電気通信大学

第1期プログラミング教室

Djangoで始める

Webプログラミング8(2of2)

日時　2017年3月5日（日）9:00~11:00

場所　国立大学法人電気通信大学

80周年記念会館リサージュ 3F フォーラム

準備

- サンプルリポジトリに移動して、

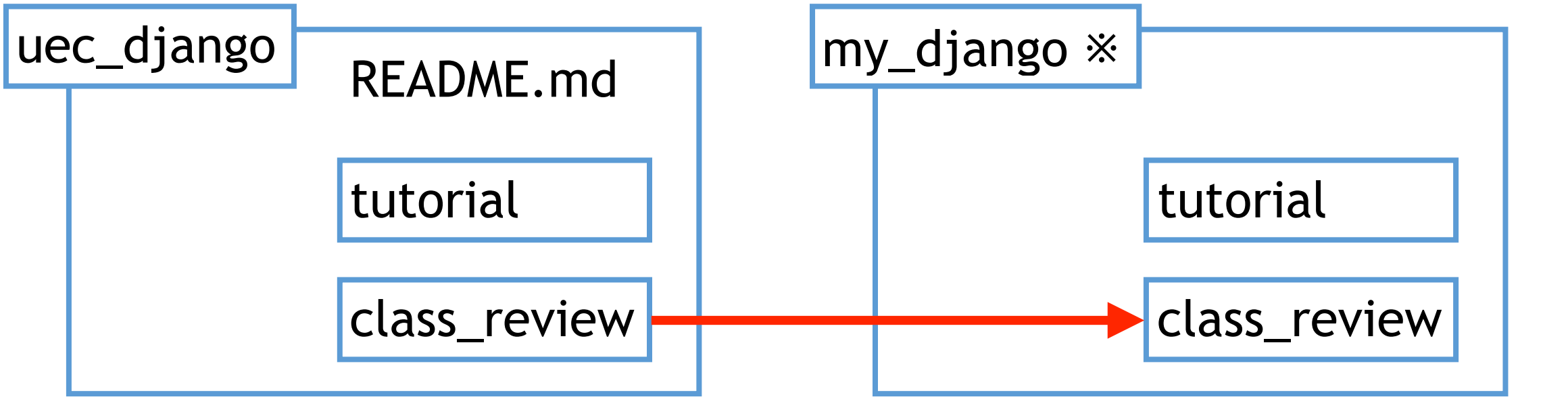
```
$ git pull
```

を実行

- サンプルリポジトリから今日の資料をコピー

ディレクトリをコピー

- サンプルリポジトリから”class_review”ディレクトリをコピー。



※自分で作っているDjangoディレクトリ

今日の流れ

1. Djangoチュートリアル8-2 (説明5分+実習20分+解説10分) 35分
   休憩 5分
2. Djangoチュートリアル8-2 (説明5分+実習20分+解説10分) 35分
   休憩 5分
3. Webアプリの仕様を考える (説明5分+実習20分+発表10分) 35分
4. 受講アンケート記入 5分

電気通信大学

第1期プログラミング教室

Djangoで始める

Webプログラミング8(2of2)

日時　2017年3月5日（日）9:00~11:00

場所　国立大学法人電気通信大学

80周年記念会館リサージュ 3F フォーラム

UEC

The Programming Class for Children and Students

Lesson 8-2:

What is Model?

投票アプリ完成までの道のり

1. It worked! & アプリ追加
2. 管理サイトを作る
3. 利用者側サイトを作る
4. 投票システムを作る

5. Formでもっと簡潔に
6. BootStrapで見栄え良く
7. テスト
8. **Model ＆ Shell** ←今ここ
9. **完成**

今日の目的

- Modelについて理解する。
- なぜ？：Webアプリで優先して作るのはModel。
  これがないとViewもTemplateも作れない。
  作りたいWebアプリを形にする為にModelを自作する力を付ける。

