
Table of Contents

Introduction	1.1
Getting Started	1.2
Demo Part 1	1.3
Demo Part 2	1.4
Advance	1.5
Azure	1.6
Bonus	1.7

Django Cupcake shop 파이콘장고튜토리얼

Beta version



cc. **Designed by Freepik**

이 튜토리얼은 Creative Commons Attribution-ShareAlike 4.0 International 저작권을 따르고 있습니다. 라이선스 전문은 <http://creativecommons.org/licenses/by-sa/4.0> 에서 확인하세요.

번역

이 튜토리얼은 열정적인 장고걸스서울 코치와 자원봉사자들의 수고로 번역되었습니다.

번역 : 이수진, 이소은

도움주신 분들

장고걸스서울의 운영진과 코치들이 함께 튜토리얼을 준비했습니다

하산 아비드, 박진우, 이소은, 이수진, 박조은, 김민선 , 송다운, 최장호, 장지호

튜토리얼에서 무엇을 배우게 되나요?

This tutorial will walk you through all the steps in creating a cupcake menu site. By the end, you'll be confident enough to make your website!

소개

장고가 처음이신 분들이라면, [공식 장고 튜토리얼 문서](#)를 보는 것이 가장 좋습니다. 장고 튜토리얼 문서는 웹 프로그래밍 경험이 있는 사람들을 위해 쓰여진 가이드입니다. 입문자를 대상으로 모든 설치 방법과 과정이 설명된 쉬운 튜토리얼이 있을까요? 네, 모두가 알고 계시는 바로 [장고걸스 튜토리얼](#)입니다. 아마 이 글을 읽으시는 많은 분들이 집에서 장고걸스 튜토리얼을 해보셨으라 생각되는데요, 하지만 튜토리얼을 끝낸 이후 어디서부터 프로젝트를 시작해야할지 막막 하시리라 생각합니다. 이런 분들을 위해 이번 PYCON APAC동안 함께 컵케이크 가게 메뉴 사이트를 만드는 모든 과정을 실습해보며 고민을 하나씩 풀어보고자 합니다. 튜토리얼을 함께 실습하시면서 나도 웹사이트를 만들 수 있다는 자신감을 가지게 되실 거예요! 그럼 컵케이크와 하이파이브 해봅시다!

설명

들어가며

장고(쟁고: Django)는 전 세계에서 가장 인기 있는 언어인 파이썬으로 작성된 웹 프레임워크로 다양하고 복잡한 기능을 지원하고 있습니다. 여러분은 장고로 어플리케이션과 사이트 개발을 할 수 있어요. 장고튜토리얼을 한번 마친 입문자들의 경우, 그 다음 무엇을 만들어봐야할지 막막하기만 합니다. 나만의 웹사이트를 만들기 위해 어디서부터 시작해야할까요? 우리는 튜토리얼에서 이미 어느정도 만들어진 리퍼지토리(소스)를 깃헙에서 다운 받아 여러 기능을 추가하고 코드를 조금씩 수정해볼 거예요. 이를 통해 장고 웹프레임워크를 활용한 웹 개발 과정을 조금씩 맛보고자 합니다.

장고걸스에서는 모든 이메일 끝에 ‘컵케이크와 하이파이브!’(Cupcakes and high fives)라는 메시지를 붙인답니다. 이 튜토리얼에서 우리는 장고컵케이크가게 사이트를 만들어볼 거예요. 가상환경과 `pip install django`, 그리고 다른 패키지들을 생성하는 방법을 살펴볼 거예요. 모델에서는 이미지 입력(ImageField)과 사용자가 데이터를 입력하는 모델 폼을 추가하는 방법도 알아 볼 거예요. 마지막으로 시간이 허락된다면, 만든 여러분의 사이트를 PythonAnywhere 또는 Azure를 배포해 전 세계 모든 사람들이 볼 수 있도록 만들어봐요!

준비사항

장고에 입문하시는 여러분들을 환영합니다. 좀더 나은 실습환경을 위해 미리 파이썬, 장고, 코드 에디터를 설치하고 오세요. 다음 링크 [[영어](#), [한국어](#)]을 참고하세요. 어느정도 하루정도 장고걸스튜토리얼를 해보신 분들을 진행되니 사전에 장고 공식 튜토리얼, 파이썬 기초 등을 학습하시고 오셔도 좋습니다. 튜토리얼 진행은 **영어로** 진행 합니다.

튜토리얼의 목적

Basic

- 장고프로젝트를 시작하기 전에 가상환경을 올바른 방법으로 다룰 수 있습니다.
- 모델에 이미지필드를 생성할 수 있습니다.
- 장고 관리자의 역할과 기능에 대해 이해할 수 있습니다.
- 기본 템플릿과 확장 템플릿을 다룰 수 있습니다.
- 장고 폼을 생성하고 이를 템플릿에 활용할 수 있습니다.
- Git/Github의 기본 명령어로 프로젝트를 Github에 배포할 수 있습니다.
- PythonAnywhere 서버에서 `DEBUG=False mode` 와 함께 사이트를 배포할 수 있습니다.
- 장고 사이트를 위한 간단한 테스트를 작성할 수 있습니다.

Advance

장고 또는 웹 개발의 경험이 있는 분 또는 더 어려운 내용을 다뤄보신 분들은 심화 내용을 학습할 수 있습니다! 심화 부분은 아래 내용을 포함하고 있습니다.:

- Django Auth (로그인과 가입하기)
- Django Model Relationships (댓글)
- Django Rest-framework
- Continuous Integration and coverage test (Travis-CI and Coveralls).
- Microsoft's `Azure` 배포 가이드

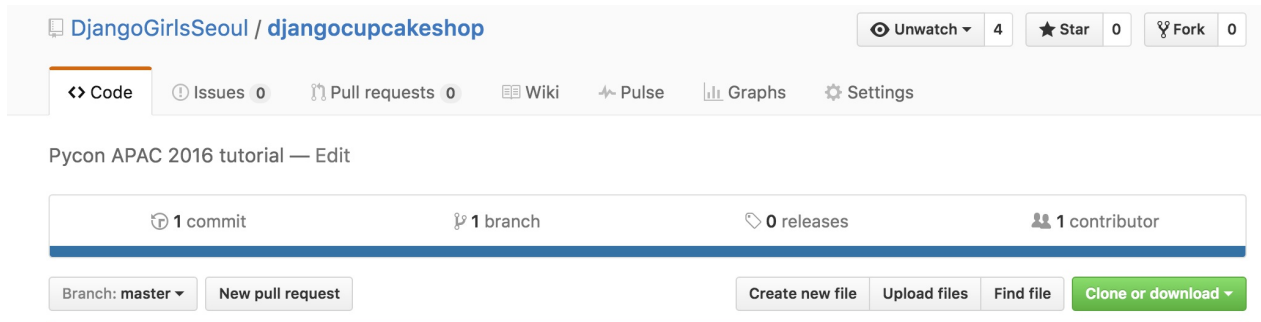
따라하기

그럼 다음 [링크](#)에서 차근차근 하나씩 따라가 보시다.

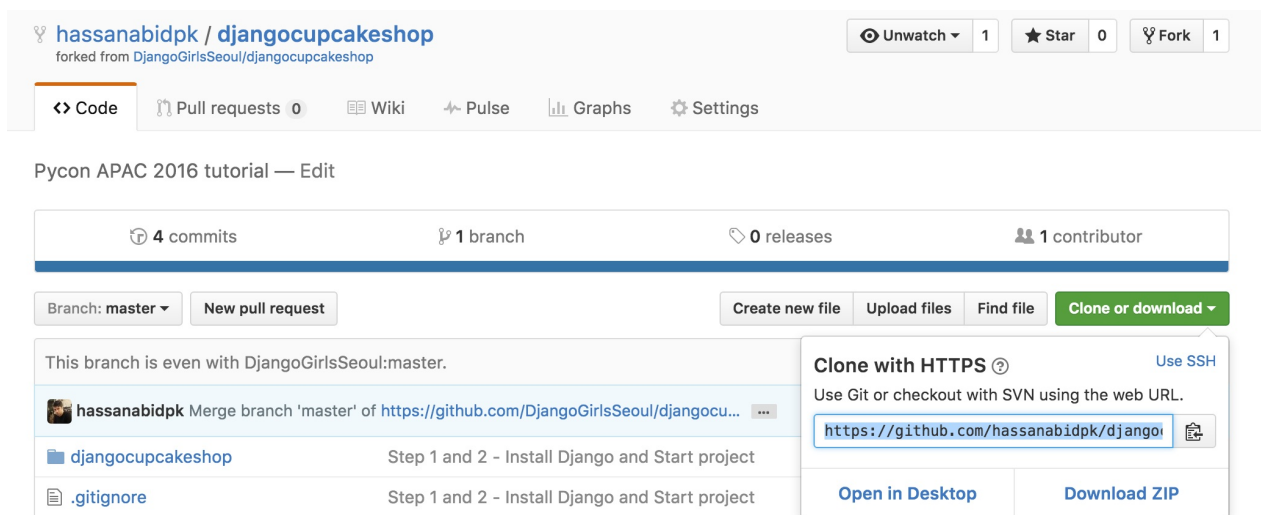
시작하기

이 튜토리얼을 찍고 내 리퍼지토리로 복사하세요.

링크로 가서 오른쪽 위에 있는 Fork 버튼을 누릅니다.



git clone 을 사용하여 여러분의 컴퓨터에 코드를 복사합니다.



```
$ git clone https://github.com/<user_name>/djangocupcakeshop.git
```

주의: `user_name` 이 부분에 여러분의 Github username을 적어주세요. 여러분의 레퍼지토리에 fork 되어야 해요!

Mac/Linux 환경에서는 terminal, window 환경에서는 prompt을 열고 djangocupcakeshop 폴더로 이동합니다.

```
cd djangocupcakeshop
```

cd djangocupcakeshop 치고 djangocupcakeshop 폴더 안으로 이동합니다.

아래 command를 콘솔에 치고 가상환경을 만듭니다.

Windows:

```
C:\Python35\python -m venv myenv
```

Mac:

```
$ python3 -m venv myenv
```

Linux

```
$ virtualenv --python=python3.4 myenv
```

가상환경을 활성화 시켜 볼까요?

Windows

```
myenv\Scripts\activate
```

Mac/Linux

```
$ source myenv/bin/activate
```

터미널 또는 prompt에서 다음과 같이 보이는지 확인합니다.

```
(myenv)...
```

아래 명령어를 이용해서 장고를 설치합니다.

```
$ pip install -r requirements.txt
```

```
python manage.py migrate
```

 통해 데이터 베이스를 만듭니다.

다음 명령어를 쳐서 데이터베이스를 만듭니다!

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

데모를 [4\(b\)](#) 단계부터 따라합니다.

