

JSON Compatible Encoder

There are some cases where you might need to convert a data type (like a Pydantic model) to something compatible with JSON (like a `dict` , `list` , etc).

For example, if you need to store it in a database.

For that, FastAPI provides a `jsonable_encoder()` function.

Using the `jsonable_encoder`

Let's imagine that you have a database `fake_db` that only receives JSON compatible data.

For example, it doesn't receive `datetime` objects, as those are not compatible with JSON.

So, a `datetime` object would have to be converted to a `str` containing the data in [ISO format](#).

The same way, this database wouldn't receive a Pydantic model (an object with attributes), only a `dict` .

You can use `jsonable_encoder` for that.

It receives an object, like a Pydantic model, and returns a JSON compatible version:

Python 3.10+

```
from datetime import datetime

from fastapi import FastAPI
from fastapi.encoders import jsonable_encoder
from pydantic import BaseModel

fake_db = {}

class Item(BaseModel):
    title: str
    timestamp: datetime
    description: str | None = None

app = FastAPI()

@app.put("/items/{id}")
def update_item(id: str, item: Item):
    json_compatible_item_data = jsonable_encoder(item)
    fake_db[id] = json_compatible_item_data
```

👉 Other versions and variants

Python 3.8+

```
from datetime import datetime
from typing import Union

from fastapi import FastAPI
from fastapi.encoders import jsonable_encoder
from pydantic import BaseModel

fake_db = {}

class Item(BaseModel):
    title: str
    timestamp: datetime
    description: Union[str, None] = None

app = FastAPI()

@app.put("/items/{id}")
def update_item(id: str, item: Item):
    json_compatible_item_data = jsonable_encoder(item)
    fake_db[id] = json_compatible_item_data
```

In this example, it would convert the Pydantic model to a `dict` , and the `datetime` to a `str` .

The result of calling it is something that can be encoded with the Python standard `json.dumps()` .

It doesn't return a large `str` containing the data in JSON format (as a string). It returns a Python standard data structure (e.g. a `dict`) with values and sub-values that are all compatible with JSON.

Note

`jsonable_encoder` is actually used by FastAPI internally to convert data. But it is useful in many other scenarios.

© 2018 Sebastián Ramírez
Licensed under the MIT License.
<https://fastapi.tiangolo.com/tutorial/encoder/>

Exported from DevDocs — <https://devdocs.io>