CHAPTER 06

웹 서버(Apache)와 연동

장고를 사용하여 웹 애플리케이션을 개발한 후에 이를 실제로 서비스하기 위해서는 운영 환경에 개발한 프로그램을 배포하고 실행해야 합니다. 이렇게 개발 환경에서 운영 환경으로 옮겨가기 위해서는 우리가 개발 시 지정했던 설정 사항을 몇 가지 변경해야 합니다. 또한, 운영 환경의 웹 서버에서도 우리가 만든 애플리케이션을 인식할 수 있도록 설정 사항 변경이 필요합니다.

6장에서는 상용 웹 서버로 현재 가장 널리 사용되는 아파치에서 장고 애플리케이션을 실행하기 위해 필요한 사항들을 설명하겠습니다. 개발 환경과 운영 환경의 차이점을 이해하고, 이에 따라 설정 사항을 변경하는 것이 주요 작업입니다.

6.1 mod_wsgi 확장 모듈

아파치Apache 웹 서버의 프로그램명은 httpd이며, 전 세계에서 운영 중인 웹 사이트의 50% 이상이 사용할 정도로 인기있는 웹 서버입니다. 아파치는 추가로 필요한 기능을 모듈로 만들어 동적 로딩 방식으로 기능을 확장할 수 있는 특징이 있습니다. 클라이언트 요청 URL을 서버 내 디렉토리로 매핑해주는 mod_alias, 사용자 인증을 위한 mod_auth, 톰캣 연동에 사용되는 mod_jk, 프록시 기능을 제공하는 mod_proxy, URL rewrite를 지원하는 mod_rewrite, PHP 및 Perl 스크립트를 실행할 수 있는 mod_php, mod_perl 등의 수많은 확장 모듈이 사용되고 있습니다. 이번 장에서

설명하는 mod_wsgi도 파이썬 웹 애플리케이션을 실행할 수 있는 확장 모듈 중 하나입니다.

mod_wsgi는 파이썬 웹 애플리케이션 표준 규격인 WSGI[Web Server Gateway Interface]를 구현한 확장 모듈로서, 파이썬 웹 애플리케이션을 아파치에서 실행하는 데 사용합니다. 장고 프레임워크에서도 기본적으로 WSGI 스펙을 준수하므로, 아파치와 장고를 연동하기 위해서 mod_wsgi 모듈을 사용하도록 하겠습니다. WSGI의 스펙은 파이썬 표준 규격인 PEP[Python Enhancement Proposals] 333에 자세하게 정의되어 있습니다.

mod_wsgi를 사용해 웹 애플리케이션(또는 WSGI 애플리케이션이라고도 함)을 실행할 때, 두 가지 방식의 실행 모드를 제공합니다. 첫 번째는 내장[embedded] 모드로 WSGI 애플리케이션을 실행하는 방식입니다. 일반적인 아파치 자식 프로세스 컨텍스트 내에서 애플리케이션이 실행된다는 점에서 mod_python 방식과 유사합니다. 이 모드로 WSGI 애플리케이션을 구동하면 같은 아파치 웹 서버에서 실행되는 다른 웹 애플리케이션과 같은 아파치 자식 프로세스를 공유하게 됩니다. 내장 모드의 단점은 WSGI 애플리케이션의 소스가 변경되어 다시 적용하려면 아파치 전체를 재기동해야 한다는 것입니다. 재기동으로 인해 다른 서비스도 영향을 받게 되고, 경우에 따라서는 아파치 재기동 권한이 없을 수도 있습니다.

두 번째는 데몬[daemon] 모드로 유닉스 기반의 아파치 2.0 이상에서 지원됩니다. 이 모드는 WSGI 애플리케이션의 전용 프로세스에서 애플리케이션이 실행된다는 점에서 FCGI[Fast CGI]/SCGI[Simple CGI]와 유사합니다. 다른 점은 WSGI 애플리케이션을 구현할 때 별도의 프로세스 관리자나 WSGI 어댑터가 필요하지 않으며, 모든 처리는 mod_wsgi가 관리한다는 것입니다. WSGI 애플리케이션이 데몬 모드로 동작하면 일반적인 다른 아파치 자식 프로세스와 별도의 프로세스에서 동작합니다. 그래서 정적인 파일을 서비스하는 프로세스나 PHP, Perl 등의 아파치 모듈로 서비스하는 다른 애플리케이션에 미치는 영향이 미미합니다. 또한, 필요하다면 WSGI 애플이케이션 간에도 서로 영향을 주지 않도록 실행 유저를 달리하여 데몬 프로세스를 기동하는 것도 가능합니다.