데모 1

Step 1. 설치하기

파이썬 3.5.x, Git, 코드에디터(atom, sublime text, visual code 중 하나)가 반드시 설치되어야 합니다. 프로젝트 시작 전 가상환경을 설치하고 실행시키세요. 그리고 아래 명령어를 입력해 django를 설치합니다.

```
$ pip install django
```

장고나 다른 패키지를 설치할 때는 requirements.txt 파일을 생성해 설치 버전을 저장해 놓는 것이 좋습니다.

```
$ pip freeze > requirements.txt
```

Step 2. 프로젝트 시작하기

이제.djangocupcakeshop 장고 프로젝트를 시작해봅시다.

```
$ django-admin startproject.djangocupcakeshop
```

Step 3. 설정 변경하기

프로젝트를 생성하고나서 settings.py 에 있는 TIME_ZONE 을 변경해야 합니다.

djangocupcakeshop/djangocupcakeshop/settings.py 에서 TIME_ZONE 을 찾아 변경하세요. 웹 사이트가 어디서 운영(host)되고 있는지 알려주는 거랍니다. 서울이라면, 아래처럼 변경해주세요.

```
$ pip install django
```

STATIC_URL 부분 아래에 동적파일과 미디어파일(images, css, javascript) 경로도 아래처럼 설정해주세요.

```
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

MEDIA_ROOT와 MEDIA_URL 설정 경로는 이미지와 파일들이 저장되는 경로를 말합니다.

데이터베이스 테이블을 생성했고 브라우저에 프로젝트가 보이는지 확인합니다. 터미널/프롬프트에서 아래 두 명령어를 실행하세요.

```
$ python manage.py migrate

$ python manage.py runserver
```

브라우저를 열고 다음 링크로 접속하세요. : <http://127.0.0.1:8000>



Step 4. Django App/Model

Relevant git branch `model`

a. `menu` 앱을 만들고 `settings.py` 의 `INSTALLED_APPS` 에 새로운 앱을 추가하세요.

```
$ manage.py startapp menu
```

```
settings.py
```

```
INSTALLED_APPS = (  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'menu',  
)
```

Demo. 지금부터 실습을 시작합니다.

b. 메뉴 앱 모델을 생성해봅시다. 여기서부터는 튜토리얼 그대로 따라하세요.

Cupcake 에 필요한 것은 이름(name), 평가(rating), 가격(price), 이미지(image), 작성자(writer), 작성일(createdAt) 의 정보입니다. 이미지필드 (ImageField)에서 필요한 것은 Pillow 패키지가 설치되어야합니다. 아래 명령어를 실행해 설치합니다.

```
(myvenv) $ pip install Pillow
```

and then update requirements file by

```
(myvenv) $ pip freeze > requirements.txt
```

윈도우에서 Pillow가 설치 시 에러가 발생하면, `pip install Pillow==3.0.0` 를 사용하세요.

먼저 import 합니다.

```
from django.contrib.auth.models import User  
from django.utils import timezone
```

아래 Cupcake 클래스를 그대로 따라해 봅시다.

```
class Cupcake(models.Model):
    name = models.CharField(max_length=200)
    price = models.CharField(max_length=20)
    rating = models.FloatField()
    image = models.ImageField(upload_to='images/cakes')
    writer = models.ForeignKey(User)
    createdAt = models.DateTimeField(default=timezone.now)

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

c. 모델을 생성한 후에, 아래 명령어를 실행해 실제 데이터 베이스를 생성합니다.

```
$ python manage.py makemigrations menu
$ python manage.py migrate
```

`makemigrations` 을 실행하면, 장고에게 모델에 수정할 것이 있는지 확인하라고 말해주는 것입니다. (이 경우에, 새로운 모델이 생성이 되었다는 것을 알려줍니다) 그리고 수정 내용을 migration하게 됩니다. `makemigrations` 이후에 `python manage.py test` 을 실행하면 수정한 데이터 베이스가 잘 반영되었는지 테스트할 수 있습니다. `makemigrations` 명령어는 모델의 수정할 내용을 알려주고 `migrate` 는 실제로 이를 데이터베이스에 반영하는 것을 말합니다.

Step 5. Django 관리자

Relevant git branch `admin`

a. 관리자에 모델을 추가해 관리자 사이트에서 컵케익을 등록할 수 있게 만듭시다.
`menu/admin.py` 파일을 열고 아래 내용을 추가하세요.

```
from django.contrib import admin
from .models import Cupcake

admin.site.register(Cupcake)
```

b. 이제 사이트에 컵케이크 메뉴를 더 추가해 봅시다. 이를 하기 위해, 관리자 계정이 필요한데요. 아래 설명을 읽어보면서 명령 프롬프트에 실행합니다.

```
python manage.py createsuperuser
```

`python manage.py runserver` 명령어를 실행해 서버를 다시 시작해봅시다.
<http://127.0.0.1:8000/admin>에 접속해 로그인하고 컵케이크를 추가하세요!

The screenshot shows the Django administration interface. The top header is 'Django administration' with a welcome message for 'ADMIN'. The left sidebar shows 'Site administration' with sections for 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'MENU' (Cupcakes). The main content area shows the 'Add cupcake' form. The form fields are: Name (Chocolate), Price (2000), Rating (4), Image (Choose File cup_cake_1.jpg), Writer (admin), and CreatedAt (Date: 2016-07-26, Time: 19:36:57). At the bottom right, there are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

Step 6. Django Urls

Relevant git branch `django-urls`

a. 홈페이지에 url을 추가해야합니다. 먼저, 메뉴 앱에 홈페이지 URL를 추가합니다.
`djangocupcakeshop/urls.py` 폴더를 열어 아래 내용을 작성하세요.

```
urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'', include('menu.urls', namespace='menu')),
]
```

b. 다음으로, 아래 `view` 함수에 URL을 추가로 넣을 거예요. `menu` 디렉터리안에 `urls.py` 새 파일을 만드세요. 그리고 아래 코드를 넣으세요.

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.cupcake_list, name="cupcake_list"),
]
```

이제 서버를 실행해 홈페이지에 접속하면, 에러가 발생할 거예요. 에러 내용을 잘 살펴 보면 `cupcake_list` 뷰 함수가 없다는 것을 알게 될 거예요. 다음 번에 뷰 함수를 추가할 거니 걱정하지 마세요.



Step 7. Django Views

장고에서는 `view`를 통해 웹 페이지와 콘텐츠를 전달해줍니다. 각각 뷰는 파이썬 함수를 통해 구현되는데요. 장고는 URL을 통해서 특정 `view`를 선택해 이를 보여줍니다. 앞서 우리는 홈페이지 `url` 을 만들고 이를 보여줬는데요. `menu/views.py` 파일에 `cupcakes_list` 리스트 함수를 만들 거예요.

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

def cupcake_list(request):
    return render(request, "menu/list.html", {})
```

서버를 실행해서 홈페이지로 접속해보세요. <http://127.0.0.1:8000> 이런! 에러가 발생했네요! 다음 단계에서 이를 찾아봅시다.



Step 8. Django Templates

a. 앞에서 발생한 에러를 보면, `menu/list.html` 템플릿이 없는 것을 알 수 있을 거예요. 장고 템플릿은 html페이지 인데, 데이터베이스에 저장된 데이터를 사용자에게 보여줍니다. 먼저 `menu` 디렉터리에 `templates` 폴더를 만드세요. `templates` 폴더에서 해당 템플릿을 찾을 거예요. 장고는 여기서 `templates` 폴더에서 해당 템플릿을 찾을 거예요. 그 안에 다시 `menu` 디렉터리를 만드세요.

(`menu/templates/menu`) 이제 `list.html` 파일을 만드세요.

(`menu/templates/menu/list.html` .)

우리는 부트스트랩을 사용해 html 페이지를 만들 거예요. [여기](#)에서 부트스트랩 활용 예제들을 확인할 수 있어요. `style.css` 파일에서 템플릿을 수정하고 멋진 이미지들을 추가할 수 있어요! 무료 이미지를 찾고 싶으면 [이 곳](#)을 이용해보세요. 동적 파일을 추가하려면, `menu` 폴더 안에 `static` 폴더를 만드세요. 그리고 다시 `menu` 폴더를 만드세요. 이 폴더 안에 `css`와 이미지 파일들을 넣으면 됩니다. `css`파일 경로는 `menu/static/menu/css` , 이미지 경로는 `menu/static/menu/images` 가 되겠죠.

`list.html`

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Django Cupcake Shop</title>

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
  <!-- Optional theme -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css">
  <link rel="stylesheet" href="{% static 'menu/css/style.css' %}">
</head>
<body>
```

```

<!-- Fixed navbar -->
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
ta-toggle="collapse" data-target="#navbar" aria-expanded="false"
aria-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="/">Django Cupcake Shop</a>
    </div>
    <div id="navbar" class="navbar-collapse collapse">
      <ul class="nav navbar-nav navbar-right">
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dro
pdown" role="button" aria-haspopup="true" aria-expanded="false">
Sort by <span class="caret"></span></a>
          <ul class="dropdown-menu">
            <li><a href="#">Highest</a></li>
            <li><a href="#">Lowest</a></li>
          </ul>
        </li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</nav>

<div class="container" >

  <!-- Main component for a primary marketing message or call
to action -->
  <div class="jumbotron title text-center" style="background-i
mage: url({% static 'menu/images/cupcake_cover.jpg' %});">
    <h1 class="title">Cupcakes and High Fives!</h1>
    <p>Django Girls Seoul welcomes you!</p>
    <p>Lets build an awesome Django site together</p>

```



```

        <p>
            <a class="btn btn-lg btn-primary" href="https://github.c
om/DjangoGirlsSeoul/djangocupcakeshop" role="button">Source Code
            &raquo;</a>
        </p>
    </div>

</div> <!-- /container -->

<div class="container">
    <h2 class="text-center">Choose your favorite Cupcake!</h2>
    <div class="row">
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4">
            <div class="card">
                <div class="card-img-top">
                    <div class="image" style="background-image: url({% sta
tic 'menu/images/chocolate_cupcake.jpg' %});"></div>
                </div>
                <div class="card-block">
                    <h4 class="card-title text-center">Chocolate Cupcake</
h4>
                    <p class="card-text text-center">
                        <span class="glyphicon glyphicon-star" aria-hidden=
"true"></span>
                        <span class="glyphicon glyphicon-star" aria-hidden=
"true"></span>
                        <span class="glyphicon glyphicon-star" aria-hidden=
"true"></span>
                        <span class="glyphicon glyphicon-star" aria-hidden=
"true"></span>
                        <span class="glyphicon glyphicon-star" aria-hidden=
"true"></span>
                    </p>
                    <a href="#" class="btn btn-default btn-lg btn-block">V
iew</a>
                </div>
            </div>
        </div>
        <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4">
            <div class="card">

```

```
<div class="card-img-top">
  <div class="image" style="background-image: url({% sta
tic 'menu/images/vanilla_cupcake.jpeg' %});"></div>
</div>
<div class="card-block">
  <h4 class="card-title text-center">Vanilla Cupcake</h4>

  <p class="card-text text-center">
    <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
    <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
    <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
    <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
  </p>
  <a href="#" class="btn btn-default btn-lg btn-block">V
iew</a>
</div>
</div>
</div>
<div class="col-xs-12 col-sm-6 col-md-4 col-lg-4">
  <div class="card">
    <div class="card-img-top">
      <div class="image" style="background-image: url({% sta
tic 'menu/images/blueberry_cupcake.png' %});"></div>
    </div>
    <div class="card-block">
      <h4 class="card-title text-center">Blueberry Cookie Cu
pcake</h4>
      <p class="card-text text-center">
        <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
        <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
        <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
      </p>
      <a href="#" class="btn btn-default btn-lg btn-block">V
```

```
iew</a>
    </div>
  </div>
</div>
</div>
</div>

<footer class="footer">
  <div class="container">
    <p class="text-muted">Pycon 2016 Tutorial.</p>
  </div>
</footer>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11
.3/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/j
s/bootstrap.min.js" ></script>

