Python Django入門 (3)

Python　Django

Djangoを始める

ドキュメント

- Django ドキュメント

「何もないところから始めるなら：概要」と「チュートリアルはこちら：その１～７」を一読するのがよいのですが、まずはこの講座を先にやってみてもよいでしょう。

- 基本的にコピペでOKです。
- どう動くのか、なんとなくわかったら、公式のチュートリアルをやってみて、理解を深めましょう。

Djangoで作ってみる

プロジェクトの作成

簡単な書籍を管理するアプリを作成してみます。

まず、mybook というプロジェクトを作ります。これには `django-admin startproject mybook` というコマンドを使います。

> PyCharmで、お試しでプロジェクトを作ってみると、PycharmProjectsというフォルダができるようなので、その下に作ることにします
>
> (env1) の仮想環境にいることを確認して下さい。
> 既にいる場合は、以下のMacの `workon env1` 、Windowsの `Scripts¥activate` は不要です。

Macの方は、

```
$ workon env1

(env1) $ cd ~/PycharmProjects/
(env1) $ django-admin startproject mybook
```

Windowsの方は、

```
C:¥Users¥hoge>cd Documents¥env1
C:¥Users¥hoge¥Documents¥env1> Scripts¥activate

(env1) C:¥Users¥hoge¥Documents¥env1> cd C:¥Users¥hoge¥PycharmP
(env1) C:¥Users¥hoge¥PycharmProjects> django-admin startprojec
```