성능적인 측면에서 mod_wsgi는 C 언어로 구현되어 있기 때문에 내부적으로 아파치가 직접 파이썬 API와 동작하므로 상대적으로 적은 메모리를 사용합니다. mod_wsgi의 이러한 구현방식 덕분에 아파치의 내장 파이썬 인터프리터가 동작하는 방식인 mod_python이나 FCGI/SCGI 같은 개선된 형태로, CGI에 비해서도 좋은 성능을 보이고 있습니다.

또한, 위에서 설명한 내장 모드와 데몬 모드의 동작 방식은 다르지만, 대용량 웹 애플리케이션에서의 처리 성능은 크게 다르지 않습니다. 처리 성능에 문제가 있다면 애플리케이션 자체 문제이거나

데이터베이스 처리로 인한 성능 저하가 더 큰 이슈가 됩니다. 장고에서는 안정성을 고려하여 내장 모드보다는 데몬 모드로 실행할 것을 권장하고 있습니다.

6.2 장고의 웹 서버 연동 원리

3장에서 장고 프로젝트의 뼈대를 만들 때 다음 명령을 실행하였습니다.

```
$ python manage.py startproject mysite
```

위 명령으로 장고의 여러 가지 기본 파일들과 함께 mysite/wsgi.py 파일이 만들어 집니다. 이 모듈이 바로 장고와 웹 서버를 연결하는 데 필요한 파일입니다. 이 모듈에는 WSGI 규격에 따라 호출 가능한callable 애플리케이션 객체를 정의하고 있습니다. 객체명은 반드시 **application**이어야 합니다. 물론 장고가 자동으로 만들어주므로 별도로 작성해줄 필요는 없습니다.

이 application 객체는 아파치와 같은 상용 웹 서버뿐만 아니라, 장고의 개발용 웹 서버인 runserver에서도 같이 사용하는 객체입니다. 다만 다른 점은 아파치에서는 httpd.conf 설정 파일에서 WSGIScriptAlias 지시자를 통해 지정하고, 개발용 runserver에서는 settings 모듈(mysite/settings.py)의 **WSGI_APPLICATION** 변수로 지정합니다.

웹 서버는 이 application 객체를 호출하여 장고의 애플리케이션을 실행하게 됩니다. 다만 application 객체를 호출하기 전에 현재의 프로젝트 및 프로젝트에 포함된 모든 애플리케이션들에 대한 설정 정보를 로딩하는 작업이 필요합니다. 이 설정 정보를 담고 있는 settings 모듈의 위치를 지정해주는 방법도 다른데, 아파치 등의 웹 서버는 wsgi.py 파일에서 지정해주고, 개발용 runserver는 다음과 같이 실행 옵션으로 지정해줄 수 있습니다. 별도로 지정해주지 않은 경우에는 디폴트로 **프로젝트명.settings**를 사용합니다.

```
# 상용 웹 서버 - mysite/wsgi.py 파일에서 지정
import os
os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'
```

```
# 개발용 runserver - 실행 시 지정
$ python manage.py runserver —settings=mysite.settings
```

6.3 상용 서버 적용 전 장고의 설정 변경

아파치 등과 같은 웹 서버와 연동하는 단계는 장고의 애플리케이션 개발이 완료되고, 개발용 웹 서버인 runserver에서 정상적으로 동작하는 것을 확인한 이후 시점입니다. 즉, 지금까지는 개발 모드 환경에 맞춰 설정해왔지만, 상용 서버에 적용하기 위해서는 보안, 성능 등을 고려하여 상용 환경에 맞는 설정으로 변경해야 할 사항들이 있습니다.

