**배포하기!**

이번 장은 따라오기 조금 힘들 수도 있어요. 끝까지 따라와주세요. 배포는 웹사이트 개발의 가장 중요한 부분이에요. 튜토리얼 중간에 이번 장이 있는 이유는 여러분이 코치가 여러분의 웹사이트를 온라인에서 볼 수 있게 도와주기 위해서에요. 시간이 모자르더라도 튜토리얼을 끝낼 수 있어요.

지금까지는 웹사이트를 내 컴퓨터에서만 볼 수 있었어요. 이제 어떻게 웹사이트를 배포하는지 배워봅시다. 배포(deploy)는 애플리케이션을 인터넷에 올려놓아 다른 사람들도 볼 수 있게 해주는 것 말해요. :)

앞에서 배웠듯이, 웹사이트는 서버라는 곳에 들어 갑니다. 인터넷에는 많은 종류의 서버 제공 업체들이 있어요. 우리는 이 중에 비교적 배포 과정이 간단한 [PythonAnywhere](https://pythonanywhere.com/" \t "_blank)을 사용할 거에요. PythonAnywhere는 방문자가 아주 많지 않은 소규모 어플리케이션으로 무료로 사용할 수 있답니다. 당연히 지금 우리가 만드는 것도 해당되지요.

우리가 사용할 다른 서비스는 [GitHub](https://www.github.com/" \t "_blank)라는 코드 호스팅 서비스입니다. 프로그래머들이 애용하는 곳들 중 하나로 대부분의 모든 프로그래머가 GitHub 계정을 가지고 있다고 봐도 될 거예요. 이제 여러분도 하나 만들 때가 되었어요!

우리는 GitHub를 작성한 코드를 PythonAnywhere에 넣고 빼기 위한 디딤돌로 사용할 거에요. 즉, 코드를 GitHub에 저장하고 GitHub에 저장된 코드를 PythonAnywhere에서 가져다가 작동하는 방법이에요.

**Git**

Git은 많은 프로그래머들이 사용하고 있는 "버전 제어 시스템(version control system)"이에요. 이 소프트웨어는 변경 내용을 추적할 수 있어 나중에 특정 버전을 다시 호출 할 수 있어요. 마이크로소프트 워드에 있는 "변경 내용 추적하기(track changes)"와 같은 기능이지만 이보다 훨씬 더 강력합니다.

**Git 설치하기**

이미 설치를 끝냈다면, 다시 할 필요가 없어요. 이 부분을 넘기고 Git 저장소를 만드는 것부터 시작하세요.

**윈도우**

Git은 [git-scm.com](https://git-scm.com/)에서 다운로드 할 수 있습니다. 다운받은 파일을 설치하는 것은 참 쉬어요. 딱 한 단계만 빼고 기본 설정대로 "다음 다음 다음"을 계속 눌러주면 쉽게 설치할 수 있어요. ; 주의해야할 단계는 다섯 번째 "PATH 환경 설정(Adjusting your PATH environment)" 화면이에요. "윈도우 커맨드라인에서 Git과 유닉스 도구를 실행(Run Git and associated Unix tools from the Windows command-line)"을 선택하세요. (옵션 아래쪽에 있어요.) 나머지는 모두 기본 설정대로 해도 됩니다. 윈도우 스타일, 유닉스 스타일로 라인엔딩 커밋하기(commit Unix-style line endings) 를 체크하는 것도 좋습니다.

**MacOS**

[git-scm.com](https://git-scm.com/)에서 맥용 설치파일을 다운받은 다음 실행해 나오는 지시 사항에 따라 설치하시면 됩니다.

