

TUTORIAL PROJECT



Django Tutorial—Build a Travel Blog with Goorm IDE and Bootstrap 4

By Hojun Lee, Suwon Choi
H.M Cho Translation

Author

Author Hojun Lee

CEO of Paul lab ICT institute of Science & Technology

CEO of Paul lab ICT computer school

CEO of Sado publishing company

Jeju Coding Basecamp (<http://www.jejuencodingcamp.com>)

Jeju Coding Hackathon

Jeju Coding Festival

Author Suwon Choi

Togathers web and app developer

suwoni-codelab.com blog

Preface

A student came to me when I was teaching software programming. Learning computer software (SW) became a burden to this student not only because of the poverty under which students live but also because of the expensive education fees. Moreover it was hard to study just by himself.

The use of software is part of our every days life (internet, word etc ...) therefore software programming should be easy to approach. Of course, it is not easy to find solutions as a professional SW education administrator however learning should not be a burden to students who just want to reach their dreams .

It should also be easy to learn. Thick and complicated books are not always the best choice for a beginner.

Those tutorials series are quick to read and you will improve your skills a lot , all by yourself.

Everyone can enjoy a great education with our tutorial, even someone with a poor background. So that you could enjoy learning and have the pleasure of solving problems. We will do our best to help you.

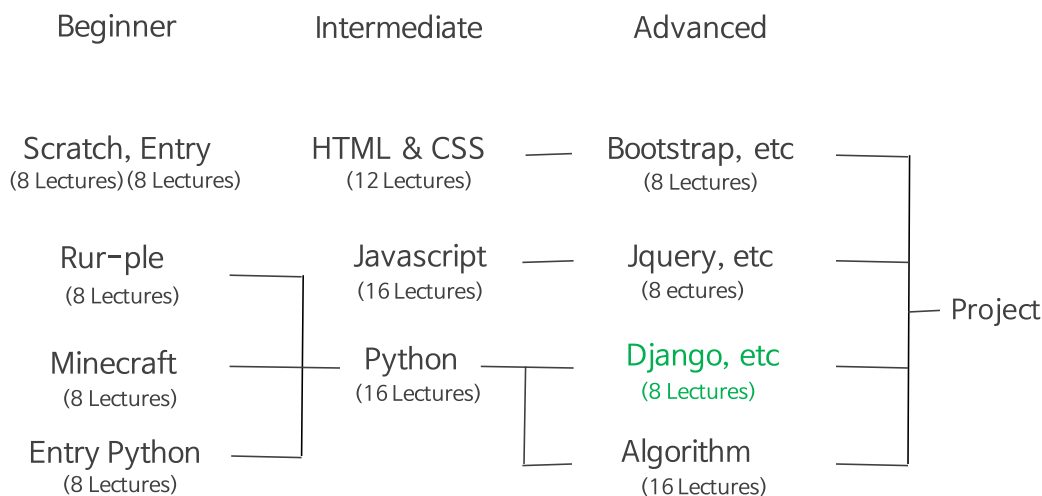
1. Check our lecture road map out on the next page and choose the one for you
2. Watch the video of the road map on our website to know what you need to learn (<http://paullab.co.kr/curriculum.html>)
3. Step through the tutorial project.

I hope that going through this steps will help you to reduce the ambiguity of software programming and reach your dreams.

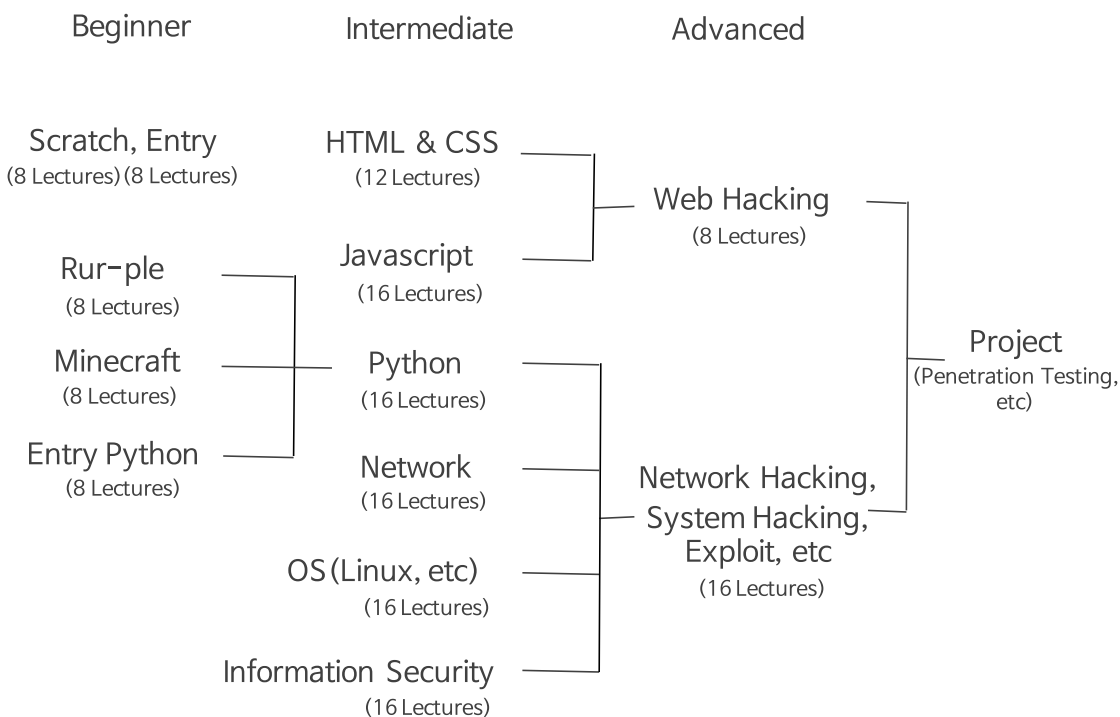
Hojun Lee

Road Map

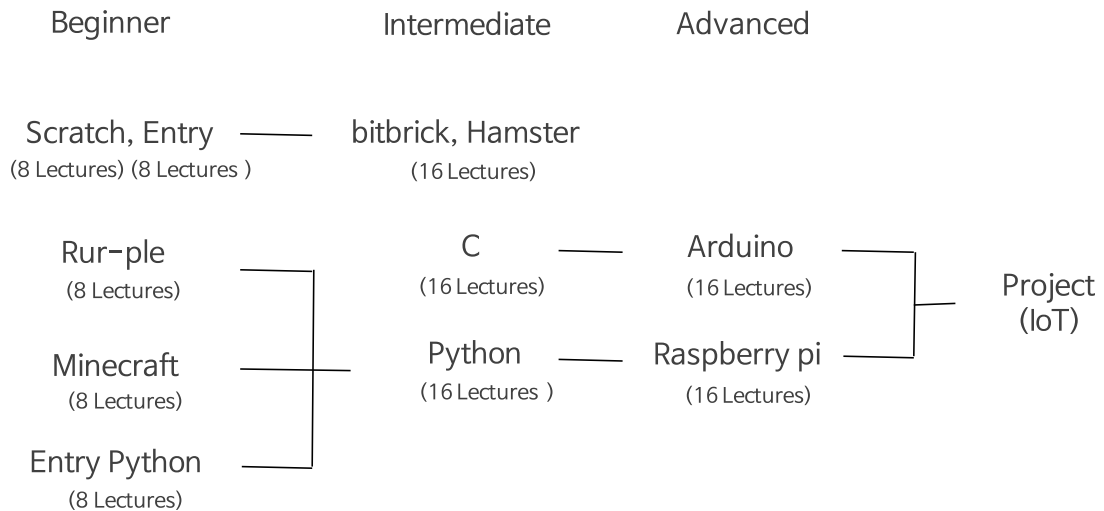
1. Web development(Front-end & Back-end : Full-Stack)



2. Information Security



3. Physical Computing



4. Advanced class

4.1 Big data Analysis in Python(8 Lectures)

4.2 Self Driving Car with Python(8 Lectures)

4.3 Information Security in Python(8 Lectures)

4.4 PyQt with Python GUI Programming(8 Lectures)

4.5 TensorFlow Tutorial(8 Lectures)

4.6 Django Tutorial(8 Lectures)

4.7 Android App Development(22 Lectures – Include java class)

4.8 Advanced Front-end programming (16 Lectures)

4.9 Advanced Back-end programming(16 Lectures)

Who should read this book

This book is recommended for those who want to program with Django. Django is a very popular framework among Python programmers. We hope it will be a good opportunity to expand your perspectives on web development

About the Book

This book is a tutorial on Django that will teach you to use it while creating a simple travel blog with Python. So instead of learning with Django in details,

We focused on a simple way (step by step) to create a web service.

If you just follow along, you will be able to create a web service. You will be able to download source code example files of each example that you will use.

The front-end programming part of this tutorial is from the Bootstrap section of the 'Travel Blog Tutorial'.

*** Download templates sources from the paullab.co.kr/templates.zip**

Next steps after the tutorial?

After learning Python and Django with us, you will be able to start creating an actual web service in the real world. When you're fixing an error, check the parts that you did not know and take a note of the solution.

If you are interested in Python itself, you may want to study about data analysis, visualization, and artificial intelligence.

Contents

1. Intro & Environment Setup	10
1.1 Introduction	11
1.2 Environment setting with Goorm IDE	15
2. A Main Page with Django	31
3. A Blog Page with Django	46
4. A Post Page with Django	59
5. Comment, Tag	76
6. Bootstrap, Deploy!	93
6.1 Bootstrap Introduction	94
6.2 Environment setting with Goorm IDE	95
6.3 Bootstrap Introduction	96

6.4 Bootstrap Basic	98
6.5 Finishing touches	105
6.6 Deployment	118

7. Summary	121
------------	-----

Chapter 1

Environment setting with Goorm IDE

1.1 Introduction

1.2 Environment setting with Goorm IDE

TUTORIAL 1

Introduction

Django is a free and open source web application framework written in Python which is used by Instagram, NASA, Disqus and etc.

A Framework is a supporting structure around which web services can be developed in a faster and easier way using a collection of given components.

Django started as a free and open source framework in 2005, and Django 2.0 released in December of 2017, and there is no Django 2.0 tutorial book in Korea yet. In this tutorial, we will build a simple blog in order to understand the whole process of website development.

If you want to know about django in detail refer to the documentation on the official website.

<https://www.djangoproject.com/> The official homepage provides an introduction to django, downloads, and supporting documentation.

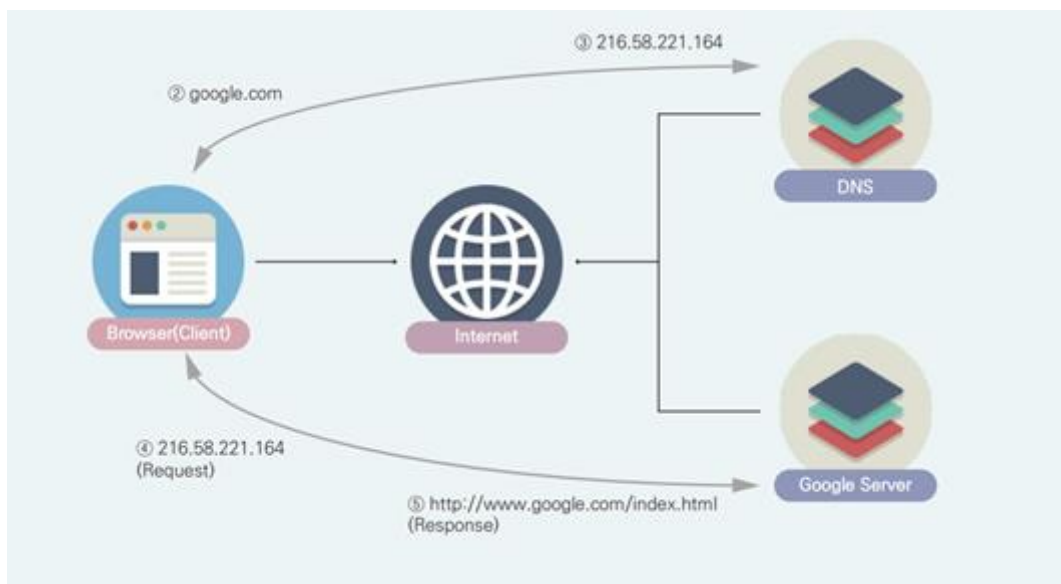
*<https://media.readthedocs.org/pdf/django/2.0.x/django.pdf>

1.1 Django installation & Environment setting(cont.)

Here is the description of how the internet works.

Below is a summary of the steps that internet is going through when you type “google” into your browser.

The web site we are going to create will work that way. We’ll be using different languages such as HTML, CSS, JavaScript, Python, and we’ll use the Bootstrap library on Django framework.



0. You type `www.google.com` into your browser

1. The Web browser gets the domain name (`www.google.com`) from the clients (you)

2. The Web browser looks for the IP address of the domain through the domain name server(DNS)

3. The Web browser requests a page to the web server through the IP address

4. The web server sends the requested page back to The web browser

5. The Web browser displays the page

1.1 Django installation & Environment setting(cont.)

We will use the Goorm IDE cloud service.

It is Linux based and it uses Ubuntu 14.

Most of you are used to use the mouse in the graphical GUI (graphic user interface) environment. (windows, mac)

Therefore the Linux CLI environment (command-line interface) might be unfamiliar to you as you command everything on a black screen. (no mouse)

Around the world The Linux server's market share represent at least 90% . That's the reason why we recommend to use it.

Now let's compare three different cloud service models:

The Cloud IDE we will use is PaaS model.

1. **SaaS (Software as a Service):** SaaS is the most user-friendly service and it is a service that allows you to take advantage of application functions over the network.
2. **Platform as a Service (PaaS):** PaaS is a cloud service that comes with Windows and Linux-like operating systems and a development platform.
3. **IaaS (infrastructure as a service):** IaaS is a cloud infrastructure service. It is mostly used professionally.

1.1 Django installation & Environment setting(cont.)

I think the reason why you’ve chosen this tutorial is because you are fascinated by Python's powerful library, intuitive and excellent extensibility. So, now python is one of the most popular programming language.

Then what about framework? Below is a description of the python framework from Wikipedia. In addition to Django, there are some popular frameworks like Flask. Django is a full-stack web framework , whereas Flask is lightweight and extensible.

Depending on your project ,make the right choice among those frameworks.

Popular Full-Stack Frameworks

A web application may use a combination of a base HTTP application server, a storage mechanism such as a database, a template engine, a request dispatcher, an authentication module and an AJAX toolkit. These can be individual components or be provided together in a high-level framework.

These are the most popular high-level frameworks. Many of them include components listed on the [WebComponents](#) page.

Name	Latest version	Latest update date	description
 Django	1.11	2017-04-04	The Web framework for perfectionists (with deadlines). Django makes it easier to build better Web apps more quickly and with less code. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It lets you build high-performing, elegant Web applications quickly. Django focuses on automating as much as possible and adhering to the DRY (Don't Repeat Yourself) principle. See Django
 TurboGears	2.3.10	2016-12-04	the rapid Web development webframework you've been looking for. Combines  SQLAlchemy (Model) or  Ming (MongoDB Model),  Genshi (View),  Repoze and  Tosca Widgets. Create a database-driven, ready-to-extend application in minutes. All with designer friendly templates, easy AJAX on the browser side and on the server side, with an incredibly powerful and flexible Object Relational Mapper (ORM), and with code that is as natural as writing a function. After reviewing the  Documentation, check out  the Tutorials
 web2py	2.14.6	2016-05-10	* Python 2.6 to 2.7, Python 3.x friendly (compile but not tested no support yet) * All in one package with no further dependencies. Development, deployment, debugging, testing, database administration and maintenance of applications can be done via the provided web interface, but not required. * web2py has no configuration files, requires no installation, can be run off a USB drive. * web2py uses Python for the Model, View and the Controller * Built-in ticketing system to manage errors * Internationalization engine and pluralisation, caching system * Flexible authentication system (LDAP, MySQL, janrain etc) * NIX(Linux, BSD), Windows, Mac OSX, tested on EC2, Webfaction * works with MySQL, PostgreSQL, SQLite , Firebird, Oracle, MSSQL and the Google App Engine via an ORM abstraction layer. * Includes libraries to handle HTML/XML, RSS, ATOM, CSV, RTF, JSON, AJAX, XMLRPC, Wiki markup. * Production ready, capable of upload/download of very large files * Emphasis on backward compatibility.

Popular Non Full-Stack Frameworks

These projects provide the base "application server", either running as its own independent process, upon Apache or in other environments. On many of these you can then introduce your own choice of templating engines and other components to run on top, although some may provide technologies for parts of the technology stack.

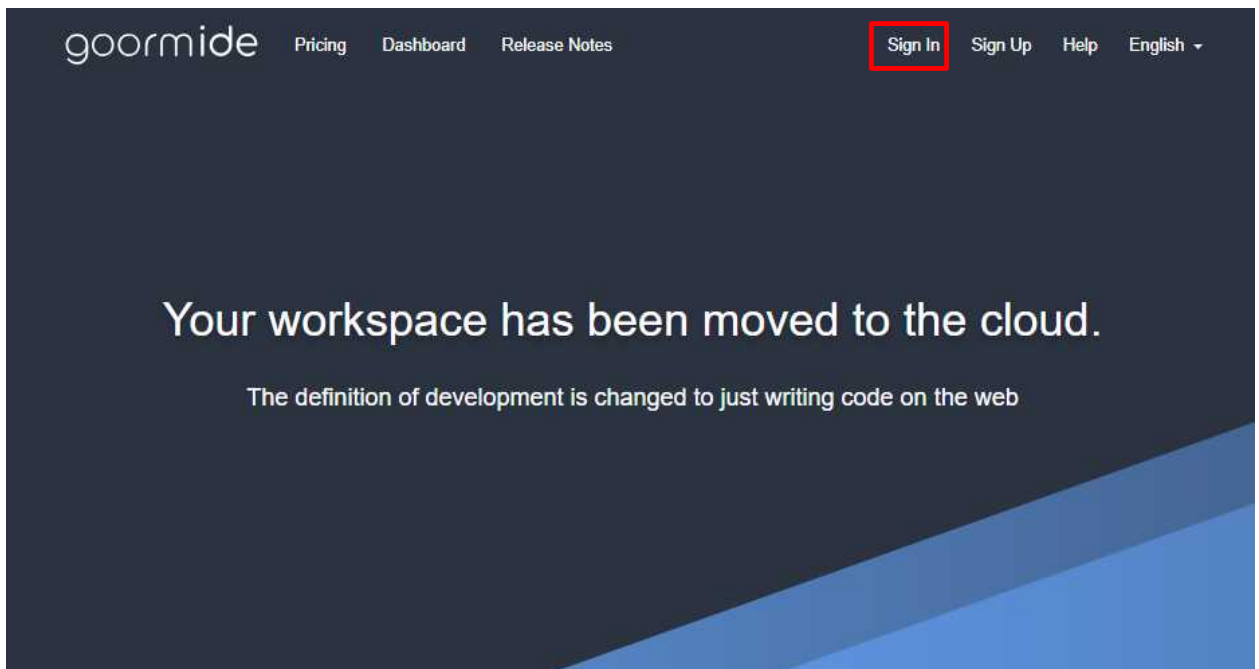
- »  Bottle (0.12.13 Released 2017-01-09) is a fast and simple micro-framework for small web-applications. It offers request dispatching (Routes) with url parameter support, Templates, key/value Databases, a build-in HTTP Server and adapters for many third party WSGI/HTTP-server and template engines. All in a single file and with no dependencies other than the Python Standard Library.
- »  CherryPy (10.2.1 Released 2017-03-13) is a pythonic, object-oriented HTTP framework. CherryPy powered web applications are in fact stand-alone Python applications embedding their own multi-threaded web server. [TurboGears](#), [web2py](#) (see above) also use [CherryPy](#).
- »  Flask (0.12.1 Released 2017-03-31) is "a microframework for Python based on Werkzeug, Jinja 2 and good intentions." Includes a built-in development server, unit testing support, and is fully Unicode-enabled with RESTful request dispatching and WSGI compliance.
- »  Hug (2.2.0 Released 2016-10-17) Embrace the APIs of the future. Hug aims to make developing APIs as simple as possible, but no simpler. It's one of the first fully future looking frameworks: only supporting Python3+.
- »  Pyramid (1.8.3 Released 2017-03-12) a small, fast, down-to-earth, open source Python web development framework. It makes real-world web application development and deployment more fun, more predictable, and more productive. Pyramid is a Pylons Project, and is the successor to the Pylons web framework.

<https://wiki.python.org/moin/WebFrameworks>

TUTORIAL 1.2

Environment setting with IDE

Let's try setting the Goorm IDE environment. Just follow the red square with us.



<https://ide.Goorm.io/>

1.2 Environment setting with Goorm IDE(cont.)

Sign up for Goorm IDE first. A free account is also a powerful service that allows you to use five containers. When you register, you will enter the dashboard automatically. If you are on the main page, click Dashboard.

Sign in to goorm

E-mail

Password

Sign in

☐ Save this e-mail

Sign in with other services

f

G

N

P

Reset password

Sign up

1.2 Environment setting with Goorm IDE(cont.)

Here is the dashboard page. A container is like setting one computer. Select Python and click Create New Container to create the container.