개발 모드에서는 에러 발생 시 디버그를 위해 브라우저에 여러 가지 정보를 출력하여 보여주었습니다. 이런 디버그 정보는 프로젝트에 관련된 중요 정보들이므로, 상용 모드에서는 settings 모듈의 DEBUG 설정값을 False로 셋팅하여 디버그 정보가 노출되지 않도록 해야 합니다.

```
DEBUG = False
```

DEBUG = False로 설정되어 있으면, 반드시 settings 모듈의 ALLOWED_HOSTS 항목을 설정해야 합니다. 악의적인 공격자가 HTTP Host 헤더를 변조하여 CSRF(Cross Site Request Forgery) 공격을 할 수 있기 때문에 이를 방지하기 위한 것입니다.

```
ALLOWED_HOSTS = [ u'192.168.56.101' ]
```

개발 서버에서는 이미지, 자바스크립트, CSS 등의 정적 파일들을 알아서 찾아 주었지만, 상용 모드에서는 아파치와 같은 웹 서버가 정적 파일들이 어디에 있는지 알 수 있도록 해주어야 합니다. settings 모듈의 STATIC_ROOT 항목은 장고의 **collectstatic** 명령 실행 시 정적 파일들을 한 곳에 모아주는 디렉토리입니다. 여기서는 /home/shkim/pyBook/ch6/www_static 디렉토리에 정적 파일들을 모아줍니다. **6.4 내장 모드 실행**에서 보게 될 아파치 설정 파일에서도 **Alias /static/** 설정이 필요합니다.

```
STATIC_ROOT = os.path.join(BASE_DIR, "www_static")
```

collectstatic 명령은 다음과 같이 실행합니다. 이 명령을 사용 시 주의할 점은 settings 모듈의 STATICFILES_DIRS 항목에 STATIC_ROOT 항목에서 정의된 디렉토리가 포함되면 안 된다는 것입니다. **STATICFILES_DIRS** 항목에 정의된 디렉토리에서 정적 파일을 찾아 **STATIC_ROOT** 디렉토리에 복사해주기 때문입니다.

```
$ python manage.py collectstatic
```

개발 모드에서는 runserver를 실행시킨 사용자의 권한으로 데이터베이스 파일이나 로그 파일을 액세스합니다. 그러나 상용 모드에서는 웹 서버 프로세스의 권한자인 apache 사용자 권한으로 해당 파일들을 액세스할 수 있어야 합니다. 이를 위해 settings 모듈의 DATABASES 항목에서 NAME 속성값의 경로를 db/db.sqlite3로 변경해주고,

```
DATABASES = {
        . . .
        'NAME': os.path.join(BASE_DIR, 'db/db.sqlite3'),
```

해당 디렉토리 및 파일의 액세스 권한을 아래처럼 변경해줘야 합니다. SQLite 데이터베이스 파일의 위치를 옮기고, SQLite 데이터베이스가 있는 디렉토리 및 파일에 apache 사용자가 접근/읽기/쓰기 가능하도록 설정한 것입니다.

```
$ cd /home/shkim/pyBook/ch6
$ mkdir db
$ mv db.sqlite3 db/
$ chmod 777 db/
$ chmod 666 db/db.sqlite3
```

settings 모듈의 LOGGING 항목에 로깅 관련 사항이 정의되어 있고, 여기에 로그 파일의 위치가 설정되어 있습니다. 로그 파일에 apache 사용자가 읽기/쓰기 가능하도록 설정한 것입니다.

```
$ cd /home/shkim/pyBook/ch6
$ chmod 777 logs/
$ chmod 666 logs/logfile
```

NOTE_ 이 외에도 상용화하기 전 확인 사항에 대한 자세한 내용은 다음 페이지를 참고하기 바랍니다.
https://docs.djangoproject.com/en/1.7/howto/deployment/checklist/

6.4 내장 모드로 실행

mod_wsgi 모듈이 아파치 프로세스에 내장되어 실행되는 방식을 먼저 살펴보겠습니다.

6.4.1 아파치 설정

아파치 웹 서버에서 mod_wsgi 모듈을 이용해 파이썬 웹 애플리케이션을 실행할 수 있도록 아파치 설정 파일인 httpd.conf에 mod_wsgi 관련 설정을 추가해야 합니다. 그리고 아파치에 대한 설정 및 웹 서버(httpd) 실행은 루트root 권한으로 작업합니다.

