**Тема занятия № 15: Модуль 9. Контроллеры-функции**

1. Введение в контроллеры-функции

Контроллер запускается при получении клиентского запроса по определенному интернет-адресу и в ответ генерирует соответствующую веб-страницу. Контроллеры реализуют основную часть логики сайта.Контроллеры-функции реализуются в виде обычных функций Python. Такая функция обязана принимать следующие параметры (указаны в порядке очередности их объявления):

□ экземпляр класса httprequest (объявлен В модуле django.http), хранящий сведения о полученном клиентском запросе. Традиционно этот параметр носит имя Request;

□ набор именованных параметров, имена которых совпадают с именами URL-параметров, объявленных в связанном с контроллером маршруте.

Контроллер-функция должен возвращать в качестве результата экземпляр класса Httpresponse, Также объявленного В модуле django.http, или какого-либо из его подклассов. Этот экземпляр класса представляет ответ, отсылаемый клиенту (вебстраница, обычный текстовый документ, файл, данные в формате JSON, перенаправление или сообщение об ошибке).

При создании нового приложения утилита manage.py записывает в пакет приложения модуль views.py, в котором, как предполагают разработчики фреймворка, и будет находиться код контроллеров. Однако ничто не мешает сохранить код контроллеров в других модулях с произвольными именами.

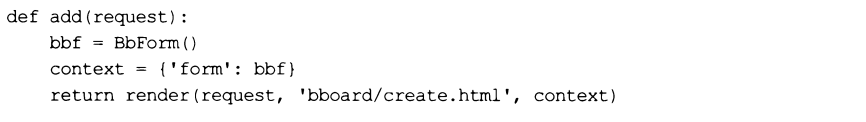
1. Как пишутся контроллеры-функции

Принципы написания контроллеров-функций достаточно просты. Однако здесь имеется один подводный камень, обусловленный самой природой Django.

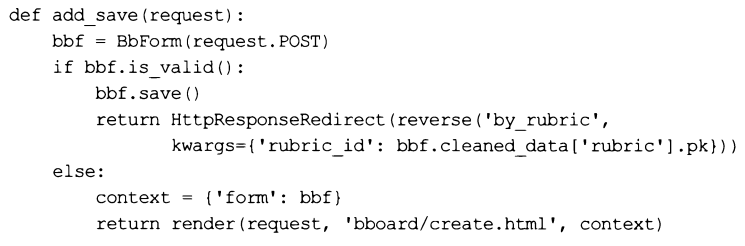
Контроллеры, выполняющие одну задачу если контроллер-функция должен выполнять всего одну задачу — скажем, вывод веб-страницы, то все очень просто. Для примера рассмотрим код — он объявляет контроллер-функцию by rubric (), которая выводит страницу с объявлениями, относящимися к выбранной посетителем рубрике.

Если нужно выводить на экран страницу для добавления объявления и потом сохранять это объявление в базе данных, понадобятся два контроллера такого рода.

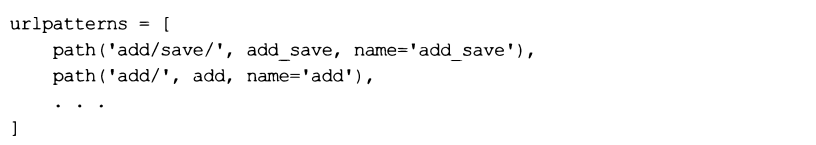
Напишем контроллер-функцию, который создаст форму и выведет на экран страницу добавления объявления.

****

Далее напишем контроллер-функцию add save (), который сохранит введенное объявление в базе.

****

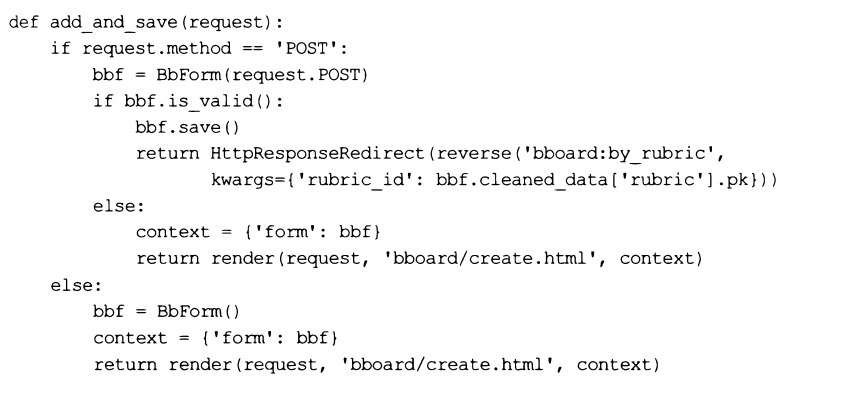
Объявим маршруты, ведущие на эти контроллеры:



