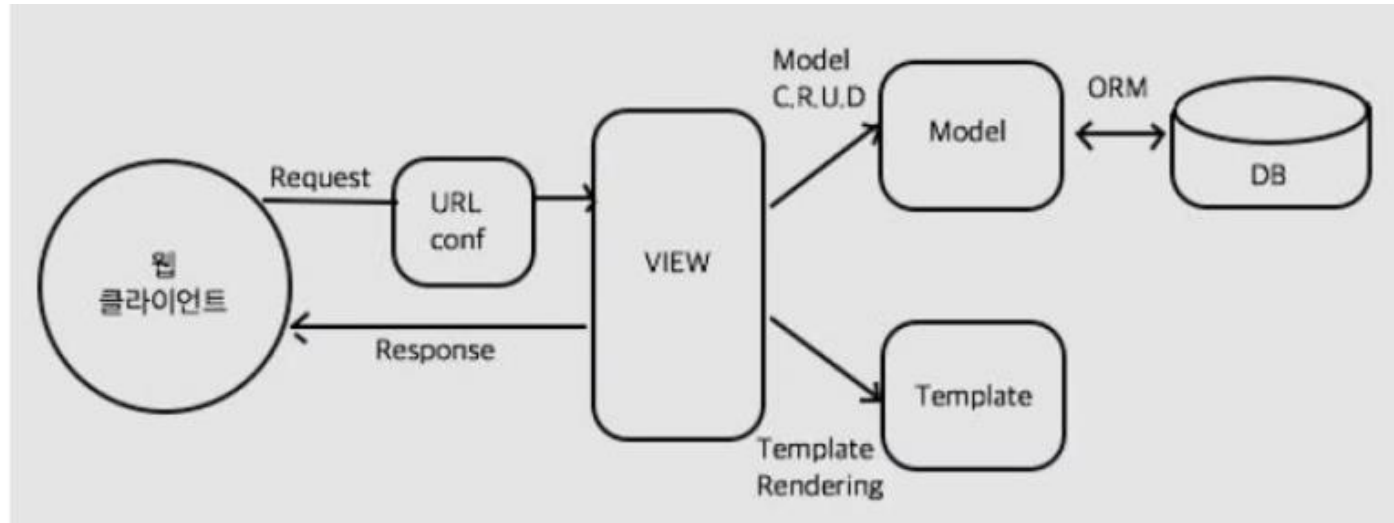
□ 프로젝트 만들기

- ▣ django 설치 : `pip install django`
- ▣ New->Other-PyDev-PyDev Django Project
- ▣ 프로젝트 이름 : `pyweb01`
- ▣ `pyweb01` 디렉토리가 2개 만들어진다
- ▣ `d:\python\work\pyweb01`
- ▣ `d:\python\work\pyweb01\pyweb01-python` 웹프로젝트의 설정 디렉토리

□ Django

- 파이썬 기반의 무료 오픈소스 웹 애플리케이션 프레임워크(Open Source Web Application framework)
- 장점
 - Python 기반 프레임워크로 배우기 쉬움
 - 빠른 개발속도, 개발 비용 절감
 - 코드 완성도를 높게 유지할수 있으며 확장성이 좋음
 - 사용자 인증, 사용자관리 등 기능이 기본적으로 구현되어 있음
- 성공적인 도입 사례 – Instagram
- Django 패키지 설치
 - `pip install django`

■ MTV 패턴



■ MVC pattern과 MTV pattern의 비교

| MVC | MTV | 설명 |
|------------|----------|------------------------------|
| Model | Model | 데이터베이스와 관련된 처리를 담당하는 코드 |
| View | Template | 사용자가 보게되는 화면을 정의하는 코드 |
| Controller | View | 데이터를 처리한 후 결과를 템플릿에게 전달하는 코드 |

2. 북마크 앱

□ 애플리케이션 설계하기

▣ 화면 UI 설계

- 화면설계는 주로 템플릿 코딩에 반영되고, *templates/*디렉토리 하위의 *.html 파일에 코딩
- 리스트 - bookmark_list.html
- 상세페이지 - bookmark_detail.html

▣ 테이블 설계

- 테이블설계 내용은 모델 코딩에 반영되고, *models.py* 파일에 코딩

| 필드명 | 타입 | 제약 조건 | 설명 |
|-------|----------------|--------------------|-------------|
| id | integer | PK, Auto Increment | Primary key |
| title | CharField(100) | Blank, Null | 북마크제목 |
| url | URLField | Unique | 북마크 URL |

□ 애플리케이션 설계

▣ 로직 설계

- 로직 설계는 처리 흐름을 설계하는 것
- 웹 프로그래밍에서는 URL을 받아서 최종 HTML템플릿 파일을 만드는 과정이 하나의 로직
- 리다이렉션이나 템플릿 파일에서 URL요청이 발생하는 일련의 과정을 모두 고려하여 표현하는 것

| URL | View | Template |
|-------------------------------------|----------------------|----------------------|
| /bookmark/ | BookmarkLV.as_view() | bookmark_list.html |
| /bookmark/99/ | BookmarkDV.as_view() | bookmark_detail.html |
| <i>Bookmark 앱 URL-뷰-템플릿 간의 처리흐름</i> | | |

□ 애플리케이션 설계

▣ URL 설계

- URL설계 내용은 URLconf 코딩에 반영
- urls.py파일에 코딩
- URL패턴, 뷰 이름, 템플릿 파일 이름 정의
- 뷰에서 어떤 제네릭 뷰를 사용할 것인지 정의

| URL패턴 | 뷰 이름 | 템플릿 파일 이름 |
|--|------------------------|----------------------|
| /bookmark/ | BookmarkLV(ListView) | bookmark_list.html |
| /bookmark/??/* | BookmarkLV(DetailView) | bookmark_detail.html |
| /admin/ | (Django 제공기능) | |
| **DetailView의 ??자리에는 PK인 id가 전달 되어 들어간다 | | |

