ボディ - 更新

`PUT` による置換での更新

項目を更新するにはHTTPの `PUT` 操作を使用することができます。

`jsonable_encoder` を用いて、入力データをJSON形式で保存できるデータに変換することができます（例: NoSQLデータベース）。例えば、 `datetime` を `str` に変換します。

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

既存のデータを置き換えるべきデータを受け取るために `PUT` は使用されます。

置換についての注意

つまり、 `PUT` を使用して以下のボディで項目 `bar` を更新したい場合は:

```
{
    "name": "Barz",
    "price": 3,
    "description": None,
}
```

すでに格納されている属性 `"tax": 20.2` を含まないため、入力モデルのデフォルト値は `"tax": 10.5` です。

そして、データはその「新しい」 `10.5` の `tax` と共に保存されます。

`PATCH` による部分的な更新

また、HTTPの `PATCH` 操作でデータを*部分的に*更新することもできます。

つまり、更新したいデータだけを送信して、残りはそのままにしておくことができます。

!!! Note "備考" `PATCH` は `PUT` よりもあまり使われておらず、知られていません。

```
また、多くのチームは部分的な更新であっても`PUT`だけを使用しています。

**FastAPI** はどんな制限も課けていないので、それらを使うのは **自由** です。

しかし、このガイドでは、それらがどのように使用されることを意図しているかを多かれ少なかれ、示しています。
```

Pydanticの `exclude_unset` パラメータの使用

部分的な更新を受け取りたい場合は、Pydanticモデルの `.dict()` の `exclude_unset` パラメータを使用すると非常に便利です。

`item.dict(exclude_unset=True)` のように。

これにより、 `item` モデルの作成時に設定されたデータのみを持つ `dict` が生成され、デフォルト値は除外されます。

これを使うことで、デフォルト値を省略して、設定された（リクエストで送られた）データのみを含む `dict` を生成することができます:

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

Pydanticの `update` パラメータ

ここで、 `.copy()` を用いて既存のモデルのコピーを作成し、 `update` パラメータに更新するデータを含む `dict` を渡すことができます。

`stored_item_model.copy(update=update_data)` のように:

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

部分的更新のまとめ

まとめると、部分的な更新を適用するには、次のようにします:

- (オプションで) `PUT` の代わりに `PATCH` を使用します。
- 保存されているデータを取得します。
- そのデータをPydanticモデルにいれます。
- 入力モデルからデフォルト値を含まない `dict` を生成します（ `exclude_unset` を使用します）。
    - この方法では、モデル内のデフォルト値ですでに保存されている値を上書きするのではなく、ユーザーが実際に設定した値のみを更新することができます。
- 保存されているモデルのコピーを作成し、受け取った部分的な更新で属性を更新します（ `update` パラメータを使用します）。
- コピーしたモデルをDBに保存できるものに変換します（例えば、 `jsonable_encoder` を使用します）。
    - これはモデルの `.dict()` メソッドを再度利用することに匹敵しますが、値をJSONに変換できるデータ型、例えば `datetime` を `str` に変換します。
- データをDBに保存します。
- 更新されたモデルを返します。

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

!!! tip "豆知識" 実際には、HTTPの `PUT` 操作でも同じテクニックを使用することができます。

しかし、これらのユースケースのために作成されたので、ここでの例では`PATCH`を使用しています。

!!! note "備考" 入力モデルがまだ検証されていることに注目してください。

そのため、すべての属性を省略できる部分的な変更を受け取りたい場合は、すべての属性をオプションとしてマークしたモデルを用意する必要があります（デフォルト値または`None`を使用して）。

更新 のためのオプション値がすべて設定されているモデルと、**作成** のための必須値が設定されているモデルを区別するには、[追加モデル](extra-models.md){.internal-link target=_blank}で説明されている考え方を利用することができます。