예제 6-1 아파치 설정 – 내장 모드로 실행하는 경우

```
# cd /etc/httpd/conf
# vi httpd.conf

# 상단 내용 동일 ----------❶
WSGIScriptAlias / /home/shkim/pyBook/ch6/mysite/wsgi.py ----------❷
WSGIPythonPath /home/shkim/pyBook/ch6 ----------❸

<Directory /home/shkim/pyBook/ch6/mysite> ----------
<Files wsgi.py>                                    |
Require all granted                                |---❹
</Files>                                           |
</Directory> ----------
```

```
Alias /static/ /home/shkim/pyBook/ch6/www_static/   ----------❺
<Directory /home/shkim/pyBook/ch6/www_static>   ----------┐
Require all granted                                       ---❻
</Directory>   ----------┘
```

위 설정 내용을 라인별로 설명하겠습니다.

❶ 기존의 설정 내용은 변경사항이 없고, 파일의 끝에 다음 내용을 추가합니다.

❷ 아파치 웹 서버로 서비스하는 URL(/)과 wsgi.py 파일의 위치를 매핑해줍니다. 루트(/) URL로 시작하는 모든 요청은 wsgi.py 파일에서 정의된 WSGI application에서 처리한다는 의미입니다.

❸ 파이썬 임포트 경로를 지정합니다. 즉, import mysite 등의 문장이 정상으로 동작하도록 합니다.

❹ 아파치가 wsgi.py 파일을 액세스할 수 있도록 mysite 디렉토리 및 wsgi.py 파일에 대한 접근 권한을 설정합니다.

❺ /static/ URL에 대한 처리를 위해 static 파일이 위치한 디렉토리를 매핑해줍니다. 이 디렉토리는 장고의 collectstaic 명령에 의해 static 파일을 모아둔 디렉토리입니다. 이는 settings.py 파일의 STATIC_ROOT 항목에 정의된 디렉토리이기도 합니다.

❻ 아파치가 www_static 디렉토리에 액세스할 수 있도록 디렉토리 접근 권한을 설정합니다.

6.4.2 지금까지 작업 확인하기

mod_wsgi 모듈에 대한 아파치 설정이 끝나면 아파치를 기동해서 동작을 확인합니다.

아파치를 기동하기 전에 한 가지 할 일이 있습니다. 설정을 정확하게 했는데도 시스템에 따라 서비스가 안 되는 경우가 있습니다. 즉, 장고의 runserver를 실행하면 정상적으로 서비스가 되는데, 아파치 웹 서버로는 서비스가 안 되는 경우가 이에 해당됩니다. 이런 경우에는 SELinux(Security Enhanced Linux) 정책이 적용되어 보안 문제로 서비스가 안 되는 것이므로, SELinux 보안 정책을 변경해줘야 합니다.

```
# setenforce permissive
```

아파치 기동 명령은 다음과 같습니다.

```
# service httpd start
```

아파치를 기동한 후에 브라우저를 통해 루트(/) URL로 접속합니다. 아파치 웹 서버는 80 포트를 사용하는데, 포트번호는 생략할 수 있습니다.

```
http://192.168.56.101:80/
```

다음과 같이 프로젝트의 첫 화면이 나타나면 정상입니다.

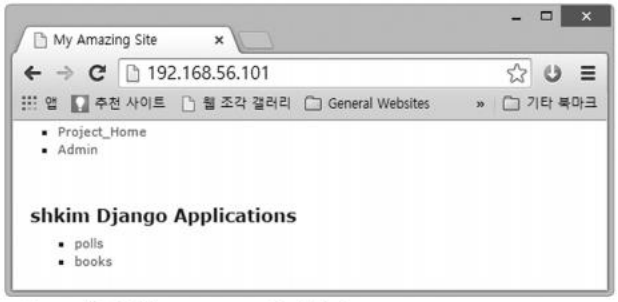

그림 6-1 아파치 실행 – mysite 프로젝트 첫 화면

또한, 아파치의 접속 로그를 통해서도 정상적으로 처리된 것을 확인할 수 있습니다. access_log를 보면 루트(/) URL에 대한 GET 요청에 응답 코드 200으로 성공 응답을 보내주고 있습니다.

```
[root@localhost logs]# cd /etc/httpd/logs
[root@localhost logs]# tail access_log
192.168.56.1 - - [04/Jan/2015:00:53:32 +0900] "GET / HTTP/1.1" 200
ML, like Gecko) Chrome/39.0.2171.95 Safari/537.36"
```

그림 6-2 아파치 실행 – apache access_log

NOTE_ SELinux 운용 모드

SELinux 정책은 아래와 같이 세 가지 모드로 운용될 수 있습니다. 정책 변경은 setenforce 명령으로 할 수도 있고, /etc/selinux/config 파일에서도 변경할 수 있습니다. setenforce 명령으로 변경하면 현재 쉘에서 작업하는 동안만 임시적으로 변경하는 것이며, config 파일에서 변경하면 시스템 부팅 시에 운용 모드가 적용됩니다.

