**Тема занятия № 51: Модуль 30. Разработка Web-служб REST. REST framework. Django REST Framework**

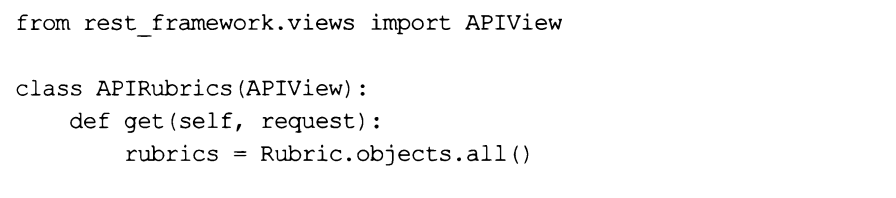
1. Контроллеры-классы Django REST framework

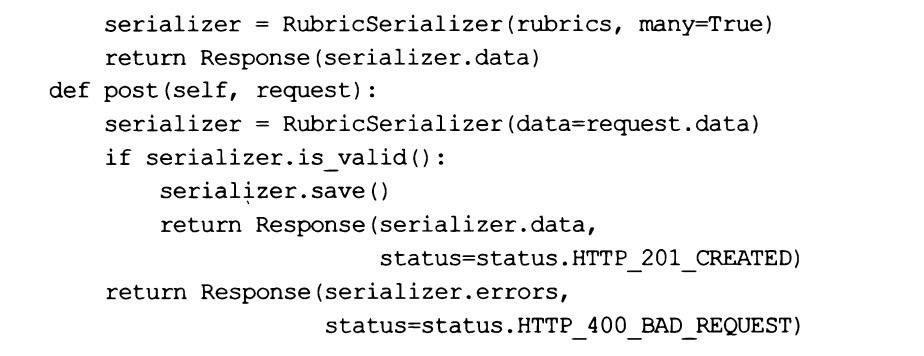
**Контроллер-класс низкого уровня**

Контроллер-класс Apiview из модуля rest framework.views всего лишь, как и аналогичный ему класс view — кстати, его суперкласс, — осуществляет диспетчеризацию по HTTP-методу: при получении запроса по методу GET вызывает метод get (), при получении запроса по методу POST — метод post о и т. Д. Если был получен запрос по методу PATCH, а метод patch () отсутствует, будет выполнен метод put ().

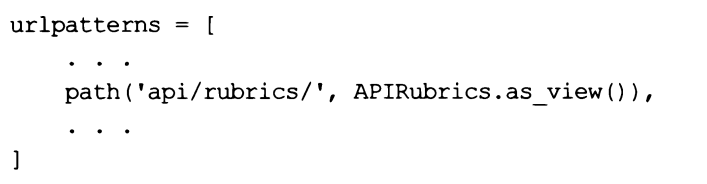
Класс apiview реализует веб-представление самостоятельно, так что декоратор api view () указывать не нужно.

Приведен код контроллера-класса apirubrics, производного от Apiview и выполняющего вывод перечня рубрик и добавление новой рубрики.





Маршрут, указывающий на такой контроллер-класс, записывается аналогично маршрутам на контроллеры-классы, рассмотренные ранее:

****

Код контроллера-класса apirubrics очень похож на код контроллеров-функций, выполняющих те же задачи. Так что контроллер-класс apirubricdetaii, который реализует вывод сведений о выбранной рубрике, правку и удаление рубрик, вы, уважаемые читатели, можете написать самостоятельно.

**Контроллеры-классы высокого уровня:**

**Комбинированные и простые**

Еще Django REST framework предоставляет набор высокоуровневых классов, объявленных в модуле rest framework.generics. Прежде всего, это четыре комбинированных контроллера-класса, которые могут выполнять сразу два или три действия, в зависимости от HTTP-метода, которым был отправлен запрос:

□ listcreateapiview— выполняет выдачу ресурса-перечня сущностей и создание нового ресурса (т. Е. Обрабатывает HTTP-методы GET и POST);

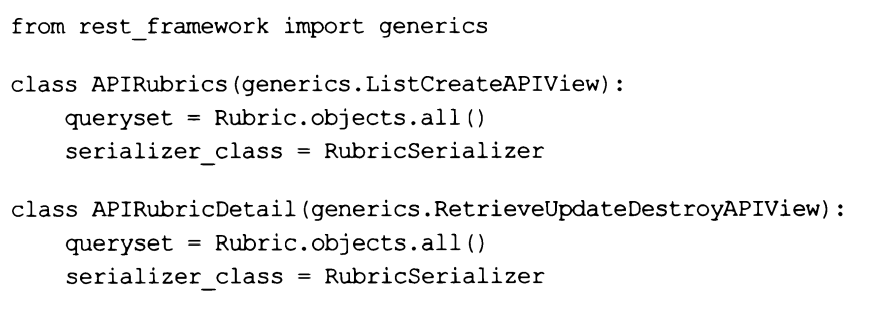
□ retrieveupdatedestroyapiview— выполняет выдачу ресурса-отдельной сущности, правку и удаление ресурса (обрабатывает методы GET, PUT, PATCH И DELETE);

□ retrieveupdateapiview — выполняет выдачу ресурса-отдельной сущности и правку ресурса (методы GET, PUT и PATCH);

□ retrievedestroyapiview —выполняет выдачу ресурса-отдельной сущности и удаление ресурса (методы GET и DELETE).

Как минимум, в таких классах нужно задать набор записей, который будет обрабатываться, и сериализатор, который будет применяться для пересылки данных фронтенду. Набор записей указывается в атрибуте queryset, а сериализатор — в атрибуте serializer\_class.

Код показывает новую реализацию контроллеров-классов apirubrics и apirubricdetail — основанную на классах listcreateapiview И retrieveupdatedestroyapiview.

