These slides are part of the downloadable resources of ***The Complete Guide to Django REST Framework and Vue JS*** **Udemy course.**

Let’s keep in touch:

►YouTube Channel: https://www.youtube.com/channel/UCxPWtz5N--X3IyYJ13Zr99A?sub_confirmation=1

►Twitter: https://www.twitter.com/pymike00

►GitHub: https://github.com/pymike00

Web APIs

Objectives

Web APIs - Objectives

In this section we are going to build our first API with Django!

We are going to cover all the most important topics that are relevant to understand how REST API work, such as the HTTP protocol, which is the foundation of data communication for the modern Web.

The knowledge and skills that you will acquire in this unit will then allow you to understand how to effectively use Django REST Framework and Vue JS later on in the next sections, giving you a framework to properly understand what is going on as we dive in more complicated concepts!

Web APIs - Objectives

- Understand the concepts of API, Web API and REST
- Familiarity with the most important HTTP methods and Status Codes
- Familiarity with the JSON file format
- How to use the Requests module to interact with Web APIs using Python
- Build simple Web APIs with “pure” Django

Web APIs

Lessons

- APIs, JSON, Endpoints
- REST, HTTP, Status Codes
- The Requests Module
- Our First Django API pt. 1
- Our First Django API pt. 2
- Competency Assessment & Solution

Web APIs - Objectives

Let’s get started!

APIs, JSON, ENDPOINTS

What is an API?

Web APIs - APIs, JSON, ENDPOINTS

API = Application Programming Interface

In general terms, an API is a set of clearly defined methods of communication between different software components.

A good API makes it easier to develop software by providing all the building blocks, which are then put together by developers.

Web APIs - APIs, JSON, ENDPOINTS

An API declares an **interface** that allows us to interact with its logic without having to know what happens “under the hood”.

APIs are everywhere: operating systems, programs, programming languages and software libraries, the web, computer hardware...

Web APIs - APIs, JSON, ENDPOINTS

… therefore, there are different kinds of API!

In this course we are going to talk about **Web APIs**.

You can think about Web APIs as a set of protocols for communication between either different web applications or different parts of the same application, and that allow external clients (such as a smartphone app!) to connect and communicate with our Web App.

Web APIs - APIs, JSON, ENDPOINTS

The most common kind of Web API nowadays is **REST**, and the APIs that we are going to build in this course are going to be **REST APIs**.

We will talk about REST and it's features in the next lesson, now it's important to answer the following questions:

Why would we want to develop an API in the first place?

What are APIs *really* useful for?

Web APIs - APIs, JSON, ENDPOINTS

Let’s take as example Instagram and Twitter!

They both have an official website you can access from the browser, and they also provide applications for smartphones with different Operating Systems, such as Android and iOS, and each one of these applications has **access to the same synchronized contents**.

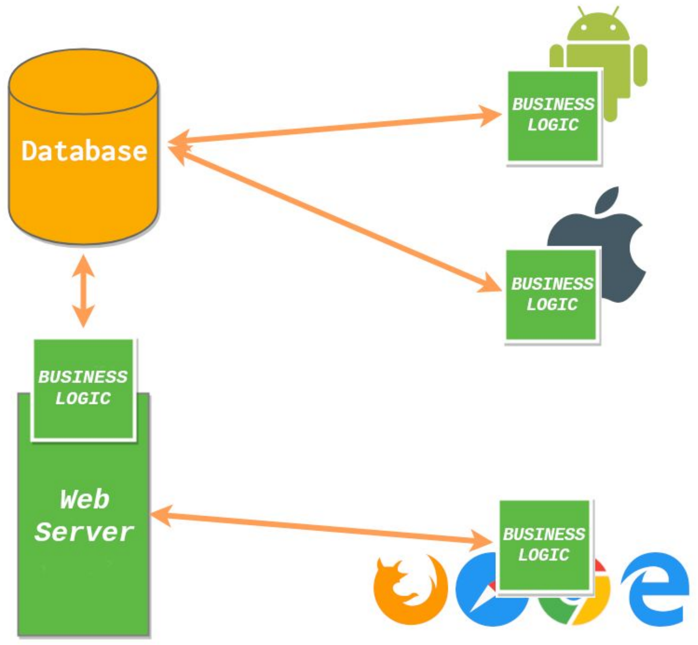
Web APIs - APIs, JSON, ENDPOINTS

Without a Web API, a basic architecture for a similar web service may look like the one shown in this diagram.