Create a new container

Source from

☒ Template ☐ Github ☐ Bitbucket ☐ Git / SVN ☐ ZIP/TAR file

Name

Description

Choose a software stack

Search

C/C++

Python

Django

Flask

Jupyter Notebook

TensorFlow

PyQt

JAVA

Maven

Python

TemplatePython Project

OS

Python3

Python

pip3

pip

Jupyter

Django

Flask

TensorFlow

✓ Ubuntu 14.04 LTS

✓ 3.6.5

✓ 2.7.6

✓ 9.0.1

✓ 9.0.1

✓ 4.3.0

✓ 1.11.12 LTS

✓ 0.12

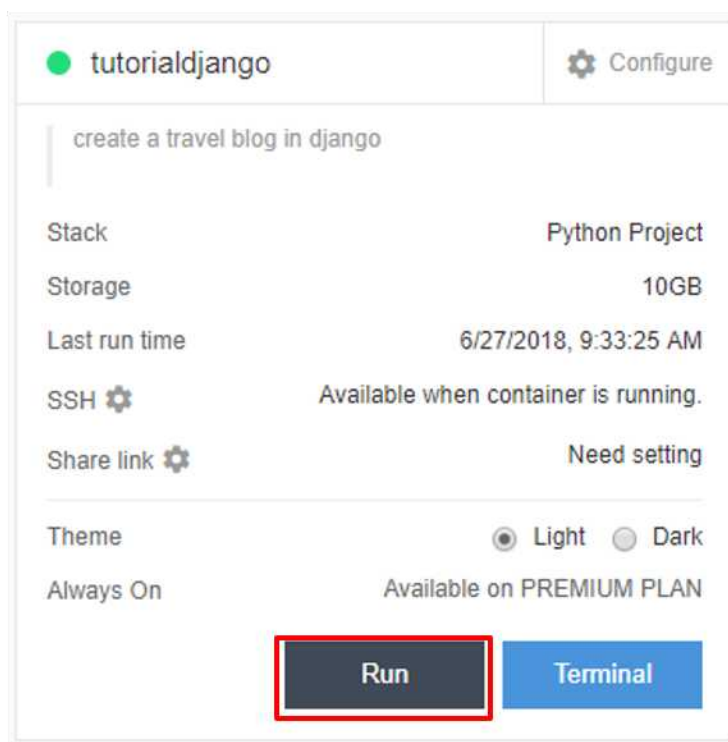
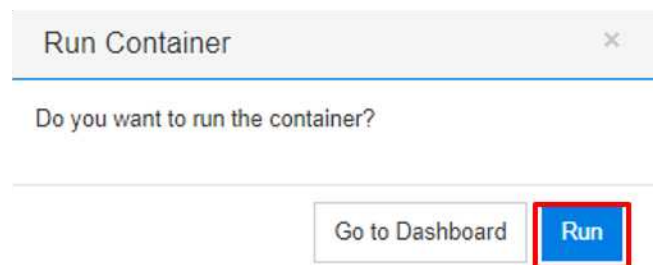
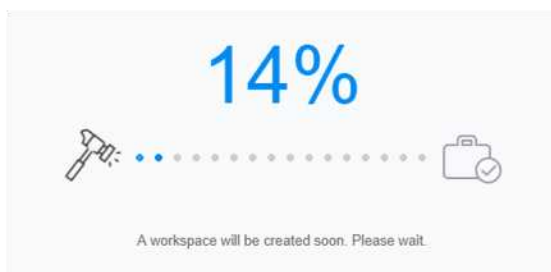
✓ 1.3.0

Cancel

Create

1.2 Environment setting with goorm IDE(cont.)

You will see that the container is being created and soon you will see that the container creation is complete. Click “Run Container” to enter the container.



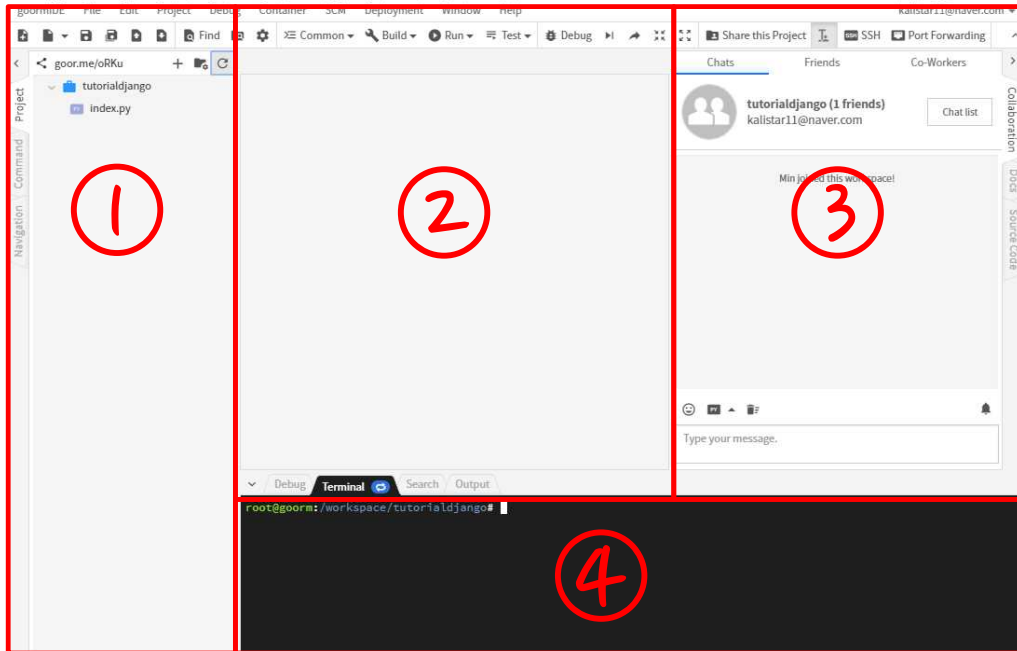
1.2 Environment setting with Goorm IDE(cont.)

This is the screen on which the container is loaded. Usually this page shows you coding tips or famous people while it's loading.



Before deleting a variable or a function, you can confirm that the variable or function is being used by the 'Right Click > Safe Delete'.

1.2 Environment setting with Goorm IDE(cont.)

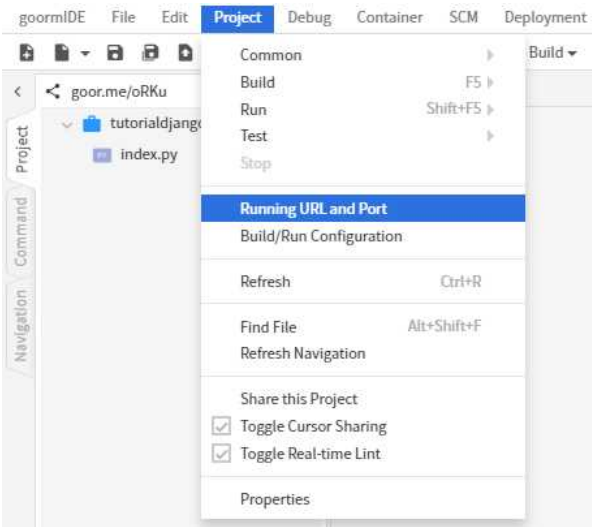


1. This space shows you the folder structure. In the top right corner there is a refresh button. If you have done something and can not see it, refresh it with this command.
2. txt, html, py files, and etc
3. Here is a space for collaboration, chat, etc. We are not going to use this options
4. Console window. We will enter most of the commands here.

1.2 Environment setting with Goorm IDE(cont.)

Click on the project and click Run URL and Port. The red squares below are the URLs we will use later. It will not run yet.

* If you are not registered, enter the URL you want below, and set the PORT to 80.



Project Properties

Property
Information
Running URL and Port
SCM
Ignore Manager
Lint Configuration
Build/Run Configuration
tutorialdjango

Running URL and Port

You can register domain address which will be used to run the server.

URL .run.goorm.io

PORT Register

Registered URL and Port (Maximum of 3 per container)

For run	URL	Port	
☒	https://tutorialdjango-keunz.run.goorm.io	80	X

Apply

Cancel OK

1.2 Environment setting with Goorm IDE(cont.)

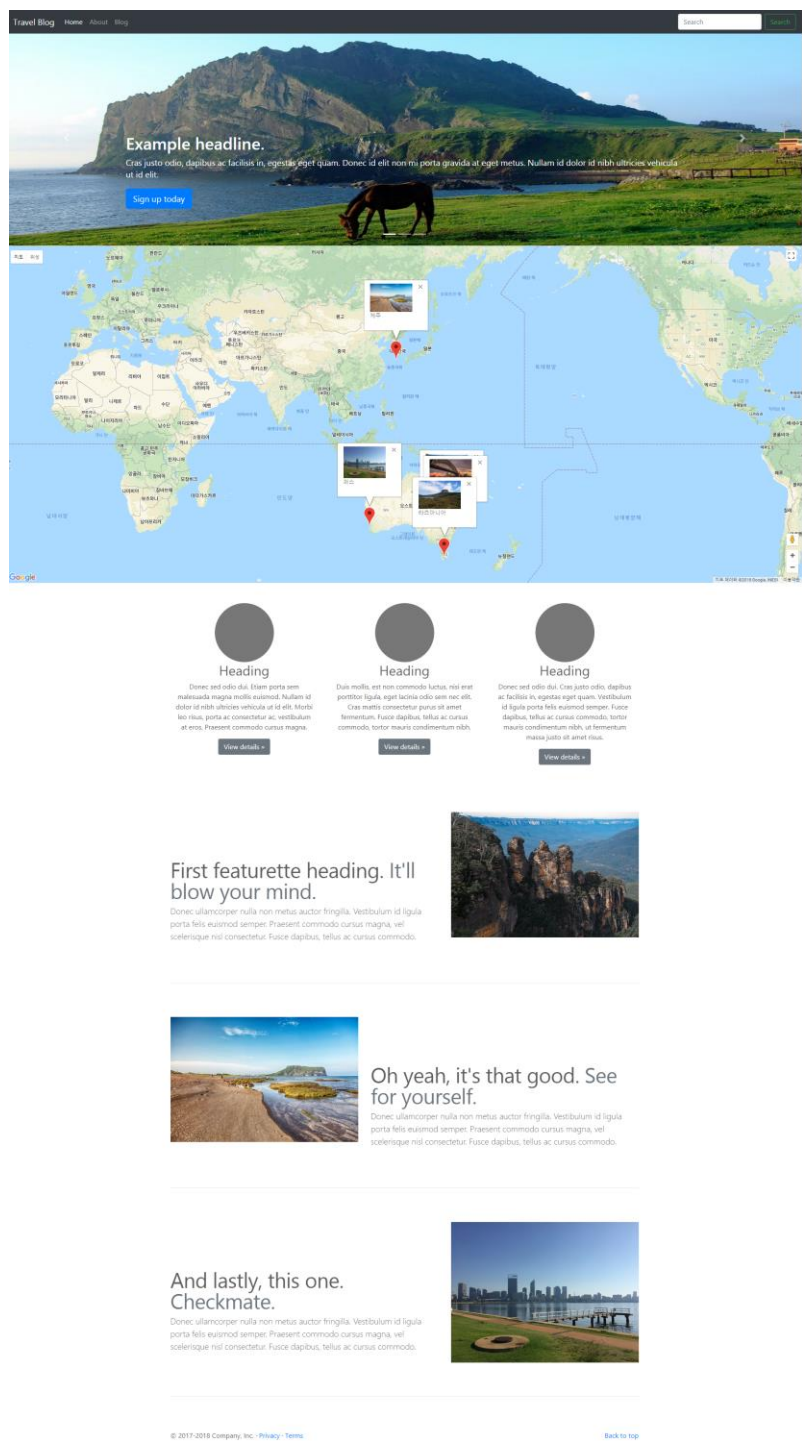
The first thing to do is planning a website. This tutorial does not cover planning steps (Please refer to the tutorial " Bootstrap Tutorial: Learn how to create Travel Blog using Bootstrap")

Let's take a look at the site structure.

1. - main page [https:// user_specified_domain .run.Goorm.io /](https://user_specified_domain.run.Goorm.io/)
 - main page, service introduction, google map api, recent post introduction
 - Template: index.html
2. - blog page [https:// user_specific_domain .run.Goorm.io / blog](https://user_specific_domain.run.Goorm.io/blog)
 - Go to the main page when you click blog, post List page
 - Template: blog.html
3. - poset page [https:// user_specified_domain .run.Goorm.io / post number](https://user_specified_domain.run.Goorm.io/post_number)
 - Move when you click post on blog page
 - Template: contents.html

1.2 Environment setting with Goorm IDE(cont.)

Here is the Main page. We did put lorem ipsum dummy text in paragraphs.



1.2 Environment setting with Goorm IDE(cont.)

Here is the Blog page.

Travel Blog

HomeAboutBlog

Search

Search

Album example

Something short and leading about the collection below—its contents, the creator, etc. Make it short and sweet, but not too short so folks don't simply skip over it entirely.

Main call to action

Secondary action



This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

View

Edit

9 mins



This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

View

Edit

9 mins



This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

View

Edit

9 mins



This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

View

Edit

9 mins



This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

View

Edit

9 mins

© 2017-2018 Company, Inc. · [Privacy](#) · [Terms](#)

[Back to top](#)

24

1.2 Environment setting with Goorm IDE(cont.)

Here is the Post page . We did put lorem ipsum dummy text in paragraphs.

Travel BlogHomeAboutBlog

Search

Search

From the Firehose

Sample blog post

January 1, 2014 by Mark

This blog post shows a few different types of content that's supported and styled with Bootstrap. Basic typography, images, and code are all supported.

Cum sociis natoque penatibus et magnis **dis parturient montes**, nascetur ridiculus mus. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Sed posuere consectetur est at lobortis. Cras mattis consectetur purus sit amet fermentum.

Curabitur blandit tempus porttitor. **Nullam quis risus eget urna mollis** ornare vel eu leo. Nullam id dolor id nibh ultricies vehicula ut id elit.

Etiam porta *sem malesuada magna* mollis euismod. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur.

Heading

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.

Sub-heading

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Example code block

Aenean lacinia bibendum nulla sed consectetur. Etiam porta sem malesuada magna mollis euismod. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa.

OlderNewer

© 2017-2018 Company, Inc. · Privacy · Terms

About

Etiam porta *sem malesuada magna* mollis euismod. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur.

이미지 캐러셀1

<

>

이미지3

내용3

— — —

Back to top

25

1.2 Environment setting with Goorm IDE(cont.)

Let's get started. Type commands as below step by step in the console window.

```
root@Goorm /workspace/tutorialdjango# mkdir mysite
```

```
root@Goorm /workspace/tutorialdjango# cd mysite
```

```
root@Goorm /workspace/tutorialdjango/mysite#
```

```
apt-get install python-virtualenv
```

```
root@Goorm /workspace/tutorialdjango/mysite#
```

```
virtualenv --python=python3.4 myvenv
```

```
root@Goorm /workspace/tutorialdjango/mysite# source myvenv/bin/activate
```

```
(myvenv)root@Goorm /workspace/tutorialdjango/mysite#
```

```
pip install django whitenoise
```

```
(myvenv)root@Goorm /workspace/tutorialdjango/mysite#
```

```
django-admin startproject tutorialdjango .
```

```
(myvenv)root@Goorm /workspace/tutorialdjango/mysite#
```

```
python manage.py migrate
```

- Please be careful ‘.’ is followed by 'startproject container name'
- Please type your container name instead of tutorialdjango

1.2 Environment setting with Goorm IDE(cont.)

```
root@goorm:/workspace/tutorialdjango# mkdir mysite
root@goorm:/workspace/tutorialdjango# cd mysite/
root@goorm:/workspace/tutorialdjango/mysite# apt-get install python-virtualenv
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
다음 새 패키지를 설치할 것입니다:
  python-virtualenv
0개 업그레이드, 1개 새로 설치, 0개 제거 및 118개 업그레이드 안 함.
```

```
root@goorm:/workspace/tutorialdjango/mysite# virtualenv --python=python3.4 myenv
Running virtualenv with interpreter /usr/bin/python3.4
Using base prefix '/usr'
New python executable in myenv/bin/python3.4
Also creating executable in myenv/bin/python
Installing setuptools, pip...done.
```

```
root@goorm:/workspace/tutorialdjango/mysite# source myenv/bin/activate
(myenv)root@goorm:/workspace/tutorialdjango/mysite#
```

```
(myenv)root@goorm:/workspace/tutorialdjango/mysite# pip install django whitenoise
Downloading/unpacking django
  Downloading Django-2.0.6-py3-none-any.whl (7.1MB): 7.1MB downloaded
Downloading/unpacking whitenoise
  Downloading whitenoise-3.3.1-py2.py3-none-any.whl
Downloading/unpacking pytz (from django)
  Downloading pytz-2018.4-py2.py3-none-any.whl (510kB): 510kB downloaded
Installing collected packages: django, whitenoise, pytz
Successfully installed django whitenoise pytz
Cleaning up...
```

```
(myenv)root@goorm:/workspace/tutorialdjango/mysite# django-admin startproject tutorialdjango .
(myenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
```

1.2 Environment setting with Goorm IDE(cont.)

Here is the brief descriptions of the commands you typed before

```
root@Goorm:/workspace/container name# python --virsion
```

Check the version of Python. Is it 2.x version? We will develop to version 3.4

```
root@Goorm:/workspace/ container name # mkdir mysite
```

mkdir is a command that creates a directory. We create a folder named mysite.

```
root@Goorm:/workspace/ container name # cd mysite
```

The cd command is the change directory command. Go to the folder mysite.

```
root@Goorm:/workspace/ container name /mysite#
```

```
apt-get install python-virtualenv
```

It is a command for setting virtual environment that makes it run as 3.X version in the environment that is currently running as 2.x version.

```
root@Goorm:/workspace/ container name /mysite#
```

```
virtualenv --python=python3.4 myvenv
```

We will use python 3.4version as a virtual environment

1.2 Environment setting with Goorm IDE(cont.)

```
root@Goorm:/workspace/ container name /mysite# source myenv/bin/activate
```

Activate virtual environment

This command should be run every time you use the cloud IDE container again.

```
(myenv)root@Goorm:/workspace/ container name /mysite#
```

```
pip install django whitenoise
```

It is a command to install up to the package. You can install only Django.

```
(myenv)root@Goorm:/workspace/ container name /mysite#
```

```
django-admin startproject 'container_name' .
```

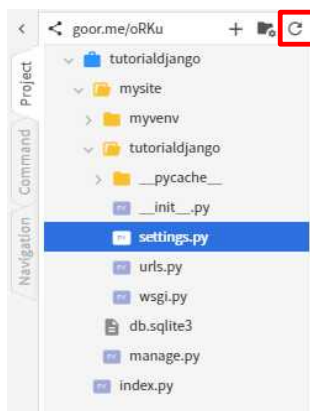
This is the command to start the project in the current folder.

```
(myenv)root@Goorm:/workspace/ container name /mysite# python manage.py migrate
```

The Migrate command will be explained later. In a nutshell, put values into DB.

1.2 Environment setting with Goorm IDE(cont.)

Now go to the settings.py file and modify the 28th line as shown below. Then type the following command into the console window.



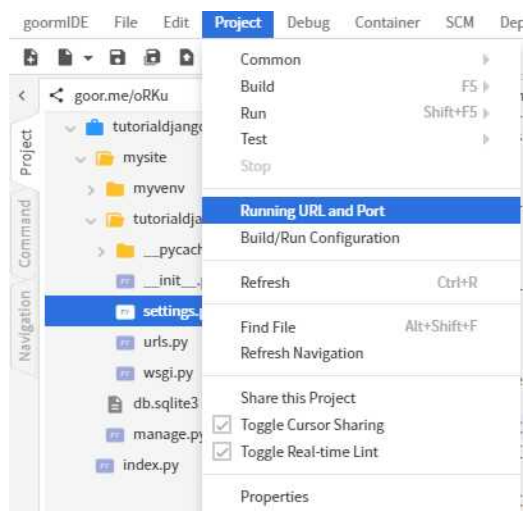
28 ALLOWED_HOSTS = ['*']

Go to Settings.py file and type the value to ‘*’ on 28th line. It will allow all users to approach.

(myvenv)root@Goorm:/workspace/container name/mysite#
python manage.py runserver 0:80

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py runserver 0:80
Performing system checks...

System check identified no issues (0 silenced).
June 27, 2018 - 01:41:32
Django version 2.0.6, using settings 'tutorialdjango.settings'
Starting development server at http://0:80/
Quit the server with CONTROL-C.
```



1.2 Environment setting with Goorm IDE(cont.)

If you click on the url in the red square, the following page will appear. Your first website is running now!

Project Properties

Property

Information

Running URL and Port

SCM

Ignore Manager

Lint Configuration

Build/Run Configuration

tutorialdjango

Running URL and Port

You can register domain address which will be used to run the server.

URL

https://

.run.goorm.io

PORT

80 ~ 65000

Register

Registered URL and Port (Maximum of 3 per container)

For run	URL		Port	
☒	https://tutorialdjango-keunz.run.goorm.io		80	

Apply

Cancel

OK

django

[View release notes for Django 2.0](#)



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

Chapter 2

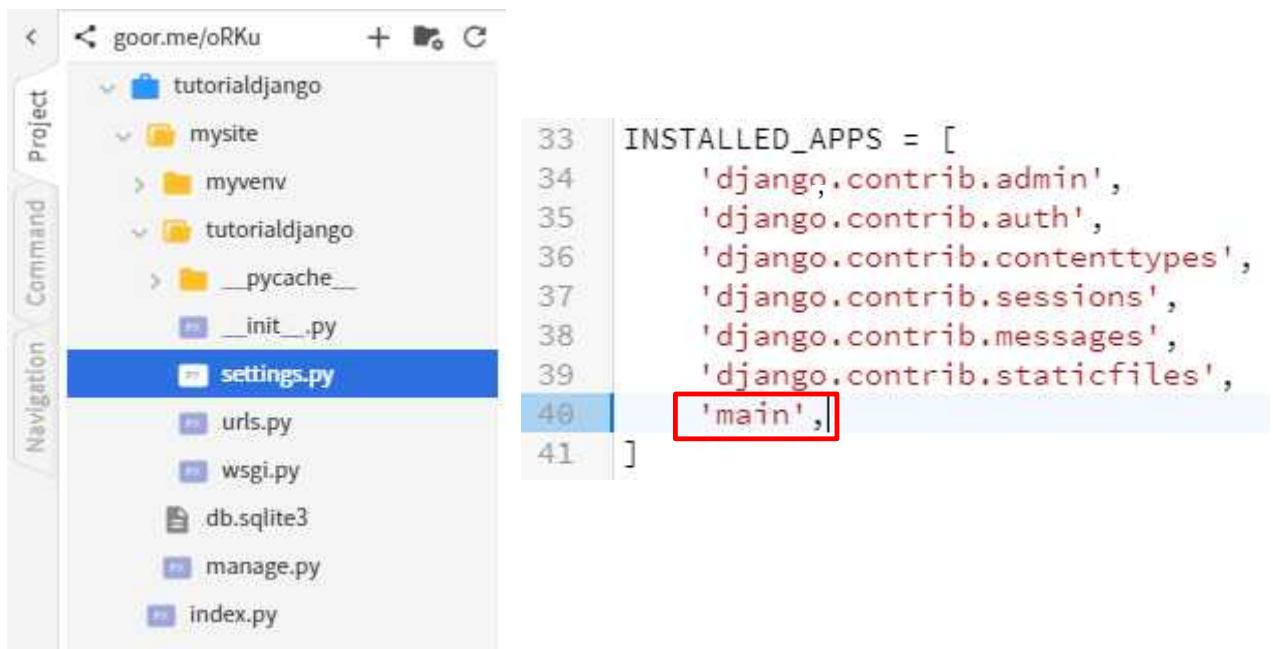
Create Main page on Django

2. Create main page(cont.)

With Django you can create and assemble multiple apps at the same time. Because we are making a simple project, we will create only one app called “main”. To create an application you need to run the command below in the console window . If you do not do this, the app will not start.

