Form Data

When you need to receive form fields instead of JSON, you can use `Form`.

Info

To use forms, first install `python-multipart`.

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

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

Import `Form`

Import `Form` from `fastapi`:

Python 3.9+

```
from typing import Annotated

from fastapi import FastAPI, Form

app = FastAPI()


@app.post("/login/")
async def login(username: Annotated[str, Form()], password: Annotated[str, Form()]):
    return {"username": username}
```

▸ 🤓 Other versions and variants

Python 3.8+

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

app = FastAPI()


@app.post("/login/")
async def login(username: Annotated[str, Form()], password: Annotated[str, Form()]):
    return {"username": username}
```

Python 3.8+ - non-Annotated

Tip

Prefer to use the `Annotated` version if possible.

```
from fastapi import FastAPI, Form

app = FastAPI()


@app.post("/login/")
async def login(username: str = Form(), password: str = Form()):
    return {"username": username}
```

Define `Form` parameters

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

Python 3.9+

```
from typing import Annotated

from fastapi import FastAPI, Form

app = FastAPI()


@app.post("/login/")
async def login(username: Annotated[str, Form()], password: Annotated[str, Form()]):
    return {"username": username}
```

▸ 🤓 Other versions and variants

Python 3.8+

```python
from fastapi import FastAPI, Form
from typing_extensions import Annotated

app = FastAPI()


@app.post("/login/")
async def login(username: Annotated[str, Form()], password: Annotated[str, Form()]):
    return {"username": username}
```

Python 3.8+ - non-Annotated

Tip

Prefer to use the `Annotated` version if possible.

```python
from fastapi import FastAPI, Form

app = FastAPI()


@app.post("/login/")
async def login(username: str = Form(), password: str = Form()):
    return {"username": username}
```

For example, in one of the ways the OAuth2 specification can be used (called "password flow") it is required to send a `username` and `password` as form fields.

The spec requires the fields to be exactly named `username` and `password`, and to be sent as form fields, not JSON.

With `Form` you can declare the same configurations as with `Body` (and `Query`, `Path`, `Cookie`), including validation, examples, an alias (e.g. `user-name` instead of `username`), etc.

Info

`Form` is a class that inherits directly from `Body`.

Tip

To declare form bodies, you need to use `Form` explicitly, because without it the parameters would be interpreted as query parameters or body (JSON) parameters.

About "Form Fields"

The way HTML forms (`<form></form>`) sends the data to the server normally uses a "special" encoding for that data, it's different from JSON.

FastAPI will make sure to read that data from the right place instead of JSON.

Technical Details

Data from forms is normally encoded using the "media type" `application/x-www-form-urlencoded`.

But when the form includes files, it is encoded as `multipart/form-data`. You'll read about handling files in the next chapter.

If you want to read more about these encodings and form fields, head to the MDN web docs for `POST`.

Warning

You can declare multiple `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 `application/x-www-form-urlencoded` instead of `application/json`.

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

Recap

Use `Form` to declare form data input parameters.