チュートリアル8 でやること

1.Model, DBとは

2.PythonコードとDBテーブルとの関係

3.レコード取得の仕組み

4.メソッドの種類

今日やること

前回の復習

- Webアプリのデータはデータベースに保存されている。データベースを操作するにはSQLが良く用いられる。しかしSQLを一から学ぶのは大変。
- そこでSQLの生成と発行も代わりにやってくれるDjangoのModelを使うと便利。ModelはPythonで書かれているので、我々には使いやすい。

前回の復習

- Webアプリのデータはデータベースに保存されている。データベースを操作するにはSQLが良く

ModelのおかげでWebアプリに必要なデータを簡単に管理できる

DjangoのModelを使うと便利。ModelはPythonで書かれているので、我々には使いやすい。

Part 1

Part1 実習の概要

- インタラクティブシェルでクラスを使ってみる。
- クラスの性質を復習する。

クラスとは

- 英単語の意味

  class：(共通の性質を有する)部類

- プログラミングでは

  オブジェクトの設計図のこと。

  オブジェクトがどの様な属性を持つのか、

  どの様な演算を行うのか自分で定義できる。

インスタンスとは

- 英単語の意味

  instance：実例

- プログラミングでは

  クラスを基に作ったオブジェクトのこと。

アトリビュートとメソッド

```
class Widget(object):  # 画面上で動く物の基本となるクラス

    def __init__(self, window, size, color, pos, speed=[0, 0]):
        self.window = window
        self.size = size
        self.color = color
        self.pos = pos
        self.speed = speed

    def acty(self):  # インスタンスを動かす
        self.window.move(self.id, self.speed[0], self.speed[1])

    def xturn(self):  # 横軸の方向転換
        self.speed[0] *= -1
```

アトリビュートとメソッド

```
class Widget(object):  # 画面上で動く物の基本となるクラス

    def __init__(self, window, size, color, pos, speed=[0, 0]):
        self.window = window
        self.size = size
        self.color = color
        self.pos = pos
        self.speed = speed

    def acty(self):  # インスタンスを動かす
        self.window.move(self.id, self.speed[0], self.speed[1])

    def xturn(self):  # 横軸の方向転換
        self.speed[0] *= -1
```

アトリビュート
（インスタンス変数）

アトリビュートとメソッド

```
class Widget(object):  # 画面上で動く物の基本となるクラス

    def __init__(self, window, size, color, pos, speed=[0, 0]):
        self.window = window
        self.size = size
        self.color = color
        self.pos = pos
        self.speed = speed

    def acty(self):  # インスタンスを動かす
        self.window.move(self.id, self.speed[0], self.speed[1])

    def xturn(self):  # 横軸の方向転換
        self.speed[0] *= -1
```

