**Тема занятия № 13: Контроллеры. Классы и функции**

1. Классы, выводящие одну запись

Контроллеры-классы ИЗ модуля django.views.generic.detail— более высокоуровневые, чем базовые. Они носят название обобщенных.

Поскольку выполняют типовые задачи и могут быть использованы в различных ситуациях.

Примесь singleobjectmixin: поиск записи класс-примесь singieobjectmixin, наследующий от contextmixin, выполняет сразу три действия:

□ извлекает из полученного интернет-адреса ключ или слаг записи, предназначенной к выводу на странице;

□ ищет запись в заданной модели по полученному ранее ключу или слагу;

□ помещает найденную запись в контекст шаблона.

Примесь поддерживает довольно много атрибутов и методов:

□ model — атрибут, задает модель;

□ queryset— атрибут, указывает либо диспетчер записей (Manager), либо набор записей (Queryset), в котором будет выполняться поиск записи;

□ get queryset (self) — метод, должен возвращать набор записей (queryset), в котором будет выполняться поиск записи.

В изначальной реализации возвращает значение атрибута queryset, если оно задано, или набор записей из модели, заданной в атрибуте model;

□ pk url kwarg — атрибут, задает имя URL-параметра, через который контроллер-класс получит ключ записи (по умолчанию: ”рк");

□ slug\_fieid— атрибут, задает имя поля модели, в котором хранится слаг (по умолчанию: "slug”);

□ get\_siug\_fieid(self) — метод, должен возвращать строку с именем поля модели, в котором хранится слаг. В реализации по умолчанию возвращает значение ИЗ атрибута slug\_field;

□ siug url kwarg — атрибут, задает имя URL-параметра, через который контроллер-класс получит слаг (по умолчанию: "slug");

□ query pk and slug — атрибут, указывает, что случится, если ключ записи не был найден в интернет-адресе. Если значение атрибута равно False, то в этом случае будет возбуждено исключение attributeerror, если True— будет выполнена попытка найти запись по слагу (если он присутствует — иначе также будет возбуждено исключение attributeerror);

□ context object name— атрибут, задает имя переменной контекста шаблона, в которой будет сохранена найденная запись;

□ get\_context\_object\_name(self, obj) — метод, должен возвращать имя переменной контекста шаблона, в которой будет сохранена найденная запись, в виде строки. В параметре obj передается объект записи.

В изначальной реализации возвращает значение атрибута context object name или, если этот атрибут хранит None, приведенное к нижнему регистру имя модели (так, если задана модель Rubric, метод вернет имя rubric);

□ get object (self, queryset=None) — метод, ВЫПОЛНЯЮЩИЙ ПОИСК записи ПО указанным критериям и возвращающий найденную запись в качестве результата.

В параметре queryset может быть передан набор записей, в котором должен выполняться поиск.

В изначальной реализации ищет запись в наборе из параметра queryset или, если он не задан, — в наборе записей, возвращенном методом get queryset ().

Значения ключа и слага получает из словаря, сохраненного в атрибуте экземпляра kwargs (его создает метод setup о), а имена необходимых URL-параметров— ИЗ атрибутов pk\_url\_kwarg И siug url kwarg. Если запись не найдена, метод возбуждает исключение Http404 из модуля django.http;

□ get\_context\_data(self, \*\*kwargs) — переопределенный метод, создающий и возвращающий контекст шаблона.

В изначальной реализации требует, чтобы в экземпляре текущего контроллера-класса присутствовал атрибут object, Хранящий найденную Запись ИЛИ None, если таковая не была найдена, а также если контроллер используется для создания новой записи. В контексте шаблона создает переменную object и переменную с именем, возвращенным методом get\_context\_object\_name(), обе переменные хранят найденную запись.

Примесь singleobjecttemplateresponsemixin:

Рендеринг шаблона на основе найденной записи класс-примесь singleobjecttemplateresponsemixin, наследующий ОТ templateresponsemixin, выполняет рендеринг шаблона на основе записи, найденной в модели. Он требует, чтобы в контроллере-классе присутствовал атрибут object, в котором хранится либо найденная запись в виде объекта модели, либо None, если запись не была найдена, а также если контроллер используется для создания новой записи.

Вот атрибуты и методы, поддерживаемые этим классом:

□ template name field— атрибут, содержащий ИМЯ ПОЛЯ модели, в котором хранится путь к шаблону. Если None, то путь к шаблону не будет извлекаться из записи модели (поведение по умолчанию);

□ template name suffix — атрибут, хранящий строку С суффиксом, который будет добавлен к автоматически сгенерированному пути к шаблону (по умолчанию: "\_detail”);

□ get\_tempiate\_names (self)— переопределенный метод, возвращающий список путей к шаблонам, заданных в виде строк.

В изначальной реализации возвращает список:

• либо из пути, извлеченного из унаследованного атрибута template name, если э тот путь указан;

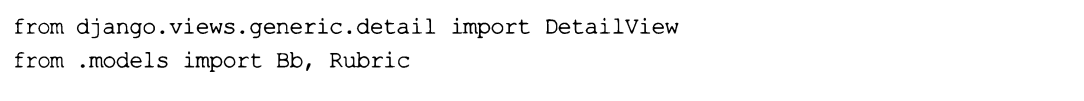
• либо из:

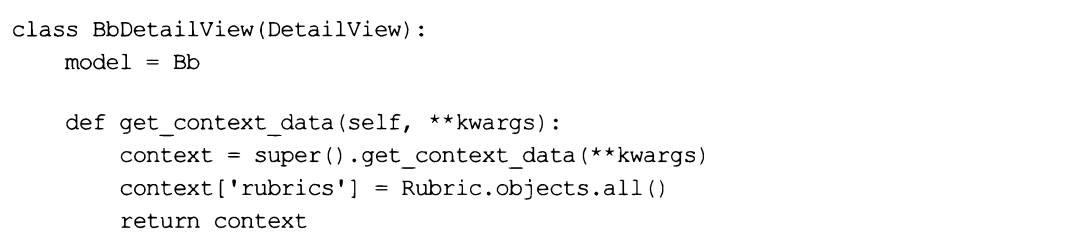
° пути, извлеченного из поля модели, имя которого хранится в атрибуте Template name field, если все необходимые данные (имя поля, запись модели и сам путь в поле этой записи) указаны;

П пути вида <псевдоним приложения>\<имя модели><суффикс>.Ыт\ (так, для модели вь из приложения bboard будет сформирован путь bboard\bb\_Detail.html).

Контроллер detailview: все вместе контроллер-класс Detailview наследует классы View, singleobj ectmixin И Singieobjecttempiateresponsemixin. Он ищет запись по полученным значениям ключа или слага, заносит ее в атрибут object (чтобы успешно работали наследуемые им примеси) и выводит на экран страницу с содержимым этой записи.

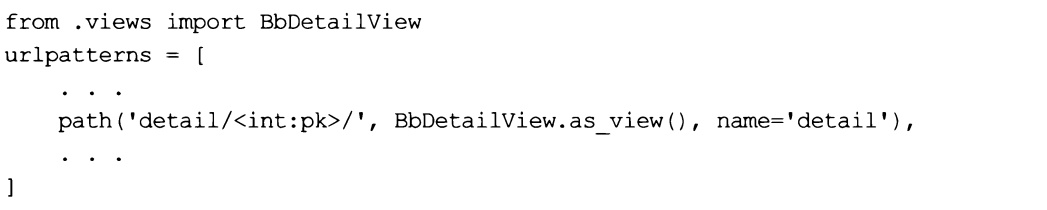
Приведен код контроллера-класса bbdetaiiview, производного от Detailview и выводящего страницу с объявлением, выбранным посетителем.





