**Disabling Django Admin**

If you don’t want Django Admin enabled, how is it turned off? There are two things to do (**Note,** do not do disable Django admin, this is just so you understand how to do it):

* Remove django.contrib.admin from INSTALLED\_APPS in the Django settings file
* Remove the admin/ URL rule from the project’s urls.py file

While on the subject of the admin URL, you can get a little extra security if you change it from the default admin/ to something else. This would prevent attackers from guessing the URL. Since the admin site is password protected anyway, this is not something you’d normally have to do, but it could help if any of your admin users have weak passwords. We’ll stick with the default admin/ in this course.

**Registering Models to the admin UI**

There are two ways to register a model and have it show in the Django admin UI. You can either register the model directly, which will just use the defaults and allow all fields to be editable. Most of the time, this is adequate, but sometimes you might want to create a model admin class by subclassing the Django admin.ModelAdmin class. This allows you to set options about how your model is displayed in the admin. Let’s examine the admin.py file and then see how to implement both these methods.

When scaffolded the admin.py file contains just one code line (and a comment, which can be removed):

from django.contrib import admin

As you can probably guess, this imports the Django admin module, ready for use. To register a model into the admin section, we use the admin.site.register function.

Let’s start by looking at the simplest method of registering a model. We’ll do this with the Tag model. First our model needs to be imported into the admin.py file.

from blog.models import Tag, Post

Then register it:

admin.site.register(Tag)

To configure how the admin site behaves with a certain model, a subclass of admin.ModelAdmin must be created. This subclass’s attributes determine how the model is displayed. First let’s look at how we’ll create one, for the Post model.

class PostAdmin(admin.ModelAdmin):

prepopulated\_fields = {"slug": ("title",)}

But, there are many other ways to customise the ModelAdmin. Some of the more common customizations are:

* exclude: a list of fields to disallow editing of in the admin. For example, we might want to prevent users from manually setting the slug, and instead compute it when saving the Model. In which case, we would set exclude to ["slug"].
* fields: this works the opposite way to exclude. If set, only fields in the fields list will be editable. Note that if a field requires a value, but is not editable (either by the use of exclude or fields), then saving the model instance will fail because the field will not be valid.
* list\_display: a list of fields to include in the admin page list view. For example, we might want to see both a Post’s title, and when it was published. We would do this by setting list\_display to ["title", "published-at"].

**Launching the Blog**

Now we’re ready to check it all out. Start the Django test server by entering the following command in the terminal. The blog should load in the top panel.

python3 manage.py runserver 0.0.0.0:8000

Once, Django is up and running, you need to navigate to the admin page. In the project URL, add a /admin and press **Enter** on the keyboard.