Dependencies in path operation decorators

In some cases you don't really need the return value of a dependency inside your *path operation function*.

Or the dependency doesn't return a value.

But you still need it to be executed/solved.

For those cases, instead of declaring a *path operation function* parameter with `Depends`, you can add a `list` of `dependencies` to the *path operation decorator*.

Add `dependencies` to the *path operation decorator*

The *path operation decorator* receives an optional argument `dependencies`.

It should be a `list` of `Depends()`:

`Python hl_lines="17" {!../../../docs_src/dependencies/tutorial006.py!}`

These dependencies will be executed/solved the same way normal dependencies. But their value (if they return any) won't be passed to your *path operation function*.

!!! tip Some editors check for unused function parameters, and show them as errors.

```
Using these `dependencies` in the *path operation decorator* you can make sure they are exec
```

```
It might also help avoid confusion for new developers that see an unused parameter in your c
```

!!! info In this example we use invented custom headers `X-Key` and `X-Token`.

```
But in real cases, when implementing security, you would get more benefits from using the in
```

Dependencies errors and return values

You can use the same dependency *functions* you use normally.

Dependency requirements

They can declare request requirements (like headers) or other sub-dependencies:

`Python hl_lines="6 11" {!../../../docs_src/dependencies/tutorial006.py!}`

Raise exceptions

These dependencies can `raise` exceptions, the same as normal dependencies:

`Python hl_lines="8 13" {!../../../docs_src/dependencies/tutorial006.py!}`

Return values

And they can return values or not, the values won't be used.

So, you can re-use a normal dependency (that returns a value) you already use somewhere else, and even though the value won't be used, the dependency will be executed:

```
Python hl_lines="9  14" {!../../../docs_src/dependencies/tutorial006.py!}
```

Dependencies for a group of *path operations*

Later, when reading about how to structure bigger applications (Bigger Applications - Multiple Files), possibly with multiple files, you will learn how to declare a single `dependencies` parameter for a group of *path operations*.

Global Dependencies

Next we will see how to add dependencies to the whole `FastAPI` application, so that they apply to each *path operation*.