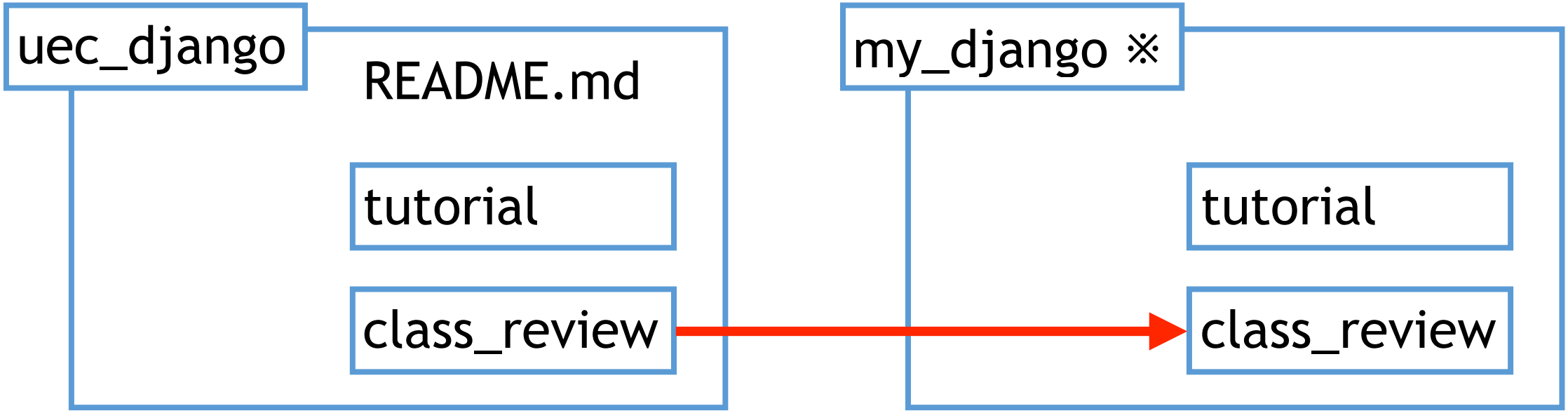
メソッド
（インスタンスメソッド）

実習⓪ サンプルリポジトリの状態を変える

```
$ git checkout check8_2_01
```

実習① ディレクトリをコピー

- サンプルリポジトリから”class_review”ディレクトリをコピー。



※自分で作っているDjangoディレクトリ

- class_reviewディレクトリの中には student.pyがある。

```python
class Student(object):
    student_number = 0  # クラス変数

    def __init__(self, name):  # インスタンス作成時に実行
        self.name = name  # インスタンス変数
        self.add()

    def myname(self):  # self.nameを出力するインスタンスメソッド
        print('私は'+self.name+'です。')

    @classmethod
    def add(cls):  # 生徒追加の際に呼び出されるクラスメソッド
        cls.student_number += 1
        print('生徒が追加されました。')

    @classmethod
    def members(cls):  # 現在の生徒数を返すクラスメソッド
        print(str(cls.student_number)+'人です。')
```

実習② ipythonの起動

```
$ ipython
```

実習③ モジュールをインポート

```
In [ ]: from student import Student
```

```
(uec) Osamu-no-MacBook-Air:0305 yokohama$ ipython
Python 3.6.0 (default, Dec 24 2016, 00:01:50)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from student import Student
```

実習④ members() メソッドを実行

```
In [ ]: Student.members()
```

```
In [2]: Student.members()
0人です。
```

実習⑤ インスタンス作成

```
In [ ]: popuko = Student('ポプ子')
```

```
In [3]: popuko = Student('ポプ子')
生徒が追加されました。
```

実習⑤ myname() メソッドを実行

```
In [ ]: popuko.myname()
```

```
In [4]: popuko.myname()
私はポプ子です。
```

実習⑥ members() メソッドを再び実行

```
In [ ]: Student.members()
```

```
In [ ]: popuko.members()
```

```
In [5]: Student.members()
1人です。

In [6]: popuko.members()
1人です。
```

実習⑦ インスタンスを追加

```
In [ ]: pipimi = Student('ピピ美')
```

```
In [7]: pipimi = Student('ピピ美')
生徒が追加されました。
```

実習⑧ myname() メソッドを再び実行

```
In [ ]: popuko.myname()
```

```
In [ ]: pipimi.myname()
```

```
In [8]: popuko.myname()
私はポプ子です。

In [9]: pipimi.myname()
私はピピ美です。
```

実習⑨ members() メソッドを三たび実行

```
In [ ]: Student.members()
```

```
In [ ]: popuko.members()
```

```
In [ ]: pipimi.members()
```

```
In [10]: Student.members()
2人です。

In [11]: popuko.members()
2人です。

In [12]: pipimi.members()
2人です。
```

やってみよう

- 実習⓪~⑨をやってみよう。
- 更に生徒を追加して、members() の結果がどうなるか見てみよう。
- Student.myname()
  を実行するとどうなるだろうか？

クラスとは

- 英単語の意味

  class：(共通の性質を有する)部類

- プログラミングでは

  オブジェクトの設計図のこと。

  オブジェクトがどの様な属性を持つのか、

  どの様な演算を行うのか自分で定義できる。

数種類ある

インスタンス変数とクラス変数

	インスタンス変数	クラス変数
値	インスタンス固有	クラス共通
例	name	student_number

popuko

name: ‘ポプ子’

student_number: 2

pipimi

name: ‘ピピ美’

student_number: 2

インスタンスメソッドとクラスメソッド

	インスタンスメソッド	クラスメソッド
付随	インスタンス	クラス
例	myname()	members()