ここから説明はMac前提に戻ります。

Windowsの方は、それぞれの環境に読み替えて下さい。

~/PycharmProjects/ にできたファイル

```
mybook/
    manage.py
    mybook/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

mybook ディレクトリの中に移動して下さい。

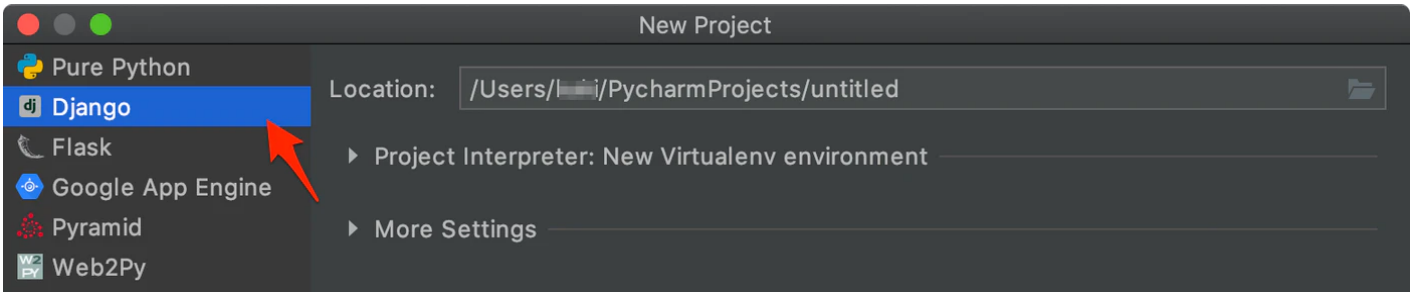
```
$ cd mybook
```

PyCharmでプロジェクトを開く

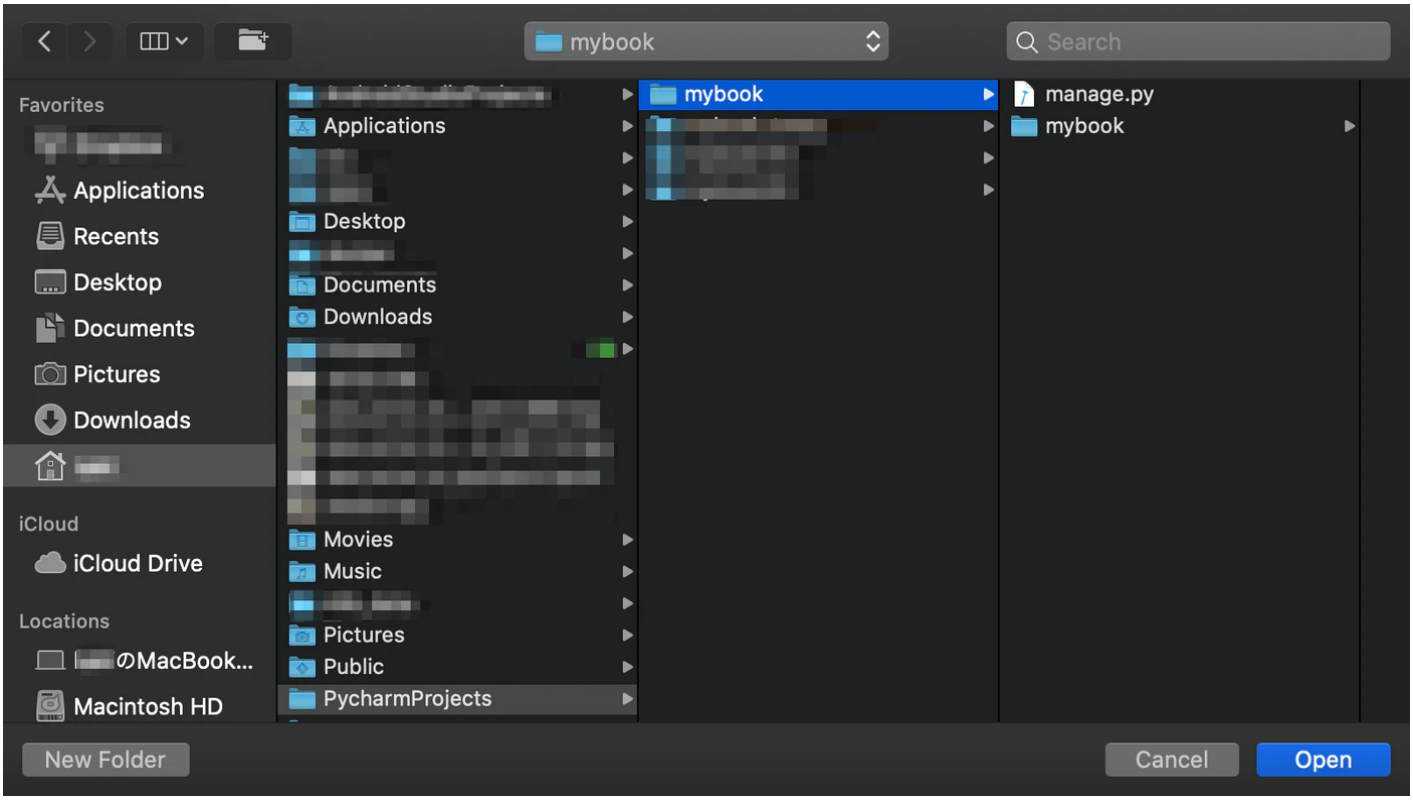