</body>
</html>
```

We also need a css file for minor adjustments in the template. Create file `style.css` in `menu/static/menu/css` folder. Add following contents to css file `style.css`

```
@import url(http://fonts.googleapis.com/css?family=Raleway:400,
800|Roboto+Slab:300);

body {
  font-family: "Roboto Slab", "Helvetica", "Arial", sans-serif;;
  padding-top: 70px;
}

.card {
  margin: 2rem auto;
}

.image {
  width: 100%;
```

```
    height: 250px;
    padding-bottom: 50%;
    transition: 0.1s linear;
    background-image: url(http://www.freeallimages.com/wp-content/
uploads/2014/09/space-cat-wallpaper-5.jpg);
    background-size: cover;
    background-position: center center;
}

.card-title {
    margin-bottom: 1rem;
    font-weight: 900;
}

.jumbotron {
    height: 450px;
}

.title {
    color: white !important;
}

.footer {
    position: relative;
    bottom: 0;
    width: 100%;
    margin-top: 150px;
    text-align: center;
    /* Set the fixed height of the footer here */
    height: 60px;
    background-color: #f5f5f5;
}

.footer > .container {
    padding-right: 15px;
    padding-left: 15px;
}

.text-muted {
    margin-top: 20px;
}
```

```
.glyphicon
{
    color:#FF5252;
}

/* navbar */
.navbar-default {
    border-color: #FF5252;
}

a {
    color: #FF5252;
}
```

menu/static/images 폴더 안에, 컵케이크 이미지 세 개를 추가하고요. 이미지 파일 이름은 `list.html` 파일과 일치해야하니 주의하세요.

b. 이제 컵케이크(`cupcake`)의 가격, 리뷰를 보여주는 상세 페이지를 만들어야 합니다. `list.html` 가 있는 동일한 디렉터리에 `detail.html` 파일을 만드세요.

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Django Cupcake Shop</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <!-- Optional theme -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css">
    <link rel="stylesheet" href="{% static 'menu/css/style.css' %}">
</head>
<body>
```

```

<!-- Fixed navbar -->
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
ta-toggle="collapse" data-target="#navbar" aria-expanded="false"
aria-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="/">Django Cupcake Shop</a>
    </div>
    <div id="navbar" class="navbar-collapse collapse">
      <ul class="nav navbar-nav navbar-right">
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dro
pdown" role="button" aria-haspopup="true" aria-expanded="false">
Sort by <span class="caret"></span></a>
          <ul class="dropdown-menu">
            <li><a href="#">Highest</a></li>
            <li><a href="#">Lowest</a></li>
          </ul>
        </li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</nav>

<div class="col-xs-12 col-sm-6 col-md-3 col-lg-3">
  <div class="card">
    <ul class="list-group">
      <li class="list-group-item"><span class="glyphicon gly
phicon-tag"></span> <strong>Chocolate Cupcake</strong></li>
      <li class="list-group-item"><span class="glyphicon gly
phicon-usd"></span> 3.00</li>
      <li class="list-group-item"><span class="glyphicon gly
phicon-pencil"></span> John</li>
      <li class="list-group-item"><span class="glyphicon gly

```

```

phicon-calendar"></span> 3rd June, 2015</li>
    <li class="list-group-item">
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
        <span class="glyphicon glyphicon-star"></span>
    </li>
    <li class="list-group-item">
        <button type="button" class="btn btn-primary" data-t
oggle="modal" data-target="#myModal">
            Order
        </button>
    </li>
</ul>
</div>
</div>
</div>
</div>
<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal"
aria-label="Close"><span aria-hidden="true">&times;</span></butt
on>
                <h4 class="modal-title" id="myModalLabel">Chocolate Cupc
ake</h4>
            </div>
            <div class="modal-body">
                Complete this website to get your Cupcake!
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismi
ss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

```

```

</div>
  <footer class="footer">
    <div class="container">
      <p class="text-muted">Pycon 2016 Tutorial.</p>
    </div>
  </footer>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" ></script>

</body>
</html>

```

C. `list.html` 와 `base.html` 파일을 보면 헤더와 푸터에 동일한 내용이 있는 것을 알 수 있는데요. 장고는 `base` 템플릿을 만들어 다른 템플릿에서도 활용될 수 있게 해준답니다. `list.html` 에 있는 폴더에 `base.html` 만들고요. `list.html` 내용을 복사하고 수정할 거예요. `<div class="container">` 부터 `footer` 전까지 내용을 삭제하고 아래 블록을 추가하세요.

```

{% block content %}
{% endblock %}

```

그러면 `base.html` 템플릿이 아래와 같을 겁니다.

```

{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Django Cupcake Shop</title>

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/boo

```



```

tstrap/3.3.6/css/bootstrap.min.css">
<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/boot
strap/3.3.6/css/bootstrap-theme.min.css">
<link rel="stylesheet" href="{% static 'menu/css/style.css' %}">
</head>
<body>
  <!-- Fixed navbar -->
  <nav class="navbar navbar-default navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" da
ta-toggle="collapse" data-target="#navbar" aria-expanded="false"
aria-controls="navbar">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="/">Django Cupcake Shop</a>
      </div>
      <div id="navbar" class="navbar-collapse collapse">
        <ul class="nav navbar-nav navbar-right">
          <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dro
pdown" role="button" aria-haspopup="true" aria-expanded="false">
Sort by <span class="caret"></span></a>
            <ul class="dropdown-menu">
              <li><a href="#">Highest</a></li>
              <li><a href="#">Lowest</a></li>
            </ul>
          </li>
        </ul>
      </div><!-- /.nav-collapse -->
    </div>
  </nav>

  {% block content %}
  {% endblock %}

```

```

<footer class="footer">
  <div class="container">
    <p class="text-muted">Pycon 2016 Tutorial.</p>
  </div>
</footer>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" ></script>
</body>
</html>

```

이제 리스트 템플릿과 와 상세 페이지 템플릿이 서로 연결되게 해야합니다.

list.html

```

{% extends 'menu/base.html' %}
{% load staticfiles %}
{% block content %}
  <div class="container">
    <!-- Main component for a primary marketing message or call
    to action -->
    <div class="jumbotron title text-center" style="background-image: url({% static 'menu/images/cupcake_cover.jpg' %});">
      <h1 class="title">Cupcakes and High Fives!</h1>
      <p>Django Girls Seoul welcomes you!</p>
      <p>Lets build an awesome Django site together</p>
      <p>
        <a class="btn btn-lg btn-primary" href="https://github.com/DjangoGirlsSeoul/djangocupcakeshop" role="button">Source Code
        &raquo;</a>
      </p>
    </div>
  </div> <!-- /container -->

  <div class="container">
    <h2 class="text-center">Choose your favorite Cupcake!</h2>

```

```
<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4">
    <div class="card">
      <div class="card-img-top">
        <div class="image" style="background-image: url({% static 'menu/images/chocolate_cupcake.jpg' %});"></div>
      </div>
      <div class="card-block">
        <h4 class="card-title text-center">Chocolate Cupcake</h4>
        <p class="card-text text-center">
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
        </p>
        <a href="#" class="btn btn-default btn-lg btn-block">View</a>
      </div>
    </div>
  </div>
  <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4">
    <div class="card">
      <div class="card-img-top">
        <div class="image" style="background-image: url({% static 'menu/images/vanilla_cupcake.jpeg' %});"></div>
      </div>
      <div class="card-block">
        <h4 class="card-title text-center">Vanilla Cupcake</h4>
        <p class="card-text text-center">
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
          <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
        </p>
      </div>
    </div>
  </div>
</div>
```

```

true"></span>
        <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
        <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
    </p>
    <a href="#" class="btn btn-default btn-lg btn-block">V
iew</a>
</div>
</div>
</div>
<div class="col-xs-12 col-sm-6 col-md-4 col-lg-4">
    <div class="card">
        <div class="card-img-top">
            <div class="image" style="background-image: url({% sta
tic 'menu/images/blueberry_cupcake.png' %});"></div>
        </div>
        <div class="card-block">
            <h4 class="card-title text-center">Blueberry Cookie Cu
pcake</h4>
            <p class="card-text text-center">
                <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
                <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
                <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
            </p>
            <a href="#" class="btn btn-default btn-lg btn-block">V
iew</a>
        </div>
    </div>
</div>
</div>
</div>
</div>
{% endblock %}

```

detail.html

```

{% extends 'menu/base.html' %}
{% load staticfiles %}

{% block content %}
    <div class="container">
        <h2 class="text-center">Order Cupcake</h2>
        <div class="row">
            <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4 col-md-of
fset-2 col-md-lg-2">
                <div class="card">
                    <div class="card-img-top">
                        <div class="image" style="background-image: url({% sta
tic 'menu/images/chocolate_cupcake.jpg' %});"></div>
                    </div>
                </div>
            <div class="col-xs-12 col-sm-6 col-md-3 col-lg-3">
                <div class="card">
                    <ul class="list-group">
                        <li class="list-group-item"><span class="glyphicon gly
phicon-tag"></span> <strong>Chocolate Cupcake</strong></li>
                        <li class="list-group-item"><span class="glyphicon gly
phicon-usd"></span> 3.00</li>
                        <li class="list-group-item"><span class="glyphicon gly
phicon-pencil"></span> John</li>
                        <li class="list-group-item"><span class="glyphicon gly
phicon-calendar"></span> 3rd June, 2015</li>
                        <li class="list-group-item">
                            <span class="glyphicon glyphicon-star"></span>
                            <span class="glyphicon glyphicon-star"></span>
                            <span class="glyphicon glyphicon-star"></span>
                            <span class="glyphicon glyphicon-star"></span>
                            <span class="glyphicon glyphicon-star"></span>
                        </li>
                        <li class="list-group-item">
                            <button type="button" class="btn btn-primary" data-t
oggle="modal" data-target="#myModal">
                                Order

```

```
        </button>
      </li>
    </ul>
  </div>
</div>
</div>
</div>
<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
aria-labelledby="myModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal"
aria-label="Close"><span aria-hidden="true">&times;</span></butt
on>
        <h4 class="modal-title" id="myModalLabel">Chocolate Cupc
ake</h4>
      </div>
      <div class="modal-body">
        Complete this website to get your Cupcake!
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismi
ss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

서버를 실행하고 <http://127.0.0.1:8000>로 가서 리스트 템플릿을 확인하세요!

다음 **Demo**

Demo Part 2

Step 9 템플릿의 동적 데이터

orm branch 관련 잇어요

- a. queryset을 이용해서 데이터베이스에 동적데이터(cupcake 정보)를 가져옵니다.
menu/views.py 안에 `cupcake_list` function에 다음과 같은 코드를 추가합니다.

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

def cupcake_list(request):
    cakes = Cupcake.objects.all().order_by('-createdAt')
    context = {"cakes": cakes}
    return render(request, "menu/list.html", context)
```

위의 `Cupcake.objects.all().order_by('-createdAt')` 부분의 query는 데이터베이스에서 `createdAt` 기준으로 내림차순으로 모든 컵케이크 정보들을 가져오는 거예요!

이 코드는 `menu/list.html` 라는 템플릿에서 로드 되고 `context`에 전달 될 거예요. `context` 는 Python 객체들에 dictionary mapping되는 템플릿 변수 이름입니다. That code loads the template called `menu/list.html` and passes it a context. The `context` is a dictionary mapping template variable names to Python objects.

