**Тема занятия № 16: Модуль 9. Контроллеры-функции**

1. Получение сведений о запросе

Полученный от посетителя запрос представляется экземпляром класса httprequest.

Он хранит разнообразные сведения о запросе, которые могут оказаться очень полезными.

Прежде всего, это ряд атрибутов, хранящих различные величины:

□ get— словарь со всеми GET-параметрами, полученными в составе запроса.

Ключи элементов этого словаря совпадают с именами GET-параметров, а значения элементов — суть значения этих параметров;

□ post— словарь со всеми POST-параметрами, полученными в составе запроса.

Ключи элементов этого словаря совпадают с именами POST-параметров, а значения элементов — суть значения этих параметров;

□ files — словарь со всеми выгруженными файлами. Ключи элементов этого словаря совпадают с именами POST-параметров, посредством которых передается содержимое файлов, а значения элементов— сами файлы, представленные экземплярами класса upioadedfiie;

□ method — обозначение HTTP-метода в виде строки, набранной прописными буквами ("GET”, "POST” и т.д.);

□ scheme —обозначение протокола ("http" или "https");

□ path И path\_infо — путь;

□ encoding — обозначение кодировки, В которой был получен запрос. Если None, запрос закодирован в кодировке по умолчанию (она задается в параметре Default charset настроек проекта);

□ content type — обозначение MIME-типа полученного запроса, извлеченное из HTTP-Заголовка Content-Type;

□ content params — словарь, содержащий дополнительные параметры MIME-типа полученного запроса, которые извлекаются из HTTP-заголовка Content-Type.

Ключи элементов соответствуют самим параметрам, а значения элементов — значениям параметров;

□ мета— словарь, содержащий дополнительные параметры в виде следующих элементов:

• content length — длина тела запроса в символах, заданная в виде строки;

• CONTENT-TYPE— MIME-ТИП тела запроса (может быть "application/x-wwwform-urlencoded", "multipart/form-data" ИЛИ "text/plain");

• HTTP—accept — строка с перечнем поддерживаемых веб-обозревателем mimeтипов данных, разделенных запятыми;

• HTTP—ACCEPT—encodings — строка с перечнем поддерживаемых веб-обозревателей кодировок, разделенных запятыми;

• http\_accept\_languages — строка с перечнем поддерживаемых веб-обозревателем языков, разделенных запятыми;

• http host — доменное имя (или IP-адрес) и номер TCP-порта, если он отличается от используемого по умолчанию, веб-сервера, с которого была загружена страница;

• http referer— интернет-адрес страницы, с которой был выполнен переход на текущую страницу (может отсутствовать, если это первая страница, открытая в веб-обозревателе);

• http user agent — строка с обозначением веб-обозревателя, запрашивающего страницу;

• query string — строка с необработанными GET-параметрами;

• remote addr — IP-адрес клиентского компьютера, запрашивающего страницу;

• remote host— доменное имя клиентского компьютера, запрашивающего страницу. Если доменное имя не удается определить, элемент хранит пустую строку;

• remote user— имя пользователя, выполнившего вход на веб-сервер. Если вход на веб-сервер не был выполнен или если используется другой способ аутентификации, этот элемент будет отсутствовать;

• request—METHOD — обозначение HTTP-метода ("get”, "post" и т.д.);

• SERVER NAME — доменное имя сервера;

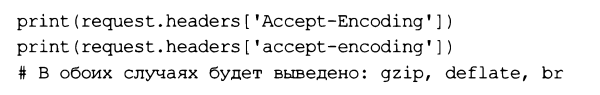
• SERVER PORT — номер TCP-порта, через который работает веб-сервер, в виде строки.

В словаре, хранящемся в атрибуте мета, могут присутствовать и другие элементы. Они формируются на основе любых нестандартных заголовков, которые имеются в клиентском запросе. Имя такого элемента создается на основе имени соответствующего заголовка, преобразованного к верхнему регистру, с подчеркиваниями вместо дефисов и с префиксом нттр\_, добавленным в начало. Так, на основе заголовка Upgrade-Insecure-Requests в словаре мета будет создан элемент С КЛЮЧОМ HTTP—UPGRADE—INSECURE—REQUESTS;

□ body — ’’сырое” содержимое запроса в виде объекта типа bytes;

□ resoiver match — экземпляр класса resoivermatch описывающий сработавший маршрут;