**리눅스**

설치가 안되었다면, 패키지 관리자로 git 설치를 해야해요. 다음과 같이 해보세요:

sudo apt-get install git

# 또는

sudo yum install git

# 또는

sudo zypper install git

**Git 저장소 만들기**

Git은 코드 저장소(줄여서 "repo"라고 합니다.)에 특정한 파일들 집합의 변화를 추적하여 관리합니다. 이제 프로젝트를 시작해 볼까요? 콘솔창을 열고 djangogirls 디렉토리에서 아래 명령어들을 실행하세요. 아래 명령 중에서 Your Name 대신 자신의 이름을, you@example.com 대신에 내 이메일 주소를 입력하세요.

저장소를 초기화 하기 전에 여러분의 현재 작업 디렉토리가 어디인지 꼭 확인하세요. 맥OS나 Linux라면 pwd 명령으로, 윈도우라면 cd 명령어를 실행하면 알 수 있을 거에요. 반드시djangogirls 폴더에서 해야합니다.

$ git init

Initialized empty Git repository in ~/djangogirls/.git/

$ git config --global user.name "Your Name"

$ git config --global user.email you@example.com

git 저장소 초기화는 프로젝트를 시작할 때 딱 한번만 필요합니다. (또한 두 번째, 세 번째 명령인 username과 email 등록 명령은 계속 적용되기 때문에 이 계정에서는 다시 입력할 필요가 없습니다.)

Git은 이 디렉토리에 모든 파일들과 폴더들의 변경점을 추적할 거에요. 무시(ignore)하도록 지정한 파일들을 제외하고는 말이죠. 기본 디렉토리에다 .gitignore라는 파일을 만들어서 특정 파일이나 폴더를 추적하지 않게 할 수 있습니다. 에디터를 열어 아래 내용을 넣어주세요. :

\*.pyc

\_\_pycache\_\_

myvenv

db.sqlite3

.DS\_Store

그리고 "djangogirls" 폴더의 맨 위에다 .gitignore 라는 파일로 저장하세요.

파일명 앞에 마침표로 시작하는 것이 중요합니다! 꼭 붙여주세요. 만약 파일을 만드는 것이 어렵다면 (예를 들어 맥은 파인더에서 마침표로 시작하는 파일을 생성하는 걸 좋아하지 않아요) "다른 이름으로 저장(Save As)" 기능을 사용하세요. 대부분 이렇게 하면 될 거에요.

git add하기 전이나 변경된 것이 있는지 잘 모를 때마다 git status 명령어를 사용하는 것은 좋은 방법이에요. 잘못된 파일이 추가된 경우, 이를 멈출 수 있게 도움을 주어요. git status 명령은 미추적/수정된/스테이지된 파일들, 브랜치 상태와 그 외 많은 정보들을 보여줍니다. 실행하면 아래와 비슷하게 나타날거에요. :

$ git status

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

blog/

manage.py

mysite/

nothing added to commit but untracked files present (use "git add" to track)

자 이제 우리가 만든 코드들을 저장소에 넣어봅시다. 콘솔창에 가서 다음 명령어를 실행하세요. :

$ git add --all .

$ git commit -m "My Django Girls app, 첫번째 커밋"

[...]

13 files changed, 200 insertions(+)

create mode 100644 .gitignore

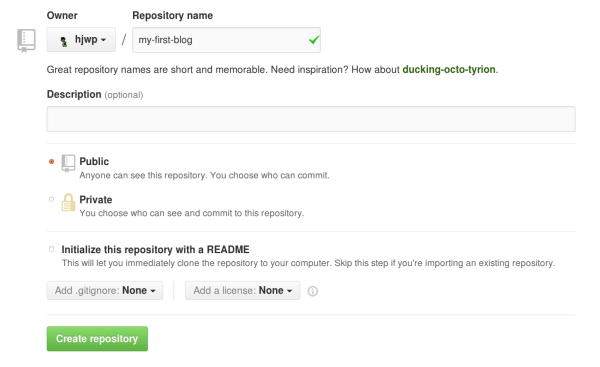
[...]