▣ 작업/코딩 순서

| 작업순서 | 관련 명령/파일 | 필요한 작업 내용 |
|--------------|-----------------|---------------------|
| 뼈대만들기 | startproject | mysite 프로젝트 생성 |
| 뼈대만들기 | settings.py | 프로젝트 설정 항목 변경 |
| 뼈대만들기 | migrate | User/Group 테이블 생성 |
| 뼈대만들기 | createsuperiser | 프로젝트 관리자인 슈퍼유저를 만든다 |
| 뼈대만들기 | startapp | 북마크 앱 생성 |
| 뼈대만들기 | settings.py | 북마크 앱 등록 |
| 모델코딩하기 | models.py | 모델(테이블) 정의 |
| 모델코딩하기 | admin.py | Admin 사이트에 모델 등록 |
| 모델코딩하기 | makemigrations | 모델을 데이터베이스에 반영 |
| 모델코딩하기 | migrate | |
| URLconf 코딩하기 | urls.py | URL 정의 |
| 뷰 코딩하기 | views.py | 뷰 로직 작성 |
| 템플릿 코딩하기 | templates 디렉토리 | 템플릿 파일 작성 |
| 그 외 코딩하기 | - | (없음) |

□ 개발 코딩하기 - 뼈대

▣ 프로젝트 생성

```
$cd /home/shkim/pyDjango/  
$django-admin.py startproject mysite
```

```
/mysite (예제에서 ch2로 변경)  
/manage.py  
/mysite  
  /__init__.py  
  /settings.py  
  /urls.py  
  /wsgi.py
```

□ 개발코딩-빠대

▣ 프로젝트 설정 파일 변경(setting.py 설정)

1. 데이터베이스 설정
2. 템플릿 설정
3. 정적파일(static url, staticfiles_dirs)
4. 타임존 지정 (UTC->Asia/Seoul)
5. 미디어 관련사항 설정(파일 업로드 기능 개발시 필요)
6. 애플리케이션 등록(setting.py)
7. 기타 설정 - 언어 등등

2. 북마크 앱

□ 기본 테이블 생성

```
cd ~/python/work/pyweb01  
python manage.py migrate
```

□ 슈퍼 유저 생성

```
cd python/work/pyweb01  
python manage.py createsuperuser
```

username : admin

email :생략 가능

password : admin1234

2. 북마크앱

□ 애플리케이션 생성

```
python manage.py startapp bookmark
```

bookmark 관련 디렉토리가 만들어짐

eclipse에서 F5를 눌러 bookmark 패키지(디렉토리) 확인

```
/mysite (예제에서 ch2로 변경)
  /manage.py
  /mysite
    /__init__.py
    /settings.py
    /urls.py
    /wsgi.py
  /bookmark
    /__init__.py
    /admin.py
    /apps.py
    /migrations
    /models.py
    /tests.py
    /views.py
```

2. 북마크앱

□ 애플리케이션 등록(setting.py)

- ▣ pyweb01/settings.py 열어서 작성

OR

Application definition

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'bookmark', # 추가  
]
```

Application definition

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'bookmark.apps.BookmarkConfig',  
]
```

□ 언어 설정

Internationalization 이하 생략

LANGUAGE_CODE = 'ko'

TIME_ZONE = 'Asia/Seoul'

USE_I18N = True

USE_L10N = True

USE_TZ = True

□ 데이터베이스 변경 사항 반영

```
cd python\work\pyweb01
```

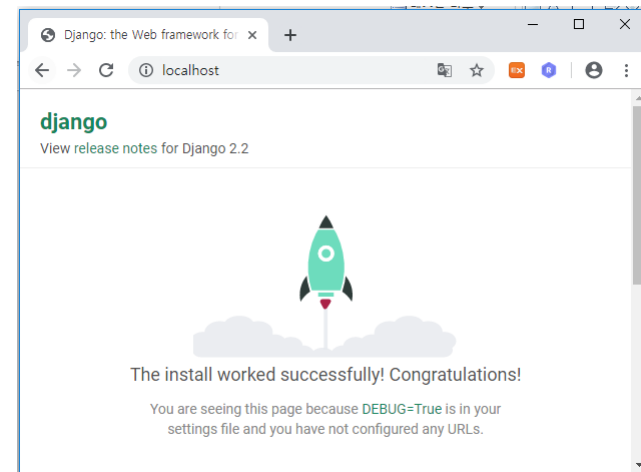
```
python manage.py makemigrations  
python manage.py migrate
```

