MIT AITI
Django Lab 2: Models

In this lab, we will set up the models and data for a blog. In Lab 1, we did the models, views, and templates at once, but in the next several labs, we will be focusing on each part individually. In larger projects, this is how things will often work – one person will do models, one will do views, and another will do the templates. A designer doesn't need to worry about the database, and the database designer doesn't need to worry about the display style.

Today we will make models to represent a blog with comments on each post. We will have a model for a post and a model for a comment.

If you get confused, look at the resources available to you:

1. Lecture slides
2. Django lab 1
3. Other students
4. Django documentation
    a. Models: https://docs.djangoproject.com/en/1.5/topics/db/models/
    b. Field reference: https://docs.djangoproject.com/en/1.5/ref/models/fields/
5. Google
6. Instructors

1. Create a new project called "`myblog`"
```
$ cd ~/Desktop
$ django-admin.py startproject myblog
$ cd myblog
```

2. Edit your settings.py file by typing
```
$ gedit myblog/settings.py
```

   Fill in these variables:

```
DATABASES = {
    ...
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'blog.db',
    ...
}
...
TIME_ZONE = 'Africa/Accra'
```

3. Create a new application (You should be in ~/Desktop/myblog right now):
```
$ django-admin.py startapp blog
$ cd blog
```

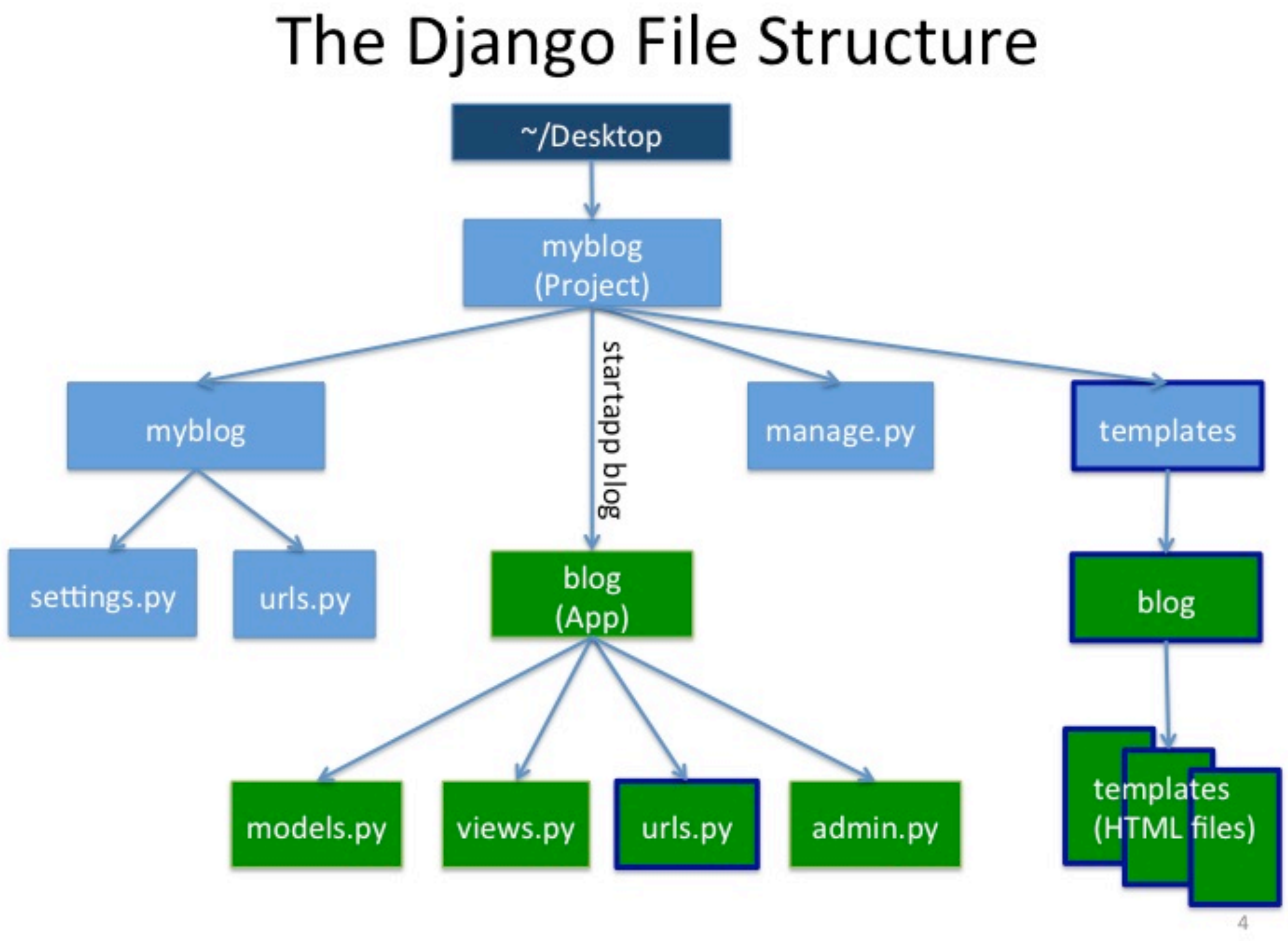
4. Edit the models file and create two models. Look at yesterday's lab and today's handout for a reminder of how to do this.
    a. `Blog`:
        i. `title` (60 characters)
        ii. `body` (large text)
        iii. `created` (date created)
        iv. `updated` (date updated)
    b. `Comment`:
        i. `body` (large text)
        ii. `author` (60 characters)
        iii. `created` (date created)
        iv. `updated` (date updated)
        v. `post` (foreign key linking `Comment` to `Blog`)
5. Configure your django project to allow the built-in admin
    a. Open `settings.py` and change the installed app setting:
    ```
    INSTALLED_APPS= (
        ...
        'blog',
        'django.contrib.admin',
    )
    ```
    b. Open `urls.py` (the one in ~/Desktop/myblog/myblog) and uncomment the admin lines:
    ```
    ...
    from django.contrib import admin
    admin.autodiscover()
    ...
    urlpatterns = patterns('',
        ...
        url(r'^admin/', include(admin.site.urls)),
    ...
    ```
6. Add admin information to the models
    a. At the top of your `models.py`, add:
    ```
    From django.contrib import admin
    ```
    b. At the end of your `models.py`, add:
    ```
    admin.site.register(Blog)
    ```
7. Update the database and start the server
    ```
    $ cd ~/Desktop/myblog
    $ python manage.py syncdb
    $ python manage.py runserver
    ```
    Remember to say "yes" to creating a superuser.