Each client app has its own embedded business logic, most likely written with a specific programming language, connecting directly to the database to get and manipulate data.

Maintaining and scaling a web service that uses such an architecture could easily become very challenging, as for each new feature you might want to add to your web service you would have to update each app accordingly.

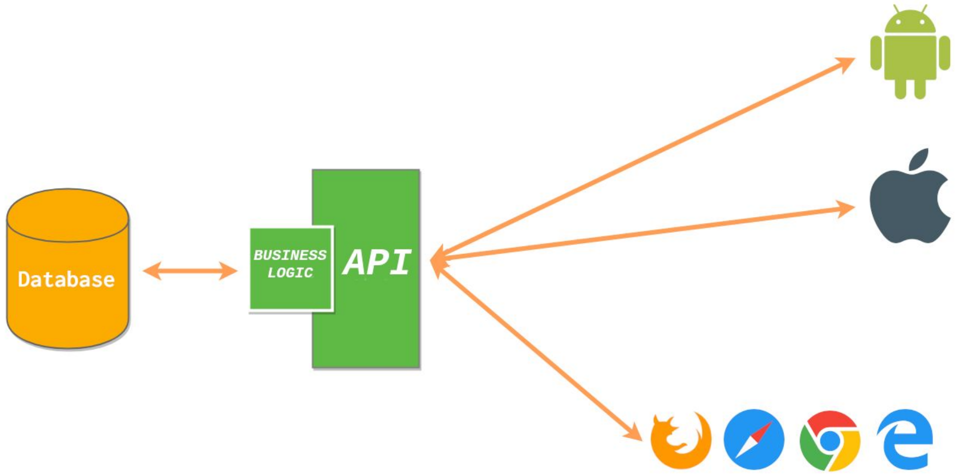
This can be a very expensive and error prone process which often leads to bugs, feature fragmentation and overall project delays. APIs allow us build a much more powerful architecture!



Web APIs - APIs, JSON, ENDPOINTS

Using an **API First approach**, the development process gets much easier and faster.

The API provides a **common interface** that can be used by all the client applications to interact with its own business logic, so that each app has feature parity and if you need to implement a new feature or update, you just make it in one place. The apps themselves will then simply become the UI layer of the service we are providing to our users.



Web APIs - APIs, JSON, ENDPOINTS

With such an architecture it's important for the different parts of our application to communicate with each other using a standard file format, that should be both human readable and fast for computers to process.

One of the most common file formats used by APIs is JSON.

JSON = **J**ava**S**cript **O**bject **N**otation

Key Value pairs surrounded by curly braces

Web APIs - APIs, JSON, ENDPOINTS

```
{
 "created_at": "Thu Jan 19 15:24:15",
 "text": "Here is my New #DRF and #VueJS Course!",
 "user": "pymike00",
 "place": "Italy"
}
```

An example of a simple JSON Object

Web APIs - APIs, JSON, ENDPOINTS

```
{
 "created_at": "Thu Jan 19 15:24:15",
 "text": "Here is my New #DRF and #VueJS Course!",
 "user": "pymike00",
 "place": "Italy",
 "hashtags": ["DRF", "VueJS"]
}
```

A JSON Object containing an array of hashtags

Web APIs - APIs, JSON, ENDPOINTS

```json
{
 "created_at": "Thu Jan 19 15:24:15",
 "text": "Here is my New #DRF and #VueJS Course!",
 "user": {
  "id": 2244994945,
  "name": "pymike00",
  "location": "Italy",
  "description": "Developer."
 },
 "place": "Italy",
 "hashtags": ["DRF", "VueJS"]
}
```

A JSON Object with another JSON Object nested within

Web APIs - APIs, JSON, ENDPOINTS

Now that we've got an idea of what a Web API is and what file format to use with it, we might ask ourselves:

How can we get access and interact with the API?

What sort of channel or access point should we use?

We can access our APIs using **Endpoints**, typically **URL patterns** that we can use to specify what particular information or service we want to access, using the file format that is used by the API.

Web APIs - APIs, JSON, ENDPOINTS

/tweets/

/tweets/536723/

/tweets/536723/hashtags/

/tweets/536723/user/

Web APIs - APIs, JSON, ENDPOINTS

In the next lecture we are going to continue our discussion about APIs, getting to the details of the technical implementation of the specific kind of API that we are going to build throughout the course: **REST APIs**!