生成したプロジェクトをPyCharmで開いてみます。

1.Create New Project を選択します

2.左側で Django を選択します。

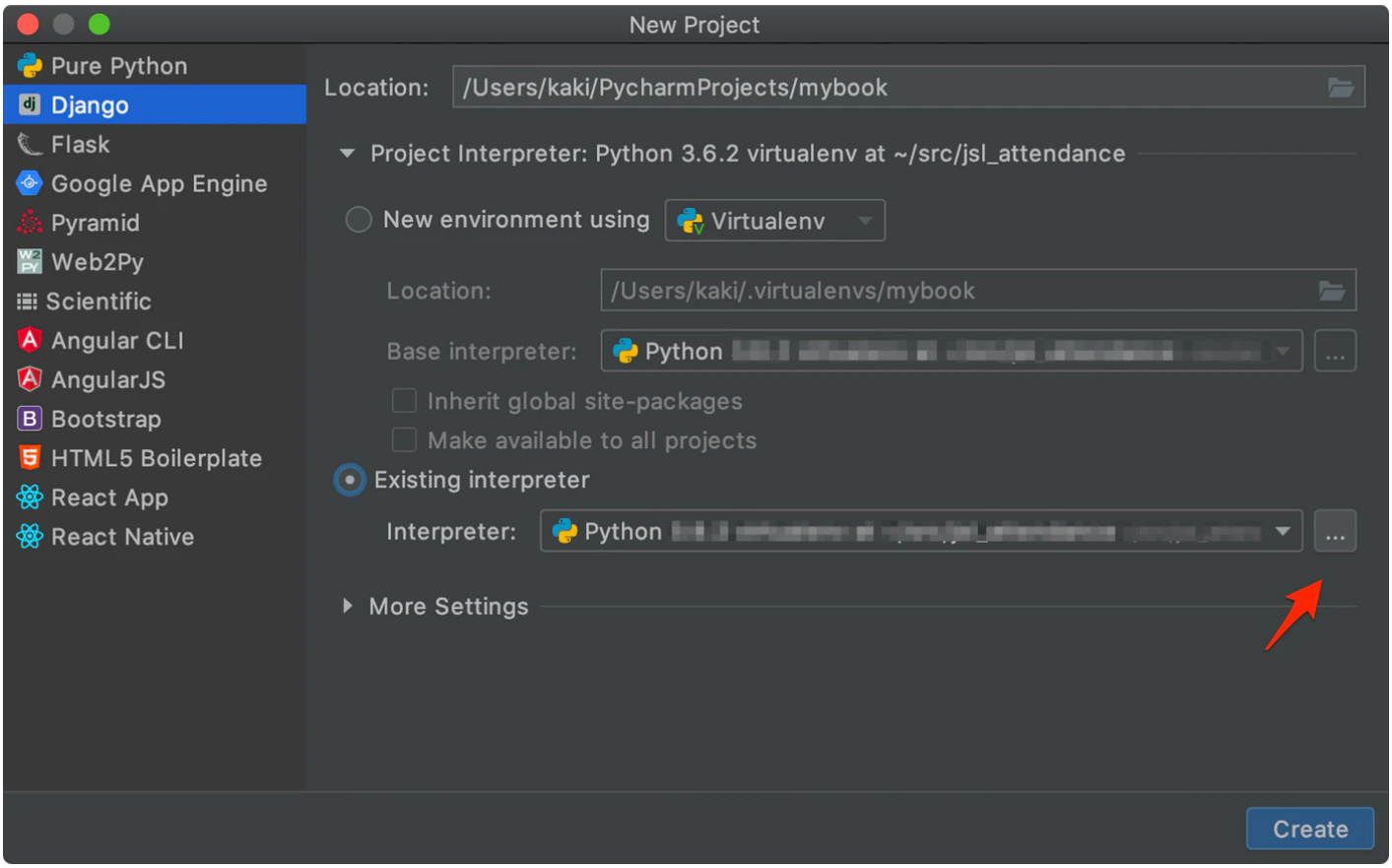


3.Location: 右側のフォルダ アイコンをクリックして、先程生成した mybook フォルダを選択します。

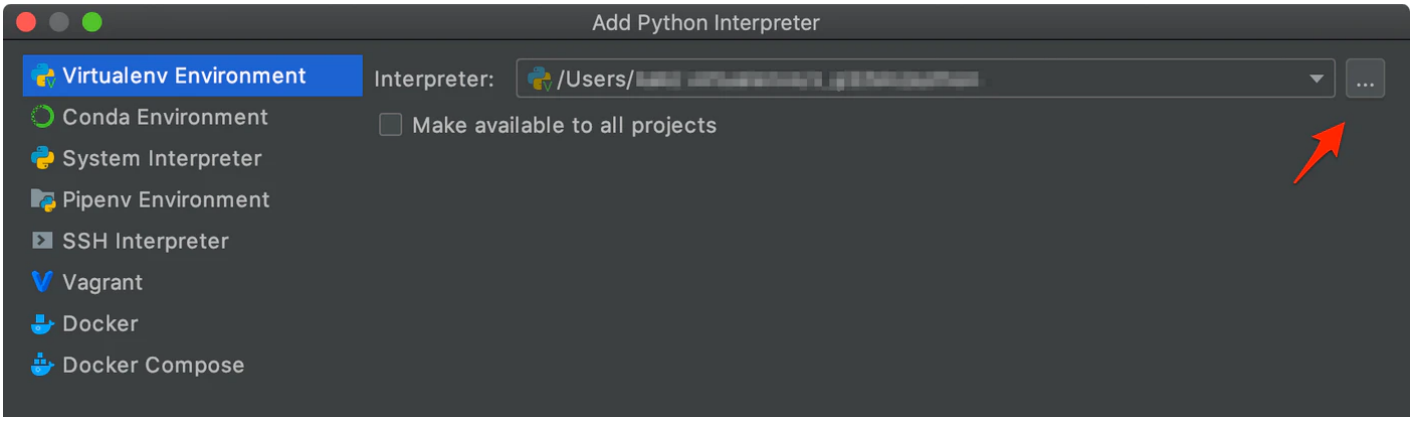


> Windowsの場合は、C:¥Users¥hoge¥PycharmProjects¥mybook になります。

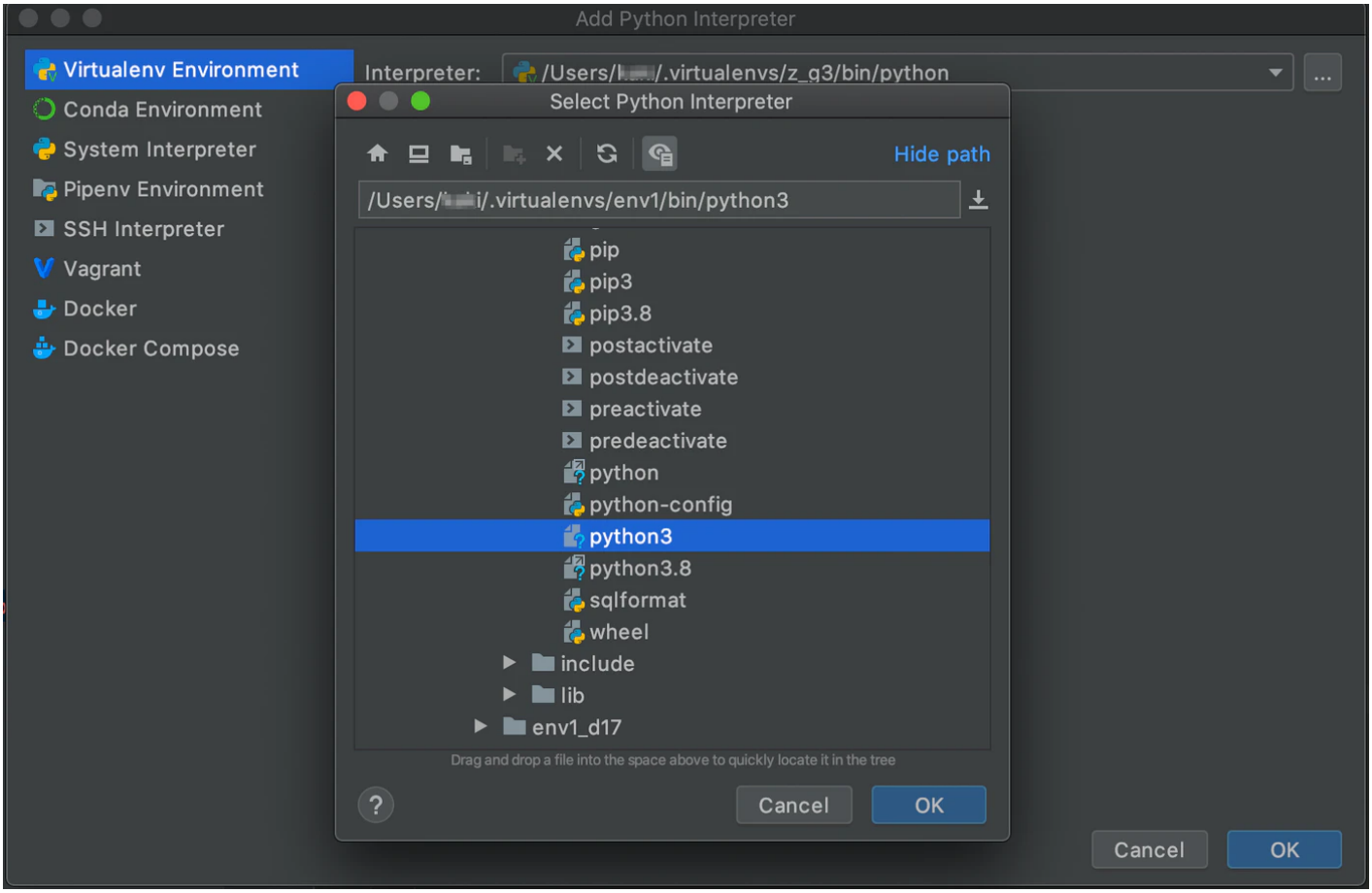