□ 웹서버 구동

```
python manage.py runserver localhost:80
```

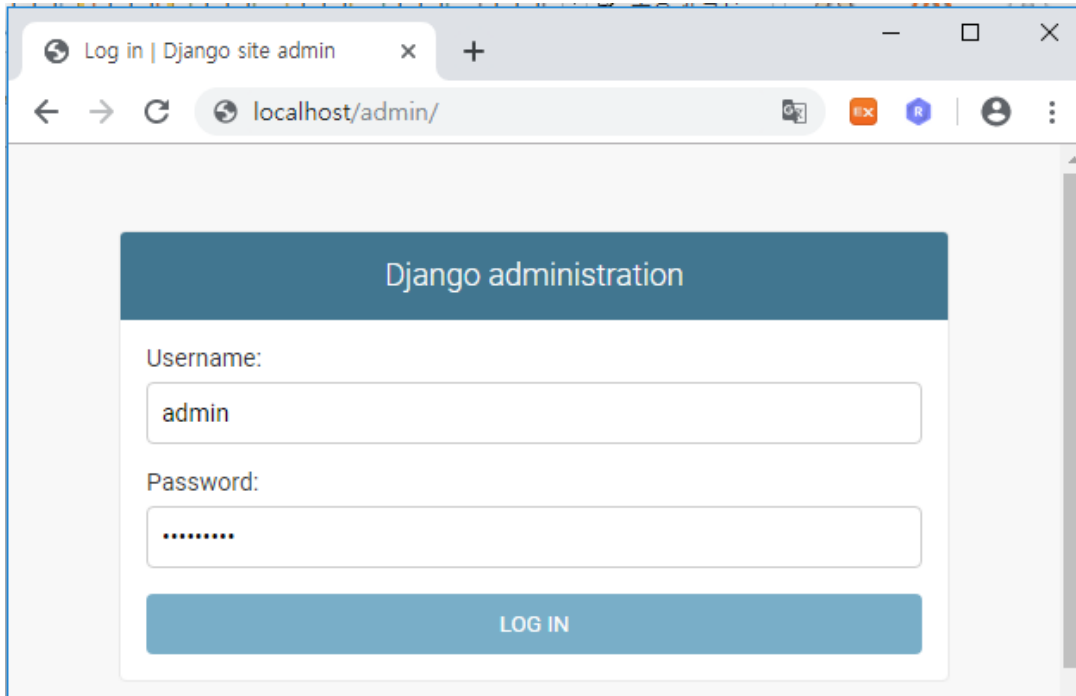
□ 웹브라우저에서 확인

```
http://localhost
```



□ 웹브라우저에서 확인

http://localhost/admin



The screenshot shows a web browser window with the address bar displaying 'localhost/admin/'. The page title is 'Log in | Django site admin'. The main content area features a 'Django administration' header, followed by 'Username:' and 'Password:' labels. The username field contains 'admin' and the password field is masked with dots. A blue 'LOG IN' button is at the bottom.

Log in | Django site admin

localhost/admin/

Django administration

Username:

admin

Password:

.....

LOG IN

□ **models.py** 설정

- ▣ 테이블을 새로 만들면 models.py와 admin.py 2개의 파일을 수정해야 함.
- ▣ models.py: 테이블에 대한 모델 클래스 정의
- ▣ admin.py : models.py에 등록한 테이블이 admin 사이트에서도 보이도록 처리

□ bookmark/models.py 작성

- 테이블을 하나의 클래스로 정의하고 테이블의 컬럼은 클래스의 변수로 매핑
- 테이블 클래스는 django.db.models.Model 클래스를 상속받아 정의
- 변수 자료형도 장고에서 미리 정의된 자료형을 사용

```
from django.db import models
```

```
# Create your models here.
```

```
class Bookmark(models.Model): #djang의 Model 클래스 상속받음
```

```
    #필드 선언,    #blank 빈값 허용 여부, null null 허용
```

```
    title=models.CharField(max_length=100, blank=True,null=True)
```

```
    #unique "primary key
```

```
    url=models.URLField("url",unique=True)
```

```
#객체를 문자열로 표현하는 함수
```

```
def __str__(self):
```

```
    return self.title
```

□ bookmark/admin.py 작성

```
from django.contrib import admin
from bookmark.models import Bookmark

# Register your models here.
#관리자 사이트에서 Bookmark 클래스 출력 모양 정의하는 코드
class BookmarkAdmin(admin.ModelAdmin):
    #관리자 화면에 출력할 필드 목록(튜플 형식)
    list_display=("title", "url")

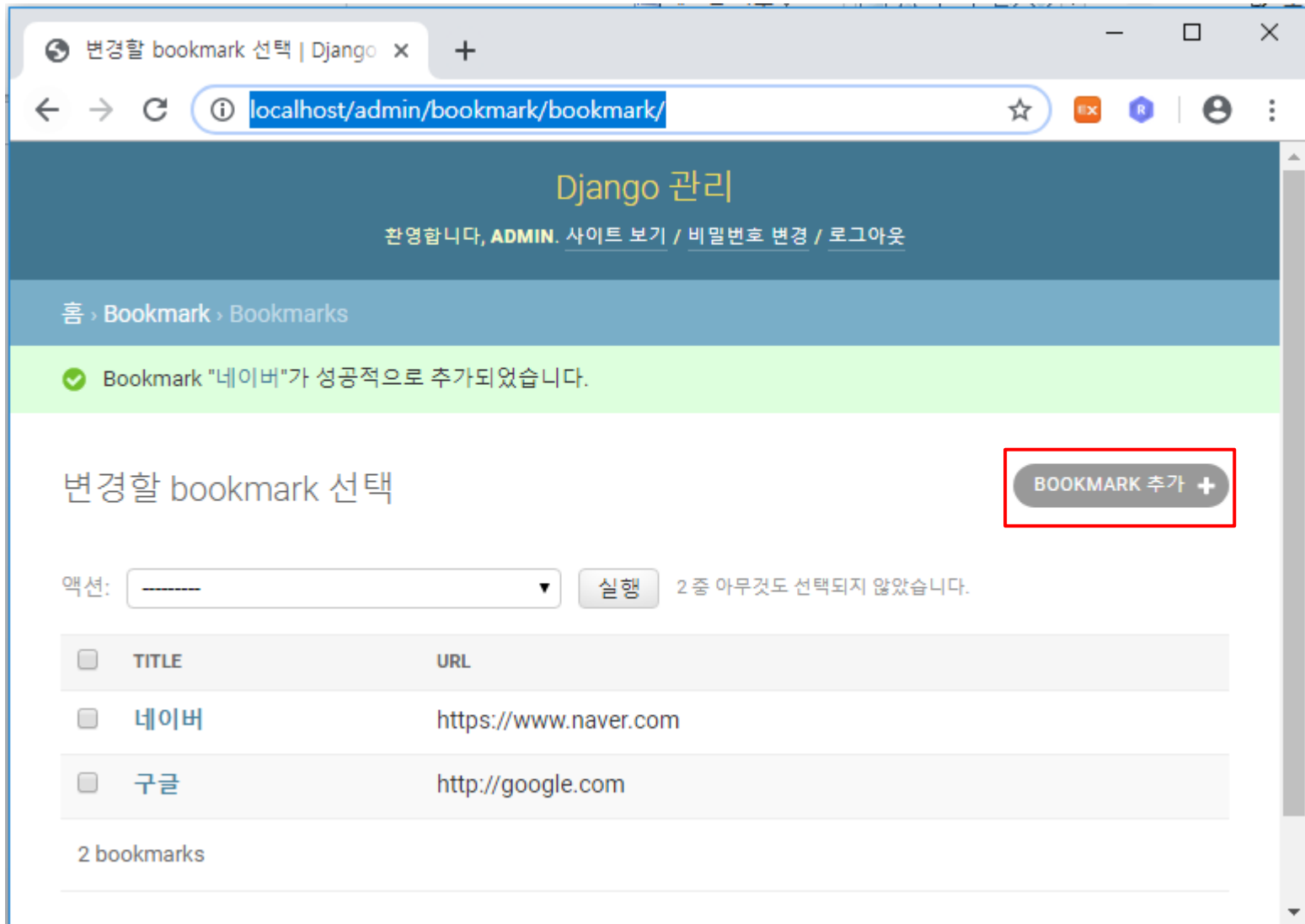
#Bookmark 클래스와 BookmaarAdmin 클래스를 등록
admin.site.register(Bookmark,BookmarkAdmin)
```