This will also give us the opportunity to talk about two very important related topics: **HTTP** and **Status Codes**.

See you there!

APIs, JSON, ENDPOINTS - Useful Links

https://en.wikipedia.org/wiki/Application_programming_interface

https://en.wikipedia.org/wiki/JavaScript_Object_Notation

https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json.html

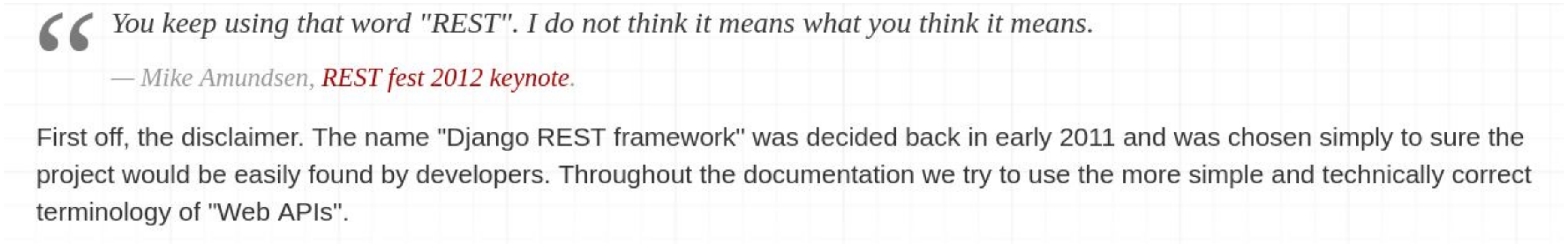
REST, HTTP, STATUS CODES

You can often hear people talking about **REST APIs**, or **RESTful APIs**

What does it mean?

Web APIs - REST, HTTP, STATUS CODES

In the Django REST Framework docs we find this puzzling quote:



> *You keep using that word "REST". I do not think it means what you think it means.*
>
> *— Mike Amundsen, REST fest 2012 keynote.*

First off, the disclaimer. The name "Django REST framework" was decided back in early 2011 and was chosen simply to sure the project would be easily found by developers. Throughout the documentation we try to use the more simple and technically correct terminology of "Web APIs".

Web APIs - REST, HTTP, STATUS CODES

The word **REST** is the acronym of **REpresentational State Transfer**.

It was conceived and used for the first time by an american computer scientist called Roy Fielding in his doctoral thesis.

Fielding is an authority in computer network architecture and he also co-founded the Apache HTTP Web Server!

You can find a link to Fielding's dissertation paper among the other Reference Links for this lecture.

Web APIs - REST, HTTP, STATUS CODES

With **REST**, Fielding describes a specific **architectural pattern** for creating Web APIs that use HTTP as their underlying communication protocol, and that must conform to the following criteria:

- Resources must be accessible via URL Endpoints
- Use of JSON or XML as file format
- Stateless (a request can not depend on any other request!)
- Must use HTTP Verbs to perform actions (GET, POST, PUT, DELETE...)

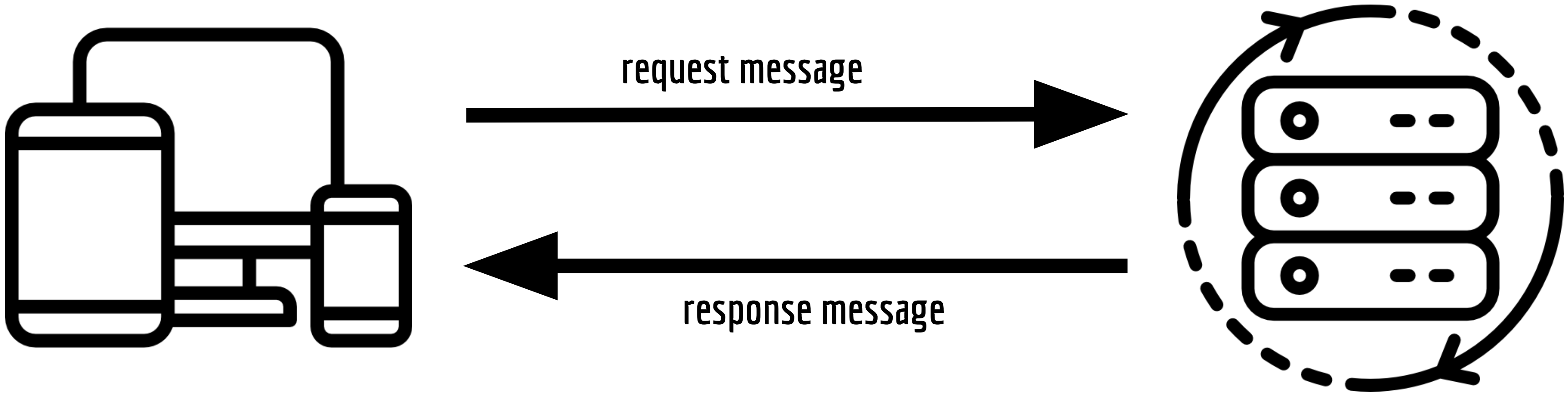
What is HTTP?