4.Project Interpreter: の三角アイコンをクリックして展開し、Existing Interpreter の方の Interpriter の「...」 を選択します。



5.Virtualenv Environment > Interpreter の「...」を更に選択します。

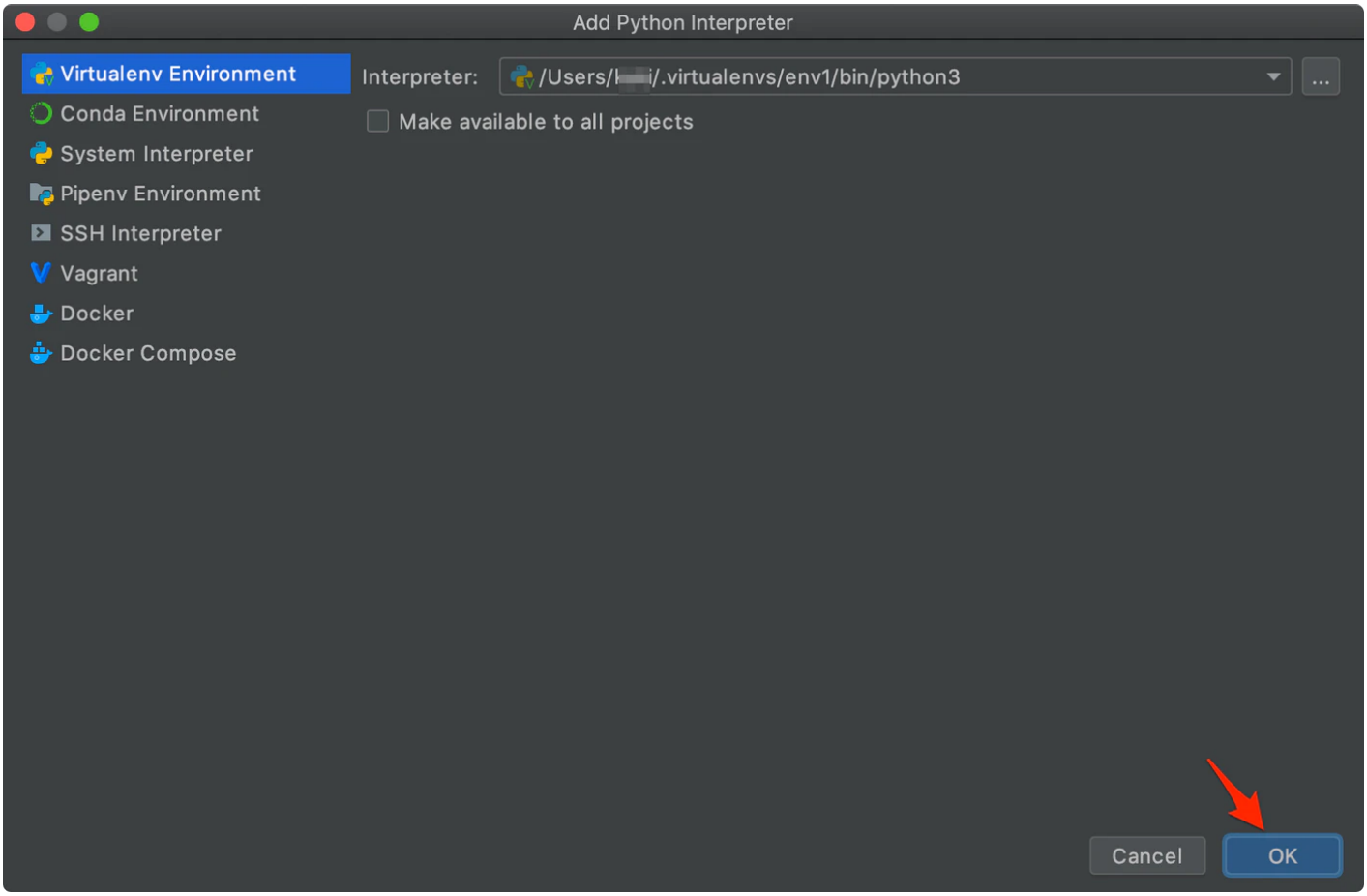


6.env1仮想環境のPython3を、Pythonインタープリターとして選択します

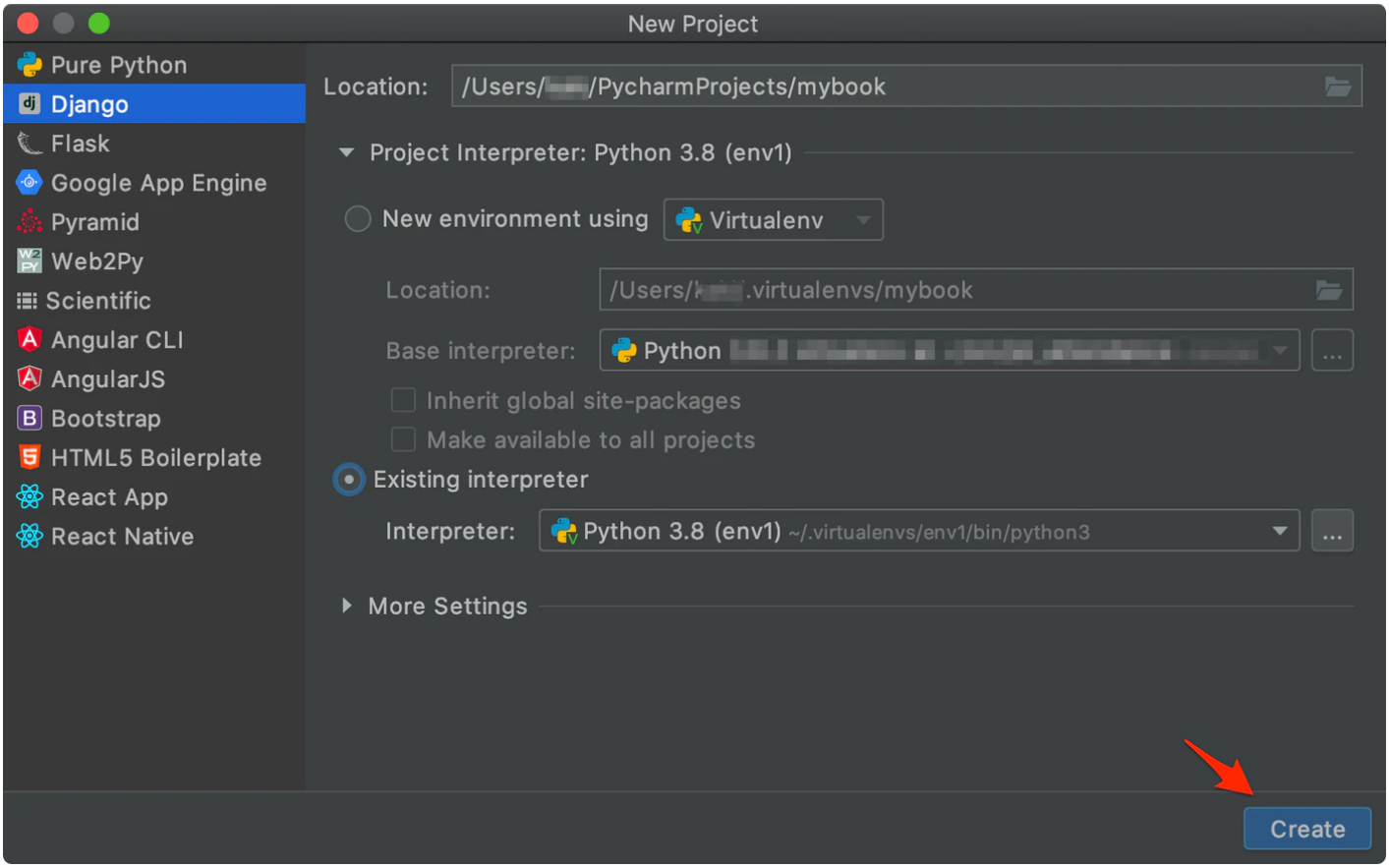