□ 데이터베이스 변경 사항 반영

```
cd ~/python/work/pyweb01  
python manage.py makemigrations  
python manage.py migrate
```

admin 작업

□ localhost/admin 사이트 데이터 추가



□ 클래스 방식 view

```
from bookmark.models import Bookmark
from django.views.generic import ListView, DetailView

# Create your views here.
#--- ListView
#템플릿파일 : 모델명소문자_list.html
class BookmarkLV(ListView):
    model=Bookmark

#--- DetailView
#템플릿파일 : 모델명소문자_detail.html
class BookmarkDV(DetailView):
    model=Bookmark
```


□ URLconf(urls.py) : path() 사용

```
from django.contrib import admin
from django.urls import path
from django.views.generic import ListView,DeleteView
from bookmark.models import Bookmark
from django.views.generic.detail import DetailView

urlpatterns = [
    path('admin/', admin.site.urls),

    path('bookmark/', ListView.as_view(model=Bookmark),name='index'),
    path('bookmark/<int:pk>/', DetailView.as_view(model=Bookmark), name='detail'),
]
```

□ URLconf(urls.py) : url() 사용

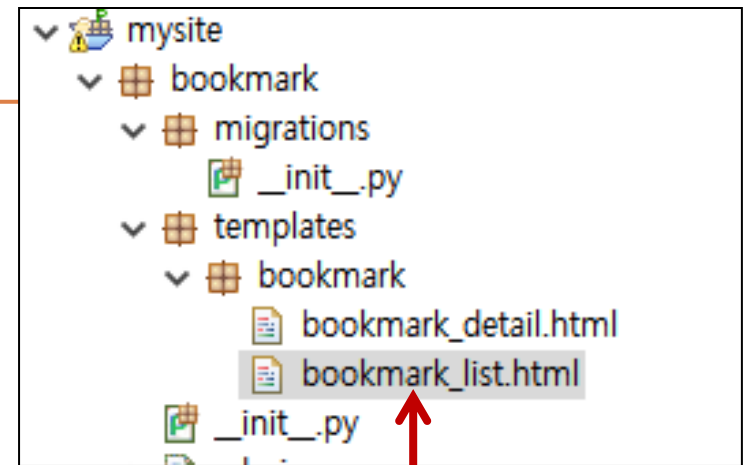
```
from django.contrib import admin
from bookmark.models import BookmarkLV, BookmarkDV
from django.conf.urls import url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),

    #Class-based views for Bookmark app
    url(r'^bookmark/$', BookmarkLV.as view(), name='index'),
    url(r'^bookmark/(?P<pk>#d+)/$$', BookmarkDV.as view(), name='detail'),
]
```

□ bookmark_list.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset= "UTF-8">
<title>Insert title here</title>
</head>
<body>
<div id= "container">
<h1>Bookmark List</h1>
<ul>
{% for bookmark in object list %}
<li> <a href= "{%url 'detail' bookmark.id%}">{{bookmark}}</a> </li>
{% endfor %}
</ul>
</div>
</body>
</html>
```



bookmark_list.html 위치

□ bookmark_detail.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset= "UTF-8">
<title>Insert title here</title>
</head>
<body>
<div id= "content">
<h1>{{dto.title}}</h1>
<ul>
<li>URL:<a href= "{{object.url}}">{{object.url}}</a> </li>
</ul>
</div>
</body>
</html>
```

□ 함수 방식- view views.py

```
from django.shortcuts import render, render to response
from bookmark.models import Bookmark
```

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

```
def home(request):
```

```
    #select * from bookmark_bookmark order by title
```

```
    urlList=Bookmark.objects.order_by("title") #-title 내림차순 정렬
```

```
    #select count(*) from bookmark_bookmark
```

```
    urlCount =Bookmark.objects.all().count()
```

```
    #list.html 페이지로 넘어가서 출력됨
```

```
    #render_to response("url",{"변수명","변수명"})
```

```
    return render(request, "list.html",{"urlList":urlList, "urlCount":urlCount)
```

□ urls.py

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import url
from bookmark import views