create mode 100644 mysite/wsgi.py

**GitHub에 코드 넣기**

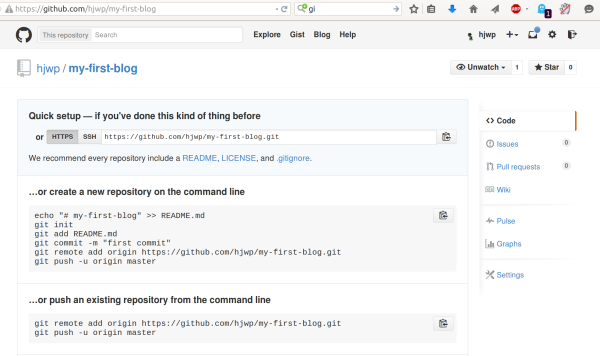
[GitHub.com](https://www.github.com/)에서 새로운 무료 계정을 만들세요. (워크샵 전에 미리 가입했다면 더 좋죠!)

그 다음 새 저장소(new repository)를 "my-first-blog"라는 이름으로 생성하세요. "초기화시 README 파일 만들기(initialize with a README)" 체크박스는 체크안 한 상태로 두세요. .gitignore 옵션도 비어있는 상태(위에서 직접 만들었죠?)로 두세요. 라이센스도 None으로 두세요.



**주의** 저장소 이름 my-first-blog는 중요합니다. -- 물론 원하는대로 이름을 바꿀 수 있지만, 매번마다 변경해서 사용해야하니 정말 불편할 거에요. 가급적이면 my-first-blog라는 저장소 이름을 그대로 사용하는 것이 편할 거에요.

다음 화면에서 저장소 복제 URL(repo's clon URL) 을 볼 수 있을 거에요. "HTTPS" 버전을 선택해서 그 주소를 복사하세요. 그리고 터미널에다 붙여넣으세요.



이제 내 컴퓨터의 Git 저장소를 GitHub에 있는 저장소로 연결해야합니다.

콘솔에 다음과 같이 입력하세요 (<your-github-username>에 꺽쇠괄호(<>)를 떼고 내 Github 유저네임을 입력하세요.

$ git remote add origin https://github.com/<your-github-username>/my-first-blog.git

$ git push -u origin master

아래와 같은 화면이 나오면 GitHub 유저네임과 암호를 입력하세요.

Username for 'https://github.com': 여러분의GitHub유저네임

Password for 'https://hjwp@github.com': 여러분의GitHub암호

Counting objects: 6, done.

Writing objects: 100% (6/6), 200 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/hjwp/my-first-blog.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

이제 여러분의 코드가 GitHub에 저장됐어요! 가서 확인해보세요! [Django](https://github.com/django/django" \t "_blank), [Django Girls Tutorial](https://github.com/DjangoGirls/tutorial" \t "_blank)뿐만 아니라 수많은 훌륭한 오픈소스 소프트웨어 프로젝트들이 GitHub에서 코드를 호스팅하고 있어요. 방금 여러분이 한 것처럼 말이죠. :)

**PythonAnywhere에 블로그 설정하기**

이미 PythonAnywhere 계정이 있으면, 이 부분을 다시 할 필요가 없어요.

다음 PythonAnywhere에서 무료 계정인 "초보자(Beginner)"로 회원가입 하세요.

* [www.pythonanywhere.com](https://www.pythonanywhere.com/)

: 여러분의 유저네임을 정할 때 블로그 주소의 일부가 여러분의 유저네임으로 되는 것을 염두하고 정하세요. 예를 들어 유저네임이 uourusername이라면 여러분의 블로그 URL은yourusername.pythonanywhere.com이 된답니다.

**GitHub에서 PythonAnywhere로 코드 가져오기**

PythonAnywhere에 가입하면 대시보드 또는 "콘솔(Consoles)" 페이지를 볼 수 있을 거에요. "배시(Bash)" 콘솔로 시작하는 옵션을 선택하세요. -- PythonAnywhere 버전의 콘솔로 PC에 있는 커맨드라인 같은 것이라 생각해도 됩니다.