만약 홈페이지에 방문하면 데이터베이스에서 가져온 데이터들이 템플릿에서 볼수 없을 거예요! 이제 템플릿에서 돌아가서 QuerySet를 보이게 해 볼까요?

- b. 템플릿에 queryset으로 가져온 데이터를 추가하기 위해 Django Template Tags를 사용할 거예요. 하드코딩된 컵케이크 코드를 제거 하고 다음과 같이 만들어 보죠!

`list.html`

```
{% extends 'menu/base.html' %}
```

```
{% load staticfiles %}
{% block content %}
    <div class="container">
        <!-- Main component for a primary marketing message or call
to action -->
        <div class="jumbotron title text-center" style="background-i
mage: url({% static 'menu/images/cupcake_cover.jpg' %});">
            <h1 class="title">Cupcakes and High Fives!</h1>
            <p>Django Girls Seoul welcomes you!</p>
            <p>Lets build an awesome Django site together</p>
            <p>
                <a class="btn btn-lg btn-primary" href="https://github.c
om/DjangoGirlsSeoul/djangocupcakeshop" role="button">Source Code
                &raquo;</a>
            </p>
        </div>
    </div> <!-- /container -->

    <div class="container">
        {% if cakes %}
            <h2 class="text-center">Choose your favorite Cupcake!</h2>
            <div class="row">
                {% for cake in cakes %}
                    <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4">
                        <div class="card">
                            <div class="card-img-top">
                                <div class="image" style="background-image: url({{ cak
e.image.url }});"></div>
                            </div>
                            <div class="card-block">

                                <h4 class="card-title text-center">{{ cake.name }}</h4>

                                <p class="card-text text-center">
                                    <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
                                </p>
                                <a href="#" class="btn btn-default btn-lg btn-block">V
iew</a>
```



```

        </div>
    </div>
</div>
{% endfor %}
</div>
{% else %}
<h2 class="text-center">No Cupcakes added yet -:(</h2>
{% endif %}
</div>
{% endblock %}

```

개발 서버를 다시 시작하고 템플릿에 데이터베이스에서 가져온 컵케이크가 잘 보이는지 홈페이지에서 확인해 보세요! :)

c. 일단 여기에 들어가 보면 한가지 문제가 있어요!. 순서를 매기는데 오직 별 한개 밖에 안보이는 거죠. Django Template에서는 Template 안에서 `range` 같은 복잡한 파이썬 함수들의 사용을 허용하지 않아요. 그래서 우리는 rating count로 looping도 하고 별 갯수도 추가 하기 위해서 커스텀한 Django Template filter를 추가 할 거예요. c. We have one problem that we have to address here. As you can see that only one star appears for rating. Django Template doesn't allow usage of complex Python functions in template such as `range`. We have to make a custom Django Template filter for looping over rating count and add number of stars based on it. We are going to create a custom Django Template Filter. You can read more about it [here](#).

그래서 `models.py` and `views.py` 파일이 있는 폴더에 `templatetags` 폴더를 만들거예요. 그 디렉토리안에 `__init__.py` 라는 빈 파일을 추가 해야 된 다는 것을 잊지 마세요!. 그리고 `templatetags` 폴더안에 `menu_extras.py` 라는 파일을 만들어 보아요.

디렉토리 구조는 다음과 같아야 됩니다.

```
menu/  
    __init__.py  
    models.py  
    templatetags/  
        __init__.py  
        menu_extras.py  
    views.py
```

menu_extras.py 안에 다음과 같은 내용을 추가합니다.

```
from django import template  
  
register = template.Library()  
  
@register.filter  
def get_range(value):  
    """  
    Filter - returns a list containing range made from given va  
    lue  
    Usage (in template):  
    """  
    return range(int(value))
```

list.html 에는 {% load staticfiles %} 의 뒷 부분에 다음과 같이 추가 하세요!

```
{% load menu_extras %}
```

커스텀한 filter를 이용해서 반복문을 돌리려면 마지막으로 추가해야 되는 게 있습니다.
html에 rating 이라는 부분을 다음과 같이 바꿔주세요.

```
list.html
```

```
<p class="card-text text-center">
    {% for i in cake.rating|get_range %}
        <span class="glyphicon glyphicon-star" aria-hidden="true"></span>
    {% endfor %}
</p>
```



개발 서버를 다시 시작하고 홈페이지에 들어가 데이터베이스에서 가져온 컵케이크들이 잘 보이는지 확인해 보세요! :)

Step 10 프로그램 어플리케이션 확장하기

`extend-app` branch와 관련 있어요!

a. 지금까지 `list.html` 템플릿을 적절하게 바꾸는 것을 했는데요. 홈페이지에서 어드민에서 추가한 post들을 볼수 있으면 잘한 거예요! 잘했어요! 이제 사용자들이 버튼을 클릭해서 `Cupcake` 에 대한 자세한 정보를 보는 기능을 만들고 싶어요! 기본으로 돌아가서 `list.html` 템플릿을 구성하던 단계를 다시 반복할 거예요.

우선 컵케이크 하나를 가리키는 `url` 를 추가하고 `menu/urls.py` 파일에 `url(r'^$', views.cupcake_list, name="cupcake_list")` 아래에 `menu/urls.py` 이 부분을 추가해 볼거예요!

```
url(r'^cupcake/(?P<pk>\d+)/$', views.cupcake_detail, name="cupcake_detail")
```

This part `^cupcake/(?P<pk>\d+)/$` looks scary, but no worries - You can read explanation about it here [\[Eng, Kor\]](#).

Then add a function `cupcake_detail` in `menu/views.py` to render the template we created earlier. Any url like `cupcake/1` will be sent to view function `cupcake_detail` .

```
from django.shortcuts import render, get_object_or_404

def cupcake_detail(request, pk):
    cake = get_object_or_404(Cupcake, pk=pk)
    context = {"cake": cake}
    return render(request, "menu/detail.html", context)
```

b. There are two more things we have to do before we can see the cupcake detail page. Firstly, add a link to `list.html` template which can take us to detail page. Replace existing `<a href="#"...` with following code.

```
<a href="{% url 'menu:cupcake_detail' pk=cake.pk %}" class="btn btn-default btn-lg btn-block">View</a>
```

The mysterious `{% url 'menu:cupcake_detail' pk=cake.pk %}` will take us to the view function `cupcake_detail` which in turn will show us the detail page!

Seondly, we are going to add template tags and custom filter in `detail.html` for showing the cupcake from database.

```
{% extends 'menu/base.html' %}
{% load staticfiles %}
{% load menu_extras %}
{% block content %}
    <div class="container">
        {% if cake %}
            <h2 class="text-center">Order Cupcake</h2>
            <div class="row">
                <div class="col-xs-12 col-sm-6 col-md-4 col-lg-4 col-md-of
fset-2 col-md-lg-2">
                    <div class="card">
                        <div class="card-img-top">
                            <div class="image" style="background-image: url('{{ cak
e.image.url }}');"></div>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-xs-12 col-sm-6 col-md-3 col-lg-3">
```

```

        <div class="card">
        <ul class="list-group">
            <li class="list-group-item"><span class="glyphicon glyphicon-tag"></span> <strong>Chocolate Cupcake</strong></li>
            <li class="list-group-item"><span class="glyphicon glyphicon-usd"></span> 3.00</li>
            <li class="list-group-item"><span class="glyphicon glyphicon-pencil"></span> John</li>
            <li class="list-group-item"><span class="glyphicon glyphicon-calendar"></span> 3rd June, 2015</li>
            <li class="list-group-item">
                {% for i in cake.rating|get_range %}
                <span class="glyphicon glyphicon-star" aria-hidden="
true"></span>
                {% endfor %}
            </li>
            <li class="list-group-item">
                <button type="button" class="btn btn-primary" data-t
oggle="modal" data-target="#myModal">
                    Order
                </button>
            </li>
        </ul>
        </div>
    </div>
</div>
{% else %}
<h2 class="text-center">No Cupcake found :( </h2>
{% endif %}
</div>
<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal"
aria-label="Close"><span aria-hidden="true">&times;</span></butt
on>
                <h4 class="modal-title" id="myModalLabel">{{ cake.name }}

```

```


</h4>
</div>
<div class="modal-body">
  <p>Order completed 주문 완료되었습니다!</p>
  <p>{% now "jS F Y H:i" %}</p>
  <p>Price : {{ cake.price }}</p>
</div>
<div class="modal-footer">
  <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
</div>
</div>
</div>
{% endblock %}

```

Start development server again, and click on `view` button in home page to see detail page. Here is one example below

Django Cupcake Shop
Sort by ▾

Order Cupcake



Blueberry Cookie Cupcake
\$ 5
★ 3.0
admin
July 28, 2016, 10:52 a.m.
Order

a. menu에 템플릿 링크 만들기 그리고 menu 상세 페이지에 뷰 추가하기

Step. 11 (Django Forms 품)

Relevant branch `forms`

이제 마지막으로 해야할 일은 등록한 유저가 새로운 컵케이크를 등록할 수 있게 만드는 겁니다. 장고 어드민이 좋지만, 좀커스터마이징 하기도 어렵고 보기에다 예쁘지 않죠. 폼을 이용해 훨씬 더 멋진 인터페이스를 만들 수 있을 거예요. 여러분이 생각하는 거의 모든 것을 해볼 수 있습니다!

a. `menu` 디렉터리 안에 `forms.py` 이라는 새 파일을 만드세요. 이미 생성된 모델을 가져와 `ModelForm` 에서 활용할 거예요. `forms.py` 안에 아래 내용을 추가해주세요.

```
from django import forms
from .models import Cupcake

class CupcakeForm(forms.ModelForm):

    class Meta:
        model = Cupcake
        fields = ('name', 'rating', 'price', 'image')
```

`createdAt` 와 `writer` 은 폼에 활용하지 않을 겁니다.

`menu/urls.py` 에 새 url을 추가합니다. `url(r'^cupcake/(?P<pk>\d+)/$', views.cupcake_detail, name="cupcake_detail")`, 다음에 아래 코드를 붙여 넣으세요.

```
url(r'^cupcake/new/$', views.cupcake_new, name='cupcake_new'
),
```

b. 이제 폼이 만들어 졌으니, url로 전달합니다. 이제 해야할 일은 템플릿을 생성하고 뷰와 연결시키는 것입니다. `base.html` 에 있는 네비케이션 버튼 `+` 누르면 내용을 작성할 수 있게 할 거예요. 전에 아래 내용을 붙여서 넣으세요. `<li class="dropdown">` .

```
{% if user.is_authenticated %}
    <li><a href="{% url 'menu:cupcake_new' %}"><span class="glyphicon glyphicon-plus"></span></a></li>
{% endif %}
```

`user.is_authenticated` 은 유저가 로그인한 것을 말합니다. 작성한 내용을 보호하는 기능이 아닙니다.

`menu/templates/menu` 디렉터리 안에 `cupcake_new.html` 이라는 파일을 추가하세요. 그리고 그 안에 아래 내용을 붙여 넣으세요.