> Windowsの場合は、C:¥Users¥hoge¥Documents¥env1¥python.exe になります。

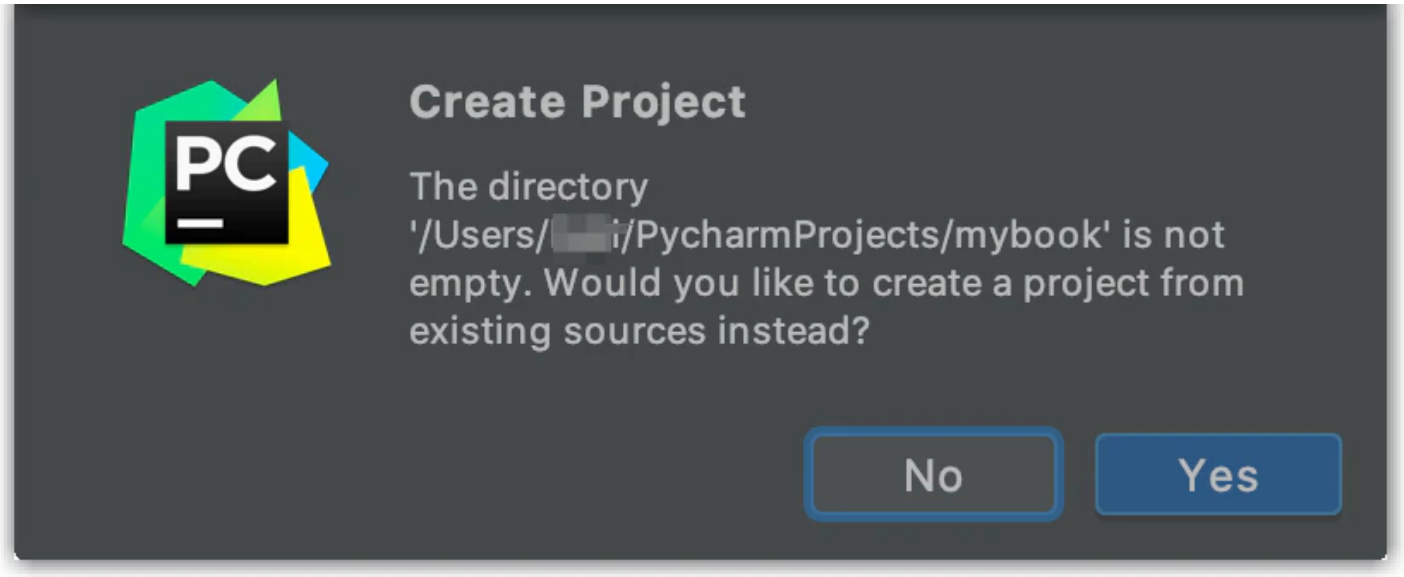
7.OK で閉じます。



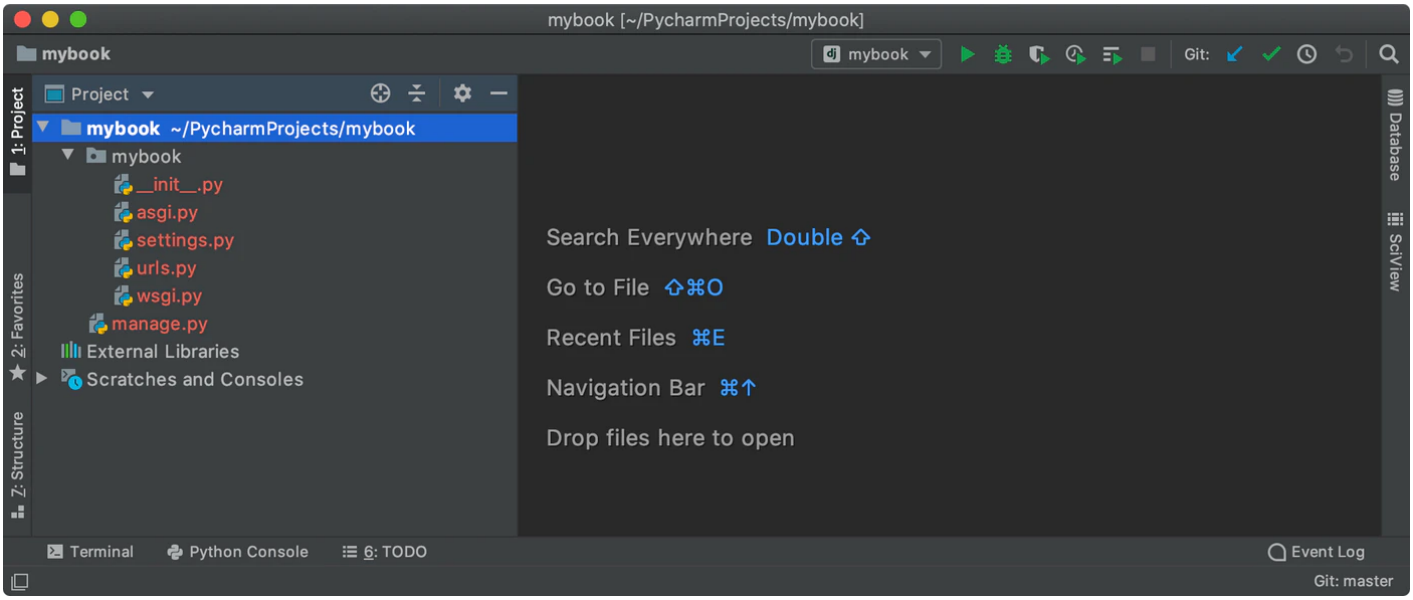
8.Createボタンを押します（左が Djangoになっているかを再確認）



9.既にあるけど、いいかと聞いてくるので、Yesと答えます



10.このようにできたら成功です



データベースのセットアップ

データベースの設定は、 `mybook/settings.py` にありますが、標準では、SQlite3を使うよう設定済です。

