Advanced Dependencies

Parameterized dependencies

All the dependencies we have seen are a fixed function or class.

But there could be cases where you want to be able to set parameters on the dependency, without having to declare many different functions or classes.

Let's imagine that we want to have a dependency that checks if the query parameter `q` contains some fixed content.

But we want to be able to parameterize that fixed content.

A "callable" instance

In Python there's a way to make an instance of a class a "callable".

Not the class itself (which is already a callable), but an instance of that class.

To do that, we declare a method `__call__`:

```
{!../../../docs_src/dependencies/tutorial011.py!}
```

In this case, this `__call__` is what **FastAPI** will use to check for additional parameters and sub-dependencies, and this is what will be called to pass a value to the parameter in your *path operation function* later.

Parameterize the instance

And now, we can use `__init__` to declare the parameters of the instance that we can use to "parameterize" the dependency:

```
{!../../../docs_src/dependencies/tutorial011.py!}
```

In this case, **FastAPI** won't ever touch or care about `__init__`, we will use it directly in our code.

Create an instance

We could create an instance of this class with:

```
{!../../../docs_src/dependencies/tutorial011.py!}
```

And that way we are able to "parameterize" our dependency, that now has `"bar"` inside of it, as the attribute `checker.fixed_content`.

Use the instance as a dependency

Then, we could use this `checker` in a `Depends(checker)`, instead of `Depends(FixedContentQueryChecker)`, because the dependency is the instance, `checker`, not the class itself.

And when solving the dependency, **FastAPI** will call this `checker` like:

```
checker(q="somequery")
```

...and pass whatever that returns as the value of the dependency in our *path operation function* as the parameter `fixed_content_included`:

```
{!../../../docs_src/dependencies/tutorial011.py!}
```

!!! tip All this might seem contrived. And it might not be very clear how is it useful yet.

```
These examples are intentionally simple, but show how it all works.

In the chapters about security, there are utility functions that are implemented in
this same way.

If you understood all this, you already know how those utility tools for security work
underneath.
```