PythonAnywhere는 리눅스 기반으로 윈도우의 커맨드라인 창과 조금 다르게 보일 수 있어요.

GitHub에 있는 코드를 끌어와 저장소를 "복제"해 PythonAnywhere로 탑재해 볼게요. PythonAnywhere에 있는 콘솔에 다음과 같이 입력하세요. (<your-github-username>대신 내 GitHub 유저네임을 입력하는 것을 잊지 마세요. :)

$ git clone https://github.com/<your-github-username>/my-first-blog.git

이렇게 PythonAnyWhere에 있는 코드 사본을 끌어올 거에요. tree my-first-blog를 입력해 확인하세요. :

$ tree my-first-blog

my-first-blog/

├── blog

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── migrations

│ │ ├── 0001\_initial.py

│ │ └── \_\_init\_\_.py

│ ├── models.py

│ ├── tests.py

│ └── views.py

├── manage.py

└── mysite

├── \_\_init\_\_.py

├── settings.py

├── urls.py

└── wsgi.py

**PythonAnywhere에서 가상환경(virtualenv) 생성하기**

PythonAnywhere에서도 내 컴퓨터에 있는 것과 같이 작동할 수 있게 가상 환경(virtualenv)을 생성할 수 있어요. 배시 콘솔(Bash console)에 다음과 같이 입력하세요. :

$ cd my-first-blog

$ virtualenv --python=python3.4 myvenv

Running virtualenv with interpreter /usr/bin/python3.4

[...]

Installing setuptools, pip...done.

$ source myvenv/bin/activate

(mvenv) $ pip install django whitenoise

Collecting django

[...]

Successfully installed django-1.8.2 whitenoise-2.0

The pip install은 설치가 될 때까지 인내심이 조금 필요해요. 하지만 몇 분 안에 완료될 거니 조금만 기다려보세요. 만약 5분 이상 시간이 지났다면, 무엇인가 잘못된거니 코치에게 물어보세요.

**정적 파일 모으기**

"백색소음"을 들어본 적이 있나요? 이 것은 "정적 파일"로 불리는 것들을 제공하는 프로그램이에요. 정적 파일이란 HTML, CSS와 같이 정기적인 수정이 없거나 프로그래밍 코드를 실행하지 않는 파일을 말해요. 서버에서 정적 파일은 컴퓨터와 다르게 작동하기 때문에 정적 파일들을 제공하기 위해서 "백색소음"과 같은 프로그램이 필요해요.

정적 파일은 튜토리얼 뒷부분에 나오는 CSS 부분에서 좀 더 자세히 알아볼게요.

이제 서버에서 추가명령인 collectstatic을 실행하세요. 이 명령은 장고가 서버에 있는 모든 정적 파일들을 모으는 것을 지시해요. 지금 이 파일들이 관리자 사이트를 예쁘게 만들어주는 것이지요.

(mvenv) $ python manage.py collectstatic

You have requested to collect static files at the destination

location as specified in your settings:

/home/edith/my-first-blog/static

This will overwrite existing files!

Are you sure you want to do this?

Type 'yes' to continue, or 'no' to cancel: yes

"yes"라고 입력하고 메시지들이 나오게 내버려두세요! 화면 가득히 텍스트가 출력되어 흘러나올건데 혹시 이런 걸 좋아하는 분 있나요? 저는 텍스트가 나타날 때마다 항상 "브릅, 브릅 브릅..." 같은 소리가 나도록 만들어 놓았어요. 멋지죠?

Copying '/home/edith/my-first-blog/mvenv/lib/python3.4/site-packages/django/contrib/admin/static/admin/js/actions.min.js'

Copying '/home/edith/my-first-blog/mvenv/lib/python3.4/site-packages/django/contrib/admin/static/admin/js/inlines.min.js'

[...]