****

Если интерфейс веб-службы предусматривает выполнение только какого-либо одного действия (например, вывода списка рубрик), комбинированные классы будут избыточными. В таких случаях удобнее задействовать более простые классы, выполняющие только одно действие:

□ listapiview — выполняет выдачу ресурса-перечня сущностей (т. Е. Обрабатывает HTTP-метод GET);

□ retrieveapiview —выполняет выдачу ресурса-отдельной сущности (метод GET);

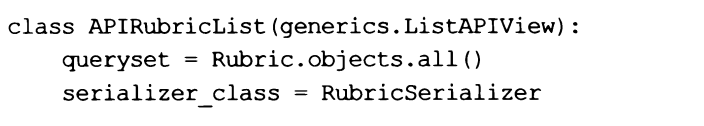
□ createapiview —выполняет создание нового ресурса (метод POST);

□ updateapiview —выполняет правку ресурса (методы PUT и PATCH);

□ destroyapiview —выполняет удаление ресурса (метод DELETE).

Используются они точно так же, как и комбинированные контроллеры-классы.

Например, код контроллера, выводящего список рубрик, может быть таким:

****

1. МетаконтроллерЫ

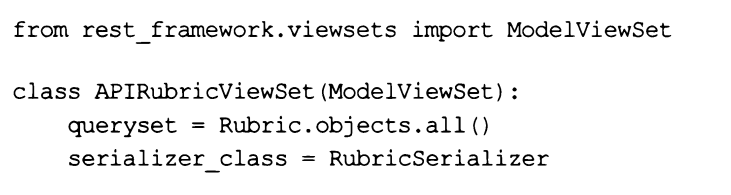
**Метаконтроллеры**

Метаконтроллер — это комбинированный контроллер-класс, выполняющий сразу все возможные действия: выдачу ресурсов-перечней, ресурсов-отдельных сущностей, добавление, правку и удаление ресурсов. Он может заменить два обычных комбинированных контроллера-класса.

Помимо этого, метаконтроллер предоставляет средства для генерирования всех необходимых маршрутов.

Метаконтроллер, связанный с моделью, создается как подкласс класса Modelviewset ИЗ модуля rest—framework.viewsets. В нем С применением атрибутов queryset И Serializer class указываются набор записей, с которым будет выполняться работа, и сериализатор, управляющий отправкой данных фронтенду.

Приведен код метаконтроллера apirubricviewset, работающего со списком рубрик. Как видим, он очень прост и компактен.

****

Чтобы сгенерировать маршруты, указывающие на отдельные функции метаконтроллера, нужно выполнить три действия:

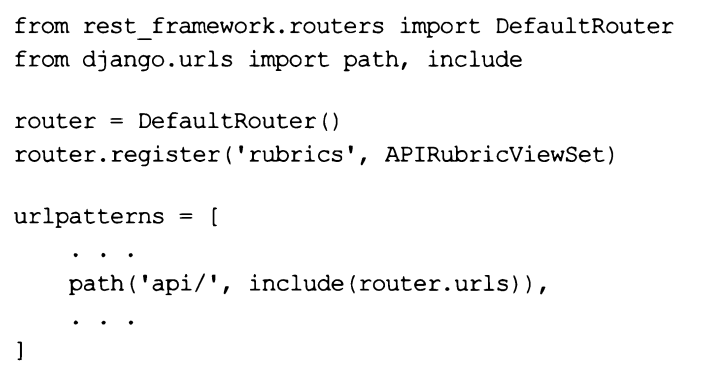
□ получить объект генератора таких маршрутов, представляющий собой экземпляр класса defaultrouter ИЗ модуля rest framework.routers. Конструктор ЭТОГО класса вызывается без параметров;

□ зарегистрировать в генераторе маршрутов метаконтроллер, связав его с выбранным префиксом. Это выполняется ВЫЗОВОМ метода register () класса defaultrouter в формате:

Register (<строка с префиксов, <ссылка на класс метаконтроллера>)

□ добавить сгенерированные маршруты в список уровня приложения или проекта, воспользовавшись функцией include о. Сами маршруты можно извлечь из атрибута urls генератора маршрутов.

Вот пример генерирования набора маршрутов для метаконтроллера apirubricviewset:

****

В результате в список будут добавлены два следующих маршрута:

□ api/rubrics/— выполняет при запросе с применением НТТР-метода:

• GET — выдачу списка рубрик;

• POST — добавление новой рубрики;

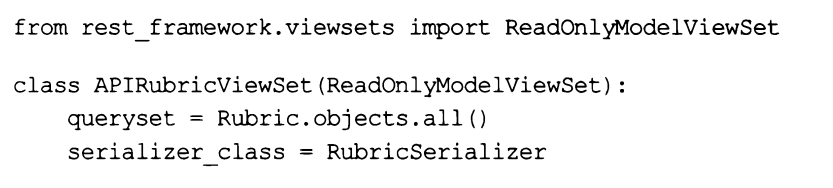
□ api/rubrics/<Jci/0Q> — выполняет при запросе с применением НТТР-метода:

• GET — выдачу рубрики с указанным ключом;

• PUT или PATCH — правку рубрики с указанным ключом;

• DELETE — удаление рубрики с указанным ключом.

Помимо класса Modelviewset, библиотека Django REST framework предлагает класс Readonlymodelviewset, объявленный В ТОМ же модуле rest\_framework.viewsets. Он реализует функциональность только по выдаче ресурса-списка сущностей и ресурса-отдельной сущности и подходит для случаев, когда фронтенды должны только получать данные от бэкенда, но не добавлять и править их. Пример метаконтроллера, обрабатывающего список рубрик и позволяющего только считывать данные, приведен.

****