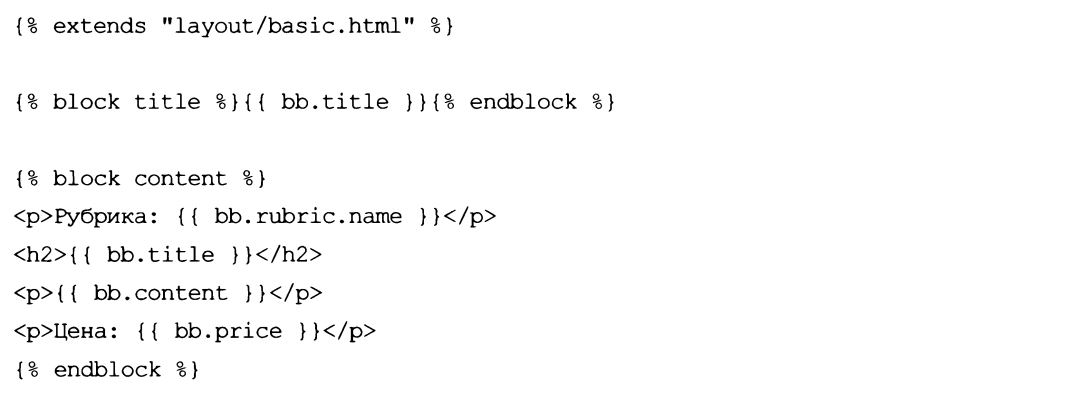
Компактность кода контроллера обусловлена в том числе и тем, что он следует соглашениям. Так, в нем не указан путь к шаблону — значит, класс будет искать шаблон со сформированным по умолчанию путем bboard\bb\_detail.html.

Добавим в список маршрутов маршрут, связанный с этим контроллером:

****

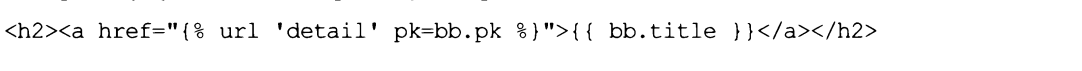
Опять же, этот маршрут написан соответственно соглашениям — у URL-параметра, содержащего ключ записи, указано имя pk, используемое классом Detailview по умолчанию.

Теперь напишем шаблон bboard\bb\_detail.html.

****

По умолчанию класс bbdetailview создаст в контексте шаблона переменную ьь, хранящую найденную запись (эту функциональность он унаследовал от базового класса Detailview). В коде шаблона мы используем эту переменную.

Осталось добавить в шаблоны bboard\index.html и bboard\by\_rubric.html гиперссылки, которые будут вести на страницу выбранного объявления:



Как видим, применяя контроллеры-классы достаточно высокого уровня и, главное, следуя заложенным в них соглашениям, можно выполнять весьма сложные действия без написания громоздкого кода.

1. Классы, выводящие наборы записей

Обобщенные классы из модуля django.views.generic.list выводят на экран целый набор записей.

**Примесь multipleobjectmixirr**

Извлечение набора записей класс-примесь muitipieobj ectmixin, наследующий от contextmixin, извлекает из модели набор записей, возможно, отфильтрованный, отсортированный и разбитый на части посредством пагинатора. Полученный набор записей он помещает в контекст шаблона.

Номер части, которую нужно извлечь, передается в составе интернет-адреса, через URL- или GET-параметр раде. Номер должен быть целочисленным и начинаться С 1. Если это правило нарушено, то будет возбуждено исключение Http404. Допустимо также указывать значение "last”, обозначающее последнюю часть.

Примесь поддерживает следующие атрибуты и методы:

□ model — атрибут, задает модель;

□ queryset — атрибут, указывает либо диспетчер записей (Manager), либо исходный набор записей (Queryset), из которого будут извлекаться записи;

□ get queryset (seif) — метод, должен возвращать исходный набор записей (Queryset), из которого будут извлекаться записи.

В изначальной реализации возвращает значение атрибута queryset, если оно задано, или набор записей из модели, которая задана в атрибуте model;

□ ordering — атрибут, задающий параметры сортировки записей. Значение может быть указано в виде:

• строки с именем поля — для сортировки только по этому полю. По умолчанию будет выполняться сортировка по возрастанию значения поля. Чтобы указать сортировку по убыванию, нужно предварить имя поля символом ’’Минус”;

• последовательности строк с именами полей — для сортировки сразу по нескольким полям.

Если значение атрибута не указано, станет выполняться сортировка, заданная в параметрах модели, или, когда таковое не указано, записи сортироваться не будут;

□ get ordering(seif) — метод, должен возвращать параметры сортировки записей. В изначальной реализации возвращает значение атрибута ordering;

□ paginate by— атрибут, задающий целочисленное количество записей в одной части пагинатора. Если не указан или его значение равно None, набор записей не будет разбиваться на части;

□ get paginate by(seif, queryset) — метод, должен возвращать число записей набора, полученного в параметре queryset, помещающихся в одной части пагинатора. В изначальной реализации просто возвращает значение из атрибута Paginate\_by;

□ page kwarg — атрибут, указывающий имя URL- или GET-параметра, через который будет передаваться номер выводимой части пагинатора, в виде строки (по умолчанию: ’’раде”);

□ paginate orphans— атрибут, задающий целочисленное минимальное число записей, которые могут присутствовать в последней части пагинатора. Если последняя часть пагинатора содержит меньше записей, то оставшиеся записи будут выведены в предыдущей части. Если задать значение 0, то в последней части может присутствовать сколько угодно записей (поведение по умолчанию);

□ get paginate orphans (self) — метод, должен возвращать минимальное число записей, помещающихся в последней части пагинатора. В изначальной реализации возвращает значение атрибута paginate orphans;

□ aiiow empty— атрибут. Значение True разрешает извлечение ’’пустой”, т.е. Не содержащей ни одной записи, части пагинатора (поведение по умолчанию). Значение False, напротив, предписывает при попытке извлечения ’’пустой” части возбуждать исключение Http404;

□ get aiiow empty(self) — метод, должен возвращать True, если разрешено извлечение ’’пустой” части пагинатора, или False, если такое недопустимо. В изначальной реализации возвращает значение атрибута aiiow empty;

□ paginator ciass — атрибут, указывающий класс используемого пагинатора (по умолчанию: Paginator ИЗ модуля django.core.paginator);

□ get paginator () — метод, должен создавать объект пагинатора и возвращать его в качестве результата. Формат объявления:

Get\_paginator(self, queryset, per\_page, orphans=0, Allоw\_empty\_first\_page=True) параметр queryset хранит обрабатываемый пагинатором набор записей, параметр per page — число записей в части, orphans — минимальное число записей в последней части пагинатора, a aiiow empty first page указывает, разрешено извлечение ’’пустой” части (True) или нет (False).

В изначальной реализации создает экземпляр класса пагинатора из атрибута Paginator ciass, передавая его конструктору все полученные им параметры;

□ paginate\_queryset (self, queryset, page\_size) —метод, разбивает набор записей, полученный в параметре queryset, на части с указанным в параметре page size числом записей в каждой части и возвращает кортеж из четырех элементов:

• объекта самого пагинатора;

• объекта его текущей части, номер которой был получен с URL- или getпараметром;

• набора записей, входящих в текущую часть (извлекается из атрибута Object list объекта текущей части пагинатора);

• True, если извлеченный набор записей действительно был разбит на части с применением пагинатора, и False — в противном случае;

□ context object name — атрибут, задает имя переменной контекста шаблона, в которой будет сохранен извлеченный набор записей;

О get\_context\_object\_name(seif, object\_iist) — метод, должен возвращать строку с именем переменной контекста шаблона, в которой будет сохранен набор записей, полученный в параметре object list.