```
root@Goorm:/workspace/container name/mysite# python manage.py startapp main
```

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py startapp main
(myvenv)root@goorm:/workspace/tutorialdjango/mysite#
```



Then find `INSTALLED_APPS` and add a line like above in the file `settings.py` to register the application.

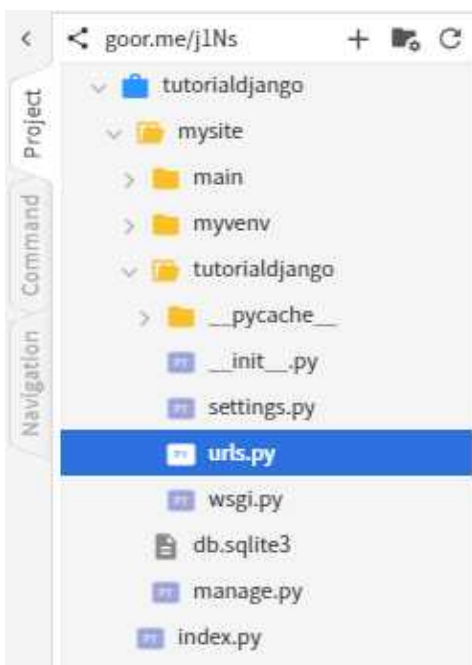
* Please check that there is a comma after “main”.

2. Create main page(cont.)

Now we are going to add lines in the urls.py file to map pages and a urls.

First, let's add 'main.view import index' line which means we will use the index function from the view.py file in main folder. We did not declare index function in the view.py file yet, so an error will occurred.

And path('', index),
means Django will now redirect everything that comes into '(your web site url/domain address)' to the index and we are going to connect the index function to the index.html file on the next page.



```
16 from django.contrib import admin
17 from django.urls import path
18 from main.views import index
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('', index)
23 ]
24
```

mysite > tutorialdjango > urls.py

2. Create main page(cont.)

Now, open the `mysite\product\views.py` file to create a function called `index`. As you can see, when you call the function it will return “render” which will render the `main/index.html` template.

`mysite > product > views.py`

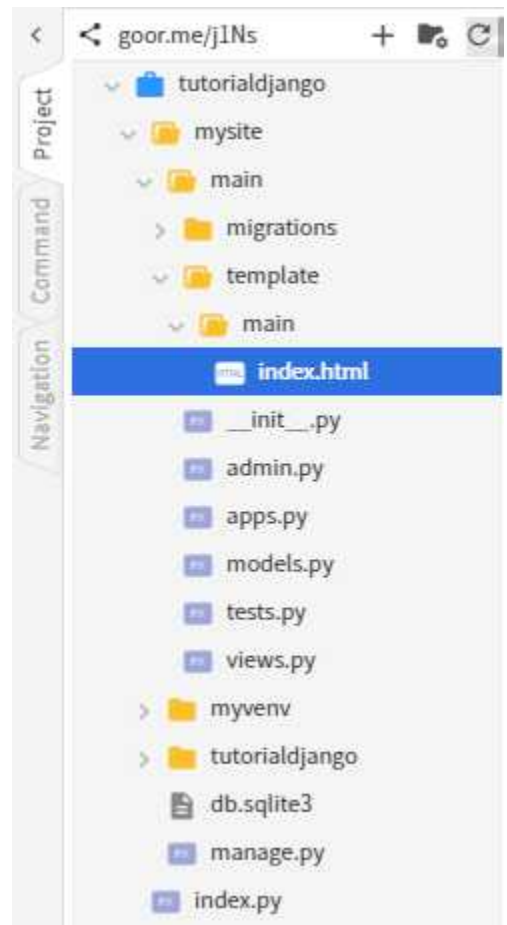
```
1 from django.shortcuts import render
2
3 def index(request):
4     return render(request, 'main/index.html')
```

2. Create main page(cont.)

In the process of connecting Template, Please make `mysite> main> templates> main> index.html` structure.

And then, Please paste codes below to the index.html file to check page loading.

```
1 <html>
2 <head>
3     <title>Django!</title>
4 </head>
5 <body>
6     <h1>Test!</h1>
7 </body>
8 </html>
```



2. Create main page(cont.)

When the web server is running check your website and see how the index.html page looks like.

```
(myenv)root@Goorm:/workspace/container name/mysite#  
python manage.py runserver 0:80
```

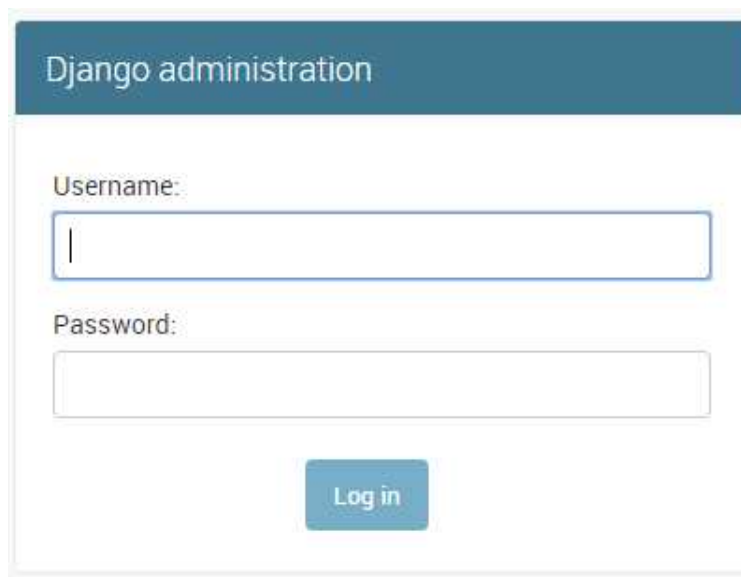
```
(myenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py runserver 0:80  
Performing system checks...  
  
System check identified no issues (0 silenced).  
June 27, 2018 - 03:00:38  
Django version 2.0.6, using settings 'tutorialdjango.settings'  
Starting development server at http://0:80/  
Quit the server with CONTROL-C.
```



Test!

2. Create main page(cont.)

This time, add `'/admin'` at the end of your url then press enter, you can see the admin page . You can post, edit, or delete posts here and manage your users as well. But now, you can not log in because you have not created a superuser yet. The `'admin/'` part you saw in the `urls.py` file before is this page. For security reasons, we recommend that you do not leave the admin page open on the web service later.

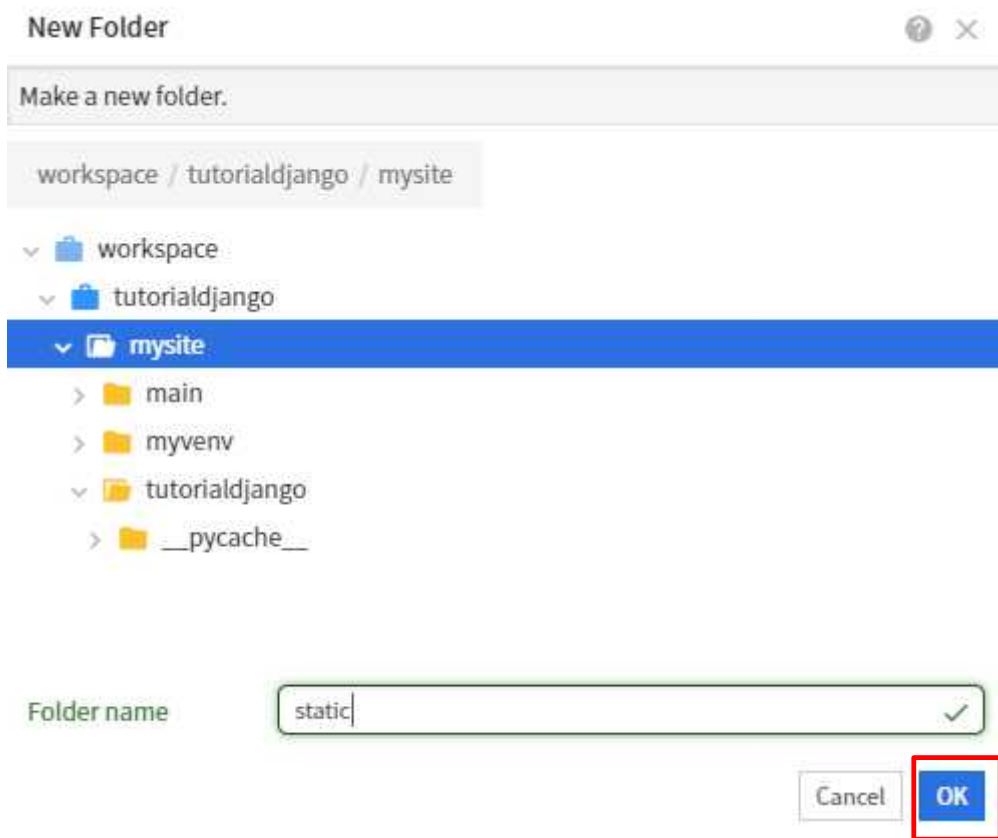
The image shows the Django administration login interface. It features a dark blue header with the text "Django administration". Below the header, there are two input fields: "Username:" and "Password:". The "Username:" field contains a single vertical bar character. Below the password field is a blue "Log in" button.

<https://yourURL/admin>

2. Create main page(cont.)

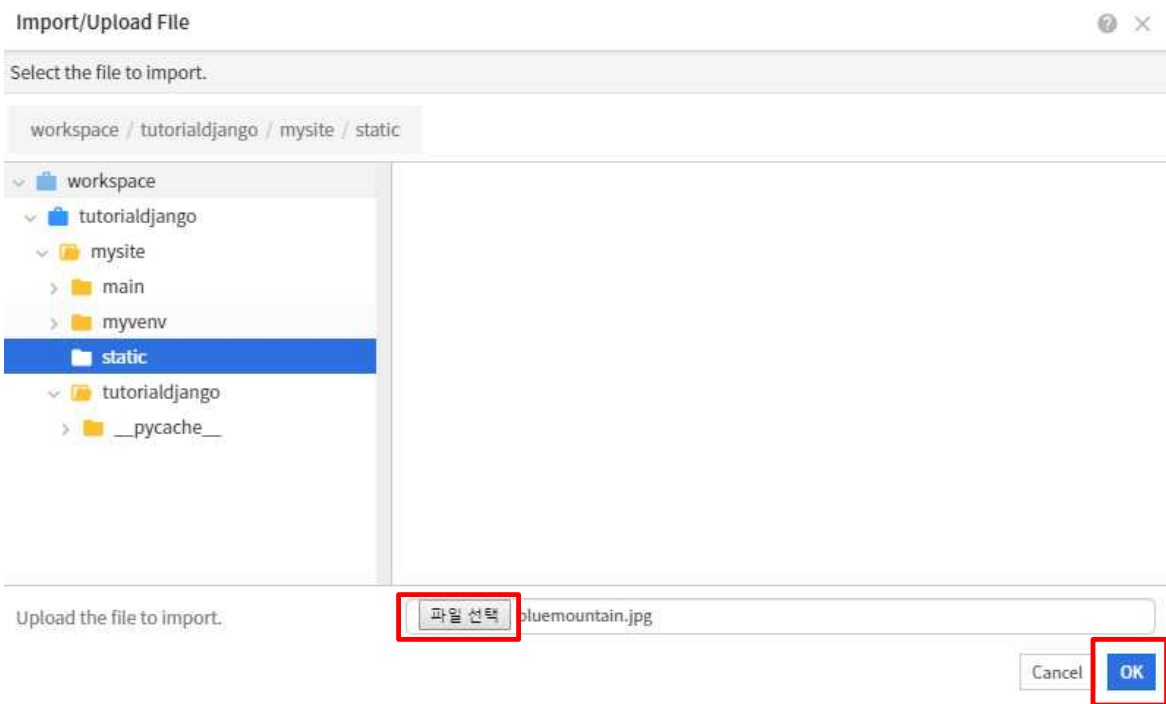
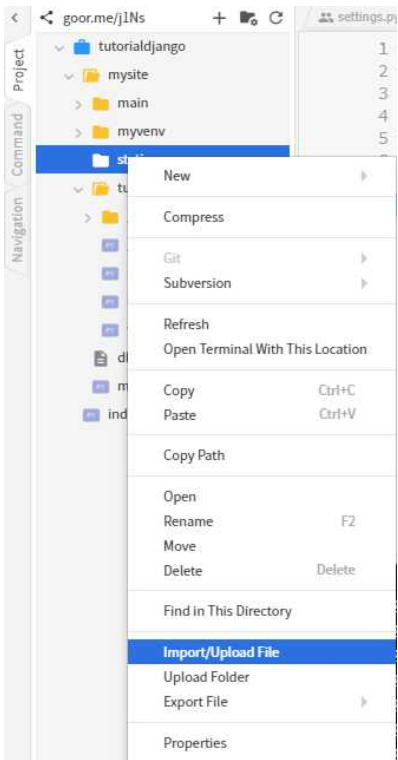
Next, I'll show you how to load images on the main page. Django does not support relative directory that HTML and CSS do support. Let's create a subfolder named 'static' in 'mysite' folder to store static files.

1) Create a new folder



2. Create main page(cont.)

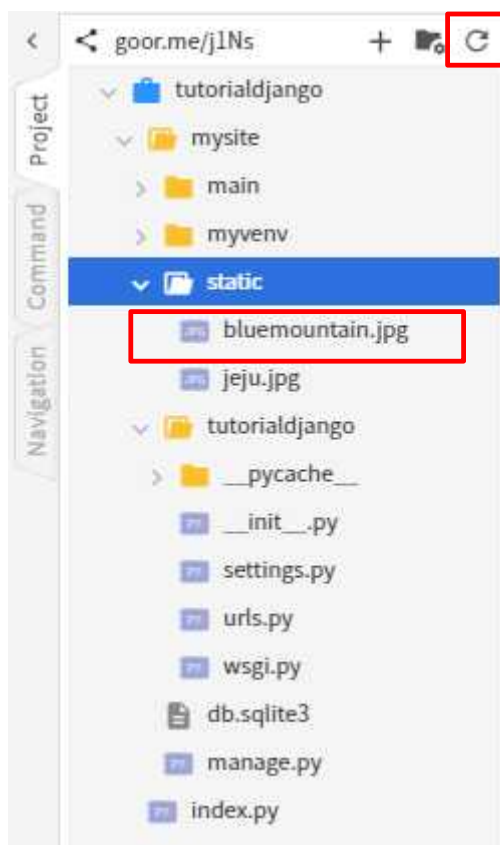
2) File upload



2. Create main page(cont.)

3) File upload confirmation

If the image does not appear, please click Refresh symbol.



2. Create main page(cont.)

The next step is to define a list of directories in the settings.py file. Add codes as below. Please make sure to type a comma at the end.

```
121 STATIC_URL = '/static/'
122 STATICFILES_DIRS = (
123     os.path.join(BASE_DIR, 'static'),
124 )
```

2. Create main page(cont.)

Now open the index.html file and type the code to calls the static file. The {% %} notation is called a template tag.

For more information, please refer Django official document (<https://docs.djangoproject.com/en/2.0/ref/templates/builtins/>).

```
1 <html>
2   <head>
3     <title>Django!</title>
4   </head>
5   <body>
6     <h1>Test!</h1>
7     {% load staticfiles %}
8     
9   </body>
10 </html>
```

tutorialdjango/mysite/main/templates/main/index.html

2. Create main page(cont.)

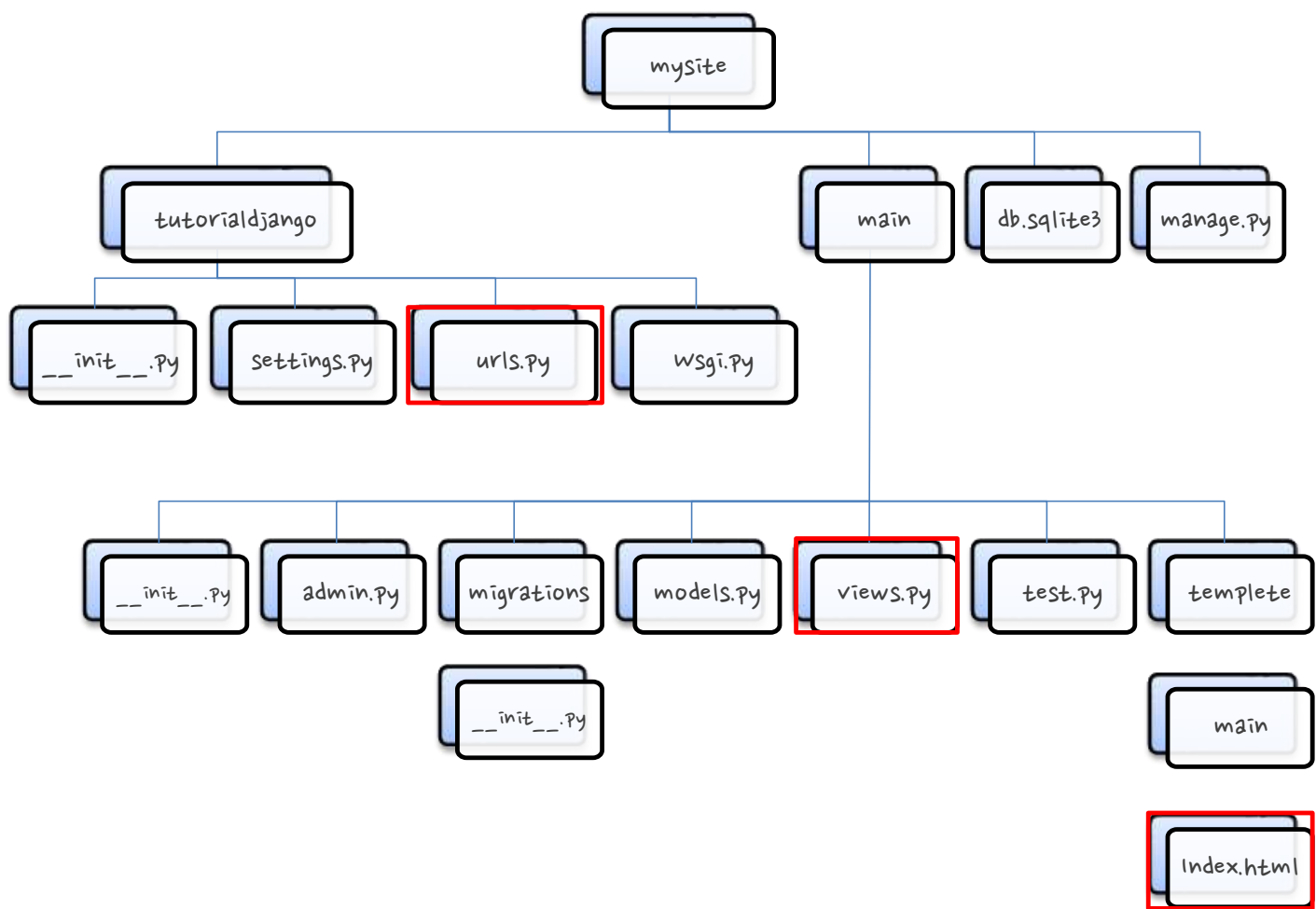
If the server is not running please type `manage.py runserver 0:80` in the console window, press enter and go to **Project** Running URL and Port menu, then click on your URL. It works!

Test!



2. Create main page(cont.)

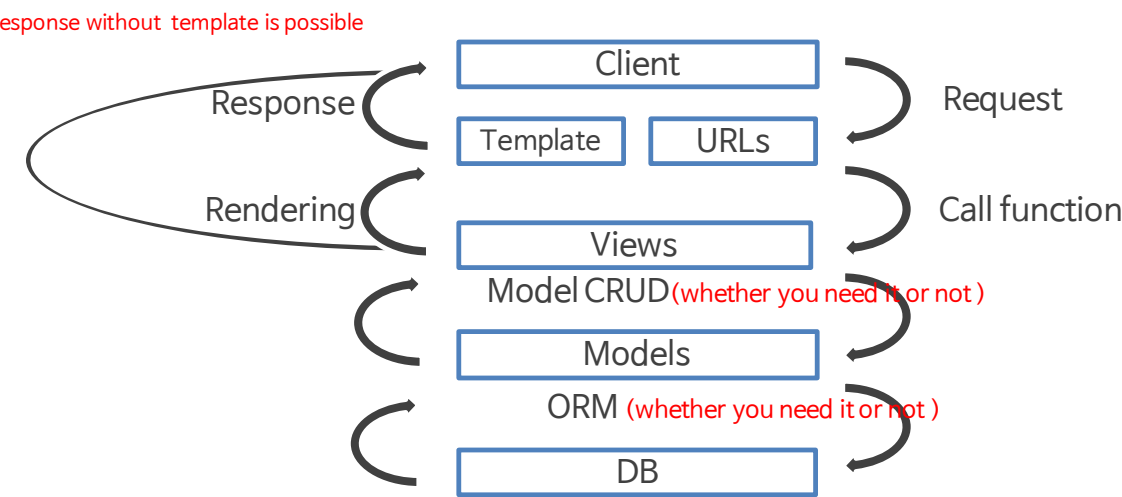
This is the file structure of our project. The red squares files are what we modified.



2. Create main page(cont.)

Django calls the MVC model to MTV. The terminology has changed, but the basic concept is the same. The table below is a diagram of how Django works. Chapter 8 covers this in more detail.

MTV



Model - Python Class connected Data Base

Template - Client View

View - Send data to Django Templates

CRUD refers to the basic data processing functions of most computer software: It stands for Create, Read, Update, and Delete.

Operation	SQL	HTTP	RESTful WS	DDS
Create	INSERT	PUT / POST	POST	write
Read (Retrieve)	SELECT	GET	GET	read / take
Update (Modify)	UPDATE	PUT / POST / PATCH	PUT	write
Delete (Destroy)	DELETE	DELETE	DELETE	dispose

https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

Chapter 3

Create Blog page on Django

3. Create Blog page(cont.)

Now, let's create a blog page that contains postings. First, edit the urls.py file and edit the views.py file as below.

```
16 from django.contrib import admin
17 from django.urls import path
18 from main.views import index, blog
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('', index),
23     path('blog/', blog),
24 ]
```

tutorialdjango/mysite/tutorialdjango/urls.py

```
3 def index(request):
4     return render(request, 'main/index.html')
5
6 def blog(request):
7     return render(request, 'main/blog.html')
```

tutorialdjango/mysite/main/views.py

3. Create Blog page(cont.)

Let's create the blog.html file in the [mysite/main/templates/main/blog.html](#) directory. Please type the codes in the file as below

```
1 <html>
2 <head>
3     <title>Django!</title>
4 </head>
5 <body>
6     <h1>Blog Page!</h1>
7 </body>
8 </html>
```

/mysite/main/templates/main/blog.html

3. Create Blog page(cont.)

Enter the command below to connect the server in the console window

`python manage.py runserver 0:80`

Go to the [Project > Running URL and Port](#) in the menu and then click the URL to see if it's working.

```
root@goorm:/workspace/tutorialdjango# cd mysite/
root@goorm:/workspace/tutorialdjango/mysite# source myenv/bin/activate
(myenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py runserver 0:80
Performing system checks...

System check identified no issues (0 silenced).
July 18, 2018 - 02:30:17
Django version 2.0.6, using settings 'tutorialdjango.settings'
Starting development server at http://0:80/
Quit the server with CONTROL-C. █
```

Executed page([https://tutorialdjango-bcrpr.run.Goorm.io/blog/](https://tutorialdjango-bcrpr.run.goorm.io/blog/))



Blog Page!