Part1 まとめ

- クラスはオブジェクトの設計図。変数とメソッドをひとまとめに定義する。
- クラスで定義する変数とメソッドにはインスタンス固有のものとクラス共通のものがある。

5分休憩

Part 2

Part2 実習の概要

- インタラクティブシェルを使ってModelを操作してみる。
- Modelにメソッドを追加し、自分で動かしてみる。

get_published_data

- データベースのQuestionテーブルから「**既に公開された物のみ**」表示するメソッド。

実習⓪ サンプルリポジトリの状態を変える

```
$ git checkout check8_2_02
```

実習① models.py にメソッドをコピー

```
@classmethod
def get_published_data(cls):  # 新たに追加したメソッド1
    return cls.objects.filter(pub_date__lte=timezone.now())
```

実習② shell_plusを起動

```
$ python manage.py shell_plus --print-sql
```

実習③ get_published_data() を実行する

```
In [ ]: Question.get_published_data()
```

```
In [1]: Question.get_published_data()
Out[1]: SELECT "polls_question"."id", "polls_question"."question_text", "polls_
question"."pub_date" FROM "polls_question" WHERE "polls_question"."pub_date" <=
 '2017-03-04 20:14:47.164742' ORDER BY "polls_question"."question_text" ASC, "p
olls_question"."pub_date" ASC LIMIT 21

Execution time: 0.002164s [Database: default]

<QuerySet [<Question: お寿司のネタは何が好き？>, <Question: どのPCを使ってる？>
, <Question: エイサイハラマスコイ踊り流行るかな？>, <Question: 修学旅行はどこへ
行きたい？>, <Question: 好きな色は？>, <Question: 海外旅行はどこへ行きたい？>]>
```

実習④ 現時点では公開されない質問を追加

実習⑤ get_published_data() を再び実行する

```
In [ ]: Question.get_published_data()
```

```
In [5]: Question.get_published_data()
Out[5]: SELECT "polls_question"."id", "polls_question"."question_text", "polls_
question"."pub_date" FROM "polls_question" WHERE "polls_question"."pub_date" <=
 '2017-03-04 20:25:28.764996' ORDER BY "polls_question"."question_text" ASC, "p
olls_question"."pub_date" ASC LIMIT 21

Execution time: 0.000195s [Database: default]

<QuerySet [<Question: お寿司のネタは何が好き？>, <Question: どのPCを使ってる？>
, <Question: エイサイハラマスコイ踊り流行るかな？>, <Question: 修学旅行はどこへ
行きたい？>, <Question: 好きな色は？>, <Question: 海外旅行はどこへ行きたい？>]>
```

実習⑥ .objects.all() を実行する

```
In [ ]: Question.objects.all()
```

```
In [6]: Question.objects.all()
Out[6]: SELECT "polls_question"."id", "polls_question"."question_text", "polls_
question"."pub_date" FROM "polls_question" ORDER BY "polls_question"."question_
text" ASC, "polls_question"."pub_date" ASC LIMIT 21

Execution time: 0.000283s [Database: default]

<QuerySet [<Question: お寿司のネタは何が好き？>, <Question: さてはアンチだなオ
メー>, <Question: どのPCを使ってる？>, <Question: エイサイハラマスコイ踊り流行
るかな？>, <Question: 修学旅行はどこへ行きたい？>, <Question: 好きな色は？>, <Q
uestion: 海外旅行はどこへ行きたい？>]>
```