В изначальной реализации возвращает имя из атрибута context object name или, если оно не указано, приведенное к нижнему регистру имя модели с добавленным суффиксом list;

□ get context data (seif, \*\*kwargs) — переопределенный метод, создающий и возвращающий контекст шаблона.

В изначальной реализации извлекает набор записей из необязательного параметра object list или, если этот параметр не указан, из атрибута object list.

После чего возвращает контекст с пятью переменными:

• object list— выводимый на странице набор записей (если используется пагинатор, это будет набор записей из его текущей части);

• переменная с именем, возвращенным методом get\_context\_object\_name(),— то же самое;

• is\_paginated— True, если применялся пагинатор, И False— В противном случае;

• paginator — объект пагинатора или None, если пагинатор не применялся;

• page obj — объект текущей страницы пагинатора или None, если пагинатор не применялся.

**Примесь multipleobjecttemplateresponsemixin:**

Рендеринг шаблона на основе набора записей класс-примесь multipleobjecttemplateresponsemixin, наследующий ОТ templateresponsemixin, выполняет рендеринг шаблона на основе извлеченного из модели набора записей. Он требует, чтобы в контроллере-классе присутствовал атрибут Object list, в котором хранится набор записей.

Вот список атрибутов и методов, поддерживаемых им:

□ template name suffix— атрибут, хранящий Строку С суффиксом, который будет добавлен к автоматически сгенерированному пути к шаблону (по умолчанию: "\_iist”);

О get tempiate names (self)— переопределенный метод, возвращающий список путей к шаблонам, заданных в виде строк.

В изначальной реализации возвращает список из:

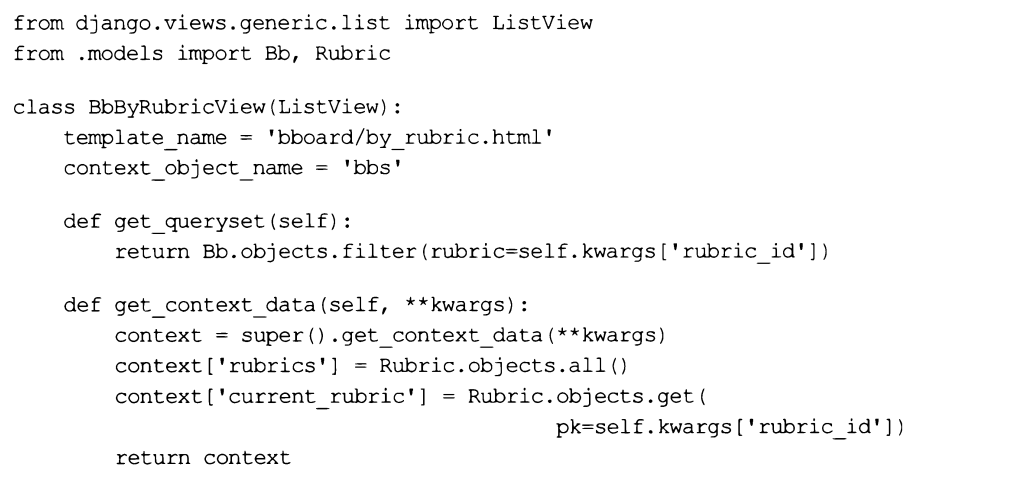
• пути, полученного из унаследованного атрибута template name, если этот путь указан;

• пути вида <псевдоним приложения>\<имя модели><суффикс>.Ыт\ (так, для модели вь из приложения bboard будет сформирован путь bboard\bb\_list.html).

**Контроллер listview**

Все вместе контроллер-класс Listview наследует классы View, muitipieobj ectmixin и Muitipieobj ecttempiateresponsemixin. Он извлекает из модели набор записей, записывает его в атрибут object list (чтобы успешно работали наследуемые им примеси) и выводит на экран страницу со списком записей.

Приведен код контроллера-класса bbbyrubricview, унаследованного от Listview и выводящего страницу с объявлениями из выбранной посетителем рубрики.

****

Код этого контроллера получился более объемным, чем у ранее написанного контроллера-функции by rubrico. Это обусловлено особенностями используемого нами шаблона bboard\by\_rubric.html. Во-первых, требуется вывести список рубрик и текущую рубрику, следовательно, придется добавить все эти данные в контекст шаблона, переопределив метод get\_context\_data ().

Во-вторых, мы используем уже имеющийся шаблон, поэтому вынуждены указать в контроллере его имя и имя переменной контекста, в которой будет храниться список объявлений.

Значение URL-параметра rubric id мы получили обращением к словарю из атрибута kwargs, содержащему все URL-параметры, указанные в маршруте. Из контекста шаблона извлечь его мы не можем.

1. Классы, работающие с формами

Обобщенные контроллеры-классы из модуля django.views.generic.edit рассчитаны на работу с формами, как связанными с моделями, так и обычными.

**Классы для вывода и валидации форм**

Классы самого низкого уровня ’’умеют” лишь вывести форму, проверить занесенные в нее данные на корректность и, в случае ошибки, вывести повторно, вместе с предупреждающими сообщениями.

**Примесь formmixin: создание формы**

Класс-примесь formmixin, производный от класса contextmixin, создает форму (неважно, связанную с моделью или обычную), проверяет введенные в нее данные, выполняет перенаправление, если данные прошли проверку, или выводит форму повторно (в противном случае).

Вот набор поддерживаемых атрибутов и методов:

□ form class — атрибут, хранит ссылку на класс используемой формы;

□ get form ciass (self) — метод, должен возвращать ссылку на класс используемой формы. В изначальной реализации возвращает значение атрибута form Class;

□ initial — атрибут, хранящий словарь с изначальными данными для занесения в только что созданную форму. Ключи элементов этого словаря должны соответствовать полям формы, а значения элементов зададут значения полей. По умолчанию хранит пустой словарь;

□ get initial (self) — метод, должен возвращать словарь с изначальными данными для занесения в только что созданную форму. В изначальной реализации просто возвращает значение атрибута initial;

□ success uri — атрибут, хранит интернет-адрес для перенаправления, если введенные в форму данные прошли проверку на корректность;

□ get success uri (self) — метод, должен возвращать интернет-адрес для перенаправления в случае, если введенные в форму данные прошли валидацию. В изначальной реализации возвращает значение атрибута success\_uri;

П prefix — атрибут, задает строковый префикс для имени формы, который будет присутствовать в создающем форму HTML-коде. Префикс стоит задавать только в том случае, если планируется поместить несколько однотипных форм в одном Теге <form>. По умолчанию хранит значение None (отсутствие префикса);

□ get\_prefix(self) — метод, должен возвращать префикс для имени формы.

В изначальной реализации возвращает значение из атрибута prefix;

П get\_form(self, form\_class=None) — метод, создающий И возвращающий объект формы.

В изначальной реализации, если класс формы указан в параметре form ciass, создает экземпляр этого класса, в противном случае — экземпляр класса, возвращенного методом get\_form\_class (). При этом конструктору класса формы передаются параметры, возвращенные методом get form kwargs ();

□ get form kwargs (self) — метод, должен создавать и возвращать словарь с параметрами, которые будут переданы конструктору класса формы в методе Get—form ().

В изначальной реализации возвращает словарь с элементами:

• initial — словарь с изначальными данными, возвращенный методом get Initial();

• prefix — префикс ДЛЯ имени формы, возвращенный методом get prefix ();

Следующие два элемента создаются только в том случае, если для отправки запроса применялись HTTP-методы POST и PUT (т. Е. При проверке введенных в форму данных):

• data — словарь с данными, занесенными в форму посетителем;

• files — словарь с файлами, отправленными посетителем из формы;

□ get—context—data (seif, \*\*kwargs) — переопределенный метод, создающий и возвращающий контекст шаблона.