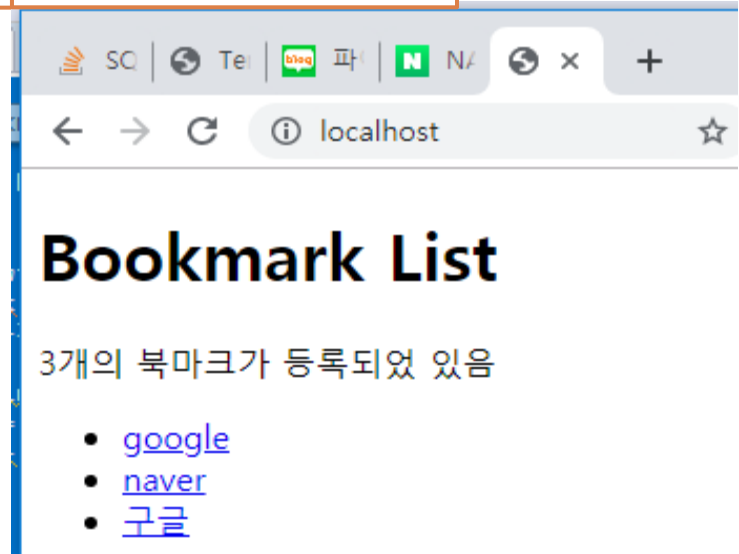
urlpatterns = [
    path('admin/', admin.site.urls),

    # r정규표현식, ^ 시작, $끝
    # http://localhost로 요청하면 views 모듈의 home 함수 실행
    url(r"^$", views.home())
]
```

□ list.html 작성

```
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<div id= "containter">
<h1>Bookmark List</h1>
{{urlCount}}개의 북마크가 등록되었 있음
<ul>
{% for row in urlList %}
<li> <a href= "detail?url={{row.url}}">{{row.title}}</a> </li>
{% endfor %}
</ul>
</div>
</body>
</html>
```

localhost 실행결과



□ detail.html 페이지 작성

▣ views.py에 detail 메소드 작성

```
from django.shortcuts import render, render_to_response
from bookmark.models import Bookmark
```

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

```
def home(request):
```

```
    #....생략
```

```
def detail(request):
```

```
    #get 방식 변수 받아오기 request.GET["변수명"]
```

```
    #post 방식 변수 받아오기 request.POST["변수명"]
```

```
    addr=request.GET[ "url" ]
```

```
    #select * from bookmark_bookmark where url="..."
```

```
    dto=Bookmark.objects.get(url=addr)
```

```
    #detail.html로 포워딩
```

```
    return render_to_response( "detail.html",{ "dto":dto} )
```


□ detail.html 페이지 작성

▣ urls.py에 detail url 추가

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import url
from bookmark import views

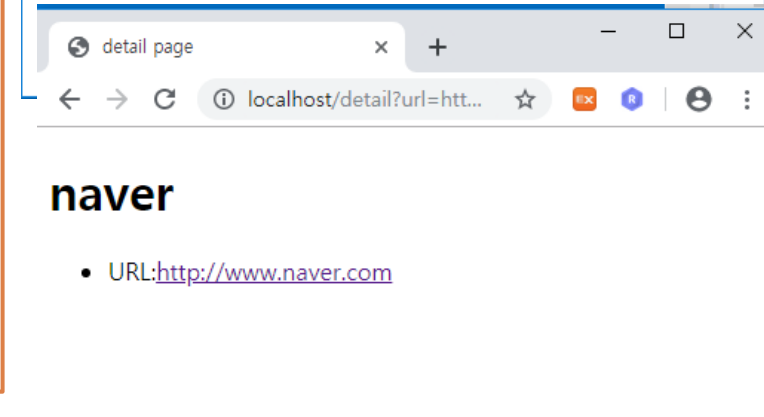
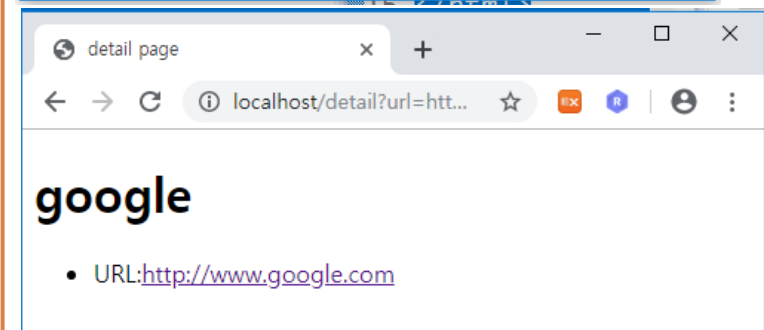
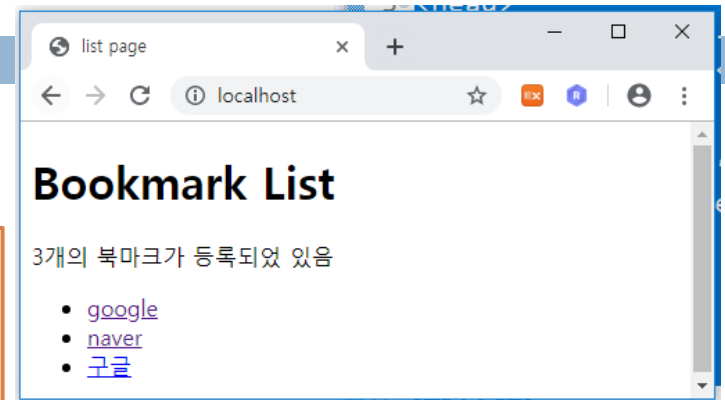
urlpatterns = [
    path('admin/', admin.site.urls),

    # r정규표현식, ^ 시작, $끝
    # http://localhost로 요청하면 views 모듈의 home 함수 실행
    url(r"^$", views.home),

    # http://localhost/detail로 요청하면 views 모듈의 home 함수 실행
    url(r"^detail$", views.detail)
]
```

□ detail.html 작성

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>detail page</title>
</head>
<body>
<div id="container">
<h1>{{dto.title}}</h1>
<ul>
<li>URL:<a href="{{dto.url}}">{{dto.url}}</a> </li>
</ul>
</div>
</body>
</html>
```



3. 설문조사

□ django-debug-toolbar 설치

- ▣ 디버깅을 위한 도구
- ▣ 패키지 설치