Copying '/home/edith/my-first-blog/mvenv/lib/python3.4/site-packages/django/contrib/admin/static/admin/css/changelists.css'

Copying '/home/edith/my-first-blog/mvenv/lib/python3.4/site-packages/django/contrib/admin/static/admin/css/base.css'

62 static files copied to '/home/edith/my-first-blog/static'.

**PythonAnywhere에서 데이터베이스 생성하기**

컴퓨터와 서버가 다른 점이 또 하나 있어요. 바로 다른 데이터베이스를 사용한다는 점이에요. 그래서 사용자 계정과 글은 서버와 여러분의 컴퓨터와 다를 수 있어요.

지난 번 내 컴퓨터에서 했던 것과 같이 서버에서도 데이터베이스를 초기화 할 거에요. migrate와createsuperuser를 사용하세요. :

(mvenv) $ python manage.py migrate

Operations to perform:

[...]

Applying sessions.0001\_initial... OK

(mvenv) $ python manage.py createsuperuser

**web app으로 블로그 배포하기**

이제 우리의 코드는 PythonAnywhere에 있고 우리의 가상환경(virtualenv)도 준비가 되었으며, 정적 파일들도 모아져 있고, 데이터베이스도 초기화되었네요. 이제 우리는 web app으로 배포할 준비가 되었어요.

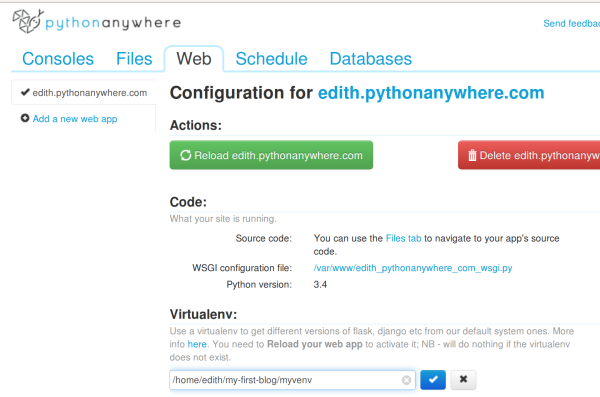
로고를 클릭해 PythonAnywhere 대시보드로 와서 **Web**을 클릭하고 **Add a new web app**를 선택하세요.

도메인 이름을 확정한 후, 대화창에 **수동설정(manual configuration)** ("Django"옵션이 *아니에요.*) 을 클릭하세요. 다음, **Python 3.4**을 선택하고 다음(Next)를 클릭하면 마법사가 종료됩니다.

" Django"가 아니라 꼭 "수동설정(Manual configuration)"을 선택하세요. 기본 PythonAnywhere Django 설정을 위해서는 이렇게 하는 것이 더 좋아요. ;-)

**가상환경(virtualenv) 설정하기**

이제 우리는 webapp을 위한 PythonAnywhere 설정 화면을 볼 수 있어요. 서버에 있는 앱에 변경사항이 생기면 필요한 화면이에요.



"가상환경(Virtualenv)" 섹션에서 "가상환경 경로를 입력해주세요(Enter the path to a virtualenv)"라고 써져있는 빨간색 글자를 클릭하고 /home/<your-username>/my-first-blog/myvenv/ 라고 입력합니다. 이동 경로 저장을 하려면 파란색 박스에 체크 표시를 하고 클릭하세요.

**주의**: 가상환경 경로를 적당한 것으로 변경하세요. 실수를 하면 PythonAnywhere가 작은 경고 메시지를 보여줄 거에요.

**WSGI 파일 설정하기**

장고는 "WSGI 프로토콜"을 사용해 작동합니다. 이 프로토콜은 파이썬을 이용한 웹사이트를 서비스하기 위한 표준으로 PythonAnywhere에서도 지원합니다. WSGI 설정을 파일을 수정해 우리가 만든 장고 블로그를 PythonAnywhere에서 인식하게 해봅시다.