`cupcake_new.html`

```
{% extends 'menu/base.html' %}
{% load staticfiles %}
{% block content %}
    <div class="container">
        <!-- Main component for a primary marketing message or call
to action -->
        <div class="jumbotron title text-center" style="height: 200px;">
            <h1 style="color:black;">Add new Cupcake!</h1>
        </div>
    </div> <!-- /container -->

    <div class="container">
        <div class="row">
            <div class="col-xs-12 col-sm-12 col-md-offset-2 col-lg-offset-3 col-md-4 col-lg-4">
                <h2 class="text-center">Fill in details and submit</h2>
                <form method="POST" class="post-form" enctype="multipart/form-data">{% csrf_token %}
                    {{ form.non_field_errors }}
                    <div class="form-group">
                        <label for="{{ form.name.id_for_label }}">Name</label>
                        <input type="text" class="form-control" id="{{ form.name.id_for_label }}" name="{{ form.name.html_name }}" placeholder="blueberry Cupcake etc.">
                        {{ form.name.errors }}
                    </div>
                    <div class="form-group">
                        <label for="{{ form.rating.id_for_label }}">Rating</label>
                        <input type="text" class="form-control" id="{{ form.rating.id_for_label }}" name="{{ form.rating.html_name }}" placeholder="5 stars">
                    </div>
                </div>
            </div>
        </div>
    </div>
```



```

ing.id_for_label }}" name="{{ form.rating.html_name }}" placeholder="1-5">
    {{ form.rating.errors }}
</div>
<div class="form-group">
    <label for="{{ form.price.id_for_label }}">Price</label>
    <input type="text" class="form-control" id="{{ form.price.id_for_label }}" name="{{ form.price.html_name }}" placeholder="$ 2.00">
    {{ form.price.errors }}
</div>
<div class="form-group">
    <label for="{{ form.image.id_for_label }}">Image</label>
    <input type="file" id="{{ form.image.id_for_label }}" name="{{ form.image.html_name }}">
    <p class="help-block">Attach an image of size of at least 360w x 250h</p>
    {{ form.image.errors }}
</div>
<button type="submit" class="btn btn-default">Submit</button>
</form>
</div>
</div>
{% endblock %}

```

템플릿안에 폼을 추가하는 방법은 아주 간단합니다. 모든 폼을 한번에 반영하고 싶다면 `<form></form>` `{{ form.as_p }}` 를 넣으세요.

c. 개발 서버를 실행하면 에러가 보일 거예요. 아직 뷰에 폼 내용을 추가하지 않았기 때문이에요. Open `menu/views.py` 파일을 열고 `from` 줄부터 아래 내용을 추가하세요.

```
from django.shortcuts import redirect
from .forms import CupcakeForm
from django.utils import timezone
from django.contrib.auth.decorators import login_required
```

그리고 아래와 같이 함수를 만들어 주세요.

```
@login_required
def cupcake_new(request):
    if request.method == "POST":
        form = CupcakeForm(request.POST, request.FILES)
        if form.is_valid():
            cake = form.save(commit=False)
            cake.createdAt = timezone.now()
            cake.writer = request.user
            cake.save()
            return redirect('menu:cupcake_detail', pk=cake.pk)
    else:
        form = CupcakeForm()
        context = {'form': form}
        return render(request, "menu/cupcake_new.html", context)
```

`@login_required` 는 로그인한 유저만 새로운 컵케이크 내용을 작성할 수 있게 만드는 것입니다.

이제 잘 작동하는지 확인해봅시다. <http://127.0.0.1:8000/cupcake/new/> 페이지로 이동해서, 이름, 평가, 가격, 이미지를 작성하고 제출하세요! 새로운 컵케이크가 추가되었고 `cupcake_detail` 페이지로 이동하는 것을 볼 수 있어요!

Fill in details and submit

Name

Blueberry Muffin

Rating

4

Price

3

Image

Choose File blueberry_muffin.png

Attach an image of size of atleast 360w x 250h

Submit

잘했습니다! :) 이제 사이트 배포만 하면 됩니다!

이제 해야할 것이 하나 더 남았습니다. 다음 단계로 넘어갑시다.

Step 12. PythonAnywhere 배포하기

Relevant branch `deploy`

잘했어요! 이제 거의 다 끝나갑니다! PythonAnywhere로 배포 전에 해야할 게 있습니다. `Github` 에 변경된 내용을 모두 커밋하고 푸시하세요. `.gitignore` 파일 내용이 아래와 같아야 합니다.

```
*.pyc
__pycache__
myenv
db.sqlite3
/static
.DS_Store
media/
```

배포시 보안 확인하기

Follow this [link](#)

Django 비밀키

`settings.py` 파일을 열면 비밀키(`secret key`)를 확인할 수 있어요.

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '(!+73cf=j*8!=r$#2à^@ibgpw8yn9pm#wa42bk&v(@*%m7nx1sg'
```

깃헙에 소스를 저장하기 때문에 모든 사람들이 비밀키를 확인할 수 있어요. 다른 사람들이 비밀키를 알지 못하게 하려면 개발과 배포 서버에 `settings.py` 를 따로 적용하면 됩니다. 비밀키 설정은 조금 까다로워요.

PythonAnywhere에 환경변수(`Environment Variables`)를 사용해 `database password` 와 `SECRET_KEY` 를 설정할 겁니다. 이전에 `settings.py` 파일에 있는 `secret_key` 를 임의로 변경해주세요. 배포 버전에서 `DEBUG=False` 를 적용할 거고요. 아래처럼 `DEBUG=True` , `SECRET_KEY` , `ALLOWED_HOSTS` 내용을 수정해주세요.

```
DEBUG = os.getenv('DJANGO_DEBUG') != 'FALSE'

# SECURITY WARNING: keep the secret key used in production secret!
if DEBUG:
    SECRET_KEY = 'Hell@World!'
else:
    SECRET_KEY = os.getenv('SECRET_KEY')

# SECURITY WARNING: don't run with debug turned on in production!

if DEBUG:
    ALLOWED_HOSTS = ['*']
else:
    ALLOWED_HOSTS = ['djangocupcakeshop.pythonanywhere.com']
```

`os.getenv('DJANGO_DEBUG')` 은 PythonAnywhere에서 사용할 환경 변수예요.

Github에 실제 비밀키가 배포되어서는 안됩니다. PythonAnywhere에서 `DEBUG=False` 이라고 설정할 거예요.

Github으로 배포하기

`git status` 명령어를 실행해 현재 상태를 확인하세요. 모든 변경된 코드를 저장하려면 아래와 같이 입력하세요.

```
$ git add --all
$ git commit -m "finished tutorial until Step 10"
```

이제까지 한 모든 작업 내용을 Github에 올리세요.

```
$ git push -u origin master
```

PythonAnywhere

PythonAnywhere은 무료계정이 있어 배포시 이를 이용하실 수 있습니다.

[PythonAnywhere.com](https://pythonanywhere.com)으로 접속해 로그인하세요.

"Beginner" 계정으로 가입하시면 무료로 이용하실 수 있습니다.

PythonAnywhere로 가시면, 대시보드 또는 "Consoles"페이지로 이동될 겁니다.

"Bash" 콘솔을 클릭하세요. -- `bash` 는 로컬 컴퓨터에 있는 콘솔과 같은 콘솔입니다

`git clone` 명령어를 입력해 Github에 있는 모든 PythonAnywhere로 이동하게 만드세요.

```
$ git clone https://github.com/<your_github_user_name>/djangocup
cakeshop.git
```

<your_github_user_name> 은 github유저 네임입니다.

DjangoGirlsSeoul 이 아니에요 :)

PythonAnywhere에서 환경 변수를 적용할 차례입니다.

```
$ cd djangocupcakeshop
$ virtualenv --python=python3.5 myenv
$ source myenv/bin/activate
(myenv) $ pip install -r requirements.txt
```

`.gitignore` 폴더에 `db.sqlite3` 파일이 있습니다. Github에서는 데이터베이스 내용이 무시되어 저장될 거예요. 그래서 PythonAnywhere에서 새로운 데이터베이스를 생성하고 `superuser` 도 다시 만들어야 합니다

```
(myenv) $ python manage.py migrate
(myenv) $ python manage.py createsuperuser
```

이제 하나 더 해야할 일이 남았습니다. 콘솔로 가서 아래 명령어를 실행해주세요.

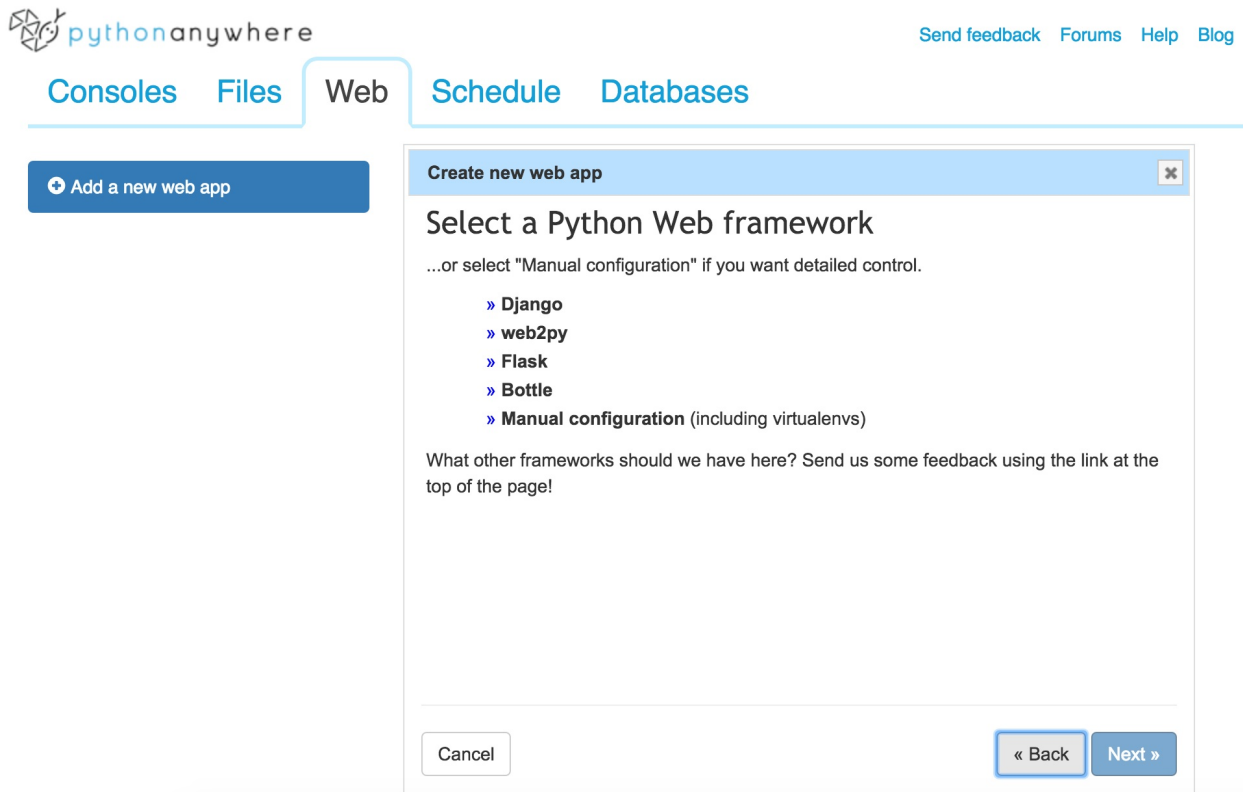
```
(myenv) $ python manage.py collectstatic
```

실행되면 `yes` 라고 입력하세요. 장고는 모든 동적 파일을 (images,css, javascript) the `STATIC_ROOT` 디렉토리로 옮길 겁니다.

이제 콘솔에서 해야할 명령어는 모두 끝났어요.

로고를 클릭해 PythonAnywhere대시보드로 돌아가고, Web 탭을 클릭하세요. 그리고 다시 new web app을 클릭하세요.

도메인 이름이 확정되면, manual configuration ("Django"옵션이 아닙니다)을 선택하세요. Python 3.5을 클릭하고 종료하세요.



virtualenv 설정

"Virtualenv" 단계에서, "Enter the path to a virtualenv"를 클릭하고 엔터를 누르세요.

/home/<your - PythonAnywhere - username>/djangocupcakeshop/myvenv/ 경로를 작성하고 파란색 클릭 버튼을 누르세요.

Code:

What your site is running.

Source code:	/home/djangocupcakeshop/djangocupcakeshop	Go to directory
Working directory:	/home/djangocupcakeshop/	Go to directory
WSGI configuration file:	/var/www/djangocupcakeshop_pythonanywhere_com_wsgi.py	
Python version:	3.5	

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones.

[More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

</home/djangocupcakeshop/djangocupcakeshop/myvenv>

[Start a console in this virtualenv](#)

`/home/<your-PythonAnywhere-username>/djangocupcakeshop` 경로를 작성하고 파란색 클릭 버튼을 누르세요.

WSGI file 설정

wsgi configuration file 링크를 클릭하고 아래 내용을 붙여 넣으세요.

wsgi