- **enforcing(1)** : SELinux 보안 정책이 실제로 동작하여 위반 시 처리에 실패함
- **permissive(0)** : SELinux 보안 정책이 동작하지는 않지만, 위반 상황 발생 시 경고(warning)로 알려줌
- **disable** : SELinux 보안 정책이 동작하지 않음, config 파일에서만 동작함

6.5 데몬 모드로 실행

장고에서는 안정성을 고려하여 내장 모드보다는 데몬 모드로 실행할 것을 권장하고 있습니다.

6.5.1 아파치 설정

내장 모드로 실행 시 설정했던 방법과 유사하게 아파치 설정 파일인 httpd.conf에 mod_wsgi 관련 설정을 추가해야 합니다.

예제 6-2 아파치 설정 - 데몬 모드로 실행하는 경우

```
# cd /etc/httpd/conf
# vi httpd.conf

# 상단 내용 동일 ----------❶
WSGIScriptAlias / /home/shkim/pyBook/ch6/mysite/wsgi.py ----------❷
WSGIDaemonProcess mysite python-path=/home/shkim/pyBook/ch6 ----------❸
WSGIProcessGroup mysite ----------❹

<Directory /home/shkim/pyBook/ch6/mysite> ----------
<Files wsgi.py>                                    |
Require all granted                                |---❺
</Files>                                           |
</Directory> ----------

Alias /static/ /home/shkim/pyBook/ch6/www_static/ ----------❻
<Directory /home/shkim/pyBook/ch6/www_static> ----------
Require all granted                                |---❼
</Directory> ----------
```

위 설정 내용을 라인별로 설명하겠습니다.

❶ 기존의 설정 내용은 변경사항이 없고, 파일의 끝에 다음 내용을 추가합니다.

❷ 아파치 웹 서버로 서비스하는 URL(/)과 wsgi.py 파일의 위치를 매핑해줍니다. 루트(/) URL로 시작하는 모든 요청은 wsgi.py 파일에서 정의된 WSGI application에서 처리한다는 의미입니다.

❸ 별도의 데몬 프로세스에서 장고를 실행하기 위해 프로세스 속성을 설정합니다. 위 예제에서는 파이썬의 모듈 임포트 경로를 지정하였습니다. 내장 모드에서 사용했던 WSGIPythonPath 지시자는 사용할 수 없습니다. 프로세스 속성으로 프로세스의 개수와 스레드의 개수 등도 지정할 수 있습니다.

❹ 프로세스 그룹을 지정합니다. 동일한 프로세스 그룹에 할당된 애플리케이션은 같은 데몬 프로세스에서 실행됩니다.

❺ 아파치가 wsgi.py 파일을 액세스할 수 있도록 mysite 디렉토리 및 wsgi.py 파일에 대한 접근 권한을 설정합니다.

❻ /static/ URL에 대한 처리를 위해 static 파일이 위치한 디렉토리를 매핑해줍니다. 이 디렉토리는 장고의 collectstaic 명령에 의해 static 파일을 모아둔 디렉토리입니다. 이는 settings.py 파일의 STATIC_ROOT 항목에 정의된 디렉토리이기도 합니다.

❼ 아파치가 www_static 디렉토리에 액세스할 수 있도록 디렉토리 접근 권한을 설정합니다.

6.5.2 지금까지 작업 확인하기

mod_wsgi 모듈에 대한 아파치 설정이 끝나면 아파치를 기동해서 동작을 확인합니다. 아파치를 기동하고, 브라우저로 접속해서 동작을 확인하는 과정은 내장 모드와 동일합니다. 내장 모드 실행의 **6.4.2 지금까지 작업 확인하기**를 참고하기 바랍니다.

