**DjagoGirls Project**

https://tutorial.djangogirls.org/

Run cmd in project folder 🡪 E:\djangoGirls>python -m venv myvenv 🡪 E:\djangoGirls>myvenv\scripts\activate 🡪 (myvenv) E:\djangoGirls>python -m pip install --upgrade pip 🡪 **make a file as:** requirements.txt in djangoGirls folder including: Django~=3.2.10 🡪 (myvenv) E:\djangoGirls>pip install -r requirements.txt 🡪 **make a site:** (myvenv) E:\djangoGirls>django-admin startproject mysite . 🡪 **make an application:** (myvenv) E:\djangoGirls>django-admin startapp blog 🡪 (myvenv) E:\djangoGirls>code .

**Run Server:** new CMD in project main folder (djangoGirls) : E:\djangoGirls>myvenv\scripts\activate 🡪 python manage.py runserver 🡪 **open your server in browser:** http://127.0.0.1:8000/

Make your site Persian: vs Code 🡪 mysite 🡪 setting.py :

Setting.py

LANGUAGE\_CODE = 'fa-ir'

TIME\_ZONE = 'Asia/Tehran'

STATIC\_URL = '/static/'

STATIC\_ROOT = BASE\_DIR / 'static'

ALLOWED\_HOSTS = ['127.0.0.1', '.pythonanywhere.com']

INSTALLED\_APPS = [

    . . .

    'blog',

]

In first CMD: (myvenv) E:\djangoGirls>python manage.py migrate

### Creating a blog post model

We want to make a blog with following properties: author, title, text, created\_date, published\_date.

# blog/models.py

from django.conf import settings

from django.db import models

from django.utils import timezone

class Post(models.Model):

    author = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

    title = models.CharField(max\_length=200)

    text = models.TextField()

    created\_date = models.DateTimeField(default=timezone.now)

    published\_date = models.DateTimeField(blank=True, null=True)

    def publish(self):

        self.published\_date = timezone.now()

        self.save()

    def \_\_str\_\_(self):

        return self.title

### Create tables for models in your database

The last step here is to add our new model to our database. First we have to make Django know that we have some changes in our model. (We have just created it!) Go to your console window and type python manage.py makemigrations blog.

(myvenv) E:\djangoGirls>python manage.py makemigrations blog

Django prepared a migration file for us that we now have to apply to our database. Type python manage.py migrate blog and the output should be as follows:

(myvenv) E:\djangoGirls>python manage.py migrate blog

**Django admin**

To add, edit and delete the posts we've just modeled, we will use Django admin.

Let's open the blog/admin.py file in the code editor and replace its contents with this:

blog/admin.py

from django.contrib import admin

from .models import Post

admin.site.register(Post)

Go to your browser and type the address <http://127.0.0.1:8000/admin/>. You will see a login page.

To log in, you need to create a superuser - a user account that has control over everything on the site. Go back to the command line, type python manage.py createsuperuser

(myvenv) E:\djangoGirls>python manage.py createsuperuser

Return to your browser. Log in with the superuser's credentials you chose; you should see the Django admin dashboard. Go to Posts and experiment a little bit with it. Add five or six blog posts.

# Deploy your website!

Until now, your website was only available on your computer. Now you will learn how to deploy it! Deploying is the process of publishing your application on the Internet so people can finally go and see your app.

As you learned, a website has to be located on a server. There are a lot of server providers available on the internet, we're going to use PythonAnywhere. PythonAnywhere is free for small applications that don't have too many visitors so it'll definitely be enough for you now.

The other external service we'll be using is GitHub, which is a code hosting service. There are others out there, but almost all programmers have a GitHub account these days, and now so will you!

## Installing Git

You can download Git from [git-scm.com](https://git-scm.com/) and install it. You can hit "next" on all steps except for two: in the step where it asks to choose your editor, you should pick Nano, and in the step entitled "Adjusting your PATH environment", choose "Use Git and optional Unix tools from the Windows Command Prompt" (the bottom option). Other than that, the defaults are fine. Checkout Windows-style, commit Unix-style line endings is good.

Do not forget to restart the command prompt or PowerShell after the installation finished successfully.

### Ignoring files in Git:

Git will track changes to all the files and folders in this directory, but there are some files we want it to ignore. We do this by creating a file called .gitignore in the base directory. Open up your editor and create a new file with the following contents:

# Python

\*.pyc

\*~

\_\_pycache\_\_

# Env

.env

myvenv/

venv/

# Database

db.sqlite3

# Static folder at project root

/static/

# macOS

.\_\*

.DS\_Store

.fseventsd

.Spotlight-V100

# Windows

Thumbs.db\*

ehthumbs\*.db

[Dd]esktop.ini

$RECYCLE.BIN/

# Visual Studio

.vscode/

.history/

\*.code-workspace

## Starting our Git repository

Git tracks changes to a particular set of files in what's called a code repository (or "repo" for short). Let's start one for our project. Open up your console and run these commands, in the djangogirls directory:

(myvenv) E:\djangoGirls>git init

Then, you should sign in in github.com. And then:

(myvenv) E:\djangoGirls>git config --global user.name "fAziz1985"

(myvenv) E:\djangoGirls>git config --global user.email fatemeh.azizzadeh1985@gmail.com

(myvenv) E:\djangoGirls>git status

Go to github.com, login and make a new repository: djangoGirlsPr.

The git status command returns information about any untracked/modified/staged files, the branch status, and much more.

And finally we save our changes. Go to your console and run these commands:

(myvenv) E:\djangoGirls>git add .

(myvenv) E:\djangoGirls>git commit -m "first commit"

(myvenv) E:\djangoGirls>git branch -M main

(myvenv) E:\djangoGirls>git remote add origin https://github.com/fAziz1985/djangoGirlsPr.git

(myvenv) E:\djangoGirls>git push -u origin main