FastAPIのバージョンについて

FastAPI は既に多くのアプリケーションやシステムに本番環境で使われています。また、100%のテストカバレッジを維持しています。しかし、活発な開発が続いています。

高頻度で新機能が追加され、定期的にバグが修正され、実装は継続的に改善されています。

これが現在のバージョンがいまだに `0.x.x` な理由であり、それぞれのバージョンは破壊的な変更がなされる可能性があります。これは、セマンティック バージョニングの規則に則っています。

FastAPI を使用すると本番用アプリケーションをすぐに作成できますが (すでに何度も経験しているかもしれませんが)、残りのコードが正しく動作するバージョンなのか確認しなければいけません。

`fastapi` のバージョンを固定

最初にすべきことは、アプリケーションが正しく動作する **FastAPI** のバージョンを固定することです。

例えば、バージョン `0.45.0` を使っているとしましょう。

`requirements.txt` を使っているなら、以下の様にバージョンを指定できます:

```
fastapi==0.45.0
```

これは、厳密にバージョン `0.45.0` だけを使うことを意味します。

または、以下の様に固定することもできます:

```
fastapi>=0.45.0,<0.46.0
```

これは `0.45.0` 以上、 `0.46.0` 未満のバージョンを使うことを意味します。例えば、バージョン `0.45.2` は使用可能です。

PoetryやPipenvなど、他のインストール管理ツールを使用している場合でも、それぞれパッケージのバージョンを指定する機能があります。

利用可能なバージョン

Release Notes{.internal-link target=_blank}で利用可能なバージョンが確認できます (現在の最新版の確認などのため)。

バージョンについて

セマンティック バージョニングの規約に従って、 `1.0.0` 未満の全てのバージョンは破壊的な変更が加わる可能性があります。

FastAPIでは「パッチ」バージョンはバグ修正と非破壊的な変更に留めるという規約に従っています。

!!! tip "豆知識" 「パッチ」は最後の数字を指します。例えば、 `0.2.3` ではパッチバージョンは `3` です。

従って、以下の様なバージョンの固定が望ましいです:

```
fastapi>=0.45.0,<0.46.0
```

破壊的な変更と新機能実装は「マイナー」バージョンで加えられます。

!!! tip "豆知識" 「マイナー」は真ん中の数字です。例えば、 `0.2.3` ではマイナーバージョンは `2` です。

FastAPIのバージョンのアップグレード

アプリケーションにテストを加えるべきです。

FastAPI では非常に簡単に実現できます (Starletteのおかげで)。ドキュメントを確認して下さい: テスト{.internal-link target=_blank}

テストを加えた後で、**FastAPI** のバージョンをより最新のものにアップグレードし、テストを実行することで全てのコードが正常に動作するか確認できます。

全てが動作するか、修正を行った上で全てのテストを通過した場合、使用している `fastapi` のバージョンをより最新のバージョンに固定できます。

Starletteについて

`Starlette` のバージョンは固定すべきではありません。

FastAPI は、バージョン毎にStarletteのより新しいバージョンを使用します。

よって、最適なStarletteのバージョン選択を**FastAPI** に任せることができます。

Pydanticについて

Pydanticは自身のテストだけでなく**FastAPI** のためのテストを含んでいます。なので、Pydanticの新たなバージョン (`1.0.0` 以降) は全てFastAPIと整合性があります。

Pydanticのバージョンを、動作が保証できる `1.0.0` 以降のいずれかのバージョンから `2.0.0` 未満の間に固定できます。

例えば:

```
pydantic>=1.2.0,<2.0.0
```