```
# Build paths inside the project like this: os.path.join(BASE_
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__f

    :

# Database
# https://docs.djangoproject.com/en/1.9/ref/settings/#database
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

mybook/settings.py を見たついでに、言語とタイムゾーンを日本にします。

> # はコメントアウトした行という意味です。コメントアウトするには、
> Macの場合は、行に位置付けて（複数行の場合は、行を選択して） command + / です。
> Windowsの場合は Ctrl + / です。
> うーん便利。

```
# LANGUAGE_CODE = 'en-us'
LANGUAGE_CODE = 'ja'
```

```
# TIME_ZONE = 'UTC'
TIME_ZONE = 'Asia/Tokyo'
```

データベースをマイグレートするコマンドを入れると、プロジェクト直下のディレクトリに `db.sqlite3` というファイルが作成されます。

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
```

```
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

以下のコマンドで、スーパーユーザーを作成します。

```
$ python manage.py createsuperuser
ユーザー名 (leave blank to use 'hoge'): admin
メールアドレス: admin@example.com
Password: hogefuga
Password (again): hogefuga
Superuser created successfully.
```

途中、以下の質問に答えます。

- ユーザー名: admin
- メールアドレス: admin@example.com ※適当でいいです
- パスワード: hogefuga
- パスワード(再入力): hogefuga