3. Create Blog page(cont.)

Now we are going to create some text fields for each posting. First, create a post name field and a contents field.

tutorialdjango/mysite/main/models.py

```
1 from django.db import models
2
3 class Post(models.Model):
4     postname = models.CharField(max_length=50)
5     contents = models.TextField()
```

In order to add new model in your database, you need to stop the web server first by pressing the Ctrl + C keys and then type these two commands in.

```
(myvenv)root@Goorm:/workspace/container name/mysite#
python manage.py makemigrations main
```

```
(myvenv)root@Goorm:/workspace/container name/mysite#
python manage.py migrate
```

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py makemigrations main
Migrations for 'main':
  main/migrations/0001_initial.py
    - Create model Post
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions
Running migrations:
  Applying main.0001_initial... OK
```

3. Create Blog page(cont.)

Now we must open the admin.py file and activate the admin page. Then you can change options if needed.

tutorialdjango/mysite/main/admin.py

```
1 from django.contrib import admin
2 from .models import Post
3
4 admin.site.register(Post)
```

3. Create Blog page(cont.)

Now, create a *superuser account* to log in the admin page. *superusers* can delete, edit, and save posts, and manage other users. Go to the console window and type the command “python manage.py createsuperuser” .

```
(myvenv)root@Goorm:/workspace/container name/mysite#  
python manage.py createsuperuser
```

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py createsuperuser  
Username (leave blank to use 'root'): tutorialdjango  
Email address: tutorialdjango@gmail.com  
Password:  
Password (again):  
Superuser created successfully.
```

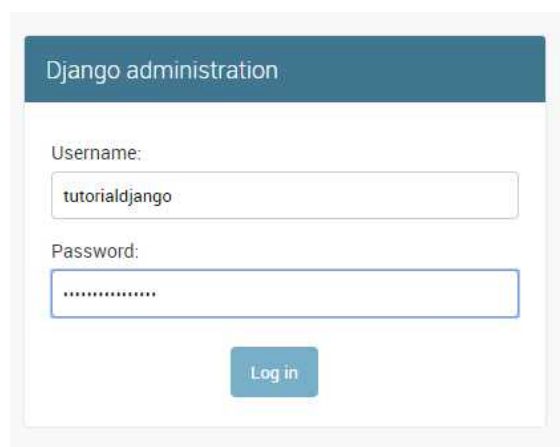
↑ I will create an admin user account.

You can't see the password when you are typing it in the console window.
Then let's start the server over again with this command.

python manage.py runserver 0:80

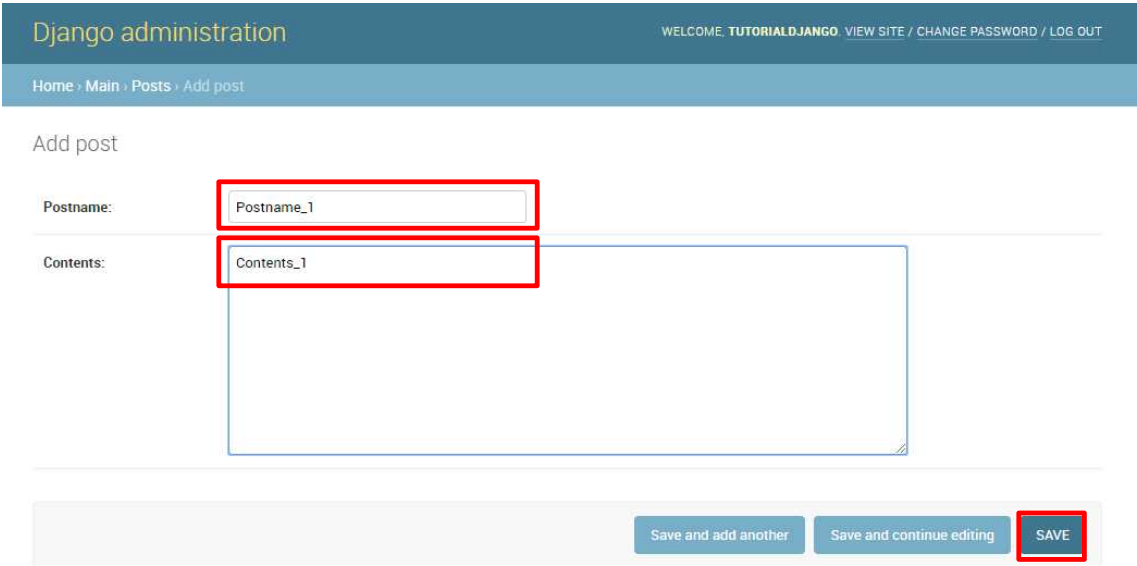
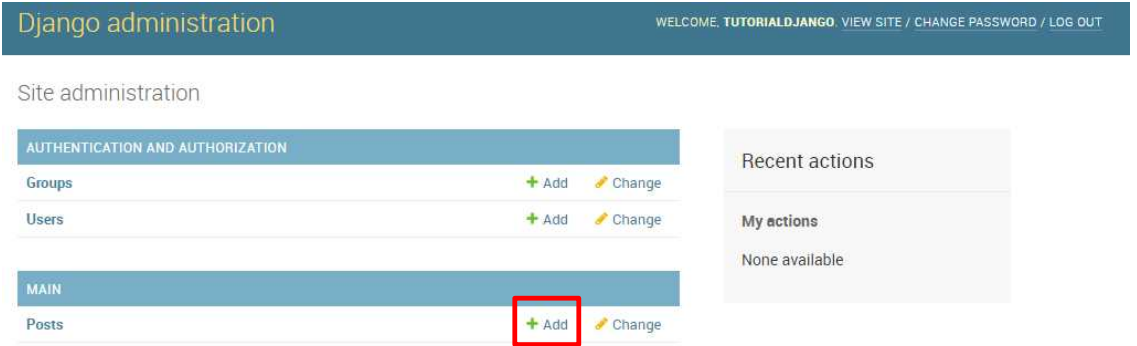
Can you see the admin dashboard when you log in?

<https://tutorialdjango-bcrpr.run.Goorm.io/admin/>



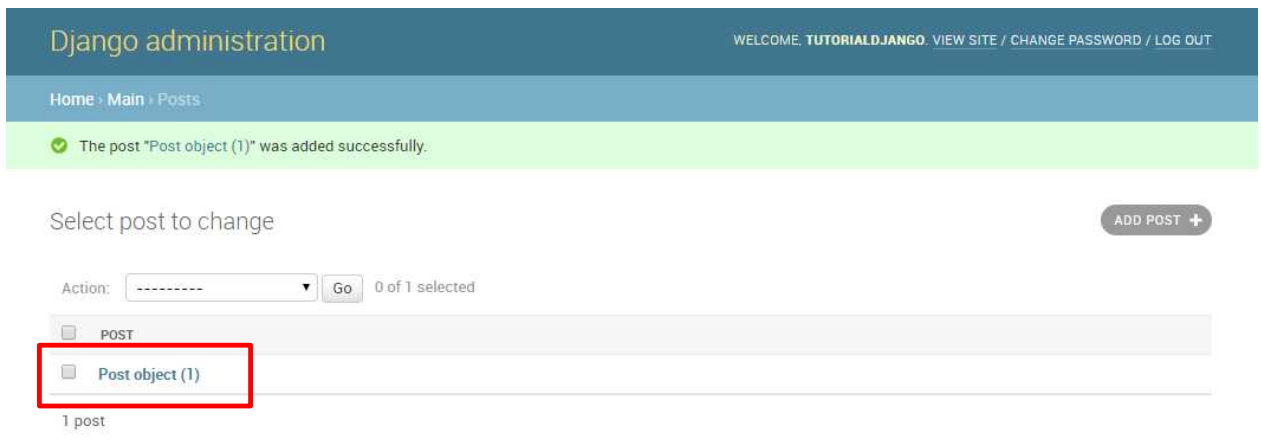
3. Create Blog page(cont.)

When you are on the dashboard try to add a post like this.



3. Create Blog page(cont.)

If you click on the SAVE button, the title is not shown as what we wrote earlier(postname_1). That is because we didn't define that "postname" is the title. Therefore let's do it: Go to the models.py file and add those codes.



mysite/product/models.py

```
1  from django.db import models
2
3  class Post(models.Model):
4      postname = models.CharField(max_length=50)
5      contents = models.TextField()
6
7      def __str__(self):
8          return self.postname
```

3. Create Blog page(cont.)

If you refresh the browser window, you will see the new title that you defined. Now let's create some new postings (posts will be deleted later).

Django administration

WELCOME, TUTORIALD.JANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts

Select post to change

ADD POST +

Action: Go 0 of 1 selected

POST

Postname_1

1 post

Add Post

Django administration

WELCOME, TUTORIALD.JANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts > Add post

Add post

Postname:

Postname_2

Contents:

Contents_2

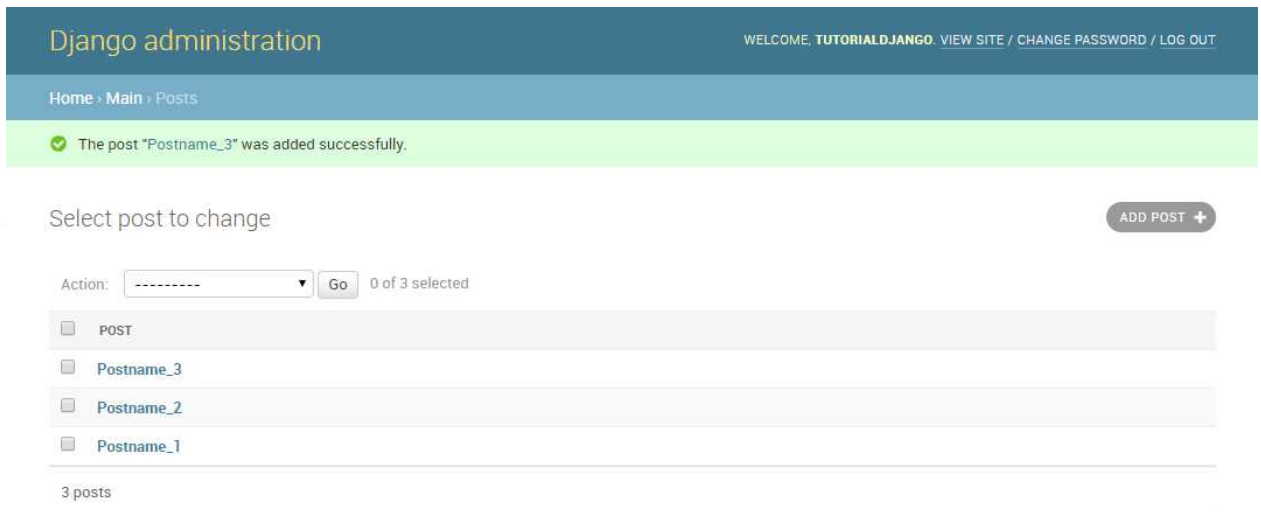
Save and add another

Save and continue editing

SAVE

3. Create Blog page(cont.)

I created three posts. Now, let's put these posts on the blog page. Please modify the views.py file as shown below.



tutorialdjango/mysite/main/views.py

```
1 from django.shortcuts import render
2 from .models import Post
3
4 def index(request):
5     return render(request, 'main/index.html')
6
7 def blog(request):
8     postlist = Post.objects.all()
9     return render(request, 'main/blog.html', {'postlist':postlist})
```

3. Create Blog page(cont.)

We use the template tag { } to get all the inserted data from database.

For more information on template tags and language please refer here

* Django documentation: <https://docs.djangoproject.com/en/2.0/>

* Templates language :

<https://docs.djangoproject.com/en/2.0/ref/templates/language/>

```
1 <html>
2 <head>
3     <title>Django!</title>
4 </head>
5 <body>
6     <h1>Blog Page!</h1>
7     <table>
8     {% for list in postlist %}
9     <tr>
10         <td>{{list.postname}}</td>
11         <td>{{list.contents}}</td>
12     </tr>
13     {% endfor %}
14 </table>
15 </body>
16 </html>
```

Blog Page!

Postname_1 Contents_1

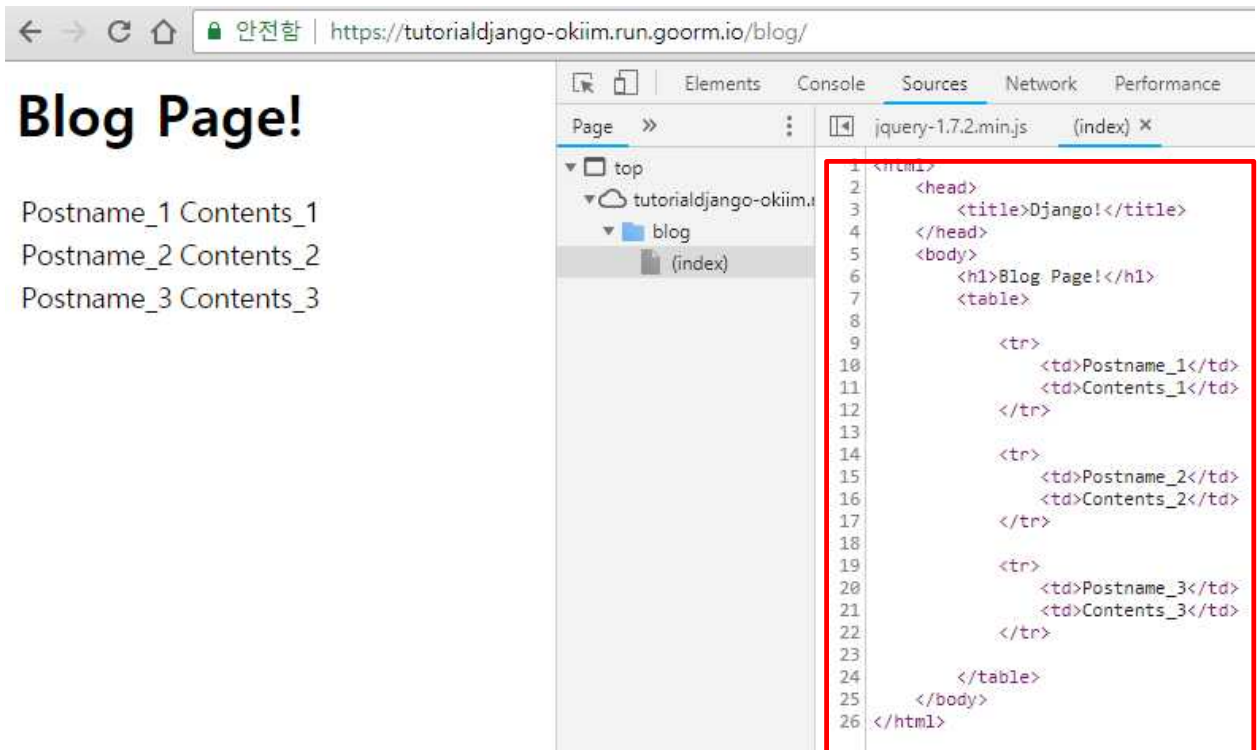
Postname_2 Contents_2

Postname_3 Contents_3

- Executed page -

3. Create Blog page(cont.)

When you check the source code with chrome development mode(F12), we can not see the template language that we created. Because it is shown to the user as a html format.



- Source code -

Chapter 4

Create Post page with Django

4. Create Post page(cont.)

Every time you come back to work on this project, you'll need to run the command below to make sure you're running the version of Python installed under myenv/ rather than your system Python. You can tell it's using this because your shell prompt will be prefixed with (myenv)

```
root@Goorm:/workspace/container name# cd mysite
```

```
root@Goorm:/workspace/container name/mysite# source myenv/bin/activate
```

```
(myenv)root@Goorm:/workspace/container name/mysite#
```

If you have been following the command without myenv, we recommend you delete the container now and start over again.

with(myenv) prefix or without is a completely different environment.

4. Create Post page(cont.)

Now, We are going to create a detailed posting page for each post that are shown on the blog page.

3. <http://username.run.Goorm.io/postnumber>

- On the blog page, click on your post to open it.
- Template: postdetails.html

```
16 from django.contrib import admin
17 from django.urls import path
18 from main.views import index, blog, postdetails
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('', index),
23     path('blog/', blog),
24     path('blog/<int:pk>/', postdetails),
25 ]
```

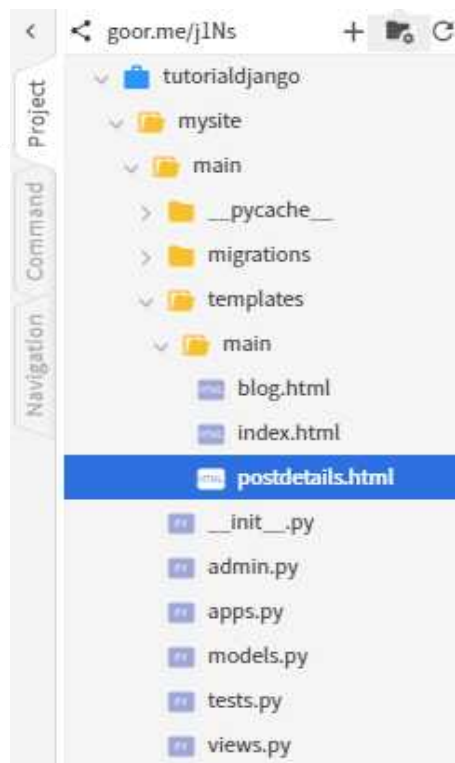
tutorialdjango/mysite/tutorialdjango/urls.py

```
1 from django.shortcuts import render
2 from .models import Post
3
4 def index(request):
5     return render(request, 'main/index.html')
6
7 def blog(request):
8     postlist = Post.objects.all()
9     return render(request, 'main/blog.html', {'postlist':postlist})
10
11 def postdetails(request, pk):
12     postlist = Post.objects.get(pk=pk)
13     return render(request, 'main/postdetails.html', {'postlist':postlist})
```

tutorialdjango/mysite/main/views.py

4. Create Post page(cont.)

As shown in the screen capture below, there are three templates in the main folder. This is the last template. (Designing blog with Bootstrap will be covered in Chapter 7.)



```
1 <html>
2 <head>
3     <title>Django!</title>
4 </head>
5 <body>
6     <h1>Postdetails Page!</h1>
7     <p>{{postlist.postname}}</p>
8     <p>{{postlist.contents}}</p>
9 </body>
10 </html>
```

tutorialdjango/mysite/main/templates/main/postdetails.html

4. Create Post page(cont.)

Add the post number at the end of the url as below, it will show you the page like the following image.

`https://tutorialdjango-okiiim.run.Goorm.io/blog/1`



Postdetails Page!

Postname_1

Contents_1

- Executed page -

4. Create Post page(cont.)

We're going to use the onclick event to move to the post detail pages when you click on the post title or contents on the blog page

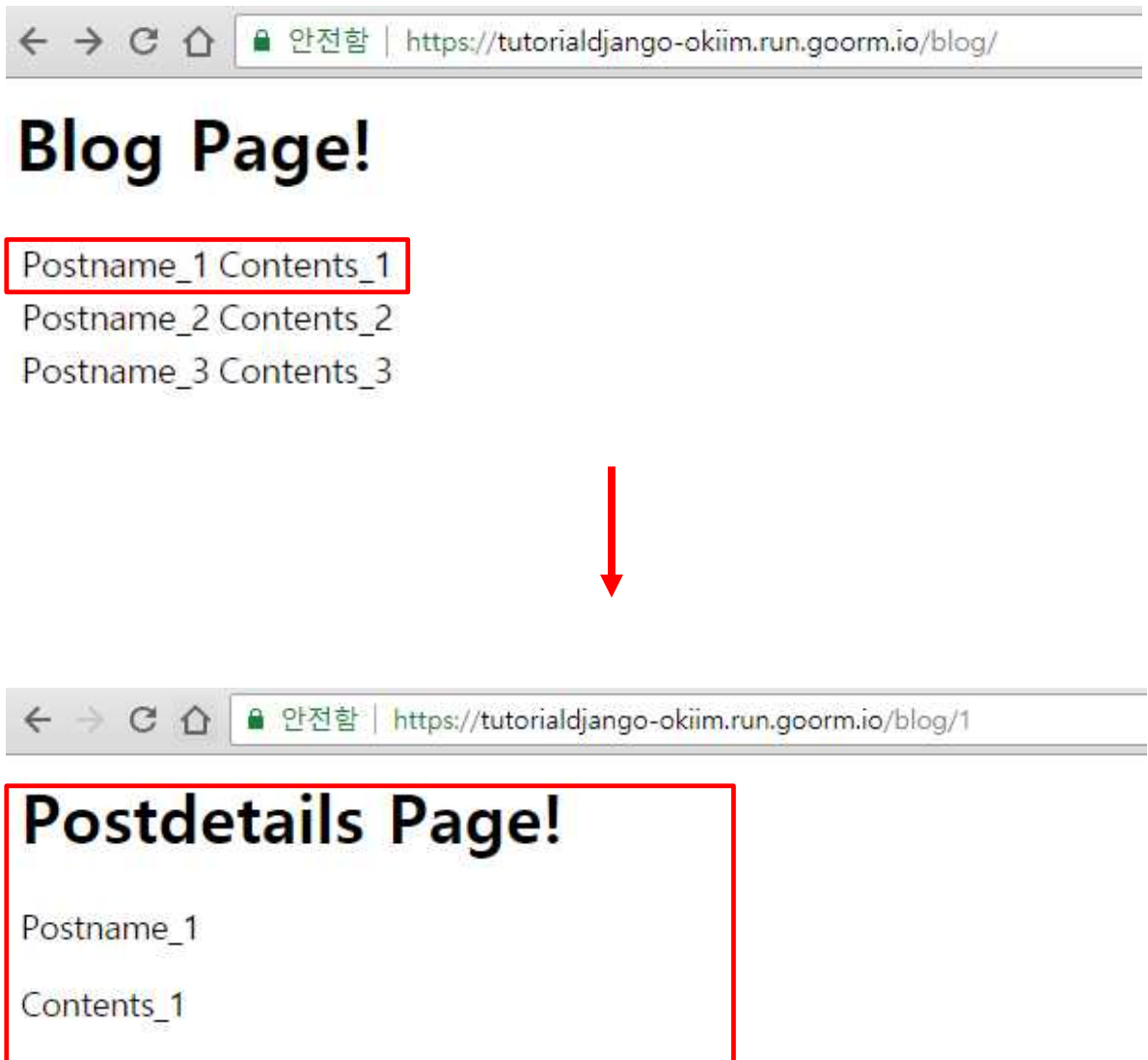
'https://tutorialdjango-bcrpr.run.Goorm.io/' Please replace this part with your URL

tutorialdjango/mysite/main/templates/main/blog.html

```
1 <html>
2 <head>
3   <title>Django!</title>
4 </head>
5 <body>
6   <h1>Blog Page!</h1>
7   <table>
8     {% for list in postlist %}
9     <tr onclick="location.href='https://tutorialdjango-bcrpr.run.goorm.io/blog/{{ list.pk }}/'">
10       <td>{{ list.postname }}</td>
11       <td>{{ list.contents }}</td>
12     </tr>
13     {% endfor %}
14   </table>
15 </body>
16 </html>
```

4. Create Post page(cont.)

Click on the post title and contents to go to the post as shown below.