```
pip install django-debug-toolbar
```

□ 프로젝트만들기

- ▣ New-Other-PyDev-PyDev Django Project
- ▣ 프로젝트 이름 : pyweb03
- ▣ pyweb03 디렉토리가 2개 만들어짐
- ▣ d:\python\work\pyweb03
- ▣ d:\python\work\pyweb03\pyweb03
 - python 웹 프로젝트 설정디렉토리

□ 기본 테이블 만들기

```
d:  
cd pyphon\work\pyweb03  
python manage.py migrate
```

□ 슈퍼유저 생성

```
d:  
cd pyphon\work\pyweb03  
python manage.py createsuperuser  
  
username : admin  
password : admin1234
```

□ 애플리케이션 생성

```
python manage.py startapp survey
```

- survey 관련 디렉토리가 만들어짐
- eclipse에서 F5를 눌러 새로고침으로 확인

□ setting.py 변경

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'survey',  
    'debug_toolbar',  
]
```

추가

```
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',  
    'debug_toolbar.middleware.DebugToolbarMiddleware',  
]
```

추가

```
INTERNAL_IPS = ('127.0.0.1',)
```

추가

□ survey/models.py

- ▣ 테이블 생성을 위해 model.py, admin.py 파일 수정
- ▣ model.py : 테이블 모델에 대한 클래스 정의
- ▣ admin.py : models.py에 등록한 테이블이 admin 사이트에서 보이도록 처리
- ▣ 테이블을 하나의 클래스로 정의, 컬럼은 클래스 변수로 매핑
- ▣ 테이블 클래스는 django.db, models.Model 클래스를 상속받아 정의, 변수 자료형도 장고에서 미리 정의된 자료형을 사용
- ▣ models.URLField('필드의 별칭', unique)

```
from django.db import models
```

```
# Create your models here.
```

```
#설문 문항 클래스
```

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

```
    #설문 인덱스
```

```
    survey_idx=models.AutoField(primary_key=True)
```

```
    #문제
```

```
    question=models.TextField(null=False)
```

```
    # 답 1-4
```

```
    ans1=models.TextField(null=True)
```

```
    ans2=models.TextField(null=True)
```

```
    ans3=models.TextField(null=True)
```

```
    ans4=models.TextField(null=True)
```

```
    #설문진행상태(y= 진행중, n=종료)
```

```
    status=models.CharField(max_length=1, default= 'y')
```

```
#설문 응답
```

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

```
    #응답아이디(자동증가 필드)
```

```
    answer_idx=models.AutoField(primary_key=True)
```

```
    #설문아이디(survey_idx 필드 참조 Foreign key)
```

```
    survey_idx=models.IntegerField()
```

```
    #응답번호
```

```
    num=models.IntegerField()
```

□ survey/admin.py 작성

▣ Admin 사이트에 테이블 반영

```
from django.contrib import admin
from survey.models import Survey, Answer

# Register your models here.
class SurveyAdmin(admin.ModelAdmin):
    list_display=("question", "ans1", "ans2", "ans3", "ans4", "status")

admin.site.register(Survey, SurveyAdmin)
admin.site.register(Answer)
```

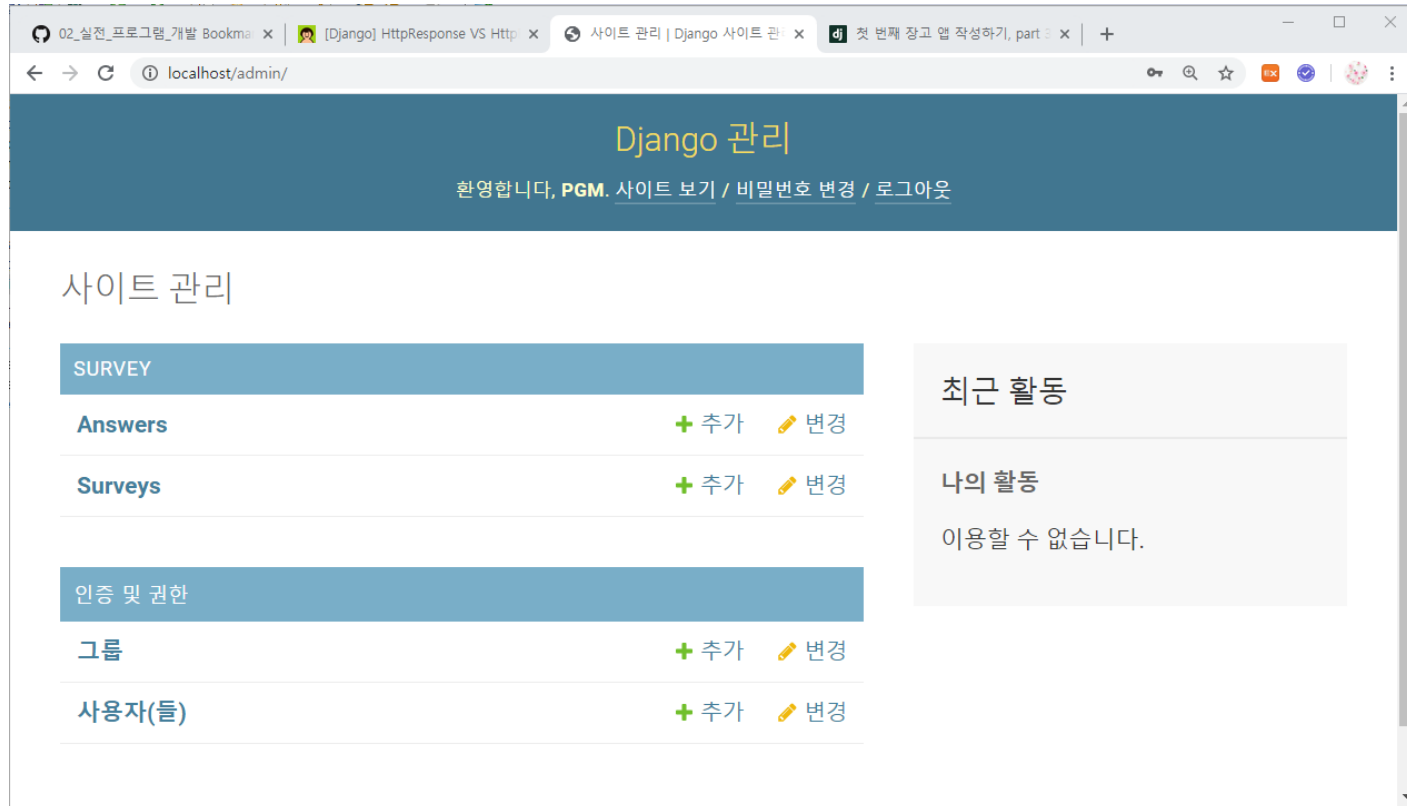
□ 데이터베이스 변경 사항 반영

```
python manage.py makemigrations
python manage.py migrate
```


