----

1 - Introduction

In this file I will describe briefly what I have done to achieve my first blog in Python, but before that let me put a little description about the project;

The blog is consisted of two Models (Post and Comment: a Post could has 0/* Comments), the user should authenticate using his login/password to access then:

-Insert a post - Only a logged-in user can create a post

-Listing posts - All users can see the posts

-Add Comment - Anyone can write reviews

The blog is coded with Python programming language and Django as the framework.

----

2 - Installation

----

1 - Download Python 3.4.3 from https://www.python.org/downloads/ and install the environment.

2 - Install the required packages (setup tools 18.0.1) for Python from

https://pypi.python.org/pypi/setuptools

- Using Windows 8(which includes PowerShell 3) it's possible to install with one simple PowerShell command. Start up PowerShell with administrative privileges and paste those commands:

```
> (Invoke-WebRequest https://bootstrap.pypa.io/ez_setup.py).Content | python -

> (Invoke-WebRequest https://bootstrap.pypa.io/ez_setup.py).Content | python - --user

> (Invoke-WebRequest https://bootstrap.pypa.io/ez_setup.py).Content | py -3 -
```

3 - Download Django 1.8.2 Web Framework from https://www.djangoproject.com/download/, I have used the first option: Get the latest official version by installing it with pip:

```
> pip install Django==1.8.2
```

----

3 - Deployment

Due to that it was my first blog (It was really interesting actually), I followed the tutorial in the

Django Official web site documentation (https://docs.djangoproject.com/), It was a good detailed tutorial that helped me greatly.

The steps needed to create the blog:

1 - Create a Django project

```
$ django-admin startproject myBlog
```

- The hierarchy should be like this:

```
myBlog/
   manage.py
   myBlog/
         __init__.py
         settings.py
         urls.py
         wsgi.py
```

2 - Like the 'code first approach in asp.net', python create a version of the database and any necessary tables by migrating the database settings in myBlog/settings.py

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

3 - Start you server by running the following command:

```
$ python manage.py runserver
```

Then access http://127.0.0.1:8000/ with the web browser.

4 - Creating models:

A-First create the directory polls (which will house the poll application) by typing this command

```
$ python manage.py startapp polls
```

B-Modify the polls/models.py by:

```
polls/models.py

from django.db import models

class Post(models.Model):

    post_text = models.CharField(max_length=200)

    pub_date = models.DateTimeField('date published')

class Comment(models.Model):

    post = models.ForeignKey(Post)

    comment_text = models.CharField(max_length=200)
```

C-Edit the mysite/settings.py file, and change the INSTALLED_APPS setting to include the string 'polls'.

```
INSTALLED_APPS = (

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'polls',

)
```

5 - To include the polls app.

```
$ python manage.py makemigrations polls
```

6 - I am really interested in the Amin site, that why I was delighted to see that Django automatically generate a coherent good structured admin site.

A-First you must create an admin user for your site

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

(I have created an admin with the following Username: admin, password: 0000) b- Some changes must be made in polls/admin.py to include the two models (Post and comment)

```
from django.contrib import admin
from .models import Post, Comment
class CommentInline(admin.TabularInline):
    model = Comment
    extra = 30
class PostAdmin(admin.ModelAdmin):
    list_display = ('post_text', 'pub_date', 'was_published_recently')
    fields = ['pub_date', 'post_text']
    list_filter = ['pub_date']
    search_fields = ['post_text']
admin.site.register(Post, PostAdmin)
admin.site.register(Comment)
```

C-Now everything should be in place, you can either create/modify/delete a post or a comment, notice that you must identify the post before creating a comment(Foreign Key), also I have included some modifications like the Post list filter and the search filter.

First, you must improve the polls/models.py code by giving the Post a few attributes:

```
class Post(models.Model):
    post_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    def __str__(self):
    return self.post_text
    def was_published_recently(self):
    return self.pub_date >= timezone.now() - datetime.timedelta(days=1)
    was_published_recently.admin_order_field = 'pub_date'
    was_published_recently.boolean = True
    was_published_recently.short_description = 'Published recently?'
```

7 - Now with the final part, which is the simple user views, you can easily create a user in the admin Dashboard. Then create a file urls.py under myBlog\polls that will include the view URL.

I have created 5 views:

-Index: Hello World

-Results: return a post by its id

-List_posts: Return all the posts in the database.

-List comments: Return all the comments in the database.

----

4 – Conclusion

I have done my best to show in this project that I could help with the best I can in the development of your honorable company and I am really willing to develop my skills and gain more experience in Python Programming Language. I am enthusiastic about this opportunity and I am certain that I will find professionals that take their job seriously and pleased to be lead by them.