Web APIs - REST, HTTP, STATUS CODES

HTTP = **H**yper**T**ext **T**ransfer **P**rotocol

HTTP is one of the most important and popular protocols used to transfer information on the Web, based on a system of request and response messages, in a typical client - server architecture.

The message exchange takes place typically between a browser accessing a web server or a client app accessing a Web API (though client and server can be anything really!)



HTTP Request/Response Cycle

The client and server communicate by sending plain-text messages. The client sends **requests** to the server and the server sends **responses**.

What does a request message look like?

Web APIs - REST, HTTP, STATUS CODES

The request message consists of the following:

- A Request Line (describes the request to implement)
- Request Header Fields (additional information about the request)
- An Empty Line (signals that the meta-information have been sent)
- An optional Message Body

Web APIs - REST, HTTP, STATUS CODES

GET http://www.google.com HTTP/1.1

An example of Request Line using the HTTP GET Method. Some people call them Request Start Line

Web APIs - REST, HTTP, STATUS CODES

So far we've seen how Web APIs expose a Web App's functionalities and components via URL Endpoints, so that they can then be accessed by external client apps or by other parts of the same Web App.

Often times these functionalities will also correspond to different types of actions to be performed, such as **retrieving** information about a specific tweet or **creating** a new blog post based on some data passed on to the API (JSON Object!)

We might also want to **update** the information about an event saved on our calendar app, or **delete** the appointment altogether.

Request Messages and REST

Web APIs - REST, HTTP, STATUS CODES

By convention, in a REST API, the HTTP request method that we use will correspond to the kind of action that we want to perform:

- GET: retrieve a resource
- POST: create a new resource
- PUT / PATCH: update a resource
- DELETE: delete a resource

In the context of a REST API, a resource represents a data entity such as a tweet, a blog post, a product...

What does a response message look like?

Web APIs - REST, HTTP, STATUS CODES

Request and Response Messages look very similar.
A Response Message consists of the following:

- A Status Line (contains the request status code - success or failure)
- Request Header Fields (additional information about the request)
- An Empty Line (Signals that the meta-information have been sent)
- An optional Message Body

Web APIs - REST, HTTP, STATUS CODES

HTTP/1.1 200 OK

An example of HTTP status line

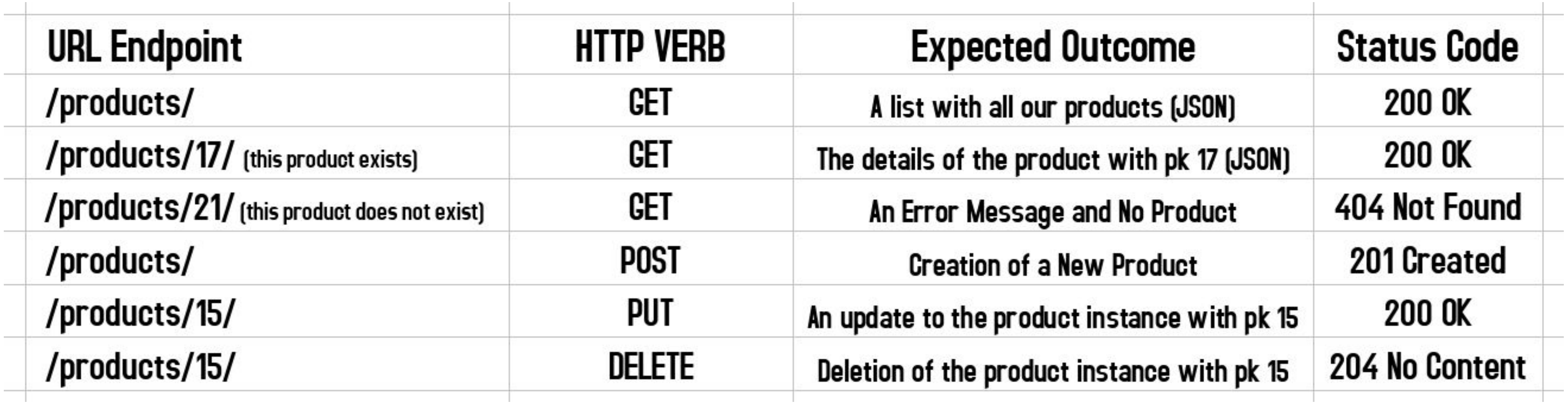
Status Code examples and Status Code Groups