□ headers (начиная с Django 2.2) — объект, хранящий все заголовки запроса. Обладает функциональностью словаря, ключи элементов которого совпадают с именами заголовков, а значениями элементов являются значения, переданные в этих заголовках. Имена заголовков можно указывать в любом регистре. Пример:

****

Начиная с Django 3.0, в именах заголовков вместо дефисов можно записывать подчеркивания:

****

Методы, поддерживаемые классом httprequest:

□ get host () — возвращает строку с комбинацией IP-адреса (или доменного имени, если его удастся определить) и номера TCP-порта, через который работает Веб-сервер;

□ get port () — возвращает строку с номером TCP-порта, через который работает Веб-сервер;

□ get\_fuii\_path () — возвращает полный путь к текущей странице;

□ buiid\_absoiute\_uri (<путь>) — строит полный интернет-адрес на основе доменного имени (IP-адреса) сервера и указанного пути:

Print(request.build\_absolute\_uri(’/test/url/’))

# Будет выведено: http://localhost:8000/test/uri/

□ is secureo — возвращает True, если обращение выполнялось по протоколу HTTPS, и False —если по протоколу HTTP;

□ is ajax () — возвращает True, если ЭТО AJAX-запрос, И False —если обычный.

AJAX-запросы выявляются фреймворком по наличию в запросе заголовка X\_Requested\_With СО Значением "xmlhttprequest".

1. Перенаправление

Очень часто приходится выполнять перенаправление клиента по-другому интернет-адресу. Так, после добавления объявления следует выполнить перенаправление на страницу списка объявлений, относящихся к рубрике, к которой принадлежит добавленное объявление.

Перенаправление такого рода называется временным. Оно просто вызывает переход на страницу с заданным интернет-адресом.

Для выполнения временного перенаправления нужно создать экземпляр класса Httpresponseredirect, являющегося подклассом класса httpresponse и объявленного в модуле django.http. Вот формат конструктора этого класса:



Целевой интернет-адрес указывается в виде строки.

Созданный таким образом экземпляр класса httpresponseredirect следует вернуть из контроллера-функции в качестве результата.

Пример выполнения временного перенаправления:



Постоянное перенаправление применяется в тех случаях, когда сайт ’’переезжает” на новый интернет-адрес, и при заходе по старому адресу посетителя нужно отправить по новому местоположению. Помимо перехода по новому адресу, вебобозреватель заменяет новым старый интернет-адрес везде, где он присутствует:

В списке истории, в избранном и др.

Постоянное перенаправление реализуется другим классом из модуля django.http —Httpresponsepermanentredirect, также ПРОИЗВОДНЫМ ОТ httpresponse:

Httpresponsepermanentredi rect (<целевой интернет-адрес>[, status=301] [,Reason=None])

Пример:

****

1. Обратное разрешение интернет-адресов

Механизм обратного разрешения формирует интернет-адреса на основе объявленных в списках именованных маршрутов.

Для формирования адресов в коде контроллеров применяется функция reverse () из модуля django.urls:

Reverse(<имя маршрута>[, args=None][, kwargs=None][, urlconf=None])

Имя маршрута указывается в виде строки. Если проект содержит несколько приложений с заданными пространствами имен, то первым параметром функции указывается Строка вида <пространство имен>: <имя маршрута>.

Если указан параметризованный маршрут, то следует задать значения URL-napaметров — одним из двух способов:

□ в параметре args — в виде последовательности. Первый элемент такой последовательности задаст значение первого по счету URL-параметра в маршруте, второй элемент задаст значение второго URL-параметра и т.д.;

□ в параметре kwargs — в виде словаря. Ключи его элементов соответствуют именам URL-параметров, а значения элементов зададут значения параметров.

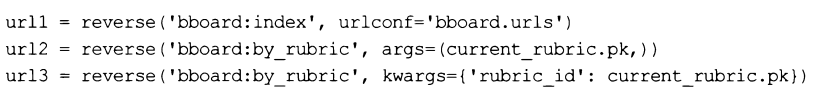
Допускается указывать только один из этих параметров: args или kwargs. Задание обоих параметров вызовет ошибку.

Параметр uriconf задает путь к модулю со списком маршрутов, который будет использоваться для обратного разрешения. Если он не указан, задействуется модуль со списком маршрутов уровня проекта, заданный в его настройках (параметр Root urlconf.).

Если в параметре uriconf указан модуль с маршрутами уровня приложения, то записывать пространство имен в первом параметре функции reverse () не нужно.