В изначальной реализации добавляет в контекст шаблона переменную form, хранящую созданную форму;

□ form\_valid (self, form) — метод, должен выполнять обработку данных, введенных в переданную через параметр form форму, в том случае, если они прошли валидацию.

В изначальной реализации просто выполняет перенаправление по адресу, возвращенному методом get-success-url ();

□ form\_invalid (seif, form) — метод, должен выполнять обработку ситуации, когда данные, введенные в переданную через параметр form форму, не проходят валидацию. В изначальной реализации повторно выводит страницу с формой на экран.

**Контроллер processformview:**

Вывод и обработка формы

Контроллер-класс processformview, производный от класса view, выводит на экран страницу с формой, принимает введенные данные и проводит их валидацию.

Он переопределяет три метода, унаследованные от базового класса:

□ get (self, request, \*args, \*\*kwargs) — ВЫВОДИТ Страницу С формой на экран;

□ post (self, request, \*args, \*\*kwargs) — получает введенные в форму данные и выполняет их валидацию. Если валидация прошла успешно, вызывает метод Form\_valid (), В ПРОТИВНОМ Случае — метод form\_invalid ();

О put (self, request, \*args, \*\*kwargs)—TO же, ЧТО И post ().

**Контроллер-класс formview:**

Создание, вывод и обработка формы

Контроллер-класс formview, производный ОТ formmixin, processformview и Tempiateresponsemixin, создает форму, выводит на экран страницу с этой формой, проверяет на корректность введенные данные и, в случае отрицательного результата проверки, выводит страницу с формой повторно. Нам остается только реализовать обработку корректных данных, переопределив метод form valid ().

Приведен код контроллера-класса bbaddview, добавляющего на виртуальную доску новое объявление.



При написании этого класса мы столкнемся с проблемой. Чтобы после сохранения объявления сформировать интернет-адрес для перенаправления, нам нужно получить значение ключа рубрики, к которой относится добавленное объявление.

Поэтому мы переопределили метод get\_form(), в котором сохранили созданную форму в атрибуте object. После этого в коде метода get\_success\_url () без проблем сможем получить доступ к форме и занесенным в нее данным.

Сохранение введенных в форму данных мы выполняем в переопределенном методе Form\_valid().

**Классы для добавления, правки и** **удаления записей**

Описанные далее высокоуровневые классы, помимо обработки форм, выполняют добавление, правку и удаление записей.

Примесь modelformmixin: создание формы, связанной с моделью класс-примесь modelformmixin, наследующий ОТ классов singleobj ectmixin и Formmixin, полностью аналогичен последнему, но работает с формами, связанными с моделями.

Вот список поддерживаемых им атрибутов и методов:

□ model — атрибут, задает ссылку на класс модели, на основе которой будет создана форма;

□ fields — атрибут, указывает последовательность имен полей модели, которые должны присутствовать в форме.

Можно указать либо модель и список ее полей в атрибутах model и fields, либо непосредственно класс формы в атрибуте form class, унаследованном от класса Formmixin, но никак не одновременно и то, и другое.

Если указан атрибут model, обязательно следует задать также и атрибут fields. Если этого не сделать, возникнет ошибка;

□ get form ciass (self) — переопределенный метод, должен возвращать ссылку на класс используемой формы.

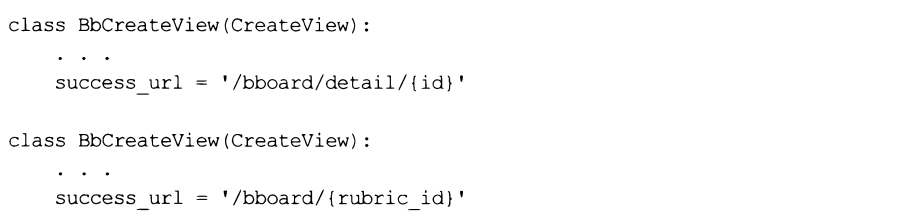
В изначальной реализации возвращает значение атрибута form class, если оно задано. В противном случае возвращается ссылка на класс формы, автоматически созданный на основе модели, которая взята из атрибута model или извлечена из набора записей, заданного в унаследованном атрибуте queryset. Для создания класса формы также используется список полей из атрибута fields;

□ success uri — атрибут, хранит интернет-адрес для перенаправления, если введенные в форму данные прошли проверку на корректность.

В отличие от одноименного атрибута базового класса formmixin, он поддерживает указание непосредственно в строке с интернет-адресом специальных последовательностей символов вида {<имя поля таблицы в базе данных>}. Вместо такой последовательности будет подставлено значение поля с указанным именем.

Отметим, что в такие последовательности должно подставляться имя не поля модели, а таблицы базы данных, которая обрабатывается моделью. Так, для вставки в адрес ключа записи следует использовать поле id, а не pk, а для вставки внешнего ключа — поле rubric id, а не rubric.

Примеры:



□ get success url (seif) — переопределенный метод, возвращает адрес для перенаправления в случае, если введенные данные прошли валидацию.

В изначальной реализации возвращает значение атрибута success uri, в котором последовательности вида {<имя поля таблицы в базе данных>} уже заменены значениями соответствующих полей. Если адрес там не указан, пытается получить его вызовом метода get\_absoiute\_uri () модели;

□ get form kwargs (self) — переопределенный метод, создает и возвращает словарь с параметрами, которые будут переданы конструктору класса формы в унаследованном методе get form ().

В изначальной реализации добавляет в словарь, сформированный унаследованным методом, элемент instance, хранящий обрабатываемую формой запись модели (если она существует, т.е. Форма используется не для добавления записи).

Эта запись извлекается из атрибута object;

□ form\_vaiid(self, form) — переопределенный метод, должен выполнять обработку данных, введенных в переданную через параметр form форму, в том случае, если они прошли валидацию.

В изначальной реализации сохраняет содержимое формы в модели, вызвав у нее метод save о, присваивает новую запись атрибуту object, после чего вызывает унаследованный Метод form valid ().

**Контроллер createview: создание новой записи**

Контроллер-класс createview наследует ОТ классов processformview, modelformmixin и singleobjecttemplateresponsemixin. Он ВЫВОДИТ форму, проверяет введенные в нее данные и создает на их основе новую запись.

Атрибут template name suffix этого класса хранит строку с суффиксом, который удет добавлен к автоматически сгенерированному пути к шаблону (по умолчанию: " form”).

Также в классе доступен атрибут object, в котором хранится созданная в модели запись или None, если таковая еще не была создана.

Пример контроллера-класса, производного от createview.

**Контроллер updateview: исправление записи**

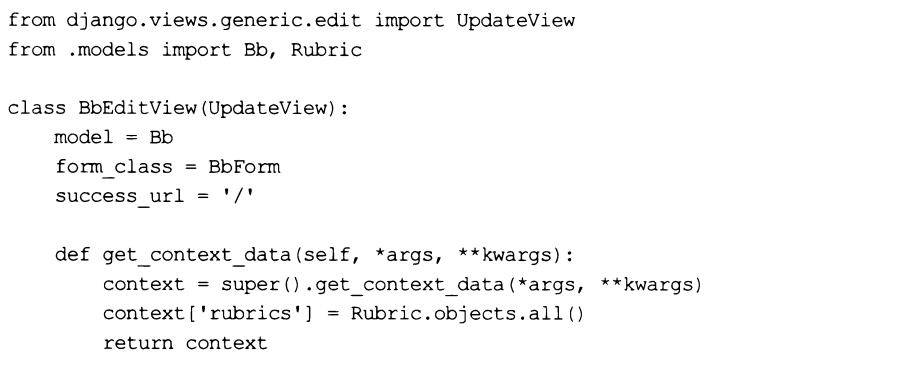
Контроллер-класс updateview наследует ОТ классов processformview, modelformmixin и singleobjecttemplateresponsemixin. Он ищет запись ПО полученным ИЗ url параметра ключу или слагу, выводит страницу с формой для ее правки, проверяет и сохраняет исправленные данные.