APPENDIX A

장고의 데이터베이스 연동

이 책의 본문에서는 SQLite 데이터베이스를 사용하여 장고 프로젝트 개발을 진행했습니다. 그런데 여러분의 서버에 또는 사용하고 있는 개발 환경에 다른 데이터베이스가 이미 설치되어 있다면 그 데이터베이스를 사용하여 장고 프로젝트를 개발할 수 있습니다.

사실 SQLite 데이터베이스는 작고 가벼워서 사용하기 쉽다는 장점이 있지만, 요즘과 같은 빅데이터 시대에 대규모 프로젝트에서는 거의 사용하지 않습니다. 즉, SQLite는 메모리, 디스크 등 서버 자원을 적게 차지하는 장점이 있는 반면, 멀티 프로세스 환경에서의 동시 접근 처리 능력 등이 약해 보통은 테스트 용도나 소규모의 프로젝트 또는 임베디드 환경에 주로 사용됩니다.

반면 큰 규모의 프로젝트에서는 다른 엔터프라이즈급 데이터베이스를 주로 사용하게 되는데, 장고에서는 SQLite 이외에도 MySQL, PostgreSQL, Oracle 데이터베이스를 공식적으로 지원하고 있습니다. 장고에서 이런 데이터베이스를 연동하는 방법을 소개하겠습니다.

MySQL 데이터베이스 연동

장고에서는 MySQL 5.0.3 이상의 버전을 지원합니다.

연동 드라이버 설치

파이썬에 MySQL 데이터베이스 연동을 위한 연동 드라이버 모듈은 여러 가지가 존재하는데, 장고는 다음과 같은 3가지 연동 드라이버를 지원합니다. 아래 3가지 중에서 자신에게 맞는 드라이버를 장고가 설치된 서버에 설치해줍니다.

- **MySQLdb** : 가장 오랫동안 사용된 드라이버로, 그만큼 안정되어 있으나 Python 3을 지원하지 않는다는 단점이 있습니다.
- **Mysqlclient** : 위의 MySQLdb를 개선한 패키지로, Python 3.3 이상의 버전도 지원하고 있어 장고에서 추천하는 드라이버입니다.
- **MySQL Connector/Python** : MySQL 개발사인 오라클에서 제공하고 있는 드라이버로, 장고의 공식 도큐먼트에는 장고의 최신 버전은 지원하지 않을 수도 있다고 되어 있는데, 필자가 확인한 바로는 장고 1.7 버전에서 잘 동작하고 있습니다.

settings.py 파일 수정

장고의 settings.py 파일의 DATABASES 항목에 MySQL 데이터베이스를 사용한다고 지정해줍니다.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'dj_mysql',
        'USER': 'root',
        'PASSWORD': 'rootpswd',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

각 항목별 의미는 다음과 같습니다.

표 A-1 MySQL 연동 시 정의하는 항목 – 디폴트 계정 사용

항목	항목 설명
ENGINE	MySQL 엔진을 사용한다는 의미로, django.db.backends.mysql처럼 장고에서 지정한대로 작성하면 됩니다.

NAME	장고에서 사용할 MySQL 내의 데이터베이스 이름을 입력합니다. 즉, 다음과 같은 MySQL 명령으로 데이터베이스를 만들어 사용하면 됩니다. `mysql> create database dj_mysql;`
USER	장고에서 MySQL 데이터베이스에 연결 시 사용할 유저 이름입니다. 즉, 다음과 같은 명령에 사용하는 유저 이름(여기서는 root)을 입력하면 됩니다. `$ mysql -uroot -p`
PASSWORD	장고에서 MySQL 데이터베이스에 연결 시 사용할 유저 이름에 대한 패스워드를 입력합니다. 즉, 다음과 같은 명령에 사용하는 패스워드입니다. `$ mysql -uroot -prootpswd`
HOST	MySQL 서버가 실행되고 있는 머신의 IP 주소를 입력합니다. 장고와 동일 머신에서 MySQL 서버가 실행되고 있으면, 127.0.0.1이라고 써줘도 되고, 생략해도 됩니다.
PORT	MySQL 서버에 접속할 때 사용하는 포트번호입니다. MySQL 서버의 디폴트 포트번호인 3306을 그대로 사용하는 경우는 생략해도 됩니다.