```
import os
import sys

path = '/home/<your_pythonanywhere_username>/djangocupcakeshop'
# use your own PythonAnywhere username here
if path not in sys.path:
    sys.path.append(path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'djangocupcakeshop.settings'
os.environ['DEBUG'] = 'FALSE'
os.environ['SECRET_KEY'] = 'MY_SECRET_KEY'

from django.core.wsgi import get_wsgi_application
from django.contrib.staticfiles.handlers import StaticFilesHandler
application = StaticFilesHandler(get_wsgi_application())
```


MY_SECRET_KEY 에 실제 비밀키를 변경해주세요!

잘했습니다! 재실행 버튼을 누르면 어플리케이션으로 이동할 거예요. 페이지 맨 위쪽에 링크가 있어요.



[Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Dashboard](#) /

[Consoles](#) [Files](#) [Web](#) [Schedule](#) [Databases](#)

All done! Your web app is now set up. Details below.

django cupcakes.pythonanywhere.com

[Add a new web app](#)

Configuration for django cupcakes.pythonanywhere.com

Reload:

[Reload django cupcakes.pythonanywhere.com](#)

Best before date:

Free sites have a limited lifespan, but you can renew that here up to a maximum of three months from today's date. You can always extend it later too! We'll send you an email a week before it expires. [See here for more details.](#)

This site will be disabled on **Friday 28 October 2016**

[Run until 3 months from today](#)

[Paying users'](#) sites do not have expiry dates.

- PythonAnywhere에서 무료 계정인 "초보자(Beginner)"로 회원가입 하세요. GitHub에서 PythonAnywhere로 코드 가져오기
- PythonAnywhere에서 가상환경(virtualenv) 생성하기. 콘솔창에서 `virtualenv --python=python3.4 myenv` 그리고 `pip install -r requirements.txt` 실행하세요. 정적 파일 모으기 `python manage.py collectstatic`
- PythonAnywhere에서 데이터베이스 생성하기 `python manage.py migrate`
- web app으로 DjangoCupcakeshop 배포하기 - 가상환경(virtualenv) 설정하기 그리고 WSGI 파일 설정하기

Step 13. Homework (숙제)

- 가격 순으로 리스트 배열: 최저가격 `highest` 최고가격 `lowest` 순으로 리스트 배열하기

도움 : `cupcakes/price/hightolow` 처럼 url을 만들고 뷰 함수를 추가해야 합니다. 데이터 베이스부터 데이터를 Get the data from Database, convert price string to int and sort using python. You can use the same template `list.html` for this homework.

b. 평점 순으로 리스트 배열: 최고 평점 highest 순으로 리스트 배열하기

심화 (Advance)

1. 로그인 및 가입하기 기능 만들기

로그인, 가입하기 기능은 꼭 필요한 기능입니다. 장고는 기본적으로 `user authentication system`이 있어요. 또 커스터마이징할 수 있어요! 우리 웹사이트에 로그인과 가입하기 기능을 넣어봅시다.

이를 위해 `accounts` 이라는 새 앱을 만듭시다.

```
$ python manage.py startapp accounts
```

그리고 이 앱에 앞으로 로그인과 가입하기에 필요한 모든 것들을 넣을 거예요.

`INSTALLED_APPS` (`settings.py`)에 `accounts` 를 추가하세요.

이미 슈퍼유저가 생성이 되어 있기 때문에 바로 로그인 기능이 잘 구현되었는지 테스트 할 수 있습니다.

그리고 `url` 을 수정해야하는데, 아래처럼 `accounts.url` 에 모든 `auth` 관련 링크가 다 연결되도록 해주세요.

```
djangocontribcupcakeshop/urls.py
```

```
url(r'^accounts/', include('accounts.urls', namespace="accounts")),
```

`accounts` 디렉터리에 `urls.py` 을 생성하고 아래처럼 코드를 수정해주세요.

```
from django.conf.urls import include, url
from . import views

urlpatterns = [
    url('^', include('django.contrib.auth.urls')),
]
```

아래 URL패턴을 사용할 수 있어요.

```
^login/$ [name='login']
^logout/$ [name='logout']
^password_change/$ [name='password_change']
^password_change/done/$ [name='password_change_done']
^password_reset/$ [name='password_reset']
^password_reset/done/$ [name='password_reset_done']
^reset/(?P<uidb64>[0-9A-Za-z_-]+)/(?P<token>[0-9A-Za-z]{1,13}-[0-9A-Za-z]{1,20})/$ [name='password_reset_confirm']
^reset/done/$ [name='password_reset_complete']
```

이제 로그인, 로그아웃 을 만들어 볼 차례예요. 등록하기 페이지를 만들어야겠죠.

디폴트 로그인 url은 `registration/login.html` 템플릿을 꼭 가지고 있어야해요. 이제 `accounts` 디렉터리안에 `registration` 이라는 새 디렉터리를 만드세요. 그리고 `registration` 디렉터리 안에 `login.html` 파일을 만들고 아래 내용을 추가하세요.

```
{% extends 'menu/base.html' %}
{% load staticfiles %}
{% block content %}
    <div class="container">
        <!-- Main component for a primary marketing message or call
to action -->
        <div class="jumbotron title text-center" style="height: 150px;">
            <h3 style="color:black;">Welcome back!</h3>
        </div>

    </div> <!-- /container -->

    <div class="container">
        <div class="row">
            <div class="col-xs-12 col-sm-12 col-md-offset-4 col-lg-offset-4 col-md-6 col-lg-6">
                {% if form.errors %}
                    <p style="color:red;">Your username and password didn'
```

```

t match. Please try again.</p>
    {% endif %}
    {% if next %}
        {% if user.is_authenticated %}
            <p>Your account doesn't have access to this page. To
proceed,
                please login with an account that has access.</p>
        {% else %}
            <p style="color:red;">Please login to see this page.
</p>
        {% endif %}
    {% endif %}
    <form class="form-horizontal" method="post" action="{% u
rl 'accounts:login' %}">
        {% csrf_token %}
        <div class="form-group">
            <label for="{{ form.username.id_for_label }}" class="c
ol-sm-2 col-md-2 col-lg-2 control-label">Username</label>
            <div class="col-sm-10 col-md-5 col-lg-5">
                <input type="text" class="form-control" id="{{ form.
username.id_for_label }}" name="{{ form.username.html_name }}" p
laceholder="username">
            </div>
        </div>
        <div class="form-group">
            <label for="{{ form.password.id_for_label }}" class="c
ol-sm-2 col-md-2 col-lg-2 control-label">Password</label>
            <div class="col-sm-10 col-md-5 col-lg-5">
                <input type="password" class="form-control" id="{{ f
orm.password.id_for_label }}" name="{{ form.password.html_name }
}" placeholder="Password">
            </div>
        </div>
        <input type="hidden" name="next" value="{{ next }}" />
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <button type="submit" class="btn btn-default">Log in
</button>
            </div>
        </div>
    </div>

```

```
        </form>
    </div>
</div>
</div>
{% endblock %}
```

서버를 재실행하고 <http://127.0.0.1/accounts/login>에 접속해보세요. 로그인 후, 에러가 보일 거예요. 왜냐하면 프로필 페이지의 뷰와 템플릿을 만들지 않았기 때문이에요. 이제 `registration` 디렉터리 안에 `profile.html` 페이지를 만들어 봅시다.

```
profile.html
```

```

{% extends 'menu/base.html' %}
{% load staticfiles %}
{% block content %}
    <div class="container">
        <div class="jumbotron title text-center" style="height: 150px;">
            <h3 style="color:black;">Hello {{ request.user.username }}!
        </h3>
        </div>
    </div>

    <div class="container">
        {% if cakes %}
            <h2 class="text-center">My Cupcakes!</h2>
            <div class="row">
                {% for cake in cakes %}
                    <div class="col-sm-6 col-md-4">
                        <div class="thumbnail" style="height:336px;">
                            <a href="{% url 'cupcake_detail' pk=cake.pk %}"></a>
                            <div class="caption">
                                <h3>{{ cake.name }}</h3>
                            </div>
                        </div>
                    </div>
                {% endfor %}
            </div>
            {% else %}
                <h2 class="text-center">No Cupcakes added yet -:</h2>
            {% endif %}
        </div>
    {% endblock %}

```

템플에서 확인하실 수 있듯이, 사용자가 컵케익을 선택할 수 있어요. 사용자 계정을 위해 view 함수를 추가해야합니다. `views.py` 에서 아래 내용을 추가하세요.

```

from django.shortcuts import render
from django.contrib.auth.decorators import login_required
from menu.models import Cupcake

@login_required
def user_profile(request):
    my_cakes = Cupcake.objects.filter(writer=request.user)
    context = {'cakes':my_cakes}
    return render(request, "registration/profile.html", context)

```

base.html 템플릿안에 로그인과 가입하기 링크를 아직 추가하지 않았는데요. 링크를 추가한 base.html 코드는 아래와 같을 겁니다.

