Return a Response Directly

When you create a **FastAPI** *path operation* you can normally return any data from it: a `dict`, a `list`, a Pydantic model, a database model, etc.

By default, **FastAPI** would automatically convert that return value to JSON using the `jsonable_encoder` explained in JSON Compatible Encoder{.internal-link target=_blank}.

Then, behind the scenes, it would put that JSON-compatible data (e.g. a `dict`) inside of a `JSONResponse` that would be used to send the response to the client.

But you can return a `JSONResponse` directly from your *path operations*.

It might be useful, for example, to return custom headers or cookies.

Return a `Response`

In fact, you can return any `Response` or any sub-class of it.

!!! tip `JSONResponse` itself is a sub-class of `Response`.

And when you return a `Response`, **FastAPI** will pass it directly.

It won't do any data conversion with Pydantic models, it won't convert the contents to any type, etc.

This gives you a lot of flexibility. You can return any data type, override any data declaration or validation, etc.

Using the `jsonable_encoder` in a `Response`

Because **FastAPI** doesn't do any change to a `Response` you return, you have to make sure it's contents are ready for it.

For example, you cannot put a Pydantic model in a `JSONResponse` without first converting it to a `dict` with all the data types (like `datetime`, `UUID`, etc) converted to JSON-compatible types.

For those cases, you can use the `jsonable_encoder` to convert your data before passing it to a response:

```
{!../../../docs_src/response_directly/tutorial001.py!}
```

!!! note "Technical Details" You could also use `from starlette.responses import JSONResponse`.

```
**FastAPI** provides the same `starlette.responses` as `fastapi.responses` just as a convenience for you, the developer. But most of the available responses come directly from Starlette.
```

Returning a custom `Response`

The example above shows all the parts you need, but it's not very useful yet, as you could have just returned the `item` directly, and **FastAPI** would put it in a `JSONResponse` for you, converting it to a `dict`, etc. All that by default.

Now, let's see how you could use that to return a custom response.

Let's say that you want to return an XML response.

You could put your XML content in a string, put it in a `Response`, and return it:

```
{!../../../docs_src/response_directly/tutorial002.py!}
```

Notes

When you return a `Response` directly its data is not validated, converted (serialized), nor documented automatically.

But you can still document it as described in Additional Responses in OpenAPI{.internal-link target=_blank}.

You can see in later sections how to use/declare these custom `Response`s while still having automatic data conversion, documentation, etc.