□ 웹서버 구동

```
python manage.py runserver localhost:80
```

□ 크롬에서 localhost/admin 실행



□ urls.py 디버깅 관련 url 등록

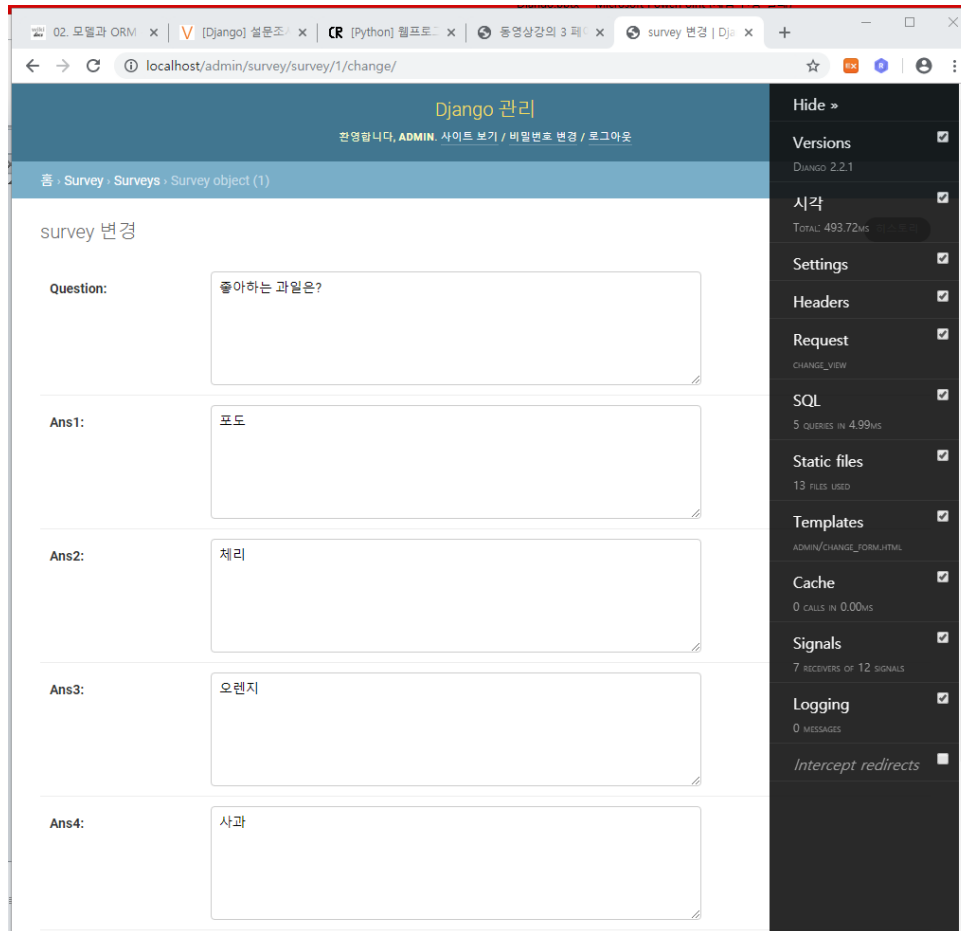
```
from django.contrib import admin
from django.urls import include,path
from django.conf import settings
from django.conf.urls import url

urlpatterns = [
    #관리자용 사이트
    path('admin/', admin.site.urls),
]

#디버깅 관련 URL
if settings.DEBUG:
    import debug_toolbar
    urlpatterns += [
        url(r'^__debug__/', include(debug_toolbar.urls)),
    ]
```

□ 관리자 사이트에서 문제 등록

▣ debug_toolbar 활용



The screenshot shows a web browser window with the Django Admin interface. The URL bar indicates the page is at `localhost/admin/survey/survey/1/change/`. The main content area displays a form titled "survey 변경" (Survey Change) for a "Survey object (1)". The form contains a "Question:" field with the text "좋아하는 과일은?" (What is your favorite fruit?) and four "Ans:" fields with the following values: "포도" (Grape), "체리" (Cherry), "오렌지" (Orange), and "사과" (Apple). On the right side of the screen, the debug_toolbar sidebar is visible, showing various debugging tools. The sidebar includes sections for Versions (Django 2.2.1), 시각 (Timing) (TOTAL: 493.72ms), Settings, Headers, Request (CHANGE_VIEW), SQL (5 QUERIES IN 4.99ms), Static files (13 FILES USED), Templates (ADMIN/CHANGE_FORM/HTML), Cache (0 CALLS IN 0.00ms), Signals (7 RECEIVERS OF 12 SIGNALS), Logging (0 MESSAGES), and Intercept redirects.

