**Тема занятия № 36: Разработка Web-служб REST. Django REST Framework**

1. Restfull Api. Паттерны и структуры backend-сервисов

REST означает REpresentational State Transfer (Википедия: «передача состояния представления»). Это популярный архитектурный подход для создания API в современном мире.

Вы изучите:

* Что такое REST?
* На чем основан REST API?
* Как используется HTTP при создании REST API?
* Что такое ресурс?
* Как вы определяете ресурсы REST API?
* Каковы лучшие практики при разработке REST API?

**Что такое REST?**

REST расшифровывается как REpresentational State Transfer. Это был термин, первоначально введен Роем Филдингом (Roy Fielding), который также был одним из создателей протокола HTTP. Отличительной особенностью сервисов REST является то, что они позволяют наилучшим образом использовать протокол HTTP. Теперь давайте кратко рассмотрим HTTP.

**Протокол HTTP**

Когда вы вводите в браузере URL-адрес, например www.google.com, на сервер отправляется запрос на веб-сайт, идентифицированный URL-адресом.

Затем этот сервер формирует и выдает ответ. Важным является формат этих запросов и ответов. Эти форматы определяются протоколом HTTP — Hyper Text Transfer Protocol.

Когда вы набираете URL в браузере, он отправляет запрос GET на указанный сервер. Затем сервер отвечает HTTP-ответом, который содержит данные в формате HTML — Hyper Text Markup Language. Затем браузер получает этот HTML-код и отображает его на экране.

Допустим, вы заполняете форму, присутствующую на веб-странице, со списком элементов. В таком случае, когда вы нажимаете кнопку «Submit» (Отправить), HTTP-запрос POST отправляется на сервер.

**HTTP и RESTful веб-сервисы**

HTTP обеспечивает базовый уровень для создания веб-сервисов. Поэтому важно понимать HTTP. Вот несколько ключевых абстракций.

**Ресурс**

Ресурс — это ключевая абстракция, на которой концентрируется протокол HTTP. Ресурс — это все, что вы хотите показать внешнему миру через ваше приложение. Например, если мы пишем приложение для управления задачами, экземпляры ресурсов будут следующие:

* Конкретный пользователь
* Конкретная задача
* Список задач

**URI ресурса**

Когда вы разрабатываете RESTful сервисы, вы должны сосредоточить свое внимание на ресурсах приложения. Способ, которым мы идентифицируем ресурс для предоставления, состоит в том, чтобы назначить ему URI — универсальный идентификатор ресурса. Например:

Создать пользователя: POST /users

Удалить пользователя: DELETE /users/1

Получить всех пользователей: GET /users

Получить одного пользователя: GET /users/1

**REST и Ресурсы**

Важно отметить, что с REST вам нужно думать о приложении с точки зрения ресурсов:

Определите, какие ресурсы вы хотите открыть для внешнего мира используйте глаголы, уже определенные протоколом HTTP, для выполнения операций с этими ресурсами.

Вот как обычно реализуется служба REST:

* Формат обмена данными: здесь нет никаких ограничений. JSON — очень популярный формат, хотя можно использовать и другие, такие как XML Транспорт: всегда HTTP. REST полностью построен на основе HTTP.
* Определение сервиса: не существует стандарта для этого, а REST является гибким. Это может быть недостатком в некоторых сценариях, поскольку потребляющему приложению может быть необходимо понимать форматы запросов и ответов. Однако широко используются такие языки определения веб-приложений, как WADL (Web Application Definition Language) и Swagger.

REST фокусируется на ресурсах и на том, насколько эффективно вы выполняете операции с ними, используя HTTP.

**Компоненты HTTP**

HTTP определяет следующую структуру запроса:

* строка запроса (request line) — определяет тип сообщения
* заголовки запроса (header fields) — характеризуют тело сообщения, параметры передачи и прочие сведения
* тело сообщения (body) — необязательное

HTTP определяет следующую структуру ответного сообщения (response):

* строка состояния (status line), включающая код состояния и сообщение о причине
* поля заголовка ответа (header fields)
* дополнительное тело сообщения (body)

**Методы HTTP-запроса**

Метод, используемый в HTTP-запросе, указывает, какое действие вы хотите выполнить с этим запросом. Важные примеры:

GET: получить подробную информацию о ресурсе

POST: создать новый ресурс

PUT: обновить существующий ресурс

DELETE: Удалить ресурс

**Код статуса ответа HTTP**

Код состояния всегда присутствует в ответе HTTP. Типичные примеры:

200 — успех

404 — cтраница не найдена

**Что такое RESTful API?**

RESTful API — это интерфейс, используемые двумя компьютерными системами для безопасного обмена информацией через Интернет. Большинство бизнес-приложений должны взаимодействовать с другими внутренними и сторонними приложениями для выполнения различных задач. Например, чтобы генерировать ежемесячные платежные ведомости, ваша внутренняя бухгалтерская система должна обмениваться данными с банковской системой вашего клиента, чтобы автоматизировать выставление счетов и взаимодействовать с внутренним приложением по учету рабочего времени. RESTful API поддерживают такой обмен информацией, поскольку они следуют безопасным, надежным и эффективным стандартам программного взаимодействия.

REST — не протокол и не стандарт. Это, как уже было сказано, архитектурное ограничение. Чтобы API считался RESTful, он должен соответствовать следующим критериям.

**Единый интерфейс**

Единый интерфейс (uniform interface) строится вокруг ресурсов с уникальными идентификаторами и уникальными URL, а каждая страница реализовывает какие-либо из HTTP-методов — GET, POST, DELETE или UPDATE — все они отвечают за различные манипуляции конкретным ресурсом.

**Архитектура “Клиент-Сервер”**

Сервер анализирует запрос клиента и отправляет ему ответ. Получив ответ от сервера, клиент определяет, как именно информация отобразится у конечных пользователей.

**Нестационарные запросы**

Не стационарность (stateless) означает, что каждый HTTP-запрос полностью изолирован. Сервер не полагается на информацию из предыдущих запросов — клиент отправляет HTTP-запрос, включающий сразу всю необходимую ему информацию. Сервер не хранит контекста между запросами, каждый запрос содержит всю необходимую информацию.

Благодаря вышеперечисленным архитектурным правилам REST API масштабируемый, переносимый и гибкий.

**Что такое Django REST Framework?**

Django — масштабируемый полнофункциональный веб-фреймворк, написанный на Python. Django берет на себя все хлопоты, зачастую связанные с веб-разработкой, такие как безопасность и доступ к базам данных.

Существует множество библиотек для Django, расширяющих его функционал. Одна из них, о которой мы поговорим сегодня, — это Django REST Framework или DRF, которая позволяет сериализовать данные из Django ORM через REST API.

Сериализация — это преобразование таблиц из базы данных в формат JSON.

Руководство написано для новичков, прошлый опыт работы с фреймворком Django не предполагается, так как вы многому научитесь прямо сейчас. Кроме того, в статье не будет обширного описания технических нюансов каждой операции. Рассматривайте руководство в качестве упражнения.