開発用サーバの起動

プロジェクトが動くかどうか、 `python manage.py runserver` というコマンドで開発用サーバを起動します。

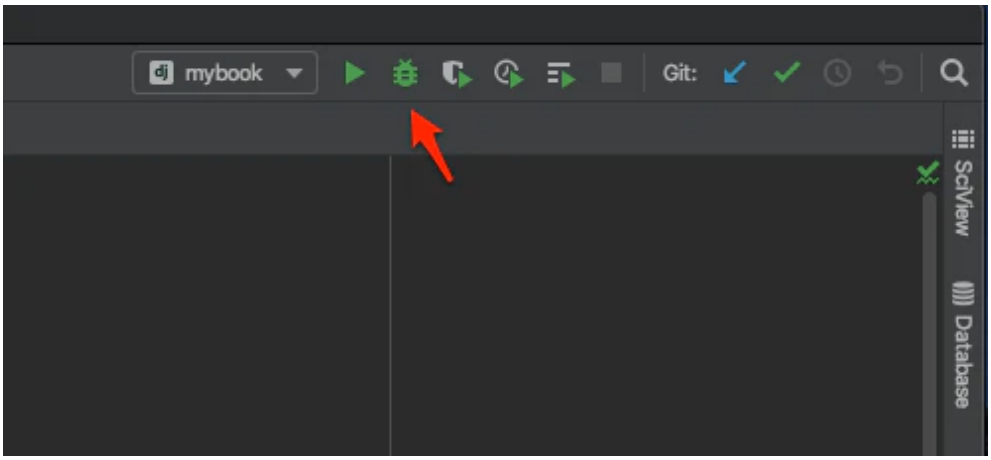
```
$ python manage.py runserver

Performing system checks...

System check identified no issues (0 silenced).
January 08, 2020 - 16:20:02
Django version 3.0.2, using settings 'mybook.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

ブラウザで http://127.0.0.1:8000/ にアクセスしてみます。

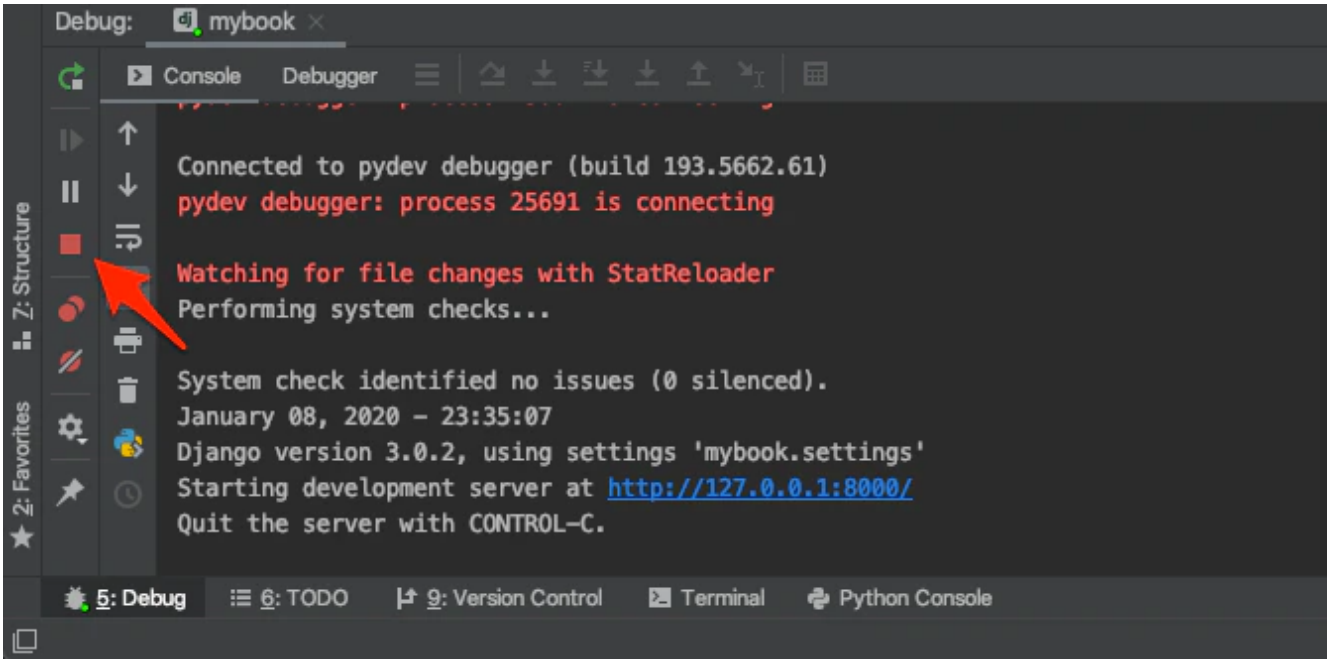
開発用サーバの終了は、 `control + c` です。

PyCharmでデバッグ実行する

画面の右上の虫マークを押すだけです。
もしくは、メニューの Run > Debug.. > mybook を選択します。

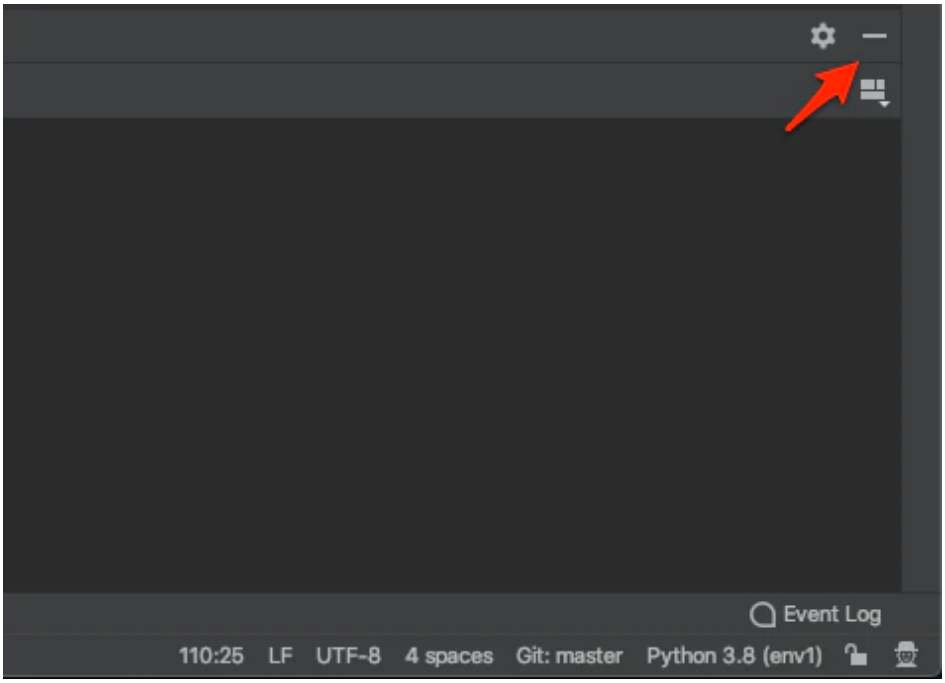


下にガバッと以下の様な表示が出ればOKです。
止めたい時は、赤い四角のボタンを押します。



一般的なIDEのように、コードの行番号の右側あたりをクリックすることで、ブレークポイントの設定や、変数の確認ができます。

止めたら、Debugウィンドウは邪魔なので、折り畳むボタンを押します。



アプリケーションの作成

プロジェクトの下に cms というアプリケーションを作成してみます。

cmsは、Contents management system、ようするにマスターメンテナンスのようなものとします。
アプリケーションは、プロジェクトを構成する機能の単位で、場合によっては再利用の対象とできる単位です。

以下のコマンドで作成します。

```
$ python manage.py startapp cms
```

mybook プロジェクトのディレクトリの下に、以下のファイルが作成されました。

```
mybook/
    cms/
        __init__.py
        admin.py
        apps.py
        migrations/
            __init__.py
        models.py
        tests.py
        views.py
    manage.py
    mybook/
       :
       :
```

モデルの作成

データベースに定義したいデータ モデルを、
`cms/models.py` に定義します。

> classの前は、2行開けないと、波線が出て警告されます。
> pep8というコードスタイルチェックの仕組みです。
> 同様に、コードの最後が改行のみで終わっていないと、また波線が出て怒られるので、気をつけましょう。

```
from django.db import models


class Book(models.Model):
    """書籍"""
    name = models.CharField('書籍名', max_length=255)
    publisher = models.CharField('出版社', max_length=255, blan
    page = models.IntegerField('ページ数', blank=True, default=

    def __str__(self):
        return self.name


class Impression(models.Model):
    """感想"""
    book = models.ForeignKey(Book, verbose_name='書籍', relatec
```

```
    comment = models.TextField('コメント', blank=True)