8. Go to admin and add 3 posts.
    http://localhost:8000/admin

Django Reference: File Structure

Here is a guide to the file and directory structure of a Django project. Use this to figure out how to navigate to each file in your project. Arrows mean one file or folder is within another folder. For example, settings.py is inside myblog, which is inside another folder named myblog, which is in Desktop. You can therefore open settings by typing:
```
gedit ~/Desktop/myblog/myblog/settings.py
```

Blue files/folders are related to the myblog *project* as a whole. Green files/folders are specific to the blog *app*. Folders with a blue outline need to be created manually; all others are created for you by Django and you just add your code in them.



Django Reference: Models

Here is an example of a model, from lab 1:

```
from django.db import models

class Notes(models.Model):
    title   = models.CharField(max_length=255)
    content = models.TextField()
    def __unicode__(self): #note: 2 underscores on each side
        return self.title
```

Some points to note:

- Every model *inherits from* models.Model
- The __unicode__ function tells Django what to call each instance of the model when displaying a list of them. For example, each Note should be called by its title.

Here are some types of Fields you can use in your models:

BooleanField
- Checkbox

CharField(max_length)
- Single-line textbox

DateField
- Javascript calendar

DateTimeField
- Javascript calendar, time picker

DecimalField(max_digits, decimal_places)
- Decimal numbers

EmailField
- Charfield that validates email address

FileField
- File upload, stores path in database

FloatField
- Floating point numbers

IntegerField
- Integer textbox

PositiveIntegerField
- Integer textbox for positive integers

TextField
- Multi-line textbox

TimeField
- Time picker

URLField
- Textbox for URLs

Special Fields denoting relationships:

ForeignKey(foreign class)
- Many-to-one

ManyToManyField(foreign class)
- Uses a temporary table to join tables together

OneToOneField(foreign class)
- Enforces uniqueness