Extending OpenAPI

!!! warning This is a rather advanced feature. You probably can skip it.

```
If you are just following the tutorial - user guide, you can probably skip this
section.

If you already know that you need to modify the generated OpenAPI schema, continue
reading.
```

There are some cases where you might need to modify the generated OpenAPI schema.

In this section you will see how.

The normal process

The normal (default) process, is as follows.

A `FastAPI` application (instance) has an `.openapi()` method that is expected to return the OpenAPI schema.

As part of the application object creation, a *path operation* for `/openapi.json` (or for whatever you set your `openapi_url`) is registered.

It just returns a JSON response with the result of the application's `.openapi()` method.

By default, what the method `.openapi()` does is check the property `.openapi_schema` to see if it has contents and return them.

If it doesn't, it generates them using the utility function at `fastapi.openapi.utils.get_openapi`.

And that function `get_openapi()` receives as parameters:

- `title`: The OpenAPI title, shown in the docs.
- `version`: The version of your API, e.g. `2.5.0`.
- `openapi_version`: The version of the OpenAPI specification used. By default, the latest: `3.0.2`.
- `description`: The description of your API.
- `routes`: A list of routes, these are each of the registered *path operations*. They are taken from `app.routes`.

Overriding the defaults

Using the information above, you can use the same utility function to generate the OpenAPI schema and override each part that you need.

For example, let's add [ReDoc's OpenAPI extension to include a custom logo](#).

Normal FastAPI

First, write all your **FastAPI** application as normally:

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

Generate the OpenAPI schema

Then, use the same utility function to generate the OpenAPI schema, inside a `custom_openapi()` function:

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

Modify the OpenAPI schema

Now you can add the ReDoc extension, adding a custom `x-logo` to the `info` "object" in the OpenAPI schema:

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

Cache the OpenAPI schema

You can use the property `.openapi_schema` as a "cache", to store your generated schema.

That way, your application won't have to generate the schema every time a user opens your API docs.

It will be generated only once, and then the same cached schema will be used for the next requests.

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

Override the method

Now you can replace the `.openapi()` method with your new function.

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

Check it

Once you go to http://127.0.0.1:8000/redoc you will see that you are using your custom logo (in this example, **FastAPI**'s logo):

Self-hosting JavaScript and CSS for docs

The API docs use **Swagger UI** and **ReDoc**, and each of those need some JavaScript and CSS files.

By default, those files are served from a CDN.

But it's possible to customize it, you can set a specific CDN, or serve the files yourself.

That's useful, for example, if you need your app to keep working even while offline, without open Internet access, or in a local network.

Here you'll see how to serve those files yourself, in the same FastAPI app, and configure the docs to use them.

Project file structure

Let's say your project file structure looks like this:

```
.
├── app
│   ├── __init__.py
│   ├── main.py
```

Now create a directory to store those static files.

Your new file structure could look like this:

```
.
├── app
│   ├── __init__.py
│   ├── main.py
└── static/
```

Download the files

Download the static files needed for the docs and put them on that `static/` directory.

You can probably right-click each link and select an option similar to `Save link as...`.

Swagger UI uses the files:

- `swagger-ui-bundle.js`
- `swagger-ui.css`

And **ReDoc** uses the file:

- `redoc.standalone.js`

After that, your file structure could look like:

```
.
├── app
│   ├── __init__.py
│   ├── main.py
└── static
    ├── redoc.standalone.js
    ├── swagger-ui-bundle.js
    └── swagger-ui.css
```

Serve the static files

- Import `StaticFiles`.
- "Mount" a `StaticFiles()` instance in a specific path.

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

Test the static files

Start your application and go to http://127.0.0.1:8000/static/redoc.standalone.js.

You should see a very long JavaScript file for **ReDoc**.

It could start with something like:

```
/*!
 * ReDoc - OpenAPI/Swagger-generated API Reference Documentation
 * -------------------------------------------------------------
 *   Version: "2.0.0-rc.18"
 *   Repo: https://github.com/Redocly/redoc
 */
!function(e,t){"object"==typeof exports&&"object"==typeof m

...
```

That confirms that you are being able to serve static files from your app, and that you placed the static files for the docs in the correct place.

