**Тема занятия № 18: Модуль 10. Контроллеры-классы**

1. Введение в контроллеры-классы

Контроллер-класс, в отличие от контроллера-функции, может самостоятельно выполнять некоторые утилитарные действия (выборку из модели записи по полученному ключу, вывод страницы и др.).

Основную часть функциональности контроллеры-классы получают из примесей (классов, предназначенных лишь для расширения функциональности других классов), от которых наследуют. Мы рассмотрим как примеси, так и полноценные классы.

Контроллер-класс записывается в маршруте не в виде ссылки, как контроллер-функция, а в виде результата, возвращенного методом as view (), который поддерживается всеми контроллерами-классами. Вот пример указания в маршруте контроллера-класса createview:

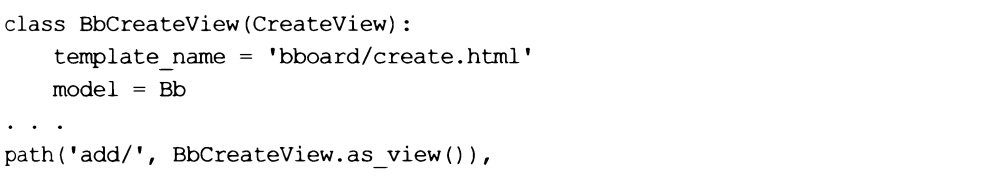


В вызове метода as viewo можно задать параметры контроллера-класса. Пример.

Указания модели и пути к шаблону (параметры model и template name соответственно):



Задать параметры контроллера-класса можно и по-другому: создав производный от него класс и указав параметры в его атрибутах:

****

Второй подход позволяет более радикально изменить поведение контроллера-класса, переопределив его методы, поэтому применяется чаще.

1. Базовые контроллеры-классы

Самые простые и низкоуровневые контроллеры-классы, называемые базовыми.

Объявлены В модуле django.views.generic.base.

**Контроллер V/eiv:**

Диспетчеризация по HTTP-методу контроллер-класс view определяет HTTP-метод, посредством которого был выполнен запрос, и исполняет код, соответствующий этому методу.

Класс поддерживает атрибут http method names, хранящий список имен допустимых HTTP-методов. По умолчанию он хранит список [’get’, ’post’, ’put', 'Patch', 'delete', 'head', 'options', 'trace'], включающий все методы, поддерживаемые протоколом HTTP.

Класс также содержит четыре метода (помимо уже знакомого нам as view ()), которые переопределяются в подклассах:

□ setup (self, request, \*args, \*\*kwargs) (начиная c Django 2.2)— выполняется самым первым и инициализирует объект контроллера.

Здесь и далее в параметре request передается объект запроса (в виде экземпляра класса httprequest), в параметре kwargs — словарь со значениями именованных URL-параметров.

Параметр args остался в ’’наследство” от первых версий Django и служит для передачи списка со значениями неименованных URL-параметров. В Django 2.0 и более поздних версиях неименованные URL-параметры не поддерживаются, поэтому данный параметр не используется.

В изначальной реализации создает в контроллере следующие атрибуты:

• request — Запрос, представленный экземпляром класса Request;

• kwargs — словарь со значениями URL-параметров.

Переопределив этот метод, можно сохранить в объекте контроллера какие-либо дополнительные данные;

□ dispatch (self, request, \*args, \*\*kwargs)—обрабатывает полученный в параметре request запрос и возвращает ответ, представленный экземпляром класса Httpresponse или его подкласса. Выполняется после метода setup ().

В изначальной реализации извлекает обозначение HTTP-метода, вызывает одноименный ему метод класса: get () — если запрос был выполнен НТТР-методом GET, post () — если запрос выполнялся методом POST, и т. П. — передавая ему все полученные параметры. Вызываемые методы должны быть объявлены в формате:

<имя метода класса> (self, request, \*args, \*\*kwargs)

Если метод класса, одноименный с HTTP-методом, отсутствует, ничего не делает. Единственное исключение — HTTP-метод HEAD: при отсутствии метода Head () вызывается метод get ();

□ http\_method\_not\_allowed (self, request, \*args, \*\*kwargs) — вызывается, если запрос был выполнен с применением неподдерживаемого НТТР-метода.

В изначальной реализации возвращает ответ типа httpresponsenotaiiowed со списком допустимых методов;

□ options (self, request, \*args, \*\*kwargs) —обрабатывает Запрос, выполненный HTTP-методом OPTIONS.

В изначальной реализации возвращает ответ с заголовком Allow, в котором записаны все поддерживаемые НТТР-методы.

