Django и REST. Обзор REST, обзор Django Rest Framework

№ урока:	9	**Курс:**	Django Starter

Средства обучения:	Персональный компьютер с установленными: Python 3.8.2 Django 3.0.4

Обзор, цель и назначение урока

Цель данного урока - ознакомиться с основами понятий REST и API. Также на уроке рассматривается какие есть инструменты в Django для построения RESTFul API и как их можно использовать. Будут рассмотрены преимущества RESTFul API и его негативные стороны. В конце урока слушатели смогут на практике научиться устанавливать все инструменты для работы с RESTFul API и создать своё первое приложение с архитектурой REST.

Изучив материал данного занятия, учащийся сможет:

- Настроить проект для работы с REST API.
- Создать своё первое приложение с архитектурой REST.
- Объяснить, какие есть преимущества и недостатки в REST.
- Объяснить, что такое REST и API.

Содержание урока

1) Архитектурный стиль REST
2) Что такое RESTful API
3) История возникновения
4) Каким требованиям отвечает RESTful API
5) Преимущества RESTful API
6) Обзор инструментов для реализации REST в Django
7) Установка и подключение

Резюме

- **REST** (от англ. Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределенного приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределенной гипермедиа-системы. В определенных случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

- REST является альтернативой RPC.

- **Удалённый вызов процедур**, реже Вызов удалённых процедур (от англ. Remote Procedure Call, RPC) — класс технологий, позволяющих компьютерным программам вызывать функции или процедуры в другом адресном пространстве (на удалённых компьютерах, либо в независимой сторонней системе на том же устройстве).

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- В сети Интернет вызов удаленной процедуры может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

- Для веб-служб, построенных с учетом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

- **REST** - набор архитектурных принципов построения сервис-ориентированных систем. RESTful - прилагательное, употребляющееся по отношению к сервисам, которые следуют принципам REST.

- В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует «официального» стандарта для RESTful веб-API. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют такие стандарты, как HTTP, URL, JSON и XML.

- **SOAP** (от англ. Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределенной вычислительной среде. Первоначально SOAP предназначался в основном для реализации удаленного вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

- Хотя данная концепция лежит в самой основе Всемирной паутины, термин «REST» был введён Роем Филдингом (англ. Roy Fielding), одним из создателей протокола «HTTP», лишь в 2000 году. В своей диссертации «Архитектурные стили и дизайн сетевых программных архитектур» («Architectural Styles and the Design of Network-based Software Architectures») в Калифорнийском университете в Ирвайне он подвел теоретическую основу под способ взаимодействия клиентов и серверов во Всемирной паутине, абстрагировав его и назвав «передачей представительного состояния» («Representational State Transfer»). Филдинг описал концепцию построения распределенного приложения, при которой каждый запрос (REST-запрос) клиента к серверу содержит в себе исчерпывающую информацию о желаемом ответе сервера (желаемом представительном состоянии), и сервер не обязан сохранять информацию о состоянии клиента («клиентской сессии»).

- Стиль «REST» развивался параллельно с «HTTP 1.1», разработанным в 1996—1999 годах, основываясь на существующем дизайне «HTTP 1.0», разработанном в 1996 году.

- Свойства архитектуры, которые зависят от ограничений, наложенных на REST-системы:
    - Производительность — взаимодействие компонентов системы может являться доминирующим фактором производительности и эффективности сети с точки зрения пользователя;
    - Масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов.

- Рой Филдинг — один из главных авторов спецификации протокола HTTP, описывает влияние архитектуры REST на масштабируемость следующим образом:
    - Простота унифицированного интерфейса;
    - Открытость компонентов к возможным изменениям для удовлетворения изменяющихся потребностей (даже при работающем приложении);
    - Прозрачность связей между компонентами системы для сервисных служб;
    - Переносимость компонентов системы, путем перемещения программного кода вместе с данными;

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Надежность, выражающаяся в устойчивости к отказам на уровне системы при наличии отказов отдельных компонентов, соединений или данных.

- **Критерии RESTFul**:
  - Client-Server. Система должна быть разделена на клиентов и на серверов. Разделение интерфейсов означает, что, например, клиенты не связаны с хранением данных, которое остается внутри каждого сервера, так что мобильность кода клиента улучшается. Серверы не связаны с интерфейсом пользователя или состоянием, так что серверы могут быть проще и масштабируемы. Серверы и клиенты могут быть заменяемы и разрабатываться независимо, пока интерфейс не изменяется.
  - Stateless. Сервер не должен хранить какой-либо информации о клиентах. В запросе должна храниться вся необходимая информация для обработки запроса и, если необходимо, идентификации клиента.
  - Cache. Каждый ответ должен быть отмечен - является ли он кэшируемым или нет, для предотвращения повторного использования клиентами устаревших или некорректных данных в ответ на дальнейшие запросы.
  - Uniform Interface. Единый интерфейс определяет интерфейс между клиентами и серверами. Это упрощает и отделяет архитектуру, которая позволяет каждой части развиваться самостоятельно.
  - Layered System. В REST допускается разделить систему на иерархию слоев, но с условием, что каждый компонент может видеть компоненты только непосредственно следующего слоя. Например, если вы вызывайте службу PayPal, а он в свою очередь вызывает службу Visa, вы о вызове службы Visa ничего не должны знать.
  - Code-On-Demand (опционально). В REST позволяется загрузка и выполнение кода или программы на стороне клиента.