Атрибут tempiate name suffix этого класса хранит строку с суффиксом, который будет добавлен к автоматически сгенерированному пути к шаблону (по умолчанию: "form”).

Также в классе доступен атрибут object, в котором хранится исправляемая запись.

Поскольку класс updateview предварительно выполняет поиск записи, в нем необходимо указать модель (в унаследованном атрибуте model), набор записей (в атрибуте queryset) ИЛИ переопределить метод get\_queryset ().

Приведен код контроллера-класса bbeditview, который выполняет исправление объявления.

****

Шаблон bboard\bb\_form.html вы можете написать самостоятельно, взяв за основу уже имеющийся шаблон bboard\create.html (там всего лишь придется поменять текст добавление на Исправление, а подпись кнопки— с Добавить на Сохранить).

Также самостоятельно вы можете записать маршрут для этого контроллера и вставить в шаблоны bboard\index.html и bboard\by\_rubric.html код, создающий гиперссылки на страницы исправления объявлений.

**Примесь deletionmixin: удаление записи**

Класс-примесь deletionmixin добавляет наследующему ее контроллеру инструменты для удаления записи. Он предполагает, что запись, подлежащая удалению, уже найдена и сохранена в атрибуте object.

Класс объявляет атрибут success\_uri и метод get\_success\_uri (), аналогичные присутствующим в примеси modelformmixin.

**Контроллер deletevfew:**

Удаление записи с подтверждением

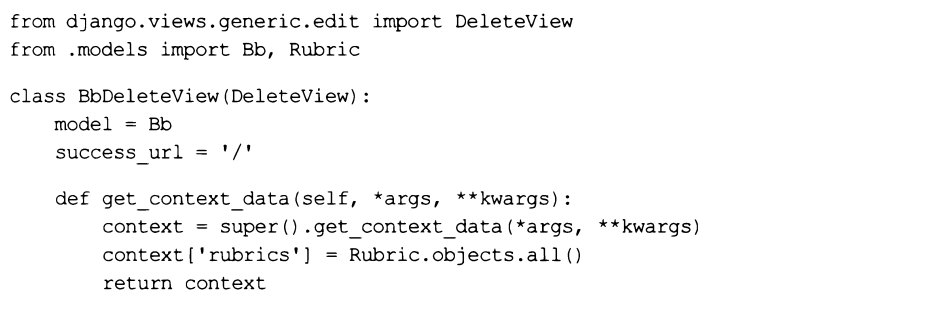
Контроллер-класс deleteview наследует ОТ классов Detailview, deletionmixin и Singieobjecttempiateresponsemixin. Он ищет запись по полученному из URL-napaметра ключу или слагу, выводит страницу подтверждения, включающую в себя форму с кнопкой удаления, и удаляет запись.

Атрибут template\_name\_suffix ЭТОГО Класса хранит строку С суффиксом, который будет добавлен к автоматически сгенерированному пути к шаблону (по умолчанию: "\_confirm\_delete”).

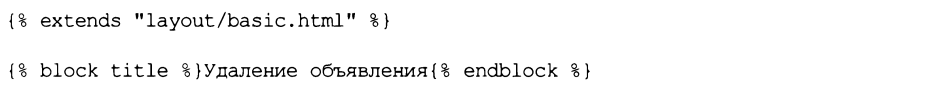
Также в классе доступен атрибут object, в котором хранится удаляемая запись.

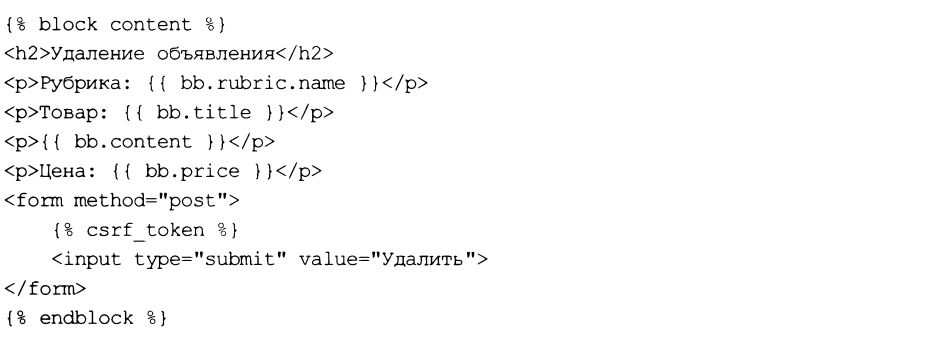
Поскольку класс Deleteview предварительно выполняет поиск записи, в нем необходимо указать модель (в атрибуте model), набор записей (в атрибуте queryset) или переопределить метод get\_queryset ().

Приведен код контроллера-класса bbdeieteview, который выполняет удаление объявления.



Код шаблона bboard\bb\_confirm\_delete.html, выводящего страницу подтверждения и форму с кнопкой Удалить, приведен.





1. Классы для вывода хронологических списков

Классы для вывода хронологических списков обобщенные классы ИЗ модуля django.views.generic.dates ВЫВОДЯТ хронологические списки: за определенный год, месяц, неделю или дату, за текущее число.

Вывод последних записей рассматриваемые далее классы выводят хронологические списки наиболее ’’свежих” записей.

Примесь datemixin: фильтрация записей по дате класс-примесь datemixin предоставляет наследующим контроллерам возможность фильтровать записи по значениям даты (или временной отметки), хранящимся в заданном поле.

Вот атрибуты и методы, поддерживаемые этим классом:

П date field --- атрибут, указывает ИМЯ ПОЛЯ модели типа datefieid или Datetimefieid, по которому будет выполняться фильтрация записей. Имя ПОЛЯ должно быть задано в виде строки;

□ get\_date\_fieid(self) — метод, должен возвращать имя поля модели, по которому будет выполняться фильтрация записей. В изначальной реализации возвращает значение атрибута date field;

П allow—future — атрибут. Значение True указывает включить в результирующий набор записи, у которых возвращенное методом get date fieido поле хранит дату из будущего. Значение False запрещает включать такие записи в набор (поведение по умолчанию);

□ get allow future (self) — метод, должен возвращать значение True ИЛИ False, говорящее, следует ли включать в результирующий набор ’’будущие” записи.

В изначальной реализации возвращает значение атрибута allow future.

Контроллер basedatelistview: базовый класс контроллер-класс basedatelistview наследует ОТ классов View, muitipieobj ectmixin и datemixin. Он предоставляет базовую функциональность для других, более специализированных классов, в частности, задает сортировку записей по убыванию значения ПОЛЯ, возвращенного методом get date field о классе datemixin.

В классе объявлены такие атрибуты и методы:

□ allow empty— атрибут. Значение True разрешает извлечение ’’пустой”, т.е. Не содержащей ни одной записи, части пагинатора. Значение False, напротив, предписывает при попытке извлечения ’’пустой” части возбудить исключение Http404 (поведение по умолчанию);

□ date list period— атрибут, указывающий, по какой части следует урезать значения даты. Должен содержать значение "year” (год, значение по умолчанию), "month" (месяц) или "day" (число, т. Е. Дата не будет урезаться);

□ get\_date\_iist\_period(seif) — метод, должен возвращать обозначение части, по которой нужно урезать дату. В изначальной реализации возвращает значение атрибута date\_list\_period;

□ get dated items (seif) — метод, должен возвращать кортеж из трех элементов:

• список значений дат, хранящихся в записях из полученного набора;

• сам набор записей;

• словарь, элементы которого будут добавлены в контекст шаблона.