Класс view используется крайне редко— обычно применяются производные от него контроллеры-классы, выполняющие более сложные действия.

**Примесь contextmixim**

Создание контекста шаблона класс-примесь contextmixin добавляет контроллеру-классу средства для формирования контекста шаблона:

□ extra context — атрибут, задающий содержимое контекста шаблона. Его значение должно представлять собой словарь, элементы которого будут добавлены в контекст;

□ get context data (self, \*\*kwargs) — метод, должен создавать и возвращать контекст шаблона. С параметром kwargs передается словарь, элементы которого должны быть добавлены в контекст шаблона.

В изначальной реализации создает пустой контекст шаблона, добавляет в него элемент view, хранящий ссылку на текущий экземпляр контроллера-класса, элементы ИЗ словарей kwargs И extra context.

**Примесь templateresponsemixin:**

Рендеринг шаблона класс-примесь templateresponsemixin добавляет наследующему классу средства для рендеринга шаблона.

На заметку!

Эта примесь и все наследующие от нее классы формируют ответ в виде экземпляра класса templateresponse, описанного В.

Поддерживаются следующие атрибуты и методы:

□ template name — атрибут, задающий путь к шаблону в виде строки;

□ get tempiate names (self) — метод, должен возвращать список путей к шаблонам, заданных в виде строк.

В изначальной реализации возвращает список из одного элемента— пути к шаблону, извлеченного ИЗ атрибута template name;

□ response class — атрибут, задает ссылку на класс, представляющий ответ (по умолчанию: templateresponse);

□ content type — атрибут, задающий MIME-тип ответа и его кодировку. По умолчанию — None (используется MIME-тип и кодировка по умолчанию);

□ render\_to\_response(self, context, \*\*response\_kwargs)—возвращает экземпляр класса, представляющего ответ. В параметре context передается контекст шаблона в виде словаря, а в параметре response kwargs — словарь, элементы которого будут переданы конструктору класса ответа в качестве дополнительных параметров.

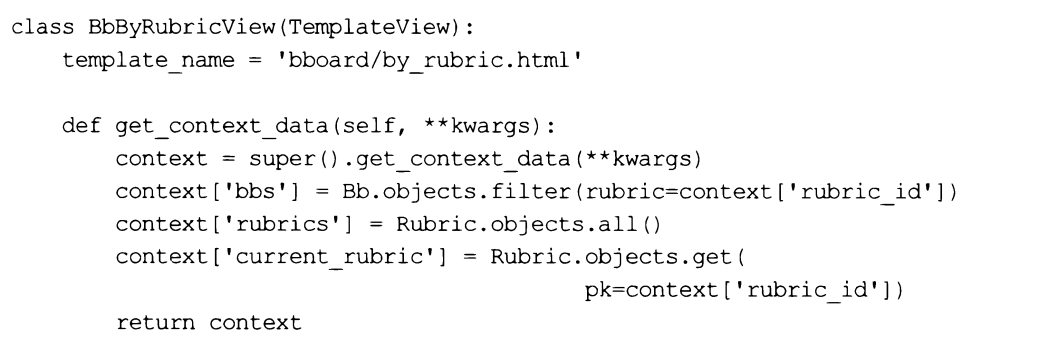
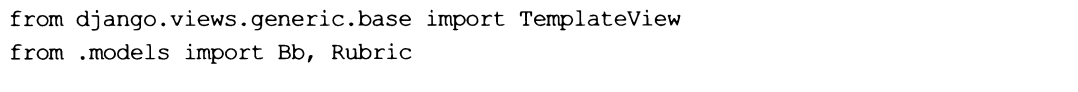
В изначальной реализации создает и возвращает экземпляр класса, указанного в атрибуте response\_class.

**Контроллер templateview**

Все вместе контроллер-класс Templateview наследует классы View, contextmixin и Templateresponsemixin. Он автоматически выполняет рендеринг шаблона и отправку ответа при получении запроса по методу GET.

В формируемый контекст шаблона добавляются все URL-параметры, которые присутствуют в маршруте, под своими изначальными именами.

Класс Templateview уже можно применять в практической работе. Приведен КОД ПРОИЗВОДНОГО ОТ Него контроллера-класса bbbyrubricview, который выводит страницу с объявлениями из выбранной рубрики.



Поскольку класс Templateview добавляет в контекст шаблона значения всех полученных им URL-параметров, мы можем извлечь ключ рубрики, обратившись к элементу rubric id контекста в переопределенном методе get context data ().

В классах, рассматриваемых далее, такой ’’номер” уже не пройдет, поскольку они не наследуют ОТ Templateview.