참고로, 다음과 같은 MySQL 명령을 사용하여 새로운 MySQL 계정과 비밀번호를 만들어서 사용할 수도 있습니다.

```
$ mysql -u root -p
mysql> create database dj_test;
mysql> grant all privileges on dj_test.* to 'django'@'localhost' identified by 'pswd';
```

이 경우에는 DATABASES 항목에 다음과 같이 입력합니다.

표 A-2 MySQL 연동 시 정의하는 항목 – 새로운 계정 사용

항목명	NAME	USER	PASSWORD
입력할 내용	dj_test	django	pswd

변경사항 장고에 반영하기

데이터베이스와 관련하여 변경사항이 발생하면, 다음과 같은 명령으로 장고에 반영해줘야 합니다. 당연히 manage.py 파일이 있는 디렉토리로 이동한 후에 명령을 실행해야겠죠?

```
$ python manage.py migrate
```

또한 데이터베이스 엔진을 새로 설정해서 초기화된 상태이므로, 장고 Admin 사이트에 로그인하기 위한 관리자를 새로 만들어줘야 합니다.

```
$ python manage.py createsuperuser
```

작업 확인하기

아래 명령으로 장고 runserver를 실행 시 에러가 나지 않는다면 데이터베이스 연동이 정상적으로 동작하는 것입니다.

```
$ python manage.py runserver 0.0.0.0:8000
```

좀 더 확실히 확인하고 싶다면, MySQL 데이터베이스에 테이블들이 잘 생성되었는지 확인하면 됩니다. 장고 Admin 사이트에서 테이블들이 잘 보이는지 확인하기 위해 브라우저로 Admin 사이트에 접속합니다. 아래는 Admin 사이트 주소의 한 가지 예시입니다.

```
http://192.168.56.101:8000/admin
```

> **NOTE_** 여기에서 살펴본 내용들은 본문에서 SQLite 데이터베이스 연동 시 설명한 내용들입니다. 복습의 개념으로 **3.4 프로젝트 뼈대 만들기**를 참고하기 바랍니다.

PostgreSQL 데이터베이스 연동

장고에서는 PostgreSQL 8.4 이상의 버전을 지원합니다.

연동 드라이버 설치

파이썬에서 PostgreSQL 데이터베이스 연동을 위한 연동 드라이버 모듈은 여러 가지가 존재하는데, 장고는 psycopg 패키지를 추천하므로 장고가 설치된 서버에 **psycopg** 최신 버전을 설치합니다.

settings.py 파일 수정

장고의 settings.py 파일의 DATABASES 항목에 PostgreSQL 데이터베이스를 사용한다고 지정해줍니다.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'dj_postgres',
        'USER': 'postgres',
        'PASSWORD': 'postgrespswd',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

각 항목별 의미는 다음과 같습니다.

표 A-3 PostgreSQL 연동 시 정의하는 항목 – 디폴트 계정 사용

항목	항목 설명
ENGINE	PostgreSQL 엔진을 사용한다는 의미로, django.db.backends.postgresql_psycopg2처럼 장고에서 지정한대로 작성하면 됩니다.
NAME	장고에서 사용할 PostgreSQL 내의 데이터베이스 이름을 사용합니다. 즉, 다음과 같은 PostgreSQL 명령으로 데이터베이스를 만들어 사용하면 됩니다. `$ createdb dj_postgres`
USER	장고에서 PostgreSQL 데이터베이스에 연결 시 사용할 유저 이름입니다. PostgreSQL 데이터베이스는 기본 사용자로 postgres 유저를 사용합니다. 즉, 다음 명령과 같이 쉘에서 PostgreSQL 데이터베이스의 기본 유저인 postgres로 로그인할 때 사용하는 유저 이름을 입력합니다. `$ su - postgres`

PASSWORD	장고에서 PostgreSQL 데이터베이스에 연결 시 사용할 유저 이름에 대한 패스워드를 입력합니다. 즉, 다음 명령과 같이 쉘에서 postgres 유저로 로그인할 때 사용하는 패스워드를 입력해주면 됩니다. `$ su - postgres`
HOST	PostgreSQL 서버가 실행되고 있는 머신의 IP 주소를 입력합니다. 장고와 동일 머신에서 MySQL 서버가 실행되고 있으면 127.0.0.1이라고 입력하거나 생략합니다.
PORT	PostgreSQL 서버에 접속할 때 사용하는 포트번호입니다. PostgreSQL 서버의 디폴트 포트 번호인 5432를 그대로 사용하는 경우는 생략해도 됩니다.