В изначальной реализации возбуждает исключение notimpiementederror. Предназначен для переопределения в подклассах;

□ get dated queryset (self, \*\*iookup) — метод, возвращает набор записей, отфильтрованный согласно заданным условиям, которые в виде словаря передаются в параметре lookup;

П get\_date\_list(self, queryset, date\_type=None, ordering='ASC') — метод, возвращает список значений даты, урезанной по части, что задана в параметре Date type (если он отсутствует, будет использовано значение, возвращенное методом get date iist periodo), для которых существуют записи в наборе, заданном в параметре queryset. Параметр ordering задает направление сортировки: "asc" (по возрастанию, поведение по умолчанию) или "desc" (по убыванию).

Класс добавляет в контекст шаблона два дополнительных элемента:

□ object list — результирующий набор записей;

□ date list — список урезанных значений дат, хранящихся в записях из полученного набора.

Контроллер archivelndexview:

Вывод последних записей контроллер-класс archivelndexview наследует ОТ классов basedatelistview и Muitipieobjecttempiateresponsemixin. Он выводит хронологический список записей, отсортированных по убыванию значения заданного поля.

Для хранения результирующего набора выводимых записей в контексте шаблона создается переменная latest. В переменной date list контекста шаблона хранится список значений дат, урезанных до года. К автоматически сгенерированному пути к Шаблону ПО умолчанию добавляется суффикс archive.

Приведён код контроллера-класса Bbindexview, основанного на классе archivelndexview. Для вывода страницы он может использовать шаблон Bboard\index.html,

****

А вот так мы можем использовать хранящийся в переменной date list контекста шаблона список дат, урезанных до года:

****

В результате на экране появится список разделенных пробелами годов, за которые в наборе имеются записи.

Вывод записей по годам следующая пара классов выводит список записей, относящихся к определенному году.

Примесь yearmixin: извлечение года класс-примесь yearmixin извлекает из URL- или GET-параметра с именем year значение года, которое будет использовано для последующей фильтрации записей.

Этот класс поддерживает следующие атрибуты и методы:

□ year format — атрибут, указывает строку с форматом значения года, поддерживаемым функцией strftimeo языка Python. Значение года будет извлекаться из URL- или GET-параметра и преобразовываться в нужный вид согласно этому формату. Значение по умолчанию: "%y" (год из четырех цифр);

□ get year format (self) — метод, должен возвращать строку с форматом значения года. В изначальной реализации возвращает значение атрибута year format;

□ year— атрибут, задает значение года в виде строки. Если None, то год будет извлекаться из URL- или GET-параметра (поведение по умолчанию);

□ get year (seif) — метод, должен возвращать значение года в виде строки. В изначальной реализации пытается вернуть значение (если оно задано): атрибута класса year, URL-параметра year, GET-параметра year. Если все попытки завершились неудачами, то возбуждает исключение Http404;

□ get previous year(self, date) — метод, возвращает значение даты, представляющей собой первый день года, который предшествует дате из параметра date. В зависимости от значений атрибутов класса aiiow empty и aiiow\_future может возбуждать исключение Http404;

□ get next year(seif, date) — метод, возвращает значение даты, представляющей собой первый день года, который следует за датой из параметра date. В зависимости от значений атрибутов класса aiiow empty и aiiow\_future может возбуждать исключение Http404.

Контроллер yeararchiveview: вывод записей за год контроллер-класс yeararchiveview наследует классы basedatelistview, yearmixiиn muitipieobjecttempiateresponsemixin. Он выводит хронологический список записей, относящихся к указанному году и отсортированных по возрастанию значения заданного поля.

Класс поддерживает дополнительные атрибут и метод:

□ make object list — атрибут. Если True, то будет сформирован и добавлен в контекст шаблона набор всех записей за заданный год. Если False, то в контексте шаблона будет присутствовать ’’пустой” набор записей (поведение по умолчанию);

□ get make object list (self) — метод, должен возвращать логический признак того, формировать ли полноценный набор записей, относящихся к заданному году. В изначальной реализации возвращает значение атрибута make objectlist.

Набор записей, относящихся к заданному году, будет храниться в переменной контекста шаблона, имеющей имя по умолчанию: object list. Кроме того, в контексте шаблона будут созданы следующие переменные:

□ date list — список значений дат, за которые в наборе существуют записи, урезанных до месяца и выстроенных в порядке возрастания;

□ year —объект типа date, представляющий заданный год;

□ previous year —объект типа date, представляющий предыдущий год;

□ next year —объект типа date, представляющий следующий год.

К автоматически сгенерированному пути к шаблону по умолчанию добавляется суффикс archiveyear.

Вывод записей по месяцам следующие два класса выводят список записей, относящихся к определенному месяцу определенного года.

Примесь monthmixin: извлечение месяца класс-примесь monthmixin извлекает из URL- или GET-параметра с именем month значение месяца, которое будет использовано для последующей фильтрации записей.

Поддерживаются следующие атрибуты и методы:

□ month format— атрибут, указывает строку с форматом значения месяца, поддерживаемым функцией strftimeo языка Python. Значение месяца будет извлекаться из URL- или GET-параметра и преобразовываться в нужный вид согласно этому формату. Значение по умолчанию: "%ь” (сокращенное наименование, записанное согласно текущим языковым настройкам);

□ get month format (seif) — метод, должен возвращать строку с форматом значения месяца. В изначальной реализации возвращает значение атрибута monthformat;

□ month— атрибут, задает значение месяца в виде строки. Если None, то месяц будет извлекаться из URL- или GET-параметра (поведение по умолчанию);

□ get month (seif) — метод, должен возвращать значение месяца в виде строки.

В изначальной реализации пытается вернуть значение (если оно задано): атрибута класса month, URL-параметра month, GET-параметра month. Если все попытки завершились неудачами, то возбуждает исключение Http404;

□ get\_previous\_month(self, date) — метод, возвращает значение даты, представляющей собой первый день месяца, который предшествует дате из параметра Date. В зависимости ОТ значений атрибутов класса allow empty И allow\_future может возбуждать исключение Http404;

□ get\_next\_month (self, date) — метод, возвращает значение даты, представляющей собой первый день месяца, который следует за датой из параметра date.

В зависимости от значений атрибутов класса allow\_empty и allow future может возбуждать исключение Http404.

Контроллер montharchiveview:

Вывод записей за месяц контроллер-класс montharchiveview наследует Классы basedatelistview, monthmixin, Yearmixin И multipleobjecttemplateresponsemixin. Он ВЫВОДИТ хронологический список записей, относящихся к указанному месяцу указанного года.

Набор записей, относящихся к заданному месяцу, будет храниться в переменной контекста шаблона, имеющей имя по умолчанию: object list.

Кроме того, в контексте шаблона будут созданы следующие переменные:

□ date list — список значений дат, за которые в наборе существуют записи, урезанных до значения числа и выстроенных в порядке возрастания;

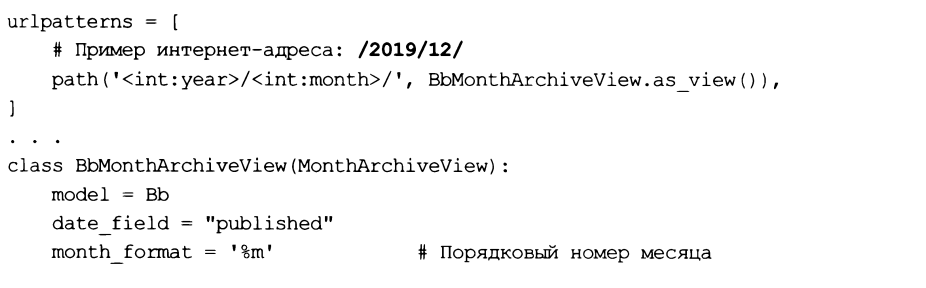
□ month —объект типа date, представляющий заданный месяц;

□ previous month —объект типа date, представляющий предыдущий месяц;

□ next month —объект типа date, представляющий следующий месяц.