```

{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Django Cupcake Shop</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <!-- Optional theme -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css">
    <link rel="stylesheet" href="{% static 'menu/css/style.css' %}">
</head>
<body>
    <!-- Fixed navbar -->
    <nav class="navbar navbar-default navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">

```



```

        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="/">Django Cupcake Shop</a>
</div>
<div id="navbar" class="navbar-collapse collapse">
    <ul class="nav navbar-nav navbar-right">
        {% if user.is_authenticated %}
        <li><p class="navbar-text">Welcome <a href="/accounts/
profile">{{ user.username }}</a></p></li>
        <li><a href="{% url 'cupcake_new' %}"><span class="gl
yphicon glyphicon-plus"></span></a></li>
        <a href="{% url "accounts:logout" %}" class="btn btn-
danger navbar-btn">Logout</a>
        {% else %}
        <a href="#" class="btn btn-primary navbar-btn">Regist
er</a>
        <a href="{% url "accounts:login" %}" class="btn btn-d
efault navbar-btn">Sign in</a>
        {% endif %}
        <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dro
pdown" role="button" aria-haspopup="true" aria-expanded="false">
Sort by <span class="caret"></span></a>
            <ul class="dropdown-menu">
                <li><a href="#">Highest</a></li>
                <li><a href="#">Lowest</a></li>
            </ul>
        </li>
    </ul>
</div><!--/.nav-collapse -->
</div>
</nav>

{% block content %}
{% endblock %}

<footer class="footer">

```

```

    <div class="container">
      <p class="text-muted">Pycon 2016 Tutorial.</p>
    </div>
  </footer>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" ></script>
  <script>
    $(function () {
      $('[data-toggle="popover"]').popover()
    })
  </script>
</body>
</html>

```

개발 서버를 재 실행하고 로그인이 잘 작동되는지 테스트해보세요. 다음으로 등록 페이지를 만들어 볼 텐데요. 로그인 기능을 만든 것 보다 좀 더 많은 작업을 해야합니다. 가장 먼저 `profile.html` 과 `login.html` 파일이 있는 같은 폴더 내에 `register.html` 을 생성하세요.

`register.html`

```

{% extends 'menu/base.html' %}
{% load staticfiles %}
{% block content %}
  <div class="container">
    <!-- Main component for a primary marketing message or call to action -->
    <div class="jumbotron title text-center" style="height: 150px;">
      <h3 style="color:black;">Cupcake and High Fives!</h3>
    </div>

  </div> <!-- /container -->

  <div class="container">
    <div class="row">

```

```

<div class="col-xs-12 col-sm-12 col-md-offset-3 col-lg-off
set-3 col-md-6 col-lg-6">
    {% if form.errors %}
        <p style="color:red;">Try again.</p>
    {% endif %}
    <form class="form-horizontal" method="post" action="{% u
rl 'accounts:register' %}">
        {% csrf_token %}
        <div class="form-group">
            <label for="{{ form.username.id_for_label }}" class="c
ol-sm-4 col-md-4 col-lg-4 control-label">Username</label>
            <div class="col-sm-10 col-md-5 col-lg-5">
                <input type="text" class="form-control" id="{{ form.
username.id_for_label }}" name="{{ form.username.html_name }}" p
laceholder="username">
                {{ form.username.errors }}
            </div>
        </div>
        <div class="form-group">
            <label for="{{ form.password1.id_for_label }}" class="
col-sm-4 col-md-4 col-lg-4 control-label">Password</label>
            <div class="col-sm-10 col-md-5 col-lg-5">
                <input type="password" class="form-control" id="{{ f
orm.password1.id_for_label }}" name="{{ form.password1.html_name
 }}" placeholder="Password">
                {{ form.password1.errors }}
            </div>
        </div>
        <div class="form-group">
            <label for="{{ form.password2.id_for_label }}" class="
col-sm-2 col-md-4 col-lg-4 control-label">Confirm Password</label>
            <div class="col-sm-10 col-md-5 col-lg-5">
                <input type="password" class="form-control" id="{{ f
orm.password2.id_for_label }}" name="{{ form.password2.html_name
 }}" placeholder="Password">
                {{ form.password2.errors }}
            </div>
        </div>
        <input type="hidden" name="next" value="{{ next }}" />

```

```

        <div class="form-group">
            <div class="col-md-offset-4 col-lg-offset-4 col-sm-off
set-2 col-sm-10">
                <button type="submit" class="btn btn-default">Regist
er</button>
            </div>
        </div>
    </form>
</div>
</div>
</div>
{% endblock %}

```

base.html 템플릿에 등록하기 페이지 링크를 추가하세요.

```

<a href="{% url 'accounts:register' %}" class="btn btn-primary n
avbar-btn">Register</a>

```

두 번째로, 뷰에 register 함수를 만들어 get/post 요청을 등록하도록 해봅시다.

accounts/views.py import 부터 시작하는 아래 내용을 추가하세요.

```

from django.http import HttpResponseRedirect
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth import authenticate, login

```

아래 내용을 이어서 추가하세요.

```
def register(request):
    if request.user.is_authenticated():
        return HttpResponseRedirect('/accounts/profile')

    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            new_user = form.save()
            print(new_user.username)
            username = request.POST['username']
            password = request.POST['password1']
            user = authenticate(username=username, password=password)

            if user is not None:
                login(request, user)
                return HttpResponseRedirect('/accounts/profile')
        else:
            form = UserCreationForm()
            return render(request, 'registration/register.html', { 'form' : form })
```

장고는 기본적으로 유저이름과 비밀번호가 포함된 `UserCreationForm` 을 제공합니다. 사용자의 이메일, 성과 이름을 필수값으로 추가하고 싶다면, custom form 을 만들어야 합니다.

서버를 재 실행하여 로그인과 등록하기 기능이 잘 작동하는지 확인하세요.

지금 `logout`(로그아웃) 링크를 클릭하면 관리자 로그아웃 링크로 연결되는 것을 볼 수 있을 텐데요. 기본값을 수정해 사용자가 로그아웃 이후에 홈페이지로 갈 수 있도록 수정해봅시다.

`accounts/urls.py` 에 아래 코드로 수정하세요.

```
url('^logout/$', views.logout_view, name="logout"),
```

그리고 `accounts/views.py` 에 아래 뷰 함수를 추가하세요.

```
from django.contrib.auth import logout

def logout_view(request):
    logout(request)
    return HttpResponseRedirect('/')
```

서버를 실행해 로그아웃 기능이 잘 작동하는지 확인해보세요.

이제 우리가 생각했던 기능들이 모두 잘 작동하고 있네요! 이제 로그인과 가입하기 페이지가 잘 되는지 테스트해 봐야겠죠?.

tests.py

```
from django.test import TestCase, Client
from django.contrib.auth.models import User

class LoginAndLogout(TestCase):
    def setUp(self):
        self.client = Client()
        u = User.objects.create_user('test_user', 'test@example.com', 'password1')
        u.save()

    def test_login_post(self):
        response = self.client.post("/accounts/login/", {"username": "test_user", "password": "password1", "next": "/"}, follow=True)
        self.assertRedirects(response, '/')
        self.assertContains(response, "test_user")

    def test_login_fail(self):
        response = self.client.post("/accounts/login/", {"username": "test_user1", "password": "password1", "next": "/"})
        self.assertContains(response, "Your username and password didn't match. Please try again.")

    def test_login_then_logout(self):
        login_response = self.client.post("/accounts/login/", {"
```

```

username": "test_user" , "password": "password1", "next": "/"}, follow=True)
    self.assertRedirects(login_response, '/')
    self.assertContains(login_response, "test_user")
    logout_response = self.client.get('/accounts/logout/', follow=True)
    self.assertRedirects(logout_response, '/')
    self.assertNotContains(logout_response, "test_user")

class Register(TestCase):
    def setUp(self):
        self.client = Client()

    def test_register_post(self):
        response = self.client.post("/accounts/register/", {"username": "test_user" , "password1": "password1", "password2": "password1"}, follow=True)
        self.assertContains(response, "test_user")
        self.assertEqual(User.objects.get(username="test_user").username, "test_user")

```

아래처럼 테스트 코드를 실행해 볼 수 있어요.

```
$ python manage.py test accounts
```

2. 댓글 기능 만들기

메뉴에 있는 컵케이크 맛은 어떤지 궁금하지 않나요? 먹어본 다른 사용자들의 리뷰 궁금하죠? 댓글 기능을 추가해 다른 사용자가 리뷰를 남길 수 있게 해볼 거예요.

댓글 모델 생성하기

컵케이크에 대해 사용자가 간단한 댓글을 남길 수 있게 만들어 봅시다. 나중에 관리자가 댓글을 승인할 수 있도록 만들 수도 있어요 :)

menu/models.py 파일에 댓글 모델을 만들어 봅시다.

```
class Comment(models.Model):
    post = models.ForeignKey(Cupcake, related_name='comments')
    writer = models.ForeignKey(User)
    text = models.TextField()
    created_date = models.DateTimeField(default=timezone.now)
    approved_comment = models.BooleanField(default=False)

    def approve(self):
        self.approved_comment = True
        self.save()

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

데이터베이스에 새 테이블을 만들기 위해 아래 명령어를 실행하세요.

```
$ python manage.py makemigrations menu
```

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

관리자에 댓글 모델 만들기

menu/admin.py 파일을 열어보세요. 이제 우리는 댓글 모델 클래스를 모두 가져와야 합니다.

```
from .models import Cupcake, Comment
```

그리고 아래 내용을 comment 모델에 등록해야 합니다.

```
admin.site.register(Comment)
```

사용자가 댓글을 작성하게 만들기

사용자가 댓글을 작성하게 폼을 만들어봅시다. 이미 앞에서 댓글 모델을 만들었는데요. 이제 작성할 수 있는 CommentForm을 만들어야 합니다. forms.py 파일을 열고 아래 코드를 추가하세요.

import Comment

```
class CommentForm(forms.ModelForm):

    class Meta:
        model = Comment
        fields = ('text',)
```

menu/templates/menu/detail.html 파일을 열고 `endblock` 바로 위에 아래 내용을 추가하세요.

```
<div class="row">
    <div class="col-xs-12 col-sm-6 col-md-7 col-lg-7 col-md-of
fset-3 col-lg-offset-2">
        <h3>Comments ({{ cake.comments.count }})</h3>
        <p></p>
        {% if user.is_authenticated %}
        <form method="POST">{% csrf_token %}
            <!-- {{ form.as_p }} -->
            <div class="form-group">
                <input type="text" name="{{ form.text.html_name }}"
class="form-control" placeholder="Write comment">
            </div>
            <button type="submit" class="btn btn-default">Submit</
button>
        </form>
        {% else %}
        <p><a href="{% url 'accounts:login' %}">Login</a> to p
ost comment</p>
        {% endif %}

        {% for comment in cake.comments.all %}
        {% if comment.approved_comment %}
        <div class="media">
            <div class="media-left">
                <a href="#">
                    </a>
</div>
<div class="media-body">
  <h4 class="media-heading">{{ comment.writer }}</h4>

  <p>{{ comment.text|linebreaks }}</p>
  <i>{{ comment.created_date }}</i>
</div>
</div>
{% endif %}
{% empty %}
  <p>No comments here yet :( </p>
{% endfor %}
</div>
</div>
{% else %}
  <h2 class="text-center">No Cupcake found :( </h2>
{% endif %}
</div>
```

CommentForm을 추가하기 위해 `menu/views.py` 파일 내 `cupcake_detail` 함수를 수정합니다. 수정한 코드는 아래와 같을 거예요.

```

from .forms import CommentForm

def cupcake_detail(request, pk):
    cake = get_object_or_404(Cupcake, pk=pk)
    if request.method == "POST":
        form = CommentForm(request.POST)
        if form.is_valid():
            comment = form.save(commit=False)
            comment.post = cake
            comment.writer = request.user
            comment.approved_comment = True
            comment.save()
            return redirect('menu.views.cupcake_detail', pk=cake
        .pk)
    else:
        form = CommentForm()
    context = {"cake": cake, "form": form}
    return render(request, "menu/detail.html", context)

```


위의 코드는 모든 댓글을 승인하도록 만들었습니다.

서버를 실행해, 컵케이크 페이지 내 댓글 기능이 잘 작동하는지 확인해보세요 :)

Comments (2)

Write comment


Submit



user1

Delicious cupcake :) I want to eat it right away!