4. Create Post page(cont.)

Let's create a button that moves from post detail pages to the post list page.
Please add anchor tag “<a>” with url like this in the postdetails.html file.

Container name/mysite/main/templates/main/productdetails.html

```
1 <html>
2   <head>
3     <title>Django!</title>
4   </head>
5   <body>
6     <h1>Postdetails Page!</h1>
7     <p>{{postlist.postname}}</p>
8     <p>{{postlist.contents}}</p>
9     <a href='https://tutorialdjango-okiim.run.goorm.io/blog/'>List(Go back)</a>
10   </body>
11 </html>
```

← → ↺ 🏠 🔒 안전함 | https://tutorialdjango-okiim.run.goorm.io/blog/1

Postdetails Page!

Postname_1

Contents_1

[List\(Go back\)](#)

4. Create Post page(cont.)

Now let's set ImageField to upload photos in the post.

Go to the tutorialdjango>mysite>main > directory and open the models.py file. If you set null=True here, the board posts will be null.

tutorialdjango/mysite/main/models.py

```
1  from django.db import models
2
3  class Post(models.Model):
4      postname = models.CharField(max_length=50)
5      mainphoto = models.ImageField(blank=True, null=True)
6      contents = models.TextField()
7
8      def __str__(self):
9          return self.postname
```

4. Create Post page(cont.)

Now that you've made the changes above, you should install a library that can process your photos. It is called Pillow and can be installed as follows.

```
(myvenv)root@Goorm:/workspace/container name/mysite#
```

```
pip install pillow==2.9.0
```

```
(myvenv)root@Goorm:/workspace/container name /mysite#
```

```
python manage.py makemigrations
```

```
(myvenv)root@Goorm:/workspace/container name /mysite#
```

```
python manage.py migrate
```

```
(myvenv)root@Goorm:/workspace/container name /mysite#
```

```
python manage.py runserver 0:80
```

```
^C(myvenv)root@goorm:/workspace/tutorialdjango/mysite# pip install pillow==2.9.0
Downloading/unpacking pillow==2.9.0
  Downloading Pillow-2.9.0.tar.gz (9.3MB): 9.3MB downloaded
  Running setup.py (path:/workspace/tutorialdjango/mysite/myvenv/build/pillow/setup.py) egg_info for package pillow

Installing collected packages: pillow
  Running setup.py install for pillow
```

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py makemigrations
Migrations for 'main':
  main/migrations/0002_post_mainphoto.py
    - Add field mainphoto to post
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions
Running migrations:
  Applying main.0002_post_mainphoto... OK
(myvenv)root@goorm:/workspace/tutorialdjango/mysite#
```

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py runserver 0:80
Performing system checks...

System check identified no issues (0 silenced).
May 02, 2018 - 07:38:06
Django version 2.0.4, using settings 'tutorialdjango.settings'
Starting development server at http://0:80/
Quit the server with CONTROL-C.
```

4. Create Post page(cont.)

* Change post

Django administration

WELCOME, TUTORIALDJANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts > Postname_1

Change post

HISTORY

Postname:

Postname_1

Mainphoto:

파일 선택

선택된 파일 없음

Contents:

Contents_1

Delete

Save and add another

Save and continue editing

SAVE

* Add post

Django administration

WELCOME, TUTORIALDJANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts > Postname_1

Change post

HISTORY

Postname:

Postname_1

Mainphoto:

파일 선택

선택된 파일 없음

Contents:

Contents_1

Delete

Save and add another

Save and continue editing

SAVE

4. Create Post page(cont.)

Now let's set the path where the photos will be saved. If you do not do this, the file will be saved directly in the mysite directory when you upload photos.

tutorialdjango/mysite/tutorialdjango/settings.py

```
126 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
127 MEDIA_URL = '/media/'
```

Let's upload 5 Image as below



Fist of all, we well post jeju

Django administration

WELCOME, TUTORIALDJANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts > Add post

Add post

Postname:

Postname_jeju

Mainphoto:

파일 선택

 jeju.jpg

Contents:

contents_jeju

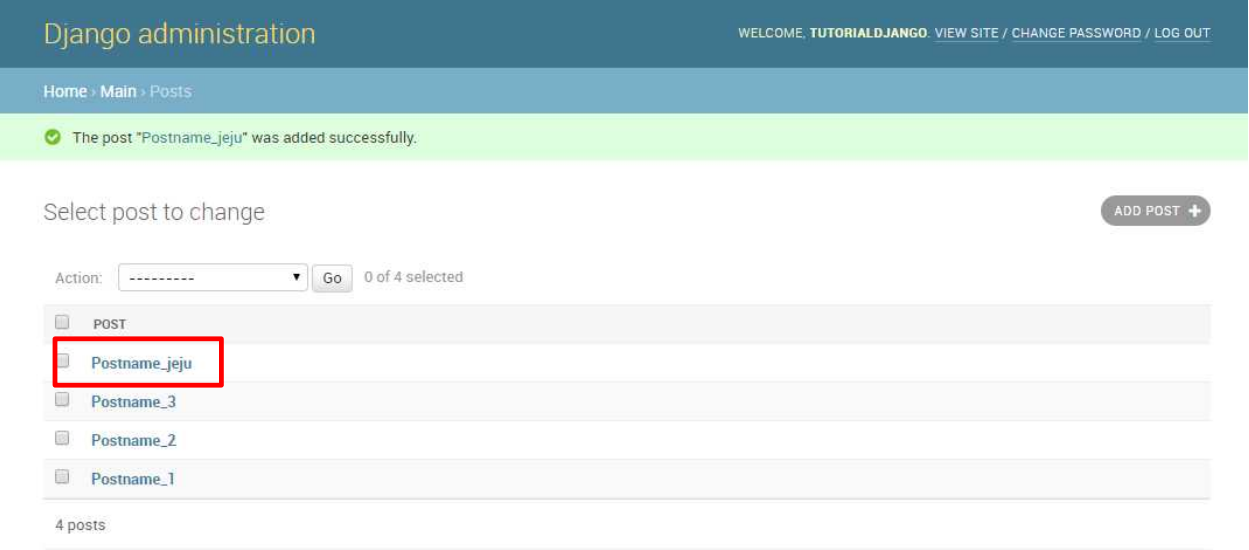
Save and add another

Save and continue editing

SAVE

4. Create Post page(cont.)

First, I uploaded a photo of JEJU. Click on the post to check the uploaded photo.



4. Create Post page(cont.)

I click on image file's name and an error page is showing up that says the page can not be found. The error is caused by the image url that can not be found. So, We have to set up the image url.

Change post HISTORY

Postname:

Postname_jeju

Mainphoto:

Currently: jeju_JLmLF8H.jpg | Clear

Change: 파일 선택 선택된 파일 없음

Contents:

contents_jeju

Delete

Save and add another

Save and continue editing

SAVE

Page not found (404)

Request Method: GET

Request URL: http://tutorialdjango-okiim.run.goorm.io/media/jeju_JLmLF8H.jpg

Using the URLconf defined in tutorialdjango.urls, Django tried these URL patterns, in this order:

- 1. admin/
- 2.
- 3. blog/
- 4. blog/<int:pk>

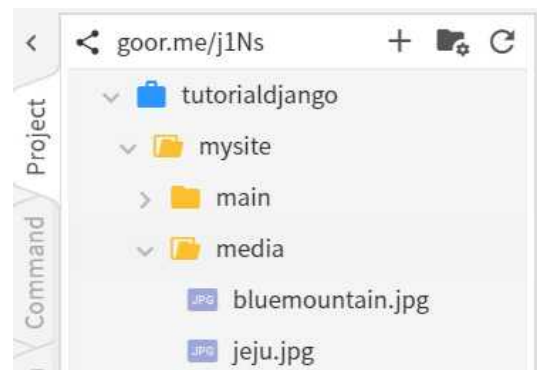
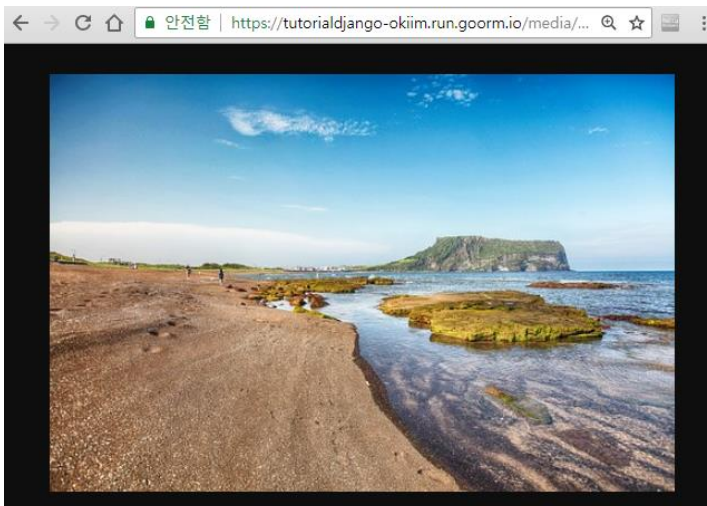
The current path, media/jeju_JLmLF8H.jpg, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

4. Create Post page(cont.)

Add urlpatterns as shown below in the urls.py file . This will allow you to see uploaded images.

```
16 from django.contrib import admin
17 from django.urls import path
18 from main.views import index, blog, postdetails
19 from django.conf.urls.static import static
20 from django.conf import settings
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('', index),
25     path('blog/', blog),
26     path('blog/<int:pk>', postdetails),
27 ]
28
29 urlpatterns += static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
```



4. Create Post page(cont.)

I'll load the image from the postdetails page. Please change the value in the red square to your website's url. For more information about using files in models, please refer to <https://docs.djangoproject.com/en/2.0/topics/files/>.

tutorialdjango/mysite/main/templates/main/postdetails.html

```
1 <html>
2 <head>
3   <title>Django!</title>
4 </head>
5 <body>
6   <h1>Postdetails Page!</h1>
7   <p>{{postlist.postname}}</p>
8   <p>{{postlist.contents}}</p>
9   {% if postlist.mainphoto %}
10    
11  {% endif %}
12  <a href='https://tutorialdjango-okiiim.run.goorm.io/blog/'>list(Go back)</a>
13 </body>
14 </html>
```

- Django official website -

Using files in models

When you use a **FileField** or **ImageField**, Django provides a set of APIs you can use to deal with that file.

Consider the following model, using an **ImageField** to store a photo:

```
from django.db import models

class Car(models.Model):
    name = models.CharField(max_length=255)
    price = models.DecimalField(max_digits=5, decimal_places=2)
    photo = models.ImageField(upload_to='cars')
```

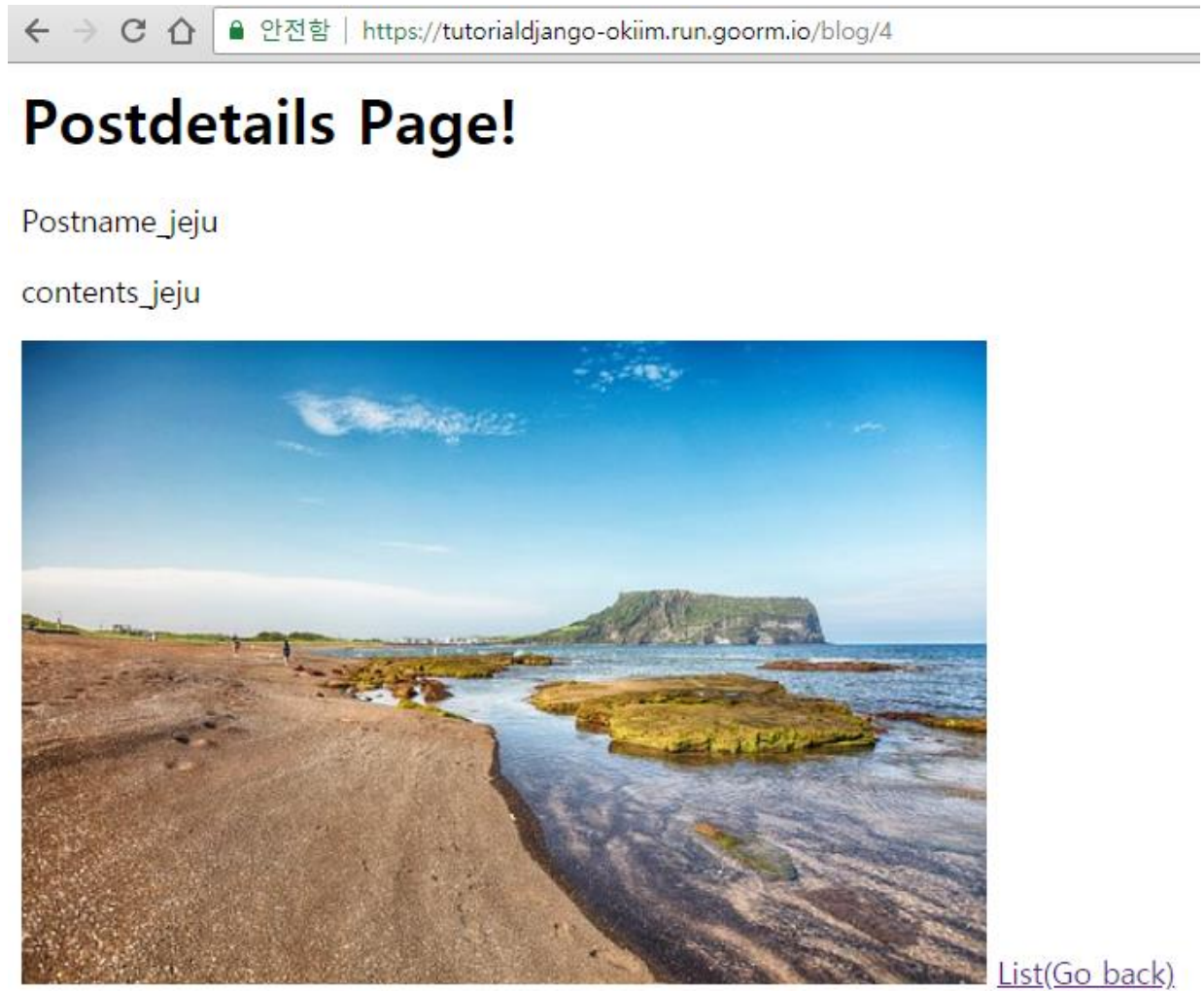
Any **Car** instance will have a **photo** attribute that you can use to get at the details of the attached photo:

```
>>> car = Car.objects.get(name="57 Chevy")
>>> car.photo
<ImageFieldFile: chevy.jpg>
>>> car.photo.name
'cars/chevy.jpg'
>>> car.photo.path
'/media/cars/chevy.jpg'
>>> car.photo.url
'http://media.example.com/cars/chevy.jpg'
```

This object – **car.photo** in the example – is a **File** object, which means it has all the methods and attributes described below.

4. Create Post page(cont.)

Now, you can see the image on the post detail page.



Chapter 5

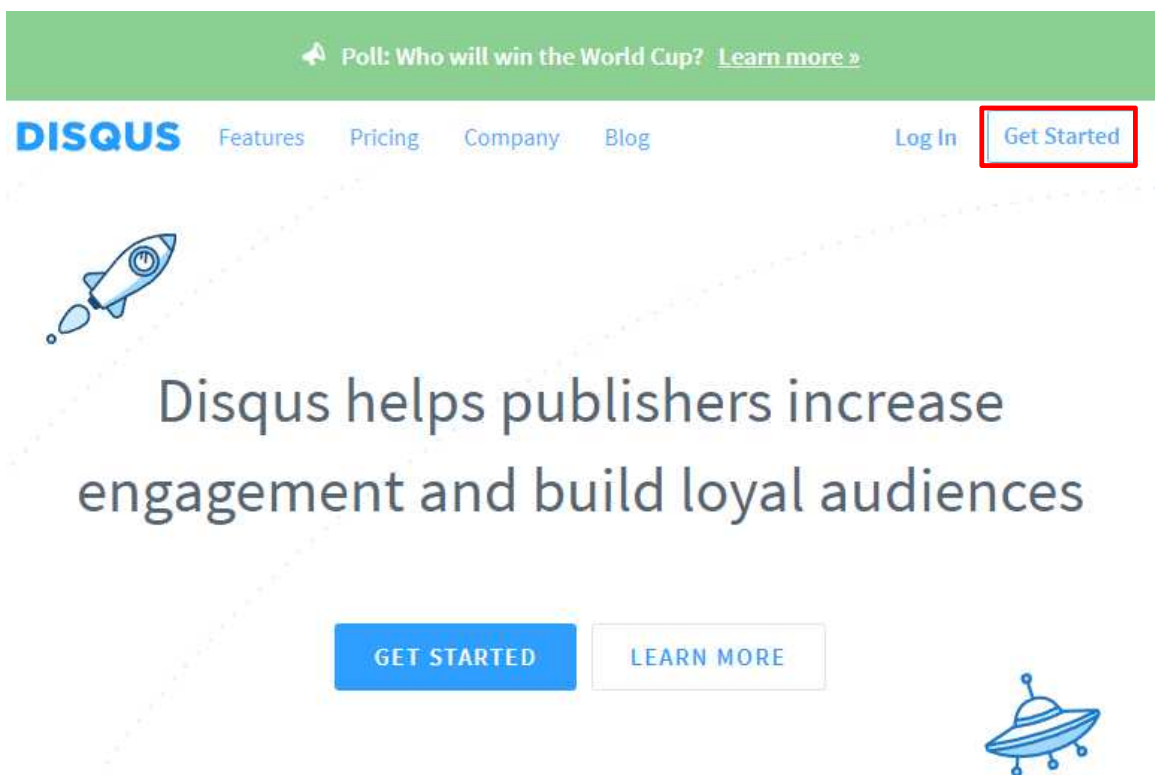
Add Comments Box & Tag

5. Add Comments Box

Now let's set it up that we can comment on it. We are going to use comments plug-in called Disqus. You can install Disqus in Django using the following command, but we will use plug-in in a more convenient way.

Install Disqus

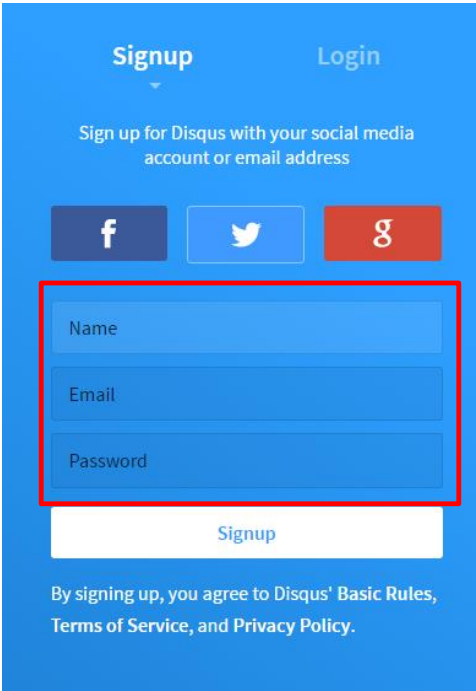
```
(myvenv)root@Goorm:/workspace/container name/mysite# pip install django-disqus
```



<https://disqus.com/>

5. Add Comments Box (cont.)

Now let's set it up so that we can comment on it.



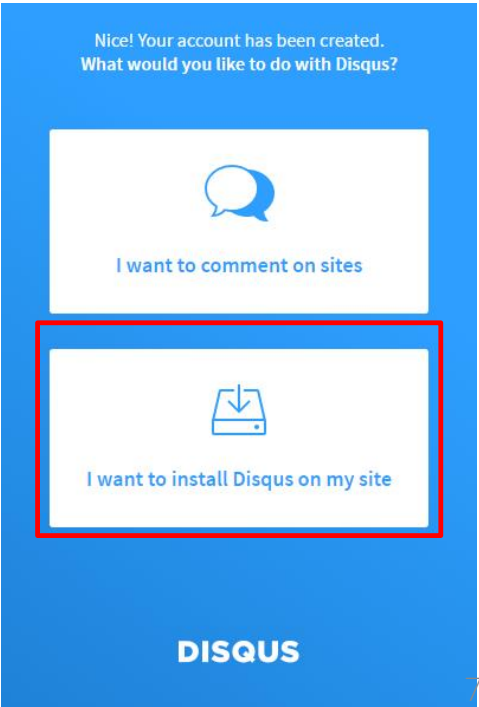
The image shows the Disqus 'Signup' page. At the top, there are links for 'Signup' and 'Login'. Below them is a heading: 'Sign up for Disqus with your social media account or email address'. There are three buttons for social media login: Facebook (f), Twitter (bird), and Google (g). Below these are three input fields: 'Name', 'Email', and 'Password'. These three fields are grouped together and highlighted with a red rectangular border. Below the input fields is a 'Signup' button. At the bottom, there is a line of text: 'By signing up, you agree to Disqus' Basic Rules, Terms of Service, and Privacy Policy.'

Name, E-mail, etc. are easy to manage
Goorm IDE ID, PW enabled

Disqus helps publishers increase engagement and build loyal audiences



The image shows two buttons. The first button is blue with the text 'GET STARTED' in white and is highlighted with a red rectangular border. The second button is white with the text 'LEARN MORE' in blue.



The image shows the Disqus page after a successful signup. At the top, it says 'Nice! Your account has been created. What would you like to do with Disqus?'. Below this are two options, each in a white box with a blue icon and text. The first option has a speech bubble icon and the text 'I want to comment on sites'. The second option has a download icon and the text 'I want to install Disqus on my site'. This second option is highlighted with a red rectangular border. At the bottom of the page is the 'DISQUS' logo in white.

