Request Forms and Files

You can define files and form fields at the same time using `File` and `Form`.

Info

To receive uploaded files and/or form data, first install `python-multipart`.

Make sure you create a virtual environment, activate it, and then install it, for example:

```
$ pip install python-multipart
```

Import `File` and `Form`

Python 3.9+

```
from typing import Annotated

from fastapi import FastAPI, File, Form, UploadFile

app = FastAPI()


@app.post("/files/")
async def create_file(
    file: Annotated[bytes, File()],
    fileb: Annotated[UploadFile, File()],
    token: Annotated[str, Form()],
):
    return {
        "file_size": len(file),
        "token": token,
        "fileb_content_type": fileb.content_type,
    }
```

▸ 🤓 Other versions and variants

Python 3.8+

```
from fastapi import FastAPI, File, Form, UploadFile
from typing_extensions import Annotated

app = FastAPI()


@app.post("/files/")
async def create_file(
    file: Annotated[bytes, File()],
    fileb: Annotated[UploadFile, File()],
    token: Annotated[str, Form()],
):
    return {
        "file_size": len(file),
        "token": token,
        "fileb_content_type": fileb.content_type,
    }
```

Python 3.8+ - non-Annotated

Tip

Prefer to use the `Annotated` version if possible.

```
from fastapi import FastAPI, File, Form, UploadFile

app = FastAPI()


@app.post("/files/")
async def create_file(
    file: bytes = File(), fileb: UploadFile = File(), token: str = Form()
):
    return {
        "file_size": len(file),
        "token": token,
        "fileb_content_type": fileb.content_type,
    }
```

Define `File` and `Form` parameters

Create file and form parameters the same way you would for `Body` or `Query`:

Python 3.9+

```
from typing import Annotated

from fastapi import FastAPI, File, Form, UploadFile

app = FastAPI()


@app.post("/files/")
async def create_file(
    file: Annotated[bytes, File()],
    fileb: Annotated[UploadFile, File()],
    token: Annotated[str, Form()],
):
    return {
        "file_size": len(file),
        "token": token,
        "fileb_content_type": fileb.content_type,
    }
```

▸ 🤓 Other versions and variants

Python 3.8+

```
from fastapi import FastAPI, File, Form, UploadFile
from typing_extensions import Annotated

app = FastAPI()


@app.post("/files/")
async def create_file(
    file: Annotated[bytes, File()],
    fileb: Annotated[UploadFile, File()],
    token: Annotated[str, Form()],
):
    return {
        "file_size": len(file),
        "token": token,
        "fileb_content_type": fileb.content_type,
    }
```

Python 3.8+ - non-Annotated

Tip

Prefer to use the `Annotated` version if possible.

```
from fastapi import FastAPI, File, Form, UploadFile

app = FastAPI()


@app.post("/files/")
async def create_file(
    file: bytes = File(), fileb: UploadFile = File(), token: str = Form()
):
    return {
        "file_size": len(file),
        "token": token,
        "fileb_content_type": fileb.content_type,
    }
```

The files and form fields will be uploaded as form data and you will receive the files and form fields.

And you can declare some of the files as `bytes` and some as `UploadFile`.

Warning

You can declare multiple `File` and `Form` parameters in a *path operation*, but you can't also declare `Body` fields that you expect to receive as JSON, as the request will have the body encoded using `multipart/form-data` instead of `application/json`.

This is not a limitation of **FastAPI**, it's part of the HTTP protocol.

Recap

Use `File` and `Form` together when you need to receive data and files in the same request.