Now we can configure the app to use those static files for the docs.

Disable the automatic docs

The first step is to disable the automatic docs, as those use the CDN by default.

To disable them, set their URLs to `None` when creating your `FastAPI` app:

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

Include the custom docs

Now you can create the *path operations* for the custom docs.

You can re-use FastAPI's internal functions to create the HTML pages for the docs, and pass them the needed arguments:

- `openapi_url`: the URL where the HTML page for the docs can get the OpenAPI schema for your API. You can use here the attribute `app.openapi_url`.
- `title`: the title of your API.
- `oauth2_redirect_url`: you can use `app.swagger_ui_oauth2_redirect_url` here to use the default.
- `swagger_js_url`: the URL where the HTML for your Swagger UI docs can get the **JavaScript** file. This is the one that your own app is now serving.
- `swagger_css_url`: the URL where the HTML for your Swagger UI docs can get the **CSS** file. This is the one that your own app is now serving.

And similarly for ReDoc...

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

!!! tip The *path operation* for `swagger_ui_redirect` is a helper for when you use OAuth2.

```
If you integrate your API with an OAuth2 provider, you will be able to authenticate
and come back to the API docs with the acquired credentials. And interact with it
using the real OAuth2 authentication.
```

```
Swagger UI will handle it behind the scenes for you, but it needs this "redirect"
helper.
```

Create a *path operation* to test it

Now, to be able to test that everything works, create a *path operation*:

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

Test it

Now, you should be able to disconnect your WiFi, go to your docs at http://127.0.0.1:8000/docs, and reload the page.

And even without Internet, you would be able to see the docs for your API and interact with it.

Configuring Swagger UI

You can configure some extra Swagger UI parameters.

To configure them, pass the `swagger_ui_parameters` argument when creating the `FastAPI()` app object or to the `get_swagger_ui_html()` function.

`swagger_ui_parameters` receives a dictionary with the configurations passed to Swagger UI directly.

FastAPI converts the configurations to **JSON** to make them compatible with JavaScript, as that's what Swagger UI needs.

Disable Syntax Highlighting

For example, you could disable syntax highlighting in Swagger UI.

Without changing the settings, syntax highlighting is enabled by default:

But you can disable it by setting `syntaxHighlight` to `False`:

```
{!../../../docs_src/extending_openapi/tutorial003.py!}
```

...and then Swagger UI won't show the syntax highlighting anymore:

Change the Theme

The same way you could set the syntax highlighting theme with the key `"syntaxHighlight.theme"` (notice that it has a dot in the middle):

```
{!../../../docs_src/extending_openapi/tutorial004.py!}
```

That configuration would change the syntax highlighting color theme:

Change Default Swagger UI Parameters

FastAPI includes some default configuration parameters appropriate for most of the use cases.

It includes these default configurations:

```
{!../../../fastapi/openapi/docs.py[ln:7-13]!}
```

You can override any of them by setting a different value in the argument `swagger_ui_parameters`.

For example, to disable `deepLinking` you could pass these settings to `swagger_ui_parameters`:

```
{!../../../docs_src/extending_openapi/tutorial005.py!}
```

Other Swagger UI Parameters

To see all the other possible configurations you can use, read the official docs for Swagger UI parameters.

JavaScript-only settings

Swagger UI also allows other configurations to be **JavaScript-only** objects (for example, JavaScript functions).

FastAPI also includes these JavaScript-only `presets` settings:

```
presets: [
    SwaggerUIBundle.presets.apis,
    SwaggerUIBundle.SwaggerUIStandalonePreset
]
```

These are **JavaScript** objects, not strings, so you can't pass them from Python code directly.

If you need to use JavaScript-only configurations like those, you can use one of the methods above. Override all the Swagger UI *path operation* and manually write any JavaScript you need.