- Четыре принципа единого интерфейса:
  - Identification of resources (основан на ресурсах). В REST ресурсом является все то, чему можно дать имя. Например, пользователь, изображение, предмет (майка, голодная собака, текущая погода) и т.д. Каждый ресурс в REST должен быть идентифицирован посредством стабильного идентификатора, который не меняется при изменении состояния ресурса. Идентификатором в REST является URI.
  - Manipulation of resources through representations. (Манипуляции над ресурсами через представления). Представление в REST используется для выполнения действий над ресурсами. Представление ресурса представляет собой текущее или желаемое состояние ресурса. Например, если ресурсом является пользователь, то представлением может являться XML или HTML описание этого пользователя.
  - Self-descriptive messages (само-документируемые сообщения). Под само-описательностью имеется ввиду, что запрос и ответ должны хранить в себе всю необходимую информацию для их обработки. Не должно быть дополнительных сообщений или кэшей для обработки одного запроса. Другими словами - отсутствие состояния, сохраняемого между запросами к ресурсам. Это очень важно для масштабирования системы.
  - HATEOAS (hypermedia as the engine of application state). Статус ресурса передается через содержимое body, параметры строки запроса, заголовки запросов и запрашиваемый URI (имя ресурса). Это называется гипермедиа (или гиперссылки с гипертекстом). HATEOAS также означает, что, в случае необходимости ссылки могут содержаться в теле ответа (или заголовках) для поддержки URI, извлечения самого объекта или запрошенных объектов.

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Филдинг указывал, что приложения, не соответствующие приведенным условиям, не могут называться REST-приложениями. Если же все условия соблюдены, то, по его мнению, приложение получит следующие преимущества:
    - Надёжность (за счёт отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
    - Производительность (за счёт использования кэша);
    - Масштабируемость;
    - Прозрачность системы взаимодействия (особенно необходимая для приложений обслуживания сети);
    - Простота интерфейсов;
    - Портативность компонентов;
    - Легкость внесения изменений;
    - Способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины).

- Проблемы связанные с REST:
    - До сих пор нет общего согласования того, что такое RESTful API
    - Словарь REST поддерживается не полностью
    - Словарь REST недостаточно насыщен
    - RESTful API трудно дебажить;
    - RESTful API привязаны к протоколу HTTP.

- Django Rest Framework (DRF) — это библиотека, которая работает со стандартными моделями Django для создания гибкого и мощного API для проекта.

- API DRF состоит из 3-х слоев: сериализатора, вида и маршрутизатора.
    - Сериализатор: преобразует информацию, хранящуюся в базе данных и определенную с помощью моделей Django, в формат, который легко и эффективно передается через API.
    - Вид (ViewSet): определяет функции (чтение, создание, обновление, удаление), которые будут доступны через API.
    - Маршрутизатор: определяет URL-адреса, которые будут предоставлять доступ к каждому виду.

- Модели Django интуитивно представляют данные, хранящиеся в базе, но API должен передавать информацию в менее сложной структуре. Хотя данные будут представлены как экземпляры классов Model, их необходимо перевести в формат JSON для передачи через API.

- Сериализатор DRF производит преобразование в формат JSON для передачи через API. Когда пользователь передает информацию (например, создание нового экземпляра) через API, сериализатор берет данные, проверяет их и преобразует в нечто, что Django может сложить в экземпляр модели. Аналогичным образом, когда пользователь обращается к информации через API, соответствующие экземпляры передаются в сериализатор, который преобразовывает их в формат, который может быть легко передан пользователю как JSON.

- Настройки fields позволяют точно указать, какие поля доступны этому сериализатору. В качестве альтернативы, может быть установлен exclude вместо fields, которое будет включать все поля модели, кроме тех, которые указаны в exclude.

- Сериализаторы — это невероятно гибкий и мощный компонент DRF. Хотя подключение сериализатора к модели является наиболее распространенным, сериализаторы могут

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

использоваться для создания любой структуры данных Python через API в соответствии с определенными параметрами.

- Сериализатор анализирует информацию в обоих направлениях (чтение и запись), тогда как ViewSet — это тот код, в котором определены доступные операции.

- Наиболее распространенным ViewSet является ModelViewSet, который имеет следующие встроенные операции:
    - Создание экземпляра: create ()
    - Получение / чтение экземпляра: retrieve ()
    - Обновление экземпляра (все или только выбранные поля): update () или partial_update ()
    - Уничтожение / Удаление экземпляра: destroy ()
    - Список экземпляров (с разбивкой по страницам по умолчанию): list ()

- И наконец, маршрутизаторы: они предоставляют верхний уровень API. Чтобы избежать создания бесконечных URL-адресов вида: «списки», «детали» и «изменить», маршрутизаторы DRF объединяют все URL-адреса, необходимые для данного вида в одну строку для каждого ViewSet.

Закрепление материала

- Что такое rest?
- Какая разница между rest и restful?
- Какие принципы restful?
- Какие есть недостатки в restful?
- Что такое сериализация?
- Какая библиотека в python используется для создания rest api приложения? Как её подключить к Django приложению?

Дополнительное задание

Задание
Подключить к приложению Django Rest Framework.

Самостоятельная деятельность учащегося

Задание 1
Изучить и понять все преимущества и недостатки от применения инструментов, которые были рассмотрены на уроке.

Задание 2
Создать простое view с помощью DRF, которое будет отвечать на запрос PING ответом PONG. Подход и способы выбрать на своё усмотрение.

Задание 3
Изучить как работает данный API (https://openweathermap.org/api). Рассмотреть документацию, какие там есть запросы, какие есть способы вытянуть данные из него и какая структура данных. Чтобы бы вы изменили, удалили или добавили?

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Рекомендуемые ресурсы

Официальная документация Django:
https://www.djangoproject.com/

API для рассмотрения:
https://openweathermap.org/api

Django Rest Framework документация:
https://www.django-rest-framework.org/

PostgreSQL документация:
https://www.postgresql.org/download/windows/

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com