□ main 페이지 작성

▣ urls.py 수정

```
from django.contrib import admin
from django.urls import include, path
from django.conf import settings
from django.conf.urls import url
from survey import views
```

```
urlpatterns = [
    #관리자용 사이트
    path('admin/', admin.site.urls),
    #설문조사 관련 url
    url(r'^$', views.main), #http://localhost
]
```

.....

□ main 페이지 작성

▣ views.py 수정

```
from django.shortcuts import render
# 설문테이블, 응답테이블 import
from survey.models import Survey, Answer

# Create your views here.
#http://localhost (시작페이지)
def main(request):
    # filter => where
    # order_by("필드") 오름차순 정렬
    # order_by("-필드") 내림차순 정렬
    survey=Survey.objects.filter(status= 'y').order_by("-survey_idx")[0]
    # main.html 페이지로 이동, 데이터 전달
    return render(request, "main.html", {'survey':survey})
```

□ main 페이지 작성

- ▣ survey->templates 폴더 작성
- ▣ templates폴더에 main.html 작성

```
<!DOCTYPE html>
<html>
<head>
<meta charset= "UTF-8">
<title>main page</title>
<script type= "text/javascript">
function show_result(){
location.href="show_result?survey_idx={{survey.survey_idx}}";
}
</script>
</head>
--계속--
```

.....

```
<body>
<h2>온라인 설문조사</h2>
<form method="post" action="save_survey">
{% csrf_token %} <!-- csrf 공격 방지 코드 -->
{{survey.question}}<br>
<input type="radio" name="num" value="1">{{survey.ans1}}<br>
<input type="radio" name="num" value="2">{{survey.ans2}}<br>
<input type="radio" name="num" value="3">{{survey.ans3}}<br>
<input type="radio" name="num" value="4">{{survey.ans4}}<br>
<br>
<input type="hidden" name="survey_idx" value="{{survey.survey_idx}}">
<input type="submit" value="투표">
<input type="button" value="결과 확인" onclick="show_result()">
</form>
</body>
</html>
```

□ 투표 결과 저장

▣ urls.py 수정

```
--생략---  
urlpatterns = [  
    #관리자용 사이트  
    path('admin/', admin.site.urls),  
    #설문조사 관련 url  
    url(r'^$', views.main), #http://localhost  
    url(r'^save_survey$', views.save_survey),  
]  
  
--생략--
```


□ 투표 결과 저장

▣ views.py 수정

```
from django.shortcuts import render, render to response
#csrf(Cross Site Request Forgery)
#크로스 사이트 요청 위조
from django.views.decorators.csrf import csrf_exempt
# 설문테이블, 응답테이블 import
from survey.models import Survey, Answer
```

```
def main(request):
    ---생략
```

```
@csrf_exempt
```

```
def save_survey(request):
    #survey_idx : 설문문항 코드
    #num : 사용자가 선택한 번호
```

```
dto=Answer(survey_idx=request.POST[ "survey_idx" ] ,num=request.POST["num"])
dto.save() #insert query 실행
# success.html로 이동
return render(request, "success.html")
```

□ 투표 결과 저장

- ▣ html 페이지(templates 폴더에 작성)

```
<!DOCTYPE html>
<html>
<head> <title>result page</title> </head>
<body>
<h2>설문조사 결과</h2>
<table border= 1>
<tr align= "center">
    <th> 문항 </th> <th> 응답수 </th> <th> 응답비율 </th>
</tr>
{% for row, ans in surveyList %}
<tr align= "center">
    <td> {{ans}} </td> <td> {{row.sum_num}} </td> <td> {{row.rate}}% </td>
</tr>
{% endfor %}
</table>
</body>
</html>
```

□ 투표결과 확인 페이지

▣ urls.py 수정

```
urlpatterns = [  
    #관리자용 사이트  
    path('admin/', admin.site.urls),  
    #설문조사 관련 url  
    url(r'^$', views.main), #http://localhost  
    url(r'^save_survey$', views.save_survey),  
    url(r'^show_result$', views.show_result),  
]
```

□ 투표결과 확인 페이지

▣ views.py 수정

```
def show_result(request):
```

```
    idx=request.GET['survey_idx']
```

```
    ans=Survey.objects.get(survey_idx=idx)
```

```
    answer=[ans.ans1, ans.ans2, ans.ans3, ans.ans4]
```

```
    surveyList=Survey.objects.raw( """
```

```
select
```

```
survey_idx,num,count(num) sum num,
```

```
round((select count(*) from survey_answer
```

```
    where survey_idx=a.survey_idx and num=a.num)*100.0 /
```

```
(select count(*) from survey_answer
```

```
    where survey_idx=a.survey_idx),1) rate
```

```
from survey_answer a
```

```
where survey_idx=%s
```

```
group by survey_idx,num
```

```
order by num
```

```
""", idx)
```

```
    surveyList=zip(surveyList,answer)
```

```
    print("surveyList:",surveyList)
```

```
    print("answer:",answer)
```

```
# select count(*) from survey_answer  
count = Answer.objects.all().count()  
return render(request,'result.html', #  
{ 'surveyList':surveyList, "count" : count})
```

□ 투표결과 확인 페이지

▣ templates 폴더에 result.html 작성

```
<!DOCTYPE html>
<html>
<head> <title>result page</title> </head>
<body>
<h2>설문조사 결과</h2>
응답인원수 : {{count}}
<table border="1">
    <tr align="center">
        <th>문항</th> <th>응답수</th> <th>응답비율</th>
    </tr>
    {% for row,ans in surveyList %}
    <tr align="center">
        <td>{{ans}}</td>    <td>{{row.sum_num}}</td> <td>{{row.rate}}</td>
    </tr>
    {% endfor %}
</table>
</body>
</html>
```