実習⑦ 取り出した結果にfilterをかける

```
In [ ]: Question.get_published_data().filter(条件)
```

```
In [7]: Question.get_published_data().filter(pk__gte=5)
Out[7]: SELECT "polls_question"."id", "polls_question"."question_text", "polls_
question"."pub_date" FROM "polls_question" WHERE ("polls_question"."pub_date" <
= '2017-03-04 20:36:22.626421' AND "polls_question"."id" >= 5) ORDER BY "polls_
question"."question_text" ASC, "polls_question"."pub_date" ASC LIMIT 21

Execution time: 0.000945s [Database: default]

<QuerySet [<Question: エイサイハラマスコイ踊り流行るかな？>, <Question: 好きな
色は？>]>
```

filterとexcludeの書き方

In[ ]: Question.objects.filter(Field名=値)

In[ ]: Question.objects.exclude(Field名=値)

条件の書き方

フィールド名__条件=値

条件の書き方

条件	Pythonの条件式	例
__gt	>	pk__gt=3
__gte	>=	pk__gte=3
__lt	<	pk__lt=3
__lte	<=	pk__lte=3
__in	1 in [1, 2, 3]	pk__in=[1, 2, 3]
__exact	‘abc’ == ‘abc’	question_text__exact=‘UEC’
__contains	‘a’ in ‘abc’	question_text__contains=‘UEC’

条件の書き方

条件	意味
__iexact	大小文字の区別をしないexact
__startswith	この文字列で始まる文字列
__endswith	この文字列で終わる文字列
__regex	正規表現による検索
__iregex	正規表現による検索(大小文字の区別をしない)

やってみよう

- 実習⓪~⑦をやってみよう。
- ↑を済ませた人は、サンプルリポジトリを”check8_2_03”タグに合わせよう。QuerySetの拡張とそれを用いたメソッドの追加を行い、使ってみよう。

リレーショナルデータベースの構造

カラム(列, 属性)

学生講師

講師番号	氏名	学年	学科
2	山根茂之	3	J
3	風間健流	3	I
4	宮澤修	3	I
5	張翌坤	2	J
6	柳裕太	2	J
7	佐藤海斗	2	M

レコード

(行, タプル, row)

表全体をテーブル(リレーションとも)と呼ぶ

テーブル
「学生講師」

学生講師

講師番号	氏名	学年	学科
2	山根茂之	3	J
3	風間健流	3	I
4	宮澤修	3	I
5	張翌坤	2	J
6	柳裕太	2	J
7	佐藤海斗	2	M

テーブル
「使ってるPC」

使ってるPC

講師番号	氏名	PC
2	山根茂之	Mac
3	風間健流	Mac
4	宮澤修	Mac
5	張翌坤	Windows
6	柳裕太	Mac
7	佐藤海斗	Windows

Modelとは

- データベース(DB)の構造と振る舞いを定義する。
- Pythonを使ってこれらの定義が出来る。

PythonコードとDBテーブルとの関係

- Modelはクラスであり、全体でDBのテーブルを表す。
- models.~Field で定義したインスタンス変数がテーブルのカラムに相当する。
- modelクラスのインスタンスがテーブルのレコードに対応する。

```python
class Question(models.Model):
    class Meta:⋯

    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
```

polls_question

id	question_text	pub_date
1	好きな寿司ネタは？	2017/01/01
2	使ってるPCは？	2017/01/15
3	希望の修学旅行先は？	2017/02/15
4	希望の海外旅行先は？	2017/03/01

メソッドの使い分け

	インスタンスメソッド	クラスメソッド
付随	インスタンス	クラス
対象	レコード	テーブル
例	was_published_recently	get_published_data

Part2 まとめ

- Modelはクラス全体がDBのテーブルと、インスタンスがレコードと対応する。
- テーブル全体に対して行う処理はクラスメソッドで、レコードに対して行う処理はインスタンスメソッドで定義する。

5分休憩

Part 3

5月から作るWebアプリの仕様を考えよう

1. 誰が・誰に・何を・何の為に 提供するのか
2. Webアプリは何をデータとして保存するか
3. 2.を基にDBの図を書いてみる
4. データに対してどの様な操作を行うか

参考資料

- Python Django チュートリアル(8)

  http://qiita.com/maisuto/items/7aec76e6f6fb906bfffa
- 電気通信大学 2016年度後期開講講座 「ソフトウェア工学」