참고로 다음과 같은 PostgreSQL 명령을 사용하여, PostgreSQL 데이터베이스 내에 새로운 계정과 비밀번호를 만들어서 사용할 수도 있습니다.

```
$ su - postgres
$ createdb dj_test
$ createuser django -P
Enter password for new role: pswd
Enter it again: pswd
$
```

이 경우에는 DATABASES 항목에 다음과 같이 입력합니다.

표 A-4 PostgreSQL 연동 시 정의하는 항목 – 새로운 계정 사용

항목명	NAME	USER	PASSWORD
입력할 내용	dj_test	django	pswd

장고에 반영 및 확인하기

이 이후에 데이터베이스 변경사항을 장고에 반영하는 방법이나, 작업 확인하기는 MySQL과 동일하므로, 앞에서 설명한 **MySQL 데이터베이스 연동**을 참고하기 바랍니다.

Oracle 데이터베이스 연동

장고에서는 Oracle Database Server 9i 이상의 버전을 지원합니다.

연동 드라이버 설치

파이썬에서 Oracle 데이터베이스 연동을 위한 연동 드라이버 모듈인 **cx_Oracle** 패키지를 최신 버전으로 설치합니다. 장고에서는 cx_Oracle 5.0.1 버전 이상을 추천하고 있습니다.

settings.py 파일 수정

장고의 settings.py 파일의 DATABASES 항목에 Oracle 데이터베이스를 사용한다고 지정해줍니다.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.oracle',
        'NAME': 'xe',
        'USER': 'scott',
        'PASSWORD': 'tiger',
        'HOST': '',
        'PORT': '',
    }
}
```

각 항목별 의미는 다음과 같습니다.

표 A-5 Oracle 연동 시 정의하는 항목 – 디폴트 계정 사용

항목	항목 설명
ENGINE	Oracle 엔진을 사용한다는 의미로, django.db.backends.oracle처럼 장고에서 지정한대로 입력합니다.
NAME	장고가 클라이언트 입장에서 접속할 Oracle 데이터베이스 서버 인스턴스의 이름, 즉 SID(Service ID)를 입력하면 됩니다. 보통은 오라클 설치 시 SID로 xe를 사용하고 있습니다.

USER	장고에서 Oracle 데이터베이스에 연결 시 사용할 유저 이름입니다. $ sqlplus scott/tiger처럼 데이터베이스에 접속할 때, 유저 이름은 scott입니다.
PASSWORD	장고에서 Oracle 데이터베이스에 연결 시 사용할 유저 이름에 대한 패스워드를 입력합니다. $ sqlplus scott/tiger처럼 데이터베이스에 접속할 때, 패스워드는 tiger입니다.
HOST	공란으로 둡니다. 공란으로 두면 오라클의 설정 파일인 tnsnames.ora 파일을 사용합니다. $ORACLE_HOME/network/admin/tnsnames.ora 파일에는 데이터베이스 서버의 HOST, PORT 항목이 정의되어 있습니다.
PORT	공란으로 둡니다. 위와 동일하게 공란으로 두면 오라클의 설정 파일인 tnsnames.ora 파일을 사용합니다.

참고로, 다음과 같은 Oracle 데이터베이스 명령을 사용하여 Oracle 데이터베이스 내에 새로운 계정과 비밀번호를 만들어서 사용할 수도 있습니다.

```
$ sqlplus system
SQL> CREATE TABLESPACE ts_django DATAFILE '$ORACLE_HOME/rdbms/dbs/ts_django.dbf' SIZE
40M ONLINE;
SQL> CREATE USER django IDENTIFIED BY pswd DEFAULT TABLESPACE ts_django;
SQL> GRANT connect, resource TO django;
```

이 경우에는 DATABASES 항목에 다음과 같이 입력합니다.

표 A-6 Oracle 연동 시 정의하는 항목 – 새로운 계정 사용

항목명	NAME	USER	PASSWORD
입력할 내용	xe	django	pswd

장고에 반영 및 확인하기

이 이후에 데이터베이스 변경사항을 장고에 반영하는 방법이나, 작업 확인하기는 MySQL과 동일하므로, 앞에서 설명한 **MySQL 데이터베이스 연동**을 참고하기 바랍니다.