К автоматически сгенерированному пути к шаблону по умолчанию добавляется суффикс \_archive\_month.

Вот небольшой пример использования контроллера montharchiveview:

****

Вывод записей по неделям эти классы выводят список записей, которые относятся к неделе с заданным порядковым номером.

Примесь weekmixin: извлечение номера недели класс-примесь weekmixin извлекает из URL- или GET-параметра с именем week номер недели, который будет использован для фильтрации записей.

Поддерживаются такие атрибуты и методы:

□ week\_format —атрибут, указывает строку с форматом номера недели, поддерживаемым функцией strftimeo языка Python. Значение номера недели будет извлекаться из URL- или GET-параметра и преобразовываться в нужный вид согласно этому формату. Значение по умолчанию: "%и" (номер недели, если первый день недели — воскресенье).

Для обработки более привычного формата номера недели, когда первым днем является понедельник, нужно указать в качестве значения формата строку ”%w";

□ get week format (self) — метод, должен возвращать строку с форматом значения недели. В изначальной реализации возвращает значение атрибута week Format;

□ week — атрибут, задает значение недели в виде строки. Если None, то номер недели будет извлекаться из URL- или GET-параметра (поведение по умолчанию);

□ get\_week(self) — метод, должен возвращать номер недели в виде строки. В изначальной реализации пытается вернуть значение (если оно задано): атрибута класса week, URL-параметра week, GET-параметра week. Если все попытки завершились неудачами, то возбуждает исключение Http404;

□ get\_previous\_week(self, date) — метод, возвращает значение даты, представляющей собой первый день недели, которая предшествует дате из параметра Date. В зависимости ОТ Значений атрибутов класса aiiow empty И allow\_future может возбуждать исключение Http404;

□ get\_next\_week(self, date) — метод, возвращает значение даты, представляющей собой первый день недели, которая следует за датой из параметра date.

В зависимости от значений атрибутов класса allow empty и aiiow\_future может возбуждать исключение Http404.

**Контроллер weekarchiveview:**

Вывод записей за неделю контроллер-класс weekarchiveview наследует классы basedatelistview, weekmixin, Yearmixin И multipleobjecttemplateresponsemixin. Он ВЫВОДИТ хронологический список записей, относящихся к указанной неделе указанного года.

Набор записей, относящихся к заданной неделе, будет храниться в переменной контекста шаблона, имеющей имя по умолчанию: object list.

В контексте шаблона также будут созданы дополнительные переменные:

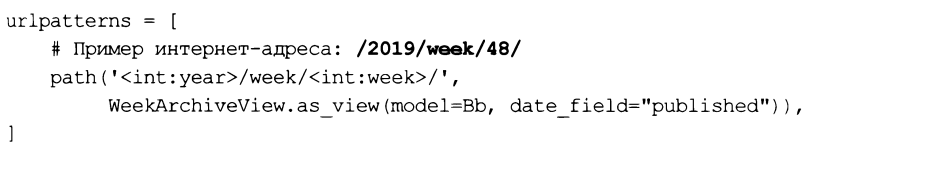
□ week —объект типа date, представляющий заданную неделю;

□ previous week —объект типа date, представляющий предыдущую неделю;

□ next week —объект типа date, представляющий следующую неделю.

К автоматически сгенерированному пути к шаблону по умолчанию добавляется суффикс archiveweek.

Пример использования контроллера weekarchiveview:

**** Вывод записей по дням следующие два класса выводят записи, относящиеся к определенному числу заданных месяца и года.

Примесь daymixin: извлечение заданного числа класс-примесь daymixin извлекает из URL- или GET-параметра с именем day число, которое будет использовано для фильтрации записей.

Атрибуты и методы, поддерживаемые классом, таковы:

□ day format— атрибут, указывает строку с форматом числа, поддерживаемым функцией strftime () языка Python. Значение числа будет извлекаться из urlили GET-параметра и преобразовываться в нужный вид согласно этому формату. Значение по умолчанию: ”%d” (число с начальным нулем);

□ get day format (self) — метод, должен возвращать строку с форматом значения числа. В изначальной реализации возвращает значение атрибута day format;

□ day — атрибут, задает значение числа в виде строки. Если None, то число будет извлекаться из URL- или GET-параметра (поведение по умолчанию);

□ get day (self) — метод, должен возвращать значение числа в виде строки. В изначальной реализации пытается вернуть значение (если оно задано): атрибута класса day, URL-параметра day, GET-параметра day. Если все попытки завершились неудачами, то возбуждает исключение Http404;

□ get previous\_day(self, date) — метод, возвращает значение даты, которая предшествует дате из параметра date. В зависимости от значений атрибутов класса aiiow empty и aiiow future может возбуждать исключение Http404;

□ get next day(seif, date) — метод, возвращает значение даты, которая следует за датой из параметра date. В зависимости от значений атрибутов класса Allow\_empty И allow\_future может возбуждать исключение Http404.

Контроллер dayarchiveview: вывод записей за день контроллер-класс dayarchiveview наследует классы basedatelistview, daymixin, Monthmixin, yearmixin И muitipieobj ecttemplateresponsemixin. Он ВЫВОДИТ xpohoлогический список записей, относящихся к указанному числу заданных месяца и Года.

Набор записей, относящихся к заданному дню, будет храниться в переменной контекста шаблона, имеющей имя по умолчанию: object list.

Дополнительные переменные, создаваемые в контексте шаблона:

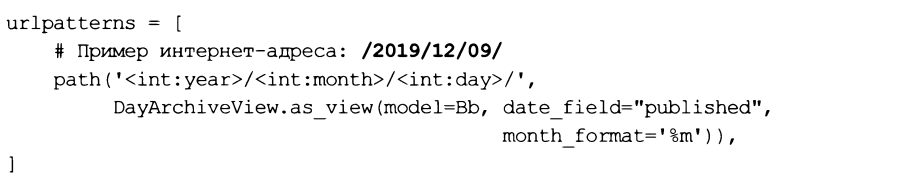
□ day —объект типа date, представляющий заданный день;

□ previous day —объект типа date, представляющий предыдущий день;

□ next day —объект типа date, представляющий следующий день.

К автоматически сгенерированному пути к шаблону по умолчанию добавляется суффикс \_archive\_day.

Пример использования контроллера dayarchiveview:

** Контроллер todayarchiveview.**

Вывод записей за текущее число контроллер-класс todayarchiveview наследует классы dayarchiveview И multipleobjecttempiateresponsemixin. Он выводит хронологический список записей, относящихся к текущему числу.

Набор записей, относящихся к текущему числу, будет храниться в переменной контекста шаблона, имеющей имя по умолчанию: object list.

В контексте шаблона будут созданы дополнительные переменные:

□ day —объект типа date, представляющий текущее число;

□ previous day —объект типа date, представляющий предыдущий день;

□ next day —объект типа date, представляющий следующий день.

К автоматически сгенерированному пути к шаблону по умолчанию добавляется суффикс \_archive\_today.