5. Add Comments Box (cont.)

Please fill out the following form (it does not matter what you write). We will install it with Basic option.

Create a new site

All fields are required.

Site Owner

cho

To associate a different account as the site owner,
[login with a different account](#)

Organization

Your Sites

The organization is the group of sites you own.
[Set an organization name](#)

Website Name

tutorialdjango

Your unique Disqus URL will be: tutorialdjango-3.disqus.com
[Customize Your URL](#)

Category

Tech

Language

English

Create Site

Basic

Free, Ads Supported

Any number of total daily pageviews

Get all of the core Disqus features with the option to configure ads.

✓ Comments Plug-in

✓ Advanced Spam Filters

✓ Moderation Tools

✓ Basic Analytics

✓ Configurable Ads

[Learn more about Disqus' Basic features](#)

Subscribe Now

Plus

\$14 \$9 per month

Under 50,000 total daily pageviews

Get everything in Disqus Basic and the option to turn ads on and off.

✓ Everything in Basic

✓ Direct Support

✓ Ads Optional

No credit card required

Start Trial (30 days)

Pro

\$99 \$89 per month

POPULAR

Under 150,000 total daily pageviews

Get everything in Disqus Basic, Plus, and extra premium features.

✓ Everything in Plus

✓ Priority Support

✓ Single Sign-On

✓ Advanced Analytics

✓ Shadow Banning

✓ Timeouts

✓ Email Subscriptions

[Learn more about Disqus' Pro features](#)

Start Trial (30 days)

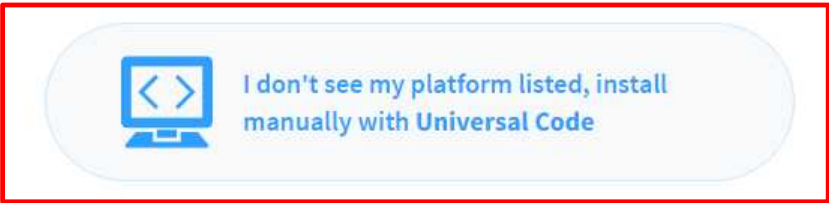
Free

For small, personal, non-commercial sites who do not run any ads. This plan includes everything in Plus except Direct Email Support.

Subscribe Now

5. Add Comments Box (cont.)

If you click Next, you will see several platforms. Click on the bottom one. Then copy all of the sources as image.1 (you will paste it to your project later).



1

Place the following code where you'd like Disqus to load:

```
<div id="disqus_thread"></div>
<script>

/**
 * RECOMMENDED CONFIGURATION VARIABLES: EDIT AND UNCOMMENT THE SECTION BELOW TO INSERT DYNAMIC VALUES FROM YOUR PLATFORM OR CMS.
 * LEARN WHY DEFINING THESE VARIABLES IS IMPORTANT:
 * https://disqus.com/admin/universalcode/#configuration-variables*/
/**
var disqus_config = function () {
this.page.url = PAGE_URL; // Replace PAGE_URL with your page's canonical URL variable
this.page.identifier = PAGE_IDENTIFIER; // Replace PAGE_IDENTIFIER with your page's unique identifier variable
};
```

Configure Disqus

Here are popular configuration settings you can adjust for your site. You can always revisit these later in the settings tab.

Appearance

Color scheme

Auto

Typeface

Auto

Website Name

tutorialdjango

Website URL

Changing domains? [Learn how.](#)

Comment Policy URL

Don't have a comment policy yet? [Check out our suggestions.](#)

Comment Policy

Summary

Ex: We welcome relevant and respectful comments. Off-topic comments may be removed.

A brief summary of your policy that will appear above the comment area. [Collapse preview.](#)

Category

Tech

Description

Language

English

[Make Disqus available in your language.](#)

Complete Setup

5. Add Comments Box (cont.)

Let's put the sources to the project. Open the postdetails.html file and paste the source above the body closing tag `</body>`.

tutorialdjango/mysite/main/templates/main/postdetails.html

```
1 <html>
2 <head>
3   <title>Django!</title>
4 </head>
5 <body>
6   <h1>Postdetails Page!</h1>
7   <p>{{ postlist.postname }}</p>
8   <p>{{ postlist.contents|linebreaks }}</p>
9   {% if postlist.mainphoto %}
10    
11  {% endif %}
12   <a href='https://tutorialdjango-bcrpr.run.goorm.io/blog/'>목록</a>
13   <div id="disqus_thread"></div>
14 <script>
15 (function() {
16   var d = document, s = d.createElement('script');
17   s.src = 'https://tutorialdjango.disqus.com/embed.js';
18   s.setAttribute('data-timestamp', +new Date());
19   (d.head || d.body).appendChild(s);
20 })();
21 </script>
22 <noscript>Please enable JavaScript to view the <a href="https://disqus
23 </a></noscript>
24 </body>
25 </html>
```

5. Add Comments Box (cont.)

When the web server is running you can see comments at the bottom of the post detail page.

안전함

https://tutorialdjango-okiim.run.goorm.io/blog/4

Postdetails Page!

Postname_jeju

contents_jeju



List(Go back)

0 Comments

tutorialdjango

1 cho

Recommend

Share

Sort by Best



Start the discussion...

Be the first to comment.

Subscribe

Add Disqus to your site

Disqus' Privacy Policy

DISQUS

5. Add Comments Box (cont.)

Now, I will add the published date and the modified date.

tutorialdjango/mysite/main/models.py

```
1  from django.db import models
2
3  class Post(models.Model):
4      postname = models.CharField(max_length=50)
5      mainphoto = models.ImageField(blank=True, null=True)
6      publishedDate = models.DateTimeField(blank=True, null=True)
7      modifiedDate = models.DateTimeField(blank=True, null=True)
8      contents = models.TextField()
9
10     def __str__(self):
11         return self.postname
```

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py makemigrations
Migrations for 'main':
  main/migrations/0003_auto_20180628_0554.py
    - Add field modifiedDate to post
    - Add field publishedDate to post
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions
Running migrations:
  Applying main.0003_auto_20180628_0554... OK
(myvenv)root@goorm:/workspace/tutorialdjango/mysite#
```

5. Add Comments Box (cont.)

You can see that the posted date field and the modified date field are added for posting.

Django administration

WELCOME, TUTORIALDJANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts > Postname_1

Change post

HISTORY

Postname:

Postname_1

Mainphoto:

파일 선택

선택된 파일 없음

PublishedDate:

Date:

Today

Time:

Now

Note: You are 9 hours ahead of server time.

ModifiedDate:

Date:

Today

Time:

Now

Note: You are 9 hours ahead of server time.

Contents:

Contents_1

Delete

Save and add another

Save and continue editing

SAVE

5. Add Comments Box (cont.)

Let's modify all posts.

Django administration

WELCOME, TUTORIALDJANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) » [Main](#) » [Posts](#) » Postname_jeju

Change post

HISTORY

Postname:

Postname_jeju

Mainphoto:

Currently: jeju.jpg

Change: 선택된 파일 없음

PublishedDate:

Date:

2018-05-02

Today

Time:

08:32:26

Now

Note: You are 9 hours ahead of server time.

ModifiedDate:

Date:

2018-05-02

Today

Time:

08:32:27

Now

Note: You are 9 hours ahead of server time.

Contents:

contents_jeju

Delete

Save and add another

Save and continue editing

SAVE

5. Add Comments Box (cont.)

If you add `modifiedDate` as below you will see them in on the blog page.

tutorialdjango/mysite/main/templates/main/blog.html

```
1 <html>
2 <head>
3     <title>Django!</title>
4 </head>
5 <body>
6     <h1>Blog Page!</h1>
7     <table>
8         {% for list in postlist %}
9         <tr onclick="location.href='https://t
10             <td>{{list.postname}}</td>
11             <td>{{list.contents}}</td>
12             <td>{{list.modifiedDate}}</td>
13         </tr>
14         {% endfor %}
15     </table>
16 </body>
17 </html>
```

<https://tutorialdjango-bcrpr.run.Goorm.io/blog/>

Blog Page!

Postname_1	Contents_1	May 2, 2018, 8:33 a.m.
Postname_2	Contents_2	May 2, 2018, 8:33 a.m.
Postname_3	Contents_3	May 2, 2018, 8:33 a.m.
Postname_jeju	contents_jeju	May 2, 2018, 8:32 a.m.

5. Add Comments Box (cont.)

We will install [Django-taggit](#) to put tag on your pages. You can use tag to search contents and you can create pages for tags only.

(myvenv)root@Goorm:/workspace/container name/mysite# pip install django-taggit

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# pip install django-taggit
Downloading/unpacking django-taggit
  Downloading django_taggit-0.22.2-py2.py3-none-any.whl (45kB): 45kB downloaded
Installing collected packages: django-taggit
Successfully installed django-taggit
Cleaning up...
```

tutorialdjango/mysite/main/models.py

```
1  from django.db import models
2  from taggit.managers import TaggableManager
3
4  class Post(models.Model):
5      postname = models.CharField(max_length=50)
6      mainphoto = models.ImageField(blank=True, null=True)
7      publishedDate = models.DateTimeField(blank=True, null=True)
8      modifiedDate = models.DateTimeField(blank=True, null=True)
9      contents = models.TextField()
10     tag = TaggableManager(blank=True)
11
12     def __str__(self):
13         return self.postname
```

5. Add Comments Box (cont.)

Let's add [taggit](#) as a application to the setting.py file as below.

```
33 INSTALLED_APPS = [  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles',  
40     'main',  
41     'taggit',  
42 ]
```

tutorialdjango/mysite/tutorialdjango/settings.py

```
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py makemigrations  
Migrations for 'main':  
  main/migrations/0004_post_tag.py  
    - Add field tag to post  
(myvenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py migrate  
Operations to perform:  
  Apply all migrations: admin, auth, contenttypes, main, sessions, taggit  
Running migrations:  
  Applying taggit.0001_initial... OK  
  Applying taggit.0002_auto_20150616_2121... OK  
  Applying main.0004_post_tag... OK
```

5. Add Comments Box (cont.)

Go to the admin page and check if tags field appears at the bottom. We will put the traveled countries names here.

Django administration

WELCOME, TUTORIALDJANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts > Postname_1

Change post

HISTORY

Postname:

Postname_1

Mainphoto:

Currently: tasmania.jpg ☐ Clear

Change: 선택된 파일 없음

PublishedDate:

Date: 2018-06-28

Today

Time: 05:57:18

Now

Note: You are 9 hours ahead of server time.

ModifiedDate:

Date: 2018-06-28

Today

Time: 05:57:20

Now

Note: You are 9 hours ahead of server time.

Contents:

Contents_1

Tags:

Australia

A comma-separated list of tags.

Delete

Save and add another

Save and continue editing

SAVE

5. Add Comments Box (cont.)

Enter the Tags, press SAVE, and you will see the tags on the postdetails page.

Tags:

A comma-separated list of tags.

[Delete](#) [Save and add another](#) [Save and continue editing](#) [SAVE](#)

tutorialdjango/mysite/main/templates/main/postdetails.html

```
1 <html>
2 <head>
3     <title>Django!</title>
4 </head>
5 <body>
6     <h1>Postdetails Page!</h1>
7     <p>{{ postlist.postname }}</p>
8     <p>{{ postlist.contents|linebreaks }}</p>
9     {% if postlist.mainphoto %}
10         
11     {% endif %}
12     <p>{{ postlist.tag.names }}</p>
13     <br>
14     <a href='https://tutorialdjango-bcrpr.run.goorm.io/blog/'>목록</a><br>
15     <div id="disqus_thread"></div>
16 <script>
17 (function() {
18     var d = document, s = d.createElement('script');
19     s.src = 'https://tutorialdjango.disqus.com/embed.js';
20     s.setAttribute('data-timestamp', +new Date());
21     (d.head || d.body).appendChild(s);
22 })();
23 </script>
24 <noscript>Please enable JavaScript to view the <a href="https://disqus.com">
25 </a></noscript>
26 <body>
27 <html>
```

5. Add Comments Box (cont.)

The tag is shown with the phrase Query Set.

Postdetails Page!

Postname_1

Contents_1



<QuerySet ['Australia']>

[List](#)([Go back](#))

0 Comments tutorialdjango

 cho ▾

 Recommend  Share

Sort by Best ▾



Start the discussion...

Be the first to comment.

5. Add Comments Box (cont.)

If you use “for loop” you can have multiple tags shown. Now we will use only one tag so put 0 at the end of the tag list.

```
1 <html>
2   <head>
3     <title>Django!</title>
4   </head>
5   <body>
6     <h1>Postdetails Page!</h1>
7     <p>{{postlist.postname}}</p>
8     <p>{{postlist.contents}}</p>
9     {% if postlist.mainphoto %}
10    
11  {% endif %}
12    <p>{{postlist.tag.names.0}}</p>
13    <a href='https://tutorialdjango-okiim.run.goorm.io/blog/'>List(Go back)</a>
14  </body>
</html>
```

Postdetails Page!

Postname_1

Contents_1



Australia

[List\(Go back\)](#)

0 Comments tutorialdjango

Recommend Share

1 cho

Sort by Best

tutorialdjango requires you to verify your email address before posting. Send verification email to kalistar11@naver.com

 Start the discussion...

Be the first to comment.

Chapter 6

Design blog with bootstrap & Deploy

6.1. Bootstrap Introduction

6.2. Bootstrap Download

6.3. Create a Template

6.4. Bootstrap Basic

6.5. Finishing touches

with template code examples

6.6. Deployment

TUTORIAL 6.1

Bootstrap Introduction

Bootstrap is a web framework. It's easy to think of a framework as a comprehensive set of components that you can use to get results quickly with minimal effort.

Bootstrap is optimized for the user's point of view, regardless of whether it looks like a big monitor screen or a small screen like a mobile phone. This site is tailored to the user's point of view and is called a responsive site. Bootstrap is optimized to build a responsive site.

Bootstrap is pre-designed with HTML, CSS, form, button, table, navigator. In Bootstrap 4, most of the tags used in bootstrap 3 are still used, but many have changed, so you should study the new version.

You can find a tutorial on bootstrap 4 at <https://www.w3schools.com/bootstrap4/>. For more information, check out the official Bootstrap homepage at <https://getbootstrap.com/docs/4.1/content/reboot/>.

TUTORIAL 6.2

Bootstrap Download

You can download bootstrap 4 at the <https://getbootstrap.com/>

* It is not necessary to download in this tutorial.

– Bootstrap Homepage in English: [https://getbootstrap.com/\(v4\)](https://getbootstrap.com/(v4))

 [Home](#) [Documentation](#) [Examples](#) [Themes](#) [Expo](#) [Blog](#) v4.1 ▾    [Download](#)

Bootstrap

Build responsive, mobile-first projects on the web with the world's most popular front-end component library.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

[Get started](#) [Download](#)

Currently v4.1.2



Students and Teachers,
save up to 60% on Adobe
Creative Cloud.
ads via Carbon



TUTORIAL 6.3

Create a Template

The default template was created using CDN (Content delivery network / content distribution network). CDN allows you to use it without downloading it.

```
1 <!doctype html>
2 <html lang="ko">
3   <head>
4     <!-- Required meta tags -->
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7
8     <!-- Bootstrap CSS -->
9     <link rel="stylesheet"
10       * href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css"
11       * integrity="sha384-9gVQ4dYFwwWSjIDZnLEWnxCjeSFWFphJiwGPXr1jddIhOegiu1Fw05qRGvFXOdJZ4"
12       * crossorigin="anonymous">
13
14     <title>Hello, world!</title>
15   </head>
16   <body>
17     <h1>Hello, world!</h1>
18
19     <!-- Optional JavaScript -->
20     <!-- jQuery first, then Popper.js, then Bootstrap JS -->
21     <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
22       * integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
23       * crossorigin="anonymous"></script>
24     <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js"
25       * integrity="sha384-cs/chFZiN24E4KMATLdqvsezeGxaGsi4hLGOzlXwp5UZB1LY//20VyM2taTB4QvJ"
26       * crossorigin="anonymous"></script>
27     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js"
28       * integrity="sha384-uefMccjFJAIv6A+rW+L4AHf99KvxDjWSu1z9VI8SKNVmz4sk7buKt/6v9KI65qnm"
29       * crossorigin="anonymous"></script>
30   </body>
31 </html>
```

001.html 24:1

001.html

If you want to use the latest version of CDN, search for the ‘bootstrap cdn’ on google and go to <https://www.bootstrapcdn.com/>. The latest version is 4.

Google

bootstrap cdn

전체

이미지

동영상

뉴스

도서

더보기

설정

도구

검색결과 약 15,600,000개 (0.50초)

Quick Start · BootstrapCDN by StackPath

<https://www.bootstrapcdn.com/>

이 페이지 번역하기

The recommended CDN for Bootstrap, Font Awesome and Bootswatch.

Quick Start

v4.1.1

Complete CSS

https://stackpath.bootstrapcdn.com/bootstrap

Click to copy

Complete JavaScript

https://stackpath.bootstrapcdn.com/bootstrap

Click to copy

Complete JavaScript Bundle

https://stackpath.bootstrapcdn.com/bootstrap

Click to copy

TUTORIAL 6.4

Bootstrap Basic

Within the body tag `<body> </ body>`, there are script tags `<script> </ script>` for the CDN . The code for designing the templates is added as shown below in the red square.

```
1  <!doctype html>
2  <html lang="ko">
3    <head>
4      <!-- Required meta tags -->
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7
8      <!-- Bootstrap CSS -->
9      <link rel="stylesheet"
10        * href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css"
11        * integrity="sha384-9gVQ4dYFwwWSjIDZnLEWnCjeSWFphJiwGPXr1jddIhOegiu1Fw05qRGvFXOdJZ4"
12        * crossorigin="anonymous">
13
14      <title>Hello, world!</title>
15    </head>
16    <body>
17      <h1>Hello, world!</h1>
18
19      <!-- Optional JavaScript -->
20      <!-- jQuery first, then Popper.js, then Bootstrap JS -->
21      <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
22        * integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
23        * crossorigin="anonymous"></script>
24      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js"
25        * integrity="sha384-cs/chFZiN24E4KMATLdqvseZGxaGsi4hLGOzlXwp5UZB1LY//20VyM2taTB4QvJ"
26        * crossorigin="anonymous"></script>
27      <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js"
28        * integrity="sha384-uefMccjFJAIiv6A+rW+L4AHf99KvxDjWSu1z9VI8SKNVmz4sk7buKt/6v9KI65qnm"
29        * crossorigin="anonymous"></script>
30    </body>
31  </html>
```

001.html 24:1

001.html

6.4 Bootstrap Grids

Bootstrap grid system provides the quick and easy way to create responsive website layouts. Bootstrap grid system allows to display up to 12 columns in a row across the page and if you do not want to use all 12 column individually, you can group the columns together to create wider columns

Bootstrap grid examples

Basic grid layouts to get you familiar with building within the Bootstrap grid system.

Five grid tiers

There are five tiers to the Bootstrap grid system, one for each range of devices we support. Each tier starts at a minimum viewport size and automatically applies to the larger devices unless overridden.

.col-4	.col-4	.col-4
.col-sm-4	.col-sm-4	.col-sm-4
.col-md-4	.col-md-4	.col-md-4
.col-lg-4	.col-lg-4	.col-lg-4
.col-xl-4	.col-xl-4	.col-xl-4

Three equal columns

Get three equal-width columns **starting at desktops and scaling to large desktops**. On mobile devices, tablets and below, the columns will automatically stack.

.col-md-4	.col-md-4	.col-md-4
-----------	-----------	-----------

Three unequal columns

Get three columns **starting at desktops and scaling to large desktops** of various widths. Remember, grid columns should add up to twelve for a single horizontal block. More than that, and columns start stacking no matter the viewport.

.col-md-3	.col-md-6	.col-md-3
-----------	-----------	-----------

Two columns

Get two columns **starting at desktops and scaling to large desktops**.

.col-md-8	.col-md-4
-----------	-----------

Full width, single column

No grid classes are necessary for full-width elements.

6.4 Bootstrap Grids (cont.)

After type the codes as shown below in the `<body> </body>` tag , save it as 002.html and execute it. The red squares represent defined grid in one row. Four small size columns are grouped in one medium column so you can see three columns.

Code

```
14      <div class="container">
15          <div class="row">
16              <div class="col-md-4">
17                  <h1>hello</h1>
18              </div>
19
20              <div class="col-md-4">
21                  <h1>hello</h1>
22              </div>
23
24              <div class="col-md-4">
25                  <h1>hello</h1>
26              </div>
27          </div>
28      </div>
```

Output

hello

4컬럼

hello

4컬럼

hello

4컬럼

002.html

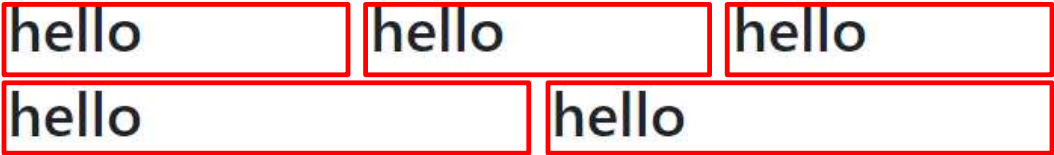
6.4 Bootstrap Grids (cont.)

We use **.col-md-4** class and **.col-md-6** class to compare how grid looks like with each class

Code

```
14      <div class="container">
15          <div class="row">
16              <div class="col-md-4">
17                  <h1>hello</h1>
18              </div>
19              <div class="col-md-4">
20                  <h1>hello</h1>
21              </div>
22              <div class="col-md-4">
23                  <h1>hello</h1>
24              </div>
25          </div>
26      <div class="row">
27          <div class="col-md-6">
28              <h1>hello</h1>
29          </div>
30          <div class="col-md-6">
31              <h1>hello</h1>
32          </div>
33      </div>
34  </div>
```

Output



003.html

6.4 Bootstrap Grids (cont.)

As you can see the official website below, it is also possible to subdivide one column unit.

Two columns with two nested columns

Per the documentation, nesting is easy—just put a row of columns within an existing column. This gives you two columns **starting at desktops and scaling to large desktops**, with another two (equal widths) within the larger column.

At mobile device sizes, tablets and down, these columns and their nested columns will stack.

.col-md-8		.col-md-4
.col-md-6	.col-md-6	

Mixed: mobile and desktop

The Bootstrap v4 grid system has five tiers of classes: xs (extra small), sm (small), md (medium), lg (large), and xl (extra large). You can use nearly any combination of these classes to create more dynamic and flexible layouts.

Each tier of classes scales up, meaning if you plan on setting the same widths for xs and sm, you only need to specify xs.