После этого останется открыть шаблон bboard\create.html, и добавить в имеющийся в его коде тег <form> атрибут action с интернет-адресом, указывающим на контроллер add save ():

****

Контроллеры, выполняющие несколько задач на практике для обработки данных, вводимых посетителями в формы, обычно применяют не два контроллера, а один. Он и выведет страницу с формой, и сохранит занесенные в нее данные. Код такого контроллера-функции add\_and\_save() приведен.

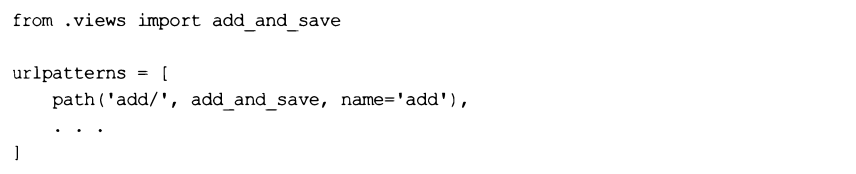


Сначала проверяется, с применением какого HTTP-метода был отправлен запрос.

Если это метод GET, значит, посетитель хочет зайти на страницу добавления нового объявления, и эту страницу, с пустой формой, нужно вывести на экран.

Если же запрос был выполнен методом POST, значит, осуществляется отсылка введенных в форму данных, и их надо сохранить в базе. Занесенные в форму данные проверяются на корректность и, если проверка прошла успешно, сохраняются в базе, в противном случае страница с формой выводится на экран повторно. После успешного сохранения осуществляется перенаправление на страницу со списком объявлений из категории, заданной при вводе нового объявления.

Поскольку мы обходимся только одним контроллером, нам понадобится лишь один маршрут:

****

В коде шаблона bboard\create.html — в теге <form>, создающем форму, — уже не нужно указывать интернет-адрес для отправки занесенных в форму данных:

****

В этом случае данные будут отправлены по тому же интернет-адресу, с которого была загружена текущая страница.

1. Формирование ответа

Основная задача контроллера— сформировать ответ, который будет отправлен посетителю. Обычно такой ответ содержит веб-страницу.

**Низкоуровневые средства** для формирования ответа

Формированием ответа на самом низком уровне занимается класс httpresponse из модуля django.http. Его конструктор вызывается в следующем формате:

Httpresponse([<содержимое>] [,][content\_type=None][,][status=200][,][reason=None])

Содержимое должно быть указано в виде строки или последовательности строк.

Необязательный параметр content type задает MIME-тип ответа и его кодировку.

Если он отсутствует, ответ получит MIME-тип text/html и кодировку из параметра Default charset, указанного в настройках проекта.

Параметр status указывает целочисленный код статуса ответа (по умолчанию — 200, т.е. Файл успешно отправлен), а параметр reason— строковый статус (по умолчанию — "ок”).

Класс ответа поддерживает атрибуты:

□ content — содержимое ответа в виде объекта типа bytes;

□ charset — обозначение кодировки;

□ status code — целочисленный код статуса;

□ reason phrase — строковое обозначение статуса;

□ streaming — если True, ЭТО ПОТОКОВЫЙ Ответ, если False — обычный. Будучи вызванным у экземпляра класса httpresponse, метод всегда возвращает False;

□ closed — True, если ответ закрыт, и False, если еще нет.

Также поддерживаются следующие методы:

□ write (<строка>) —добавляет строку в ответ;

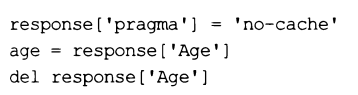
О writelines (^последовательность строк>) — добавляет к ответу строки из указанной последовательности. Разделители строк при этом не вставляются;

□ flush () — принудительно переносит содержимое буфера записи в ответ;

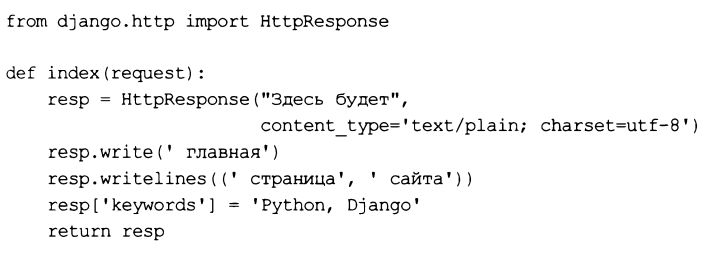
□ has\_header (<заголовок>) — возвращает True, если указанный заголовок существует в ответе, и False — в противном случае;

□ setdefault (<заголовок>, <значение>) — создает в ответе указанный заголовок с указанным значением, если таковой там отсутствует.