**Контроллер datedetailview.**

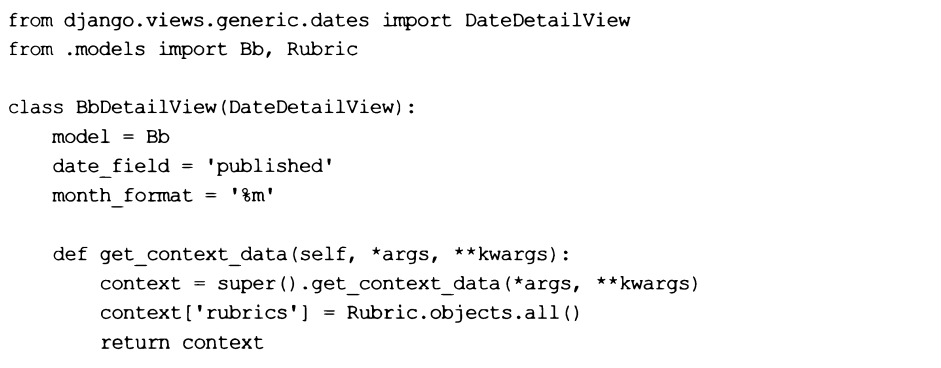
Вывод одной записи за указанное число контроллер-класс datedetailview наследует классы Detailview, datemixin, daymixin,

Monthmixin, yearmixin И singleobjecttemplateresponsemixin. Он ВЫВОДИТ однуединственную запись, относящуюся к текущему числу, и может быть полезен в случаях, когда поле даты, по которому выполняется поиск записи, хранит уникальные значения.

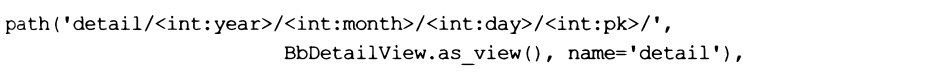
Запись, относящаяся к текущему числу, будет храниться в переменной контекста аблона, имеющей имя по умолчанию: object.

К автоматически сгенерированному пути к шаблону по умолчанию добавляется суффикс detail.

Приведён код контроллера bbdetailview, основанного на классе Datedetailview.

****

Для его использования в список маршрутов нужно добавить маршрут такого вида:

****

К сожалению, в маршрут, указывающий на контроллер datedetailview или его подкласс, следует записать URL-параметр ключа (pk) или слага (slug). Это связано с тем, что упомянутый ранее класс является производным от примеси Singieobj ectmixin, которая требует для нормальной работы один из этих urlпараметров. Такая особенность сильно ограничивает область применения контроллера-класса datedetailview.

1. Контроллер RedirectView: перенаправление

**Контроллер redirectview:**

Перенаправление контроллер-класс redirectview, производный от класса view, выполняет перенаправление по указанному интернет-адресу. Он объявлен в модуле django.views, Generic.base. Этот класс поддерживает такие атрибуты и методы:

О url — атрибут, задает строку с интернет-адресом, на который следует выполнить перенаправление.

Этот адрес может включать спецификаторы, поддерживаемые оператором % языка Python (за подробностями — на страницу https://docs.python.Org/3/Library/stdtypes.html#printf-style-string-formatting). В таких спецификаторах сследует указывать имена URL-параметров — в этом случае в результирующий интернет-адрес будут подставлены их значения;

□ pattern name — атрибут, задает имя именованного маршрута. Обратное разрешение интернет-адреса будет проведено с теми же URL-параметрами, которые получил контроллер;

□ query string— атрибут. Если True, то все GET-параметры, присутствующие в текущем интернет-адресе, будут добавлены к интернет-адресу, на который выполняется перенаправление. Если False, то GET-параметры передаваться не будут (поведение по умолчанию);

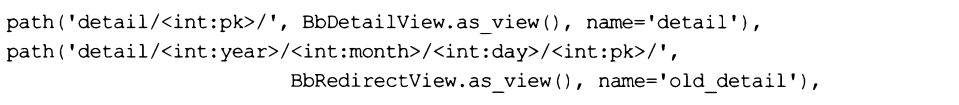
□ get\_redirect\_url(self, \*args, \*\*kwargs) —метод, должен возвращать строку с интернет-адресом, на который следует выполнить перенаправление.

В параметре kwargs передается словарь со значениями именованных URL-параметров. Параметр args в старых версиях Django хранил список со значениями неименованных URL-параметров, но в настоящее время не используется.

В реализации по умолчанию сначала извлекает значение атрибута url и выполняет форматирование, передавая значения полученных URL-параметров оператору %. Если атрибут url хранит значение None, то проводит обратное разрешение на основе значения атрибута pattern name и, опять же, значений urlпараметров. Если и эта попытка увенчалась неудачей, то отправляет ответ с кодом 410 (запрошенная страница более не существует);

□ permanent — атрибут. Если True, то будет выполнено постоянное перенаправление (с кодом статуса 301). Если False, то будет выполнено временное перенаправление (с кодом статуса 302). Значение по умолчанию: False.

В качестве примера организуем перенаправление с интернет-адресов вида 1йе12им<год>1<месяц>1<число>1<ключ>1 по адресу 1&е1ъ\\1<ключ>1. Для этого добавим в список маршруты:

****

Код контроллера bbredirectview, который МЫ используем ДЛЯ ЭТОГО, очень прост.

****

Для формирования целевого пути использован атрибут url и строка со спецификатором, обрабатываемым оператором %. Вместо этого спецификатора в строку будет подставлено значение URL-параметра pk, т. Е. Ключ записи.

1. Контроллеры-классы смешанной функциональности

**Контроллеры-классы смешанной функциональности**

Большая часть функциональности контроллеров-классов наследуется ими от классов-примесей. Наследуя классы от нужных примесей, можно создавать контроллеры смешанной функциональности.

Так, МЫ можем объявить класс, производный ОТ классов singleobj ectmixin и Listview. В результате получится контроллер, одновременно выводящий сведения о выбранной записи (функциональность, унаследованная от singleobj ectmixin) и набор связанных с ней записей (функциональность класса Listview).

Приведен КОД класса bbbyrubricview, созданного на подобном принципе и имеющего смешанную функциональность.

****

Намереваясь использовать уже существующие шаблон и маршрут, мы указали в атрибутах класса путь к нашему шаблону и имя URL-параметра, через который передается ключ рубрики.

В переопределенном методе get о вызове метода get object о, унаследованного от примеси singleobjectmixin, извлекаем рубрику с заданным ключом. Эту рубрику сохраняем в атрибуте object класса — она нам еще понадобится.

В переопределенном методе get\_context\_data() заносим в переменную Current rubric контекста шаблона найденную рубрику (взяв ее из атрибута object),

А в переменную rubrics — набор всех рубрик.

Здесь мы столкнемся с проблемой. Наш шаблон за перечнем объявлений обращается к переменной bbs контекста шаблона. Ранее, разрабатывая предыдущую редакцию контроллера, мы занесли имя этой переменной в атрибут класса contect\_object\_name. Но здесь такой "номер” не пройдет: указав новое имя для переменной в атрибуте класса contect\_object\_name, мы зададим новое имя для переменной, в которой будет храниться рубрика, а не перечень объявлений (это связано с особенностями механизма множественного наследования Python). В результате чего получим чрезвычайно трудно диагностируемую ошибку.

Поэтому мы поступили по-другому: в том же переопределенном методе get\_Context data () создали переменную bbs контекста шаблона и присвоили ей значение переменной object list, в которой по умолчанию хранится набор записей, ВЫВОДИМЫХ контроллером Listview.

И наконец, в переопределенном методе get queryset () возвращаем перечень объявлений, связанных с найденной рубрикой и полученных через диспетчер обратной связи.

Вообще лучше избегать контроллеров смешанной функциональности. Удобнее взять за основу контроллер-класс более низкого уровня и реализовать в нем всю нужную логику самостоятельно.