.col-12 .col-md-8		.col-6 .col-md-4	
.col-6 .col-md-4	.col-6 .col-md-4		.col-6 .col-md-4
.col-6		.col-6	

Mixed: mobile, tablet, and desktop

.col-12 .col-sm-6 .col-lg-8		.col-6 .col-lg-4	
.col-6 .col-sm-4	.col-6 .col-sm-4		.col-6 .col-sm-4

<https://getbootstrap.com/docs/4.1/examples/grid/>

6.4 Bootstrap Grids (cont.)

If you want to use all 12 columns as a 100% of the screen width in a grid system, use the **.container-fluid** class.

You can remove padding and margin between columns by applying the **.no-gutters** class to the **.row** class.

Below is a description of the grid system on the official website. **Nestable** means you can split a column.

We will use **md** class mainly .

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

<https://getbootstrap.com/docs/4.1/layout/grid/>

6.4 Bootstrap Grids (cont.)

If offset is used, the space of one column is floated. There are many ways to give margins, but we mainly use the offset method and the Flexbox method to sort columns. Only offset is covered here.

Code

```
14      <div class="container">
15          <div class="row">
16              <div class="col-md-5 offset-md-1">
17                  <h1>hello</h1>
18              </div>
19              <div class="col-md-5">
20                  <h1>hello</h1>
21              </div>
22          </div>
23      <div class="row">
24          <div class="col-md-6">
25              <h1>hello</h1>
26          </div>
27          <div class="col-md-6">
28              <h1>hello</h1>
29          </div>
30      </div>
31  </div>
```

Output

hello
hello

hello
hello

003.html

TUTORIAL 6.5

Finishing touches with template code examples

When you extract **templates.zip** file there will be index.html, blog.html, postdetails.html files in it. If you run index.html, you can see the map with some markers. Coordinates are required for each post in order to draw a marker on the map. **LatLng** in the codes below is latitude and longitude.

Next, we will use the template tag to insert coordinates of each post.

```
285     function initMap() {
286
287         var bluemountain = new google.maps.LatLng(-33.3493206,149.7360613);
288         var jejucityhall = new google.maps.LatLng(33.499597,126.5290653);
289         var perth = new google.maps.LatLng(-32.0388312,115.4010747);
290         var sydney = new google.maps.LatLng(-33.8567844,151.213108);
291         var tasmania = new google.maps.LatLng(-42.200633, 146.643736);
```

Index_001.html

* [Download Source Code Examples: www.paullab.co.kr/templates.zip](http://www.paullab.co.kr/templates.zip)

6.5 Finishing touches with template code examples

Let's add codes as shown below in the `models.py` file.

The official document contains blank and null references.

Official document:

<https://docs.djangoproject.com/en/2.0/ref/models/fields/#django.db.models.FloatField>

```
1 from django.db import models
2 from taggit.managers import TaggableManager
3
4 class Post(models.Model):
5     postname = models.CharField(max_length=50)
6     Lat = models.FloatField(null=True)
7     Lng = models.FloatField(null=True)
8     mainphoto = models.ImageField(blank=True, null=True)
9     publishedDate = models.DateTimeField(blank=True, null=True)
10    modifiedDate = models.DateTimeField(blank=True, null=True)
11    contents = models.TextField()
12    tag = TaggableManager(blank=True)
13
14    def __str__(self):
15        return self.postname
```

tutorialdjango/mysite/main/models.py

null

If **True**, Django will store empty values as **NULL** in the database. Default is **False**.

blank

If **True**, the field is allowed to be blank. Default is **False**.

Note that this is different than **null**. **null** is purely database-related, whereas **blank** is validation-related. If a field has **blank=True**, form validation will allow entry of an empty value. If a field has **blank=False**, the field will be required.

Django official documentation about a null & blank part

6.5 Finishing touches with template code examples (cont.)

Type these commands in the console window and execute to apply all migrations.

```
python manage.py make migrations
```

```
python manage.py migrate
```

Then let's run the web server.

```
python manage.py runserver 0:80
```

```
(myenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py makemigrations
Migrations for 'main':
  main/migrations/0005_auto_20180504_0204.py
    - Add field Lat to post
    - Add field Lng to post
(myenv)root@goorm:/workspace/tutorialdjango/mysite# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions, taggit
Running migrations:
  Applying main.0005_auto_20180504_0204... OK
```

6.5 Finishing touches with template code examples (cont.)

You can see the fields **Lat** and **Lng** for the geographical coordinates (as shown below) If you have forgotten your ID and password, you can re-create it in the console window.

Django administration

WELCOME, TUTORIALDJANGO. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Main > Posts > Postname_1

Change post

HISTORY

Postname:

Postname_1

Lat:

Lng:

Mainphoto:

Currently: tasmania.jpg ☐ Clear

Change: 선택된 파일 없음

PublishedDate:

Date: 2018-06-28

Today |

Time: 05:57:18

Now |

Note: You are 9 hours ahead of server time.

ModifiedDate:

Date: 2018-06-28

Today |

Time: 05:57:20

Now |

Note: You are 9 hours ahead of server time.

Contents:

Contents_1

Tags:

Australia

A comma-separated list of tags.

Delete

Save and add another

Save and continue editing

SAVE

6.5 Finishing touches with template code examples (cont.)

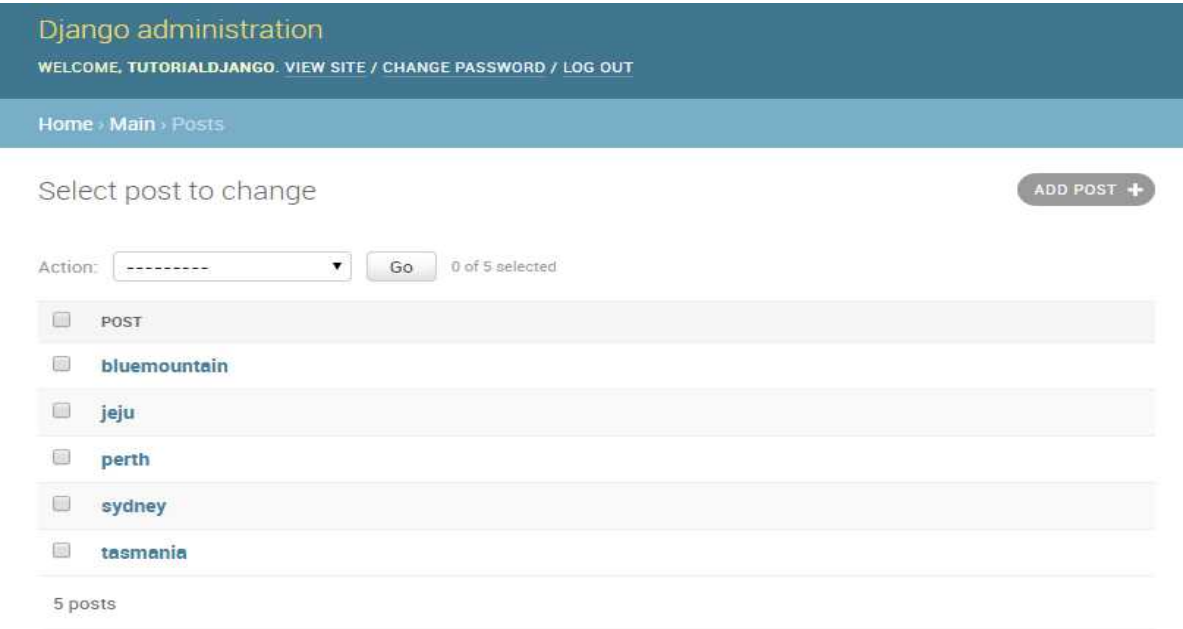
Let's put these value below to the Lat field and the Lng field of each post.

bluemountain, latitude -33.3493206, longitude 149.7360613
jejucityhall, latitude 33.499597, longitude 126.5290653
perth, latitude -32.0388312, longitude 115.4010747
sydney, latitude -33.8567844, longitude 151.213108
tasmania, latitude -42.200633, longitude 146.643736

I will put contents from the **lorem ipsum** web site and upload a photo from the template/img folder

lorem ipsum : <https://lipsum.com/>

* I will remove all posts except the 5 below.



6.5 Finishing touches with template code examples (cont.)

Now, let's check if the content that we posted is applied to the web site. First, Add the codes to the views.py file as below so that you can use postlist at the index.html. Then, copy the whole codes from the templates/Index_002.html and paste them to the index.html file in your project. The difference between the original index.html file and the index_002.html is shown below.

Then, If you restart the web server, you can see the broken image with the marker as shown below. Next, let's modify the index.html file.

```
4 def index(request):
5     postlist = Post.objects.all()
6     return render(request, 'main/index.html', {'postlist':postlist})
```

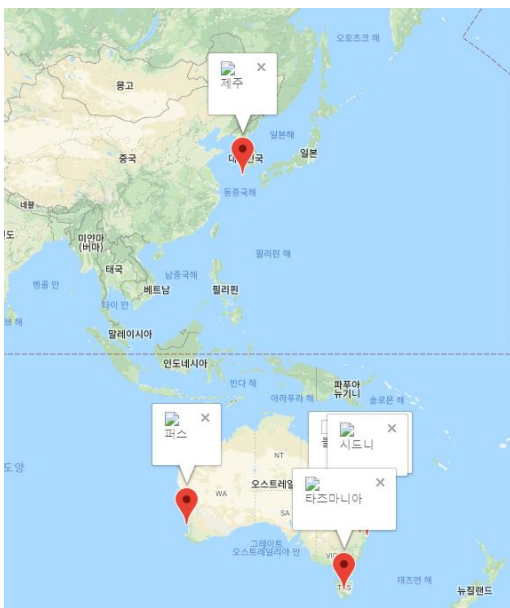
tutorialdjango/mysite/main/views.py

```
265 function initMap() {
266
267     var bluemountain = new google.maps.LatLng(-33.3493206,149.7360613);
268     var jejuicityhall = new google.maps.LatLng(-33.499597,126.5290653);
269     var perth = new google.maps.LatLng(-32.0388312,115.4018747);
270     var sydney = new google.maps.LatLng(-33.8567844,151.2131808);
271     var tasmania = new google.maps.LatLng(-42.208633, 146.643736);
272 }
```

[Index.html](#)

```
285 function initMap() {
286     % for list in postlist %
287     var {{list.postname}} = new google.maps.LatLng({{list.Lat}},{{list.Lng}});
288     % endfor %
}
```

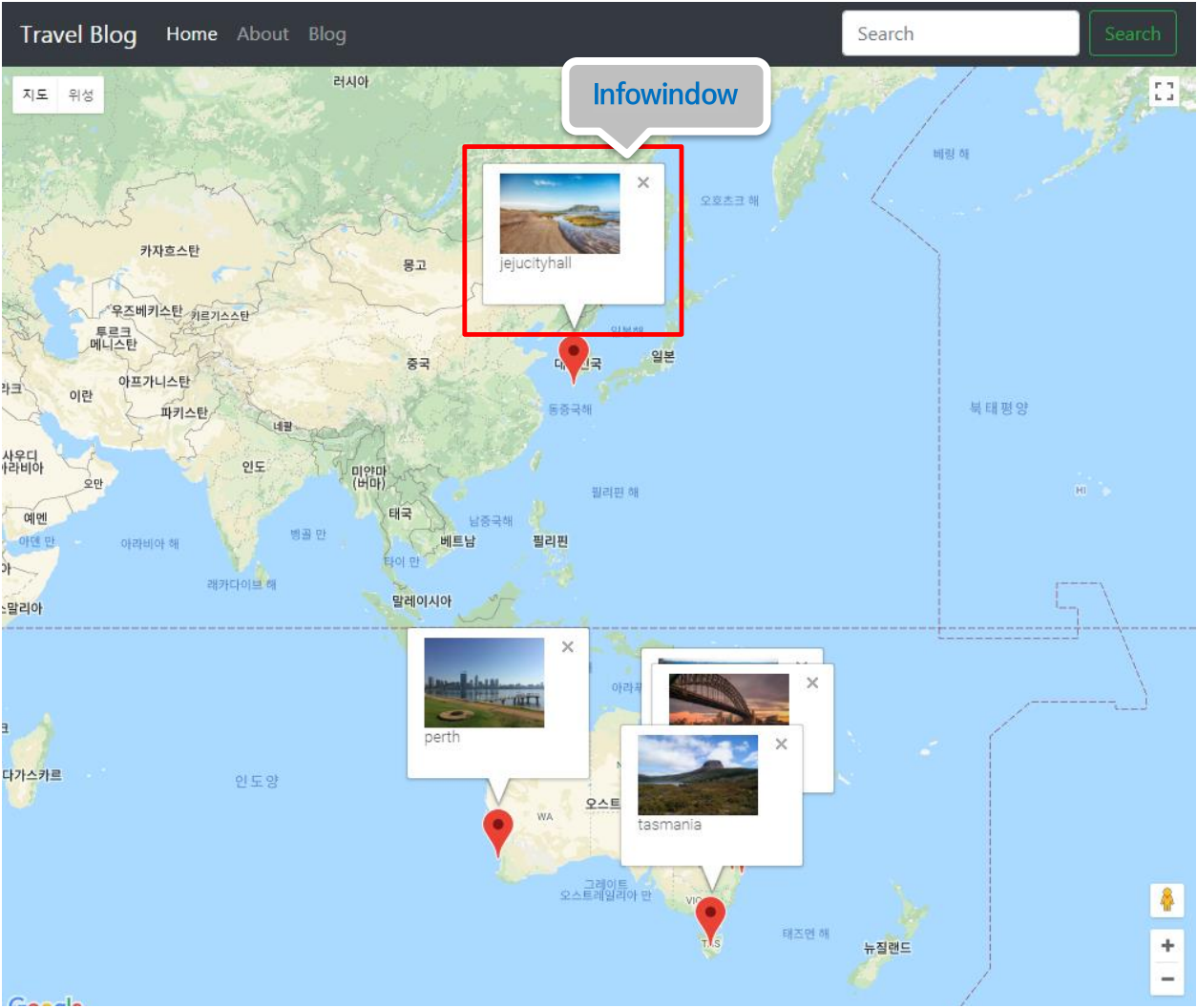
[Index_002.html](#)



If you can not see this page restart the web server

6.5 Finishing touches with template code examples (cont.)

Let's use template tag `{% %},{}` to put a photo to a [Infowindow](#) which is in the red square of each post. If you click on a photo, the blog page will change to the post detail page.



6.5 Finishing touches with template code examples (cont.)

Then, copy the entire codes from the [templates/index_003.html](#) file and paste them to the [index.html](#) file in your project.

```
295
296 var bluemountainmarker = new google.maps.Marker({
297   position: bluemountain,
298   map: map
299 });
300
301 var infowindow = new google.maps.InfoWindow({
302   content: "<a href='#'><img width='100px;' src='img/bluemountain.jpg'></a><br><
303 });
304 infowindow.open(map,bluemountainmarker);
305
306 var jejuhallmarker = new google.maps.Marker({
307   position: jejuhall,
308   map: map
309 });
310 var infowindow = new google.maps.InfoWindow({
311   content: "<a href='#'><img width='100px;' src='img/jeju.jpg'></a><br><
312 });
313 infowindow.open(map,jejuhallmarker);
314
315 var perthmarker = new google.maps.Marker({
316   position: perth,
317   map: map
318 });
319 var infowindow = new google.maps.InfoWindow({
320   content: "<a href='#'><img width='100px;' src='img/perth.jpg'></a><br>
321 });
322 infowindow.open(map,perthmarker);
323
324 var sydneymarker = new google.maps.Marker({
325   position: sydney,
326   map: map
327 });
328 var infowindow = new google.maps.InfoWindow({
329   content: "<a href='#'><img width='100px;' src='img/sydney.jpg'></a><br>
330 });
331 infowindow.open(map,sydneymarker);
332
333 var tasmaniamarker = new google.maps.Marker({
334   position: tasmania,
335   map: map
336 });
337 var infowindow = new google.maps.InfoWindow({
338   content: "<a href='#'><img width='100px;' src='img/tasmania.jpg'></a><br>
339 });
340 infowindow.open(map,tasmaniamarker);
341
```

```
295
296 {% for list in postlist %}
297 var {{list.postname}}marker = new google.maps.Marker({
298   position: {{list.postname}},
299   map: map
300 });
301 var infowindow = new google.maps.InfoWindow({
302   content: "<a href='https://tutorialdjango-bcrpr.run.goorm.io/blog/{{ list.postname }}'><img width='100px;' src='{{ list.mainphoto.url }}'></a><br><p>{{list.postname}}</p>
303 });
304
305 infowindow.open(map,{{list.postname}}marker);
306 {% endfor %}
307
```

[Index.html](#)

[Index_003.html](#)

6.5 Finishing touches with template code examples (cont.)

Now, Let's modify the blog.html file. Just copy the whole codes from the templates/blog.html file and paste them to the blog.html file in your project. The Most important part of those codes is as shown below (from the line 161 to the line 180).

Find the anchor tag `<a href=...></a>` and put your url in the red square as below.

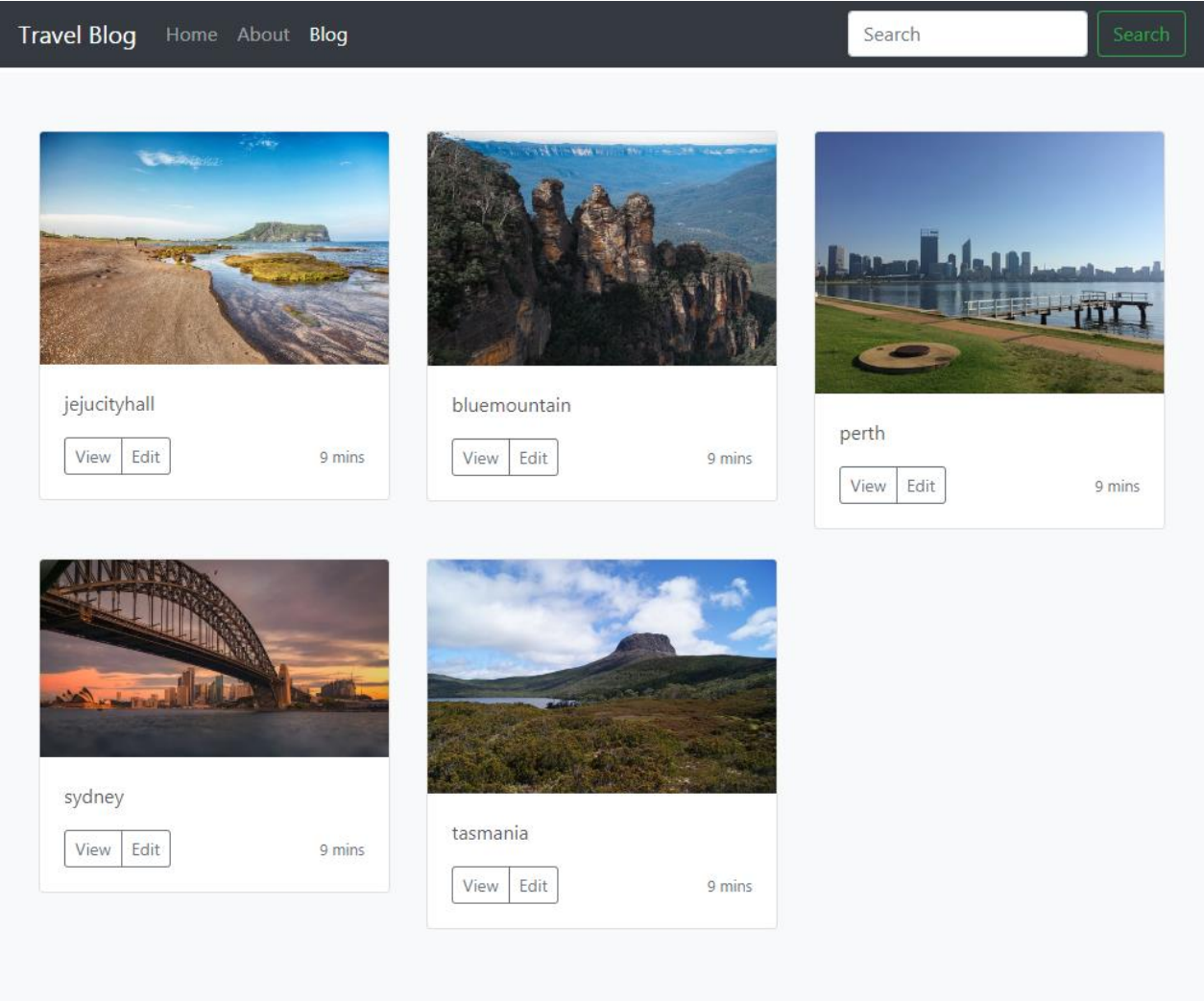
```
{% for list in postlist %}
<div class="col-md-4">
  <div class="card mb-4 box-shadow">
    {% if list.mainphoto %}
      <a href="https://tutorialdjango-bcrpr.run.goorm.io/blog/{{ list.pk
      }}"></a>
    {% endif %}
    <div class="card-body">
      <p class="card-text">{{list.postname}}</p>
      <div class="d-flex justify-content-between align-items-center">
        <div class="btn-group">
          <button type="button" class="btn btn-sm
          btn-outline-secondary">View</button>
          <button type="button" class="btn btn-sm
          btn-outline-secondary">Edit</button>
        </div>
        <small class="text-muted">9 mins</small>
      </div>
    </div>
  </div>
</div>
{% endfor %}
```

Put your url here instead of this.

blog.html

6.5 Finishing touches with template code examples (cont.)

I run the web server and execute the project's url. When you click on each photo here, the blog page will be moved to the post detail page.



The blog page

6.5 Finishing touches with template code examples (cont.)

Now let's modify `postdetails.html`. Just copy the entire codes from the `templates/blogdetails.html` file and paste them to the `postdetails.html` file in your project. We use template tags `{% %}`, `{ { }}` to use `postlist`'s element as below (from the line 147 to the line 166).

Find the anchor tag `<a href=...></a>` and put your url in the red square as below.

```
<div class="blog-post">
  <h2 class="blog-post-title">{{ postlist.postname }}</h2>
  <p class="blog-post-meta">Modified Date : {{ postlist.modifiedDate }}</p>
  <p class="blog-post-meta">Published Date : {{ postlist.publishedDate }}</p>
  {% if postlist.mainphoto %}
    
  {% endif %}

  <p>{{ postlist.contents|linebreaks }}</p>
  <h2>tag</h2>
  <p>{{ postlist.tag.names.0 }}</p>

  <div id="disqus_thread"></div>

</div><!-- /.blog-post -->
<nav class="blog-pagination">
  <a class="btn btn-outline-primary" href="https://https://tutorialdjango-okiim.run.goorm.io/blog/">
    List</a>
</nav>
```

postdetails.html

6.5 Finishing touches with template code examples (cont.)

Here is the executed post detail page

Travel Blog

Home

About

Blog

Search

Search

From the Firehose

tasmania

Modified Date : June 28, 2018, 7:42 a.m.