Класс httpresponse поддерживает функциональность словарей, которой мы можем пользоваться, чтобы указывать и получать значения заголовков:

****

Приведен код простого контроллера, использующего низкоуровневые средства для создания ответа и выводящего строку: здесь будет главная страница сайта. Кроме того, он задает в ответе заголовок keywords со значением "Python, Django".

****

**Формирование ответа на основе шаблона**

Применять низкоуровневые средства для создания ответа на практике приходится крайне редко. Гораздо чаще мы имеем дело с высокоуровневыми средствами — шаблонами.

Для загрузки нужного шаблона Django предоставляет две функции, объявленные в модуле dj ango. Template. Loader:

□ get\_tempiate (<путь к шаблону>) — загружает шаблон, расположенный по указанному пути, и возвращает представляющий его экземпляр класса Template из модуля dj ango. Template.

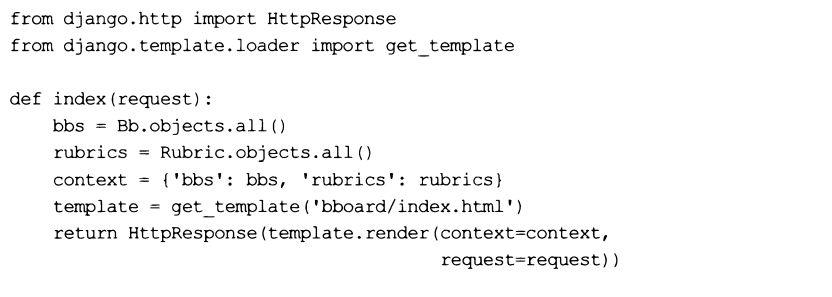
□ select\_template (Последовательность путей шаблонов>} —перебирает указанную последовательность путей шаблонов, пытается загрузить шаблон, расположенный по очередному пути, и возвращает первый шаблон, который удалось загрузить, в виде экземпляра класса Template.

Если шаблон загрузить не получилось, то будет возбуждено исключение Tempiatedoesnotexist. Если в коде шаблона встретилась ошибка, возбуждается исключение tempiatesyntaxerror. Оба класса исключений объявлены в модуле Dj ango.template.

Пути шаблонов указываются относительно папки, в которой хранятся шаблоны.

Дня получения обычной веб-страницы нужно выполнить рендеринг шаблона, вызвав один из следующих методов:

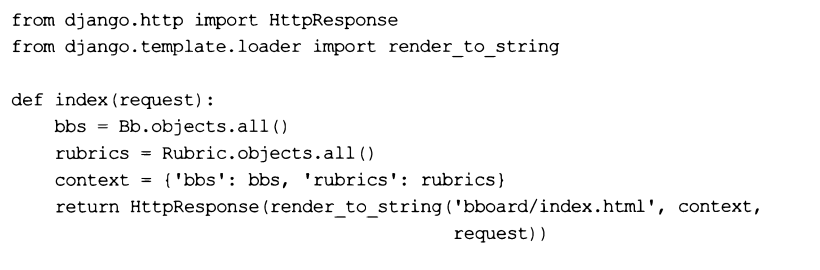
□ render ( [context= Сконтекст шаблона>] [, ] [request=<3arzpoc>] ) — метод класса Template. Выполняет рендеринг текущего шаблона на основе заданного контекста (задается в виде словаря со значениями, которые должны быть доступны в шаблоне). Если указан запрос (экземпляр класса Request), то он также будет добавлен в контекст шаблона. Возвращает строку с HTML-кодом сформированной страницы. Пример использования этого метода.

****

Параметры у метода render () можно указать не только как именованные, но и как позиционные:

Return httpresponse(template.render(context, request))

П render\_to\_string (<путь к шаблону>[, context=<kohrekct шаблона>] [, request=<запрос>])— функция из модуля django.template.loader. Загружает шаблон с указанным путем и выполняет его рендеринг с применением заданных контекста шаблона и запроса. Возвращает строку с HTML-кодом сформированной страницы. Пример использования функции render to string о приведен.

****