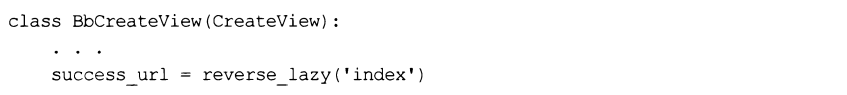
Функция возвращает строку с интернет-адресом, полученным в результате обратного разрешения.

Примеры:

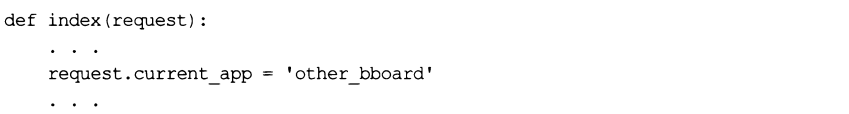
****

Функция reverse () имеет серьезный недостаток — она работает лишь после того, как список маршрутов был загружен и обработан. Из-за этого ее можно использовать только в контроллерах.

Если же нужно указать интернет-адрес где-либо еще, например в атрибуте контроллера-класса, то следует применить функцию Reverse iazy () из того же модуля django.urls. Ее формат вызова точно такой же, как и у функции reverse (). Пример:

****

Принудительно указать имя или псевдоним приложения, которое будет использоваться при обратном разрешении интернет-адресов в шаблонах, можно, занеся его в атрибут current app объекта запроса:

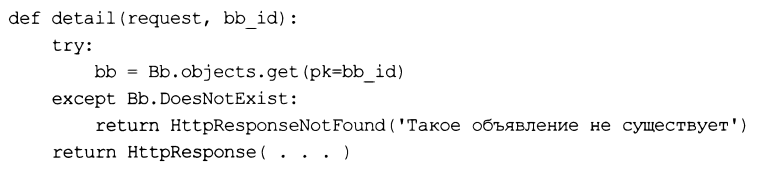
****

Для обратного разрешения интернет-адресов в шаблонах применяется тег шаблонизатора url.

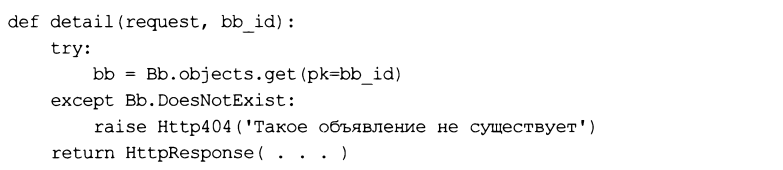
1. Выдача сообщений об ошибках и обработка особых ситуаций

Для выдачи сообщений об ошибках и уведомления клиентов об особых ситуациях (например, если страница не изменилась с момента предыдущего запроса, и ее можно загрузить из локального кэша) Django предоставляет ряд классов. Все они ЯВЛЯЮТСЯ производными ОТ класса httpresponse И объявлены В модуле django.http.

□ httpresponsenotfound([<содержимое>] [,][content\_type=None][,][status=404][,][reason=None]) — запрашиваемая страница не существует (код статуса 404):

****

Также МОЖНО возбудить исключение Http404 ИЗ модуля django.http:

****

□ httpresponsebadrequest([<содержимое>] [, ] [content\_type=None][,] [status=400] [,][reason=None]) — клиентский запрос некорректно сформирован (код статуса 400);

□ httpresponseforbidden([<содержимое>] [,][content\_type=None][,][status=403][,][reason=None]) — доступ к запрошенной странице запрещен (код статуса 403).

Также МОЖНО возбудить исключение permissiondenied ИЗ модуля django. Core, Exceptions;

О httpresponsenotallowed («^последовательность обозначений разрешенных методов>[,][content\_type=None] [, ] [status=405] [, ] [reason=None] ) — клиентский запрос был выполнен с применением недопустимого HTTP-метода (код статуса 405).

Первым параметром конструктора указывается последовательность методов, которые допустимы для запрошенной страницы. Пример:

Return httpresponsenotallowed([’GET’])

□ httpresponsegone([<содержимое>] [, ] [content\_type=None] [,] [status=410] [,] [reason=None]) — запрошенная страница удалена насовсем (код статуса 410);

О httpresponseservererror([<содержимое>] [,][content\_type=None][,][status=500][, ] [reason=None]) — ошибка в программном коде сайта (код статуса 500);

О httpresponsenotmodified([<содержимое>] [,][content\_type=None][,][status=304][, ] [reason=None]) — запрашиваемая страница не изменилась с момента последнего запроса и может быть извлечена веб-обозревателем из локального кэша (код статуса 304).