Published Date : June 28, 2018, 5:57 a.m.



Maecenas eros tellus, dapibus at sem in, lobortis consectetur erat. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam erat volutpat. Quisque ac turpis sit amet purus fringilla congue eu et dolor. Phasellus risus urna, gravida vel cursus eget, facilisis vel felis. Cras eu pulvinar ipsum. Quisque vel fringilla orci. Fusce dictum purus elit, vulputate pulvinar odio hendrerit vel. Vestibulum tristique tempus nibh, quis rutrum tellus aliquam quis. Quisque commodo ante orci, vitae tempus enim tincidunt id. Donec imperdiet, enim a dictum auctor, nisi velit condimentum erat, non pharetra orci sem ac lorem. Nullam bibendum massa nulla. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

tag

Australia

1 Comment

tutorialdjango

Login

Recommend

Share

Sort by Best

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

Name

About

Etiam porta sem malesuada magna mollis euismod. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur.

image carousel 1

image1

contents1

The postdetails page

117

6.5 Finishing touches with template code examples (cont.)

I did not use [template inheritance](#) for this tutorial. But most of time when you create a web service you will use [template inheritance](#).

Usually, we create templates of menus, headers, footers or etc. And add them in every single page through template inheritance .

Here is a template inheritance example from the official document.

year_archive.html inherits **base.html**.

```
mysite/templates/base.html
```

```
{% load static %}
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    
    {% block content %}{% endblock %}
</body>
</html>
```

```
mysite/news/templates/news/year_archive.html
```

```
{% extends "base.html" %}

{% block title %}Articles for {{ year }}{% endblock %}

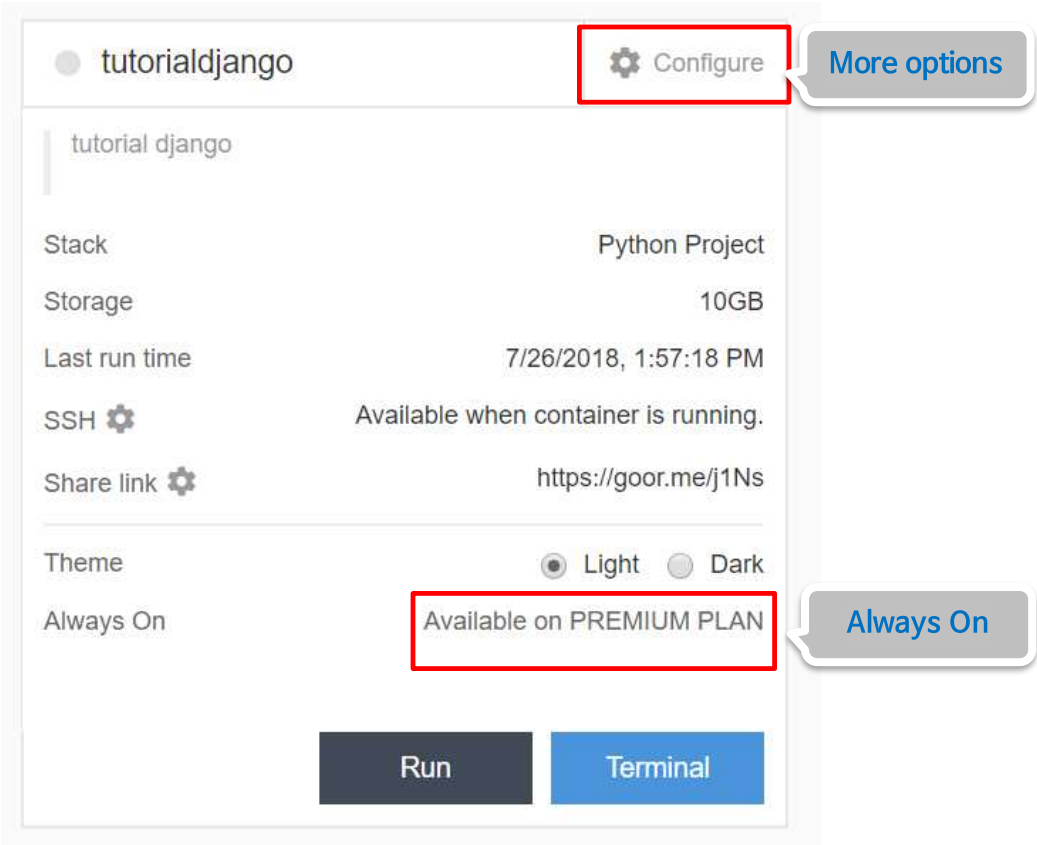
{% block content %}
<h1>Articles for {{ year }}</h1>

{% for article in article_list %}
    <p>{{ article.headline }}</p>
    <p>By {{ article.reporter.full_name }}</p>
    <p>Published {{ article.pub_date|date:"F j, Y" }}</p>
{% endfor %}
{% endblock %}
```

TUTORIAL 6.6

Deployment

In the Goorm IDE, the [Always On](#) option is only available for Premium accounts. [Always On](#) is activated when you click on it then you can always access your travel blog. Click on the button in the red square to see the detailed options.



6.6 Deployment

Press the

+ Add

 button in the second red square and add the port and the URL which you purchased. In the [Init script](#) I usually put those two commands as below.

```
source /workspace/

container name

/mysite/myenv/bin/activate
python3 /workspace / 

container name

 /mysite/manage.py runserver 0:80
```

Setting

Always-on

Init Script

Init script

Available when purchasing PREMIUM plan

Unset Set

1 Script goes here...

Stop this container

Stop

All running 'tutorialdjango' container will be stopped.

Delete this container

Delete

This action can not be undone!

Manage Network Settings

Add URL & Port

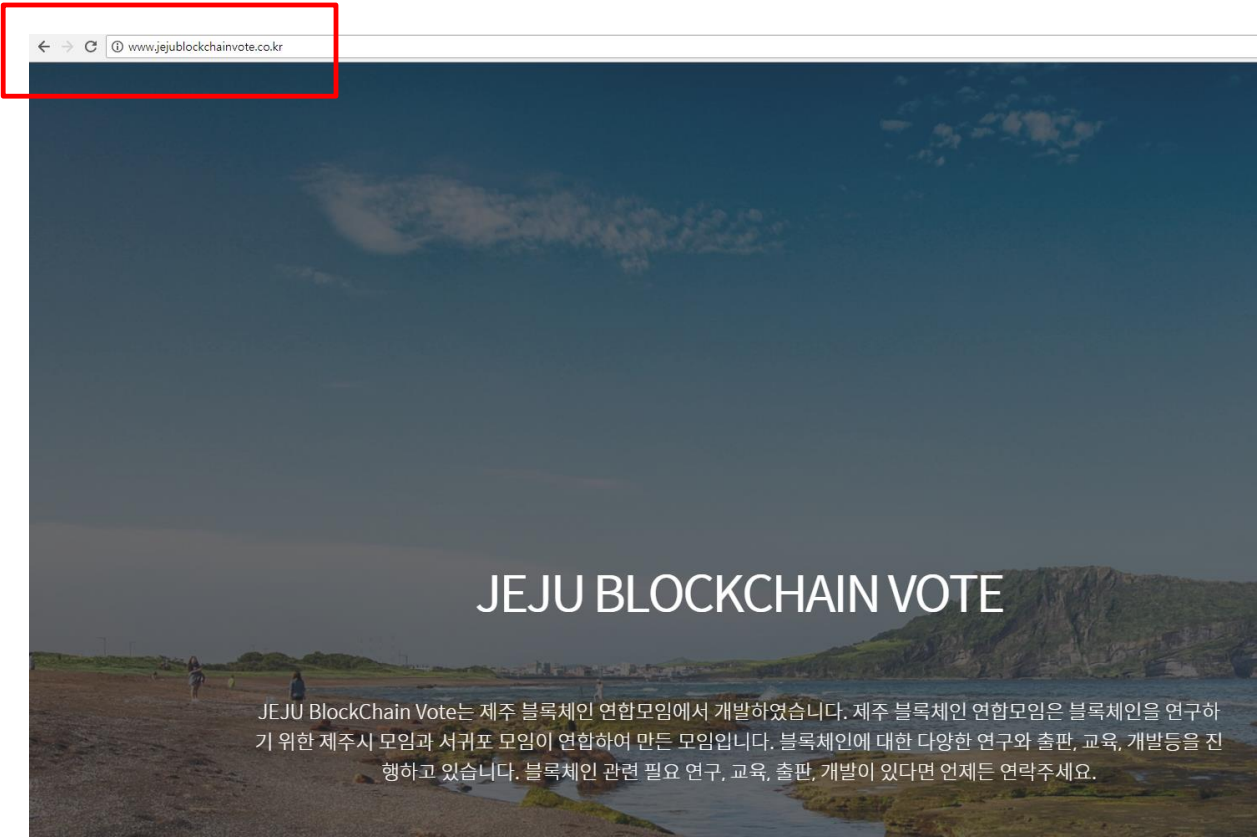
+ Add Edit

Run URL & Port

URL	Port
tutorialdjango-okiiim.run.goorm.io	80

6.6 Deployment (cont.)

Please verify that the URL is properly linked and working.



Chapter 7

Summary

Django

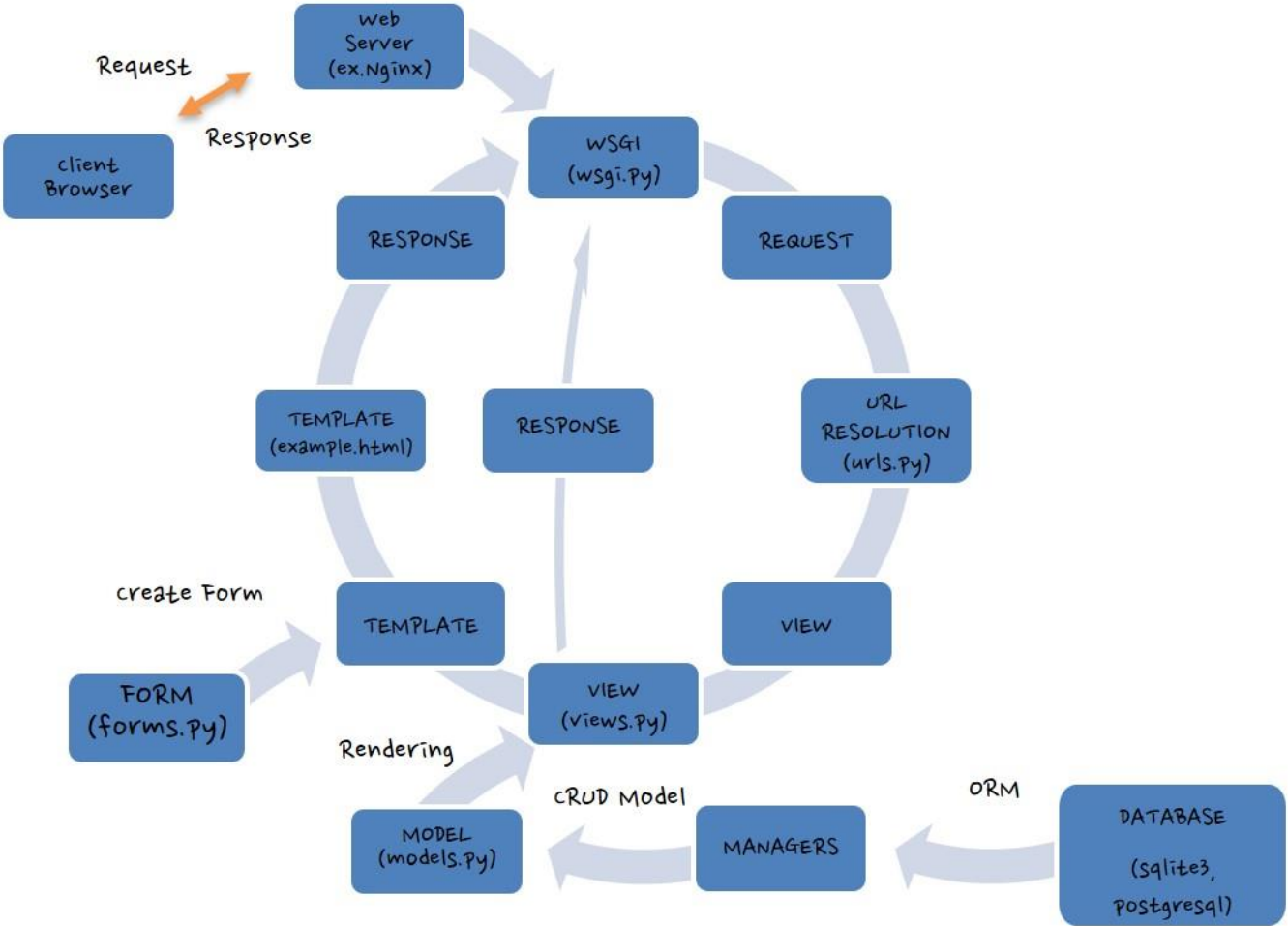
Django is a web framework that came out to develop the web quickly with Python. Most of the features are automated and the official documentation is very detailed, so once you read it, you would not need any other documentation.

We have created a simple travel blog so far with Django so far. But, it is only a small part of Django framework. If you want to use the more powerful features of Django, I recommend reading the official documentation.

Below is a Django architecture diagram and I will summarize briefly for all the steps that we did before. So, please check that you have not missed any of it.

* This book does not cover forms in Django.

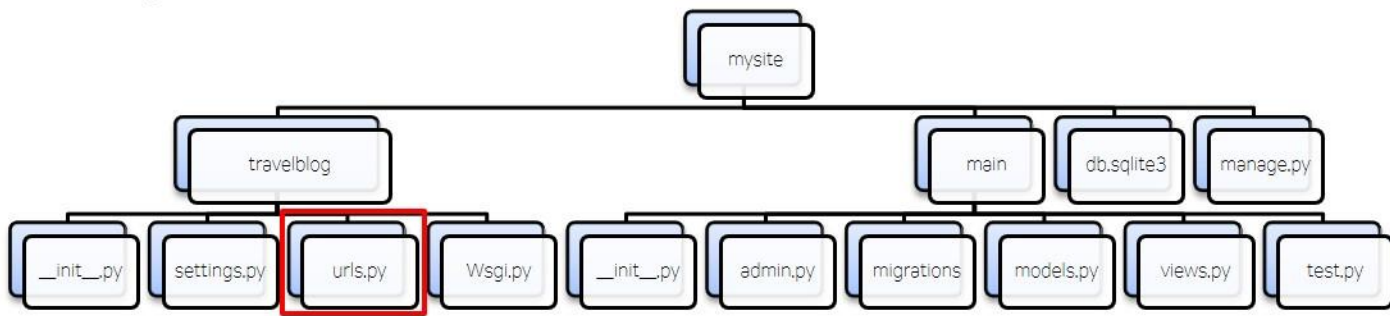
Django



urls.py

Here is a summary or “MAP” of the whole process we’ve been through. This is the order of the modifications of the **Django file system** that we did in the previous chapters (marked by red square). At first I designed URLs. A file specifies which view is called for a given URL pattern.

urls.py



```
from django.contrib import admin
from django.urls import path
from main.views import index, blog, postdetails
from django.conf.urls.static import static
from django.conf import settings

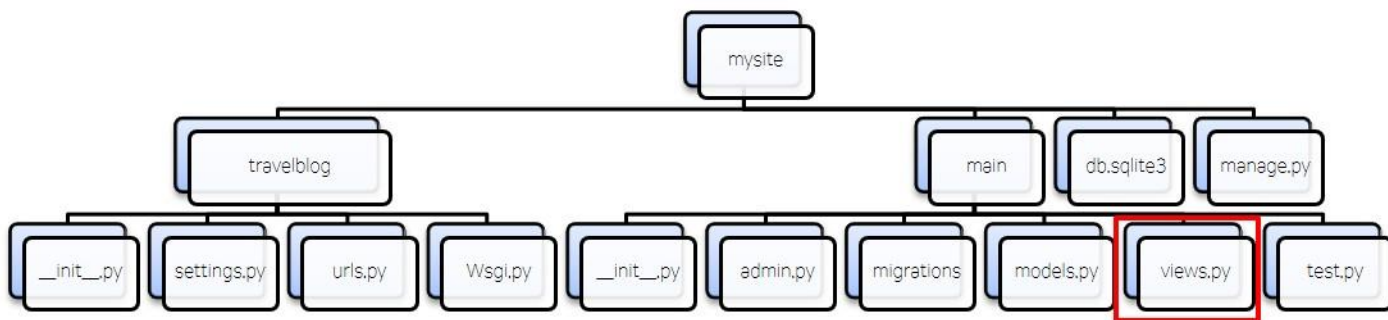
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index),
    path('blog/', blog),
    path('blog/<int:pk>/', postdetails),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

views.py

If you have finished designing the URL, you can create view functions that are linked to the URLs. Views contains the logic for pages.

views.py



```
from django.shortcuts import render
from .models import Post

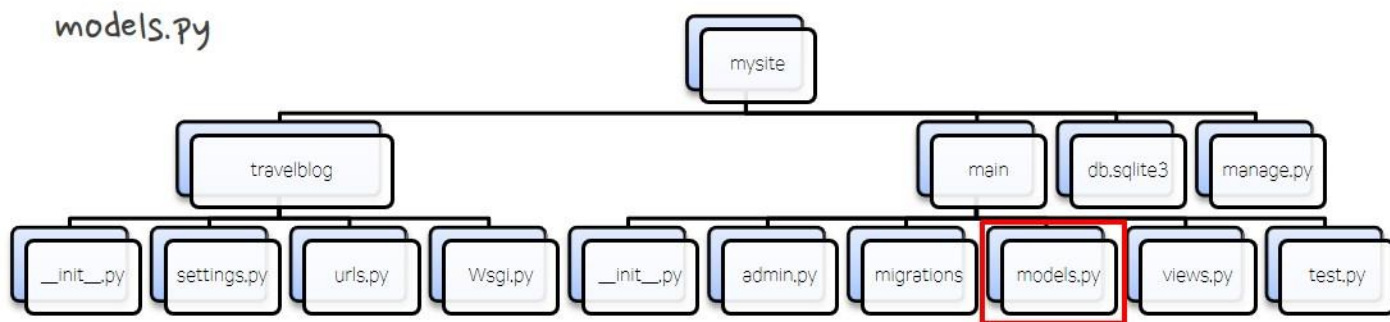
def index(request):
    postlist = Post.objects.all()
    return render(request, 'main/index.html', {'postlist':postlist})

def blog(request):
    postlist = Post.objects.all()
    return render(request, 'main/blog.html', {'postlist':postlist})

def postdetails(request, pk):
    postlist = Post.objects.get(pk=pk)
    return render(request, 'main/postdetails.html', {'postlist':postlist})
```

models.py

Class variables represents a database fields in the model. And you need migrations to keep the database in sync with model objects.



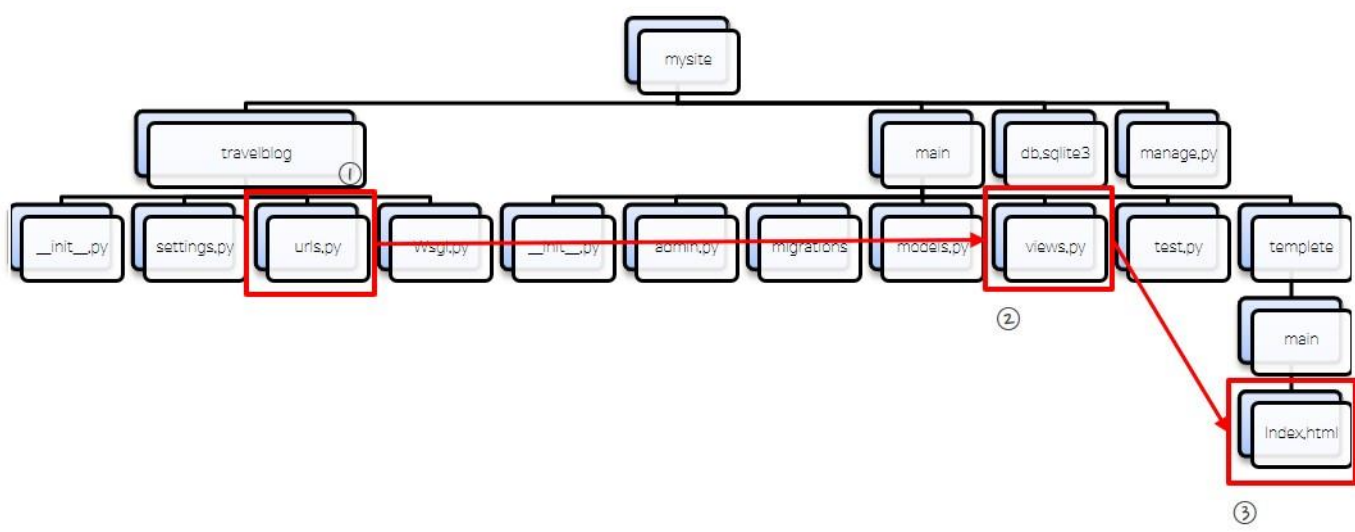
```
from django.db import models
from taggit.managers import TaggableManager

class Post(models.Model):
    postname = models.CharField(max_length=50)
    Lat = models.FloatField(null=True)
    Lng = models.FloatField(null=True)
    mainphoto = models.ImageField(blank=True, null=True)
    publishedDate = models.DateTimeField(blank=True, null=True)
    modifiedDate = models.DateTimeField(blank=True, null=True)
    contents = models.TextField()
    tag = TaggableManager(blank=True)

    def __str__(self):
        return self.postname
```

Templates

A template is simply a text file which can be generated in any text-based format(html, csv, etc.) and It describes the design of the pages.



Django Tutorial–Build a Travel Blog with Python and Bootstrap4

Published in 4th May 2017 By SADO Publisher

Address | Paul-lab, 137, Donggwang-ro, Jeju-si,

Jeju-do, Republic of Korea

Cover Design | SADO Publisher

Homepage | <http://www.paullab.co.kr>

E – mail | paul-lab@naver.com

ISBN | 979-11-88786-10-7

Copyright © 2017 SADO

All rights reserved.

TUTORIAL PROJECT

Django Tutorial-Build a Travel Blog with Python and Bootstrap 4



Who should read this tutorial

This book is recommended for those who want to program with Django.

Django is a very popular framework among Python programmers. We hope it will be a good opportunity to expand your perspectives on web development.

About the book

This book is a tutorial on Django that will teach you to use it while creating a simple travel blog with Python. So instead of learning with Django in details,

We focused on a simple way (step by step) to create a web service.

If you just follow along, you will be able to create a web service.