- 200 OK
- 201 Created
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- (...)

- 1xx - Informational
- 2xx - Success
- 3xx - Redirection
- 4xx - Client Error
- 5xx - Server Error

Here are some examples of status codes on the left; On the right the five status code groups.

In a nutshell



URL Endpoint	HTTP VERB	Expected Outcome	Status Code
/products/	GET	A list with all our products (JSON)	200 OK
/products/17/ (this product exists)	GET	The details of the product with pk 17 (JSON)	200 OK
/products/21/ (this product does not exist)	GET	An Error Message and No Product	404 Not Found
/products/	POST	Creation of a New Product	201 Created
/products/15/	PUT	An update to the product instance with pk 15	200 OK
/products/15/	DELETE	Deletion of the product instance with pk 15	204 No Content

Examples of requests to an online store's RESTful API

REST, HTTP, STATUS CODES - Reference Links

https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages

https://stackoverflow.com/questions/13200152/why-is-it-said-that-http-is-a-stateless-protocol

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

https://github.com/for-GET/http-decision-diagram

https://en.wikipedia.org/wiki/Representational_State_Transfer

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

https://www.django-rest-framework.org/topics/rest-hypermedia-hateoas/

The Requests Module

Web APIs - The Requests Module

In this lesson you are going to learn how to use the **Requests Module** to programmatically send HTTP requests using Python!

This will give you the opportunity to get a deeper understanding of all the HTTP related concepts discussed so far during the course, getting ready to build your first API with Django in the next lessons!

Let’s get started!

The Requests Module - Reference Links

http://docs.python-requests.org/en/master/

https://github.com/requests/requests

https://exchangeratesapi.io/

https://github.com/toddmotto/public-apis

DJANGO API pt.1

Web APIs - Django API Pt. 1

In this lesson you are going to learn how to build a simple API using Django, that will expose a list with all the products available in the database of a website which we are also going to create.

As the title suggests, the lesson is divided in two parts!

In this first part we are going to setup our Django Project, with models and config files, making a short recap of how to properly use Django’s “standard” views to show our products in a HTML template.

Web APIs - Django API Pt. 1

In the second part of the lesson we will proceed to the creation of the API itself, by creating the required API Views and Endpoints.

Writing a simple API without Django REST Framework’s support will help us to find out how incredibly useful the package is, understanding all the concepts explained so far in the course with a completely practical example.

We will then be ready to start using Django REST Framework in the next session!

Let’s get started!

DJANGO API pt.2

Our First Web API

Web APIs - Django API Pt. 2

We are going to see how to setup two Endpoints for our API:

1) will allow users to retrieve information about a specific product instance

2) will allow users to retrieve a list of all the available products

As we’ve already mentioned in the first part of this lesson, we are going to build our API Views only using the functions and classes provided by “pure” Django.

Web APIs - Django API Pt. 2

Our API will make use of the JSON file format.

Let’s get started!

Web APIs

Competency Assessment

Web APIs - Competency Assessment

Welcome to this unit’s assessment project!

It’s finally time to test all the new skills acquired so far!

Web APIs - Competency Assessment

For this project you will be creating two new Endpoints for our online store, similar to the ones we have already built.

1) Given a primary key, the first endpoint will allow users to retrieve all the details about a specific manufacturer (products included!)

2) The second endpoint will allow users to retrieve a list of all the available *active* manufacturers in our online store

Web APIs - Competency Assessment

In the next lesson we are going to create and discuss together one of the possible solutions for this assignment.

Happy Coding!

Web APIs

Project Solution

Web APIs - Competency Assessment

You can download all the code for this project's solution from this section's downloadable resources on Udemy, and from my GitHub Profile, at the address **https://github.com/pymike00**