"WSGI 설정 파일(WSGI configuration file)" 링크(페이지의 상단 근처에 있는 "코드(Code)" 섹션에 있습니다. -- 아마 /var/www/<your-username>\_pythonanywhere\_com\_wsgi.py와 같이 나타나있을 것입니다.)를 클릭하면 에디터를 볼 수 있을 것입니다.

모든 내용을 삭제하고 아래 내용을 넣으세요. :

import os

import sys

path = '/home/<your-username>/my-first-blog' # 여러분의 유저네임을 여기에 적어주세요.

if path not in sys.path:

sys.path.append(path)

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'mysite.settings'

from django.core.wsgi import get\_wsgi\_application

from whitenoise.django import DjangoWhiteNoise

application = DjangoWhiteNoise(get\_wsgi\_application())

<your-username>에 내 유저네임을 넣는 것을 잊지 마세요.

이 파일은 PythonAnywhere에게 작동 할 우리의 web app 위치를 알려주고 장고 세팅 파일 이름을 알려주는 역할을 하죠. 또 "백색소음" 정적 파일 툴을 설정해줍니다.

**저장(Save)**을 누르고 **웹(Web)** 탭을 누릅니다.

다 되었어요! 큰 녹색 **다시 불러오기(Reload)** 버튼을 누르면 여러분은 여러분의 어플리케이션을 볼 수 있을 거에요. 여러분은 페이지 최 상단에 가는 링크를 발견할 수 있을 거에요.

**디버깅 팁**

본인의 사이트에 접속할 때 오류가 보이면, 제일 먼저 **error log(오류 로그)**에서 디버깅 정보를 찾아보세요. PythonAnywhere [Web tab](https://www.pythonanywhere.com/web_app_setup/)에서 이 링크를 찾을 수 있어요. 여기서 오류 메세지가 있는지 확인하세요. 대부분의 가장 최근의 오류 메세지는 맨 하단에 있어요. 일반적으로 많이 일어나는 문제들은 아래와 같습니다.

* virtualenv를 생성하고 활성화할 때, Django를 설치할 때, Collestac을 돌릴 때, 데이터베이스를 설치할 때 같은 각 단계를 위한 콘솔 작업을 할 때 한 단계를 빼먹는 경우
* web tab에서 virtualenv 경로를 쓸 때 오타가 났을 때 - 이런 경우 빨간 작은 에러 메시지가 뜰 것입니다.
* WSGI 설정 파일에 실수가 있을 때 -- my-first-blog-folder 폴더의 경로를 올바르게 입력했었나요?
* Web app에서 그랬듯이 virtualenv에도 동일한 파이썬 버전을 선택했나요? 둘다 모두 3.4 버전이어야해요.
* [PythonAnywhere 위키에서 일반적인 디버깅 팁](https://www.pythonanywhere.com/wiki/DebuggingImportError)을 확인할 수 있어요.

기억하세요, 여러분의 코치들은 여러분들에게 도움을 주기 위해 여기 있답니다!

**내 웹사이트가 나왔어요!**

여러분 사이트의 기본 페이지(default page) 가 PC와 같이 "Welcome to Django"라는 메시지를 보여줘야해요. URL 뒤에 /admin/를 붙여서 다시 접속해 보세요. 이제 관리자 사이트를 볼 수 있어요. 서버에서 유저네임과 암호를 입력해서 로그인하면 새 글을 작성할 수 있음을 알 수 있어요.

이제까지 수고한 모두에게 *큰* 격려를 해주세요! 서버 배포는 웹 개발 단계 중 가장 까다로운 부분 중 하나이기 때문에 작업 완성까지 몇 일이 걸리기도 해요. 하지만 이제 우리 사이트는 실제 인터넷에서 접속이 가능해요. 쉽게 잘 해냈어요!