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

感想 は 書籍 に紐づく、子供のモデルとしています。

モデルを有効にする

cms アプリケーションをインストールしたことを、プロジェクトに教えてあげる必要があります。

cms/apps.py を開いてみると、CmsConfig というクラスが定義されています。
これを mybook/settings.py の INSTALLED_APPS の最後に 'cms.apps.CmsConfig', という文字列で追加します。

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'cms.apps.CmsConfig',   # cms アプリケーション
]
```

以下のコマンドで、models.py の変更を拾って、マイグレートファイルを作成します。

```
$ python manage.py makemigrations cms
Migrations for 'cms':
  cms/migrations/0001_initial.py
    - Create model Book
    - Create model Impression
```

このマイグレートファイルが、どのような SQL になるか、以下のコマンドで確認できます。

```
$ python manage.py sqlmigrate cms 0001
BEGIN;
--
-- Create model Book
--
CREATE TABLE "cms_book" ("id" integer NOT NULL PRIMARY KEY AUT
--
```

```
-- Create model Impression
--
CREATE TABLE "cms_impression" ("id" integer NOT NULL PRIMARY K
CREATE INDEX "cms_impression_book_id_b2966102" ON "cms_impress
COMMIT;
```

まだデータベースに反映していないマイグレートファイルを、以下のコマンドでデータベースに反映します。

```
$ python manage.py migrate cms

Operations to perform:
  Apply all migrations: cms
Running migrations:
  Applying cms.0001_initial... OK
```

> Django 1.7 からデータベースのマイグレートツールが標準で取り込まれたので、たとえば新しいテーブルや、既存のテーブルの項目を追加／変更したい場合、日常のルーチンワークは以下のようになります。
>
> 1. models.py を直す

2. $ python manage.py makemigrations app名 ... models.pyの変更を拾ってマイグレートファイルを作る
3. $ python manage.py migrate ... マイグレートファイルをデータベースに反映する

管理サイトの有効化

ここからは Django のいいところです。

Django には scaffold のようなアプリケーションのひな形生成、すなわち CRUD (Create、Read、Update、Delete) の自動生成はありません。
そのかわり、管理サイトというものがあって、ここからマスターメンテナンス的なデータ投入がすべてできてしまいます。

管理サイトを表示させてみましょう。

1. `python manage.py runserver` で開発用サーバを起動します。

2. `http://127.0.0.1:8000/admin/` にブラウザでアクセスします。

3. 初回の `$ python manage.py createsuperuser` で初期化したスーパーユーザー `admin/hogefuga` でログインします。

まだ、グループ、ユーザーしか見えていませんね。

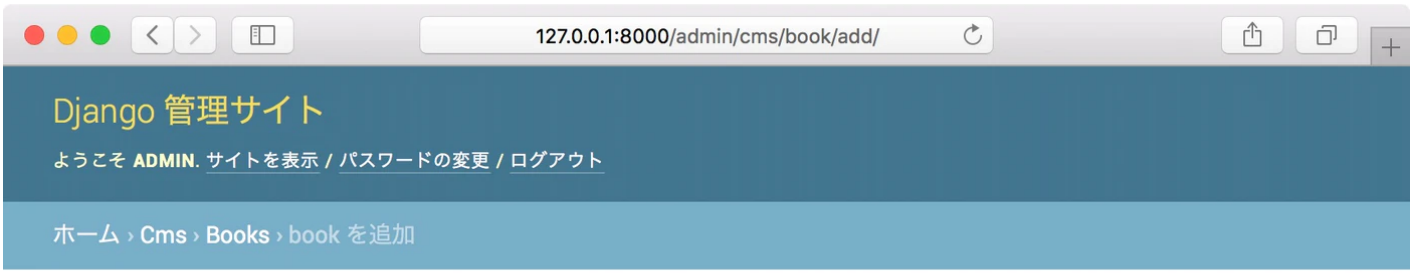
cms モデルを admin 上で編集できるようにする

cms/admin.py に、管理サイトへ表示したいモデルを追加します。

```
from django.contrib import admin
from cms.models import Book, Impression

admin.site.register(Book)
admin.site.register(Impression)
```

もう一度、 `http://127.0.0.1:8000/admin/` を見てみましょう。

ひととおり、データの追加、修正、削除を行ってみて下さい。



book を追加

書籍名:	Django入門
出版社:	GeekLab.Nagano
ページ数:	100

保存してもう一つ追加　保存して編集を続ける　保存

変更する book を選択　BOOK を追加 +

操作: --------- 実行　1個の内ひとつも選択されていません

	BOOK
☐	Django入門

1 book

管理サイトの一覧ページをカスタマイズする

管理サイトの一覧を見た時、models.py の

```
def __str__(self):
```

で設定したものが、レコード名として見えています。

もう少しレコードの項目全体が見えるよう、
`cms/admin.py` を修正してみましょう。

> classの前は、2行開けないと、波線が出て警告されます。

```
from django.contrib import admin

from cms.models import Book, Impression

# admin.site.register(Book)
# admin.site.register(Impression)
```

```
class BookAdmin(admin.ModelAdmin):
    list_display = ('id', 'name', 'publisher', 'page',)  # 一覧
    list_display_links = ('id', 'name',)  # 修正リンクでクリック

admin.site.register(Book, BookAdmin)

class ImpressionAdmin(admin.ModelAdmin):
    list_display = ('id', 'comment',)
    list_display_links = ('id', 'comment',)
    raw_id_fields = ('book',)   # 外部キーをプルダウンにしない（デ

admin.site.register(Impression, ImpressionAdmin)
```

どうでしょうか。

モデルに追加したテーブルの一覧、登録、修正、削除が簡単にできました。

CRUDをどう作るかの説明はいらないですね？

と言ってしまうと Django の解説にならないので、自分で CRUD を作る場合の説明をしましょう。

Python Django入門 (4) に続きます