Aug. 4, 2016, 5:23 p.m.



user2

Looks delicious :)

Aug. 4, 2016, 6:03 p.m.

Tip : PythonAnywhere 또는 Azure 배포한다음 댓글 기능을 다시 테스트 해보세요.
 요. `migrate` 명령어를 사용해야 합니다!

3. Travis CI와 Coveralls

코멘트를 커밋하기 전에 테스트를 하고 그 다음에 머지 하세요.

고맙게도, 지속적인 통합(Continuous Integration)은 코드와 관련된 문제를 피할 수 있도록 도와줍니다. 이 지속적인 통합(Continuous Integration)은 종종 CI라고 축약해서 불리곤하는데요, CI는 커밋이 이루어질 때 자동으로 빌드하고 테스트를 실행하도록 하는 과정을 말합니다.

우리는 Travis-CI라는 오픈소스 프로젝트를 사용해 볼거예요! 지금 <https://travis-ci.org>에 접속하고 여러분의 Github 계정을 이용해서 가입하세요. 여러분의 Github 저장소를 연결하면 TRAVIS-CI를 사용할 수 있습니다.



파일에 내용을 추가하기 전에 프로젝트의 root 디렉토리에 `.travis.yml` 라는 이름의 파일을 새로 만드세요. 그럼 이제 <https://coveralls.io>에서 가입을 완료해볼까요? 성공적으로 가입을 완료하면 여러분의 Github 저장소를 추가할 수 있습니다.

Coveralls는 시간이 지남에 따라 변경되는 코드를 추적하고, 모든 새로운 코드가 완전히 반영되었는지 확인할 수 있도록 도와주는 웹 서비스입니다.



`.travis.yml` 파일을 열고 아래의 코드를 추가하세요.

```

language: python
python:
  - '3.5'
branches:
  only:
    - 'advance'
    - 'rest-api'
install:
  - pip install -r requirements.txt
  - pip install coveralls
script:
  - python manage.py test
  - coverage run --source=djangocupcakeshop,menu,accounts manage
.py test
notifications:
  email: false
after_success:
  coveralls

```

branches 필드를 수정하는 것 잊지마세요! 우리는 advance와 rest-api라는 branches 만 추가할 거예요.

그럼 README에서 빌드와 coverage 상태를 볼 수 있는 멋진 배지를 추가해볼까요? 여러분의 프로젝트 README.md에 Travis와 Coveralls의 markup을 추가하세요. 아래의 예시가 있습니다.



그럼 변경된 사항들을 Github에 commit하고 push 하세요! 이제 자동으로 travis-ci 빌드와 coverage 테스트를 진행합니다. 여러분은 아래와 비슷한 보고서를 확인할 수 있습니다.



4. 데이터베이스(MySQL) 변경하기

여러분은 Postgres나 MySQL과 같은 다른 데이터베이스를 사용할 수 있습니다. 아래의 코드를 참고하여 settings를 수정하고 MySQL을 사용해보아요!

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'mydatabase',  
        'USER': 'mydatabaseuser',  
        'PASSWORD': 'mypassword',  
        'HOST': '127.0.0.1',  
        'PORT': '5432',  
    }  
}
```

여러분은 `mysqlclient` 를 설치해야만 MySQL을 데이터베이스로 사용할 수 있으니 주의해주세요.

5. 관리자 페이지를 내 마음대로 변경하기

여러분은 Django 관리자 페이지를 마음대로 변경할 수 있습니다! 자세한 내용은 Django 공식 튜토리얼을 참고하세요! 여러분을 도와줄 예시와 가이드가 준비되어있습니다. [official tutorial](#)

Azure

Relevant branch `azure`

Azure는 마이크로 소프트의 클라우드 서버예요. 컵케이크 사이트를 Azure에도 배포할 수 있습니다. 무료 계정을 만들어서 Azure를 체험해 보세요. 장고걸스 코치가 Azure에 장고 사이트를 배포 할 수 있도록 친절한 가이드를 이미 준비해놨답니다!

1. English Tutorial : <https://jinpark-dg.gitbooks.io/django-girls-azure/content/>
2. Korean Tutorial : <https://github.com/askdjango/azure-webapp-django-setup>

TO-DO (English)

영어로 된 [튜토리얼](#)에서 **Before we start** 부분에 zip file을 다운로드 받는 부분이 있는데요. 조금 바뀌어야 할 부분이 있어요!

web.config

web.config 파일에서 `python <add key="DJANGO_SETTINGS_MODULE" value="mysite.settings" />` 을 아래와 같이 바꿔주세요!

```
<add key="DJANGO_SETTINGS_MODULE" value="djangocupcakeshop.settings" />
```

또 이부분을 `<add input="{REQUEST_URI}" pattern="/static/.*" ignoreCase="true" negate="true" />` 아래와 같이 바꿔주세요!

```
<add input="{REQUEST_URI}" pattern="/media/.*" ignoreCase="true"
  " negate="true" />
</conditions>
```

requirements.txt 파일에서 다음과 같은 내용이 있는지 확인해 주세요!

```
Django==1.9.8
Pillow==3.0.0
```

settings.py

settings.py 파일에서 Azure 사이트를 허용하기 위해 다음과 같이 조금 바꿀꺼예요.

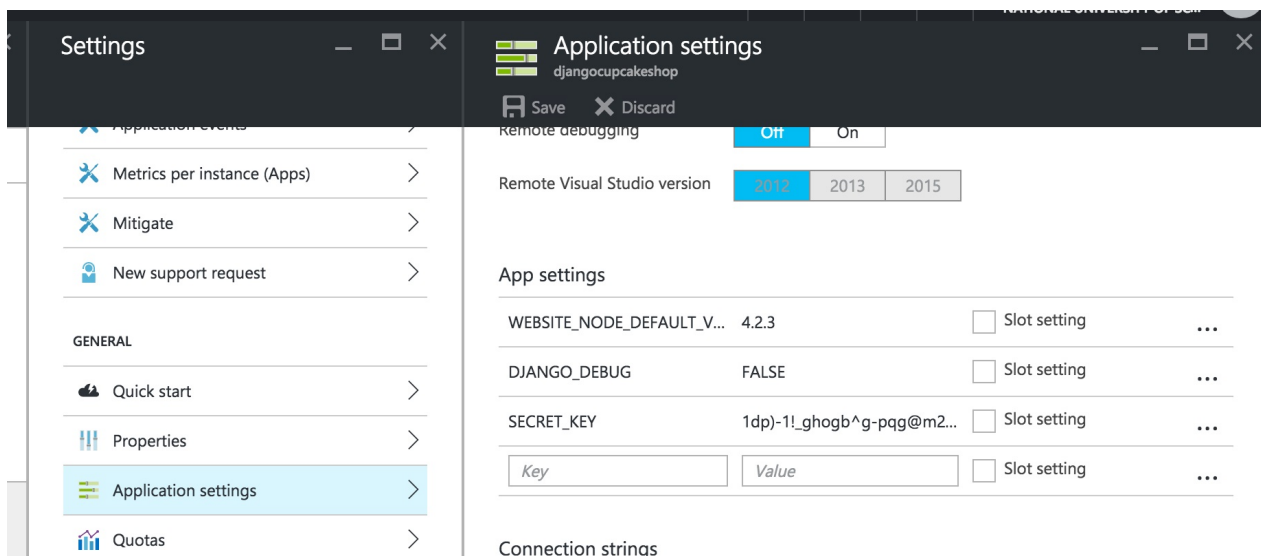
```
if DEBUG:
    ALLOWED_HOSTS = ['*']
else:
    ALLOWED_HOSTS = ['djangocupcakeshop.azurewebsites.net']
```

이제 [튜토리얼](#)로 돌아가서 **Deploying** 스텝을 마무리 해주세요!

Environment variables

한가지 더 해야 할 것이 있어요! PythonAnywhere에서 했던 것 처럼 **Environment variables** 를 추가해야 해요! Azure web app에서 **settings** 를 클릭 -->

Application settings 클릭 --> **App settings** --> 그리고 **DJANGO_DEBUG** 와 **SECRET_KEY** 값을 넣어주세요! 그리고 저장해 주세요!



이제 끝 !!!!

Azure에 배포한.djangocupcakeshop 사이트를 탐험해 볼까요!!

<https://djangocupcakeshop.azurewebsites.net/>

Bonus

Django is a powerful and sophisticated backend. Which means it can be used for a backend of apps and services. For example Instagram has the biggest deployment of Django as their backend.

In this tutorial, we are going to use Django REST Framework to add API endpoints to our model and use that API to make a simple Android app.

Installing REST framework

```
$ pip install djangorestframework
```

Add `rest_framework` to `INSTALLED_APPS` in `settings.py`

```
INSTALLED_APPS = (  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'menu',  
    'accounts',  
    'rest_framework',  
)
```

Add default Rest Framework Renderer in `settings.py`

```
REST_FRAMEWORK = {  
    'DEFAULT_RENDERER_CLASSES': (  
        'rest_framework.renderers.JSONRenderer',  
    )  
}
```

Creating a Serializer class

The first thing we need to get started on our Web API is to provide a way of serializing and deserializing the snippet instances into representations such as json

Create a file in menu directory named `serializers.py` and add the following content.

```
from rest_framework import serializers
from .models import Cupcake

class CupcakeSerializer(serializers.ModelSerializer):
    class Meta:
        model = Cupcake
        fields = ('name', 'rating', 'price', 'image', 'writer', '
createdAt')
```

Django Views using Serializer

Add a url for cupcake list in `menu/urls.py`

```
url(r'^api/v1/cupcakes/$', views.cupcakes_list),
```

Add the following code in `menu.views.py` import part.

```
from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt
from rest_framework.renderers import JSONRenderer
from rest_framework.parsers import JSONParser
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status
from .serializers import CupcakeSerializer
```

and view function

```
@api_view(['GET'])
def cupcakes_list(request):
    """
    REST Api V1
    List all cupcakes
    """
    if request.method == 'GET':
        cakes = Cupcake.objects.all().order_by('-createdAt')
        serializer = CupcakeSerializer(cakes, many=True)
        cake_json = Response(serializer.data)
        return JsonResponse(serializer.data)
```

Start development service and test API either using `curl` or `httpie`. In our case, we use `postman` to test it. You will get response like below

```
[
  {
    "name": "Another Chocolate",
    "rating": 5,
    "price": "5",
    "image": "/media/images/cakes/chocolate_cupcake_AjwFoHt.jpg"
  },
  {
    "name": "Green Tea Cupcake",
    "rating": 4,
    "price": "4",
    "image": "/media/images/cakes/cup_cake_2.jpg",
    "writer": "admin",
    "createdAt": "2016-08-01T03:19:29.333117Z"
  },
  {
    "name": "Blueberry Muffin",
    "rating": 4,
    "price": "3",
    "image": "/media/images/cakes/blueberry_muffin.png",
    "writer": "admin",
    "createdAt": "2016-07-29T15:35:40Z"
  },
  {
    "name": "Blueberry Cookie Cupcake",
    "rating": 3,
    "price": "5",
    "image": "/media/images/cakes/blueberry_cupcake.png",
    "writer": "admin",
    "createdAt": "2016-07-28T01:52:28Z"
  }
]
```

Android Cupcake App

The app with Django REST API [link](#)

iOS Cupcake App

Initial version of the iOS app [[link](#)]