FastAPI 버전들에 대하여

FastAPI 는 이미 많은 응용 프로그램과 시스템들을 만드는데 사용되고 있습니다. 그리고 100%의 테스트 정확성을 가지고 있습니다. 하지만 이것은 아직까지도 빠르게 발전하고 있습니다.

새로운 특징들이 빈번하게 추가되고, 오류들이 지속적으로 수정되고 있습니다. 그리고 코드가 계속적으로 향상되고 있습니다.

이것이 아직도 최신 버전이 `0.x.x` 인 이유입니다. 이것은 각각의 버전들이 잠재적으로 변할 수 있다는 것을 보여줍니다. 이는 유의적 버전 관습을 따릅니다.

지금 바로 **FastAPI**로 응용 프로그램을 만들 수 있습니다. 이때 (아마 지금까지 그래 왔던 것처럼), 사용하는 버전이 코드와 잘 맞는지 확인해야합니다.

`fastapi` 버전을 표시

가장 먼저 해야할 것은 응용 프로그램이 잘 작동하는 가장 최신의 구체적인 **FastAPI** 버전을 표시하는 것입니다.

예를 들어, 응용 프로그램에 `0.45.0` 버전을 사용했다고 가정합니다.

만약에 `requirements.txt` 파일을 사용했다면, 다음과 같이 버전을 명세할 수 있습니다:

```
fastapi==0.45.0
```

이것은 `0.45.0` 버전을 사용했다는 것을 의미합니다.

또는 다음과 같이 표시할 수 있습니다:

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

이것은 `0.45.0` 버전과 같거나 높으면서 `0.46.0` 버전 보다는 낮은 버전을 사용했다는 것을 의미합니다. 예를 들어, `0.45.2` 버전과 같은 경우는 해당 조건을 만족합니다.

만약에 Poetry, Pipenv, 또는 그밖의 다양한 설치 도구를 사용한다면, 패키지에 구체적인 버전을 정의할 수 있는 방법을 가지고 있을 것입니다.

이용가능한 버전들

Release Notes{.internal-link target=_blank}를 통해 사용할 수 있는 버전들을 확인할 수 있습니다.(예를 들어, 가장 최신의 버전을 확인할 수 있습니다.)

버전들에 대해

유의적 버전 관습을 따라서, `1.0.0` 이하의 모든 버전들은 잠재적으로 급변할 수 있습니다.

FastAPI는 오류를 수정하고, 일반적인 변경사항을 위해 "패치"버전의 관습을 따릅니다.

!!! tip "팁" 여기서 말하는 "패치"란 버전의 마지막 숫자로, 예를 들어 `0.2.3` 버전에서 "패치"는 `3` 을 의미합니다.

따라서 다음과 같이 버전을 표시할 수 있습니다:

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

수정된 사항과 새로운 요소들이 "마이너" 버전에 추가되었습니다.

!!! tip "팁" "마이너"란 버전 넘버의 가운데 숫자로, 예를 들어서 `0.2.3` 의 "마이너" 버전은 `2` 입니다.

FastAPI 버전의 업그레이드

응용 프로그램을 검사해야합니다.

(Starlette 덕분에), **FastAPI** 를 이용하여 굉장히 쉽게 할 수 있습니다. Testing{.internal-link target=_blank}문서를 확인해 보십시오:

검사를 해보고 난 후에, **FastAPI** 버전을 더 최신으로 업그레이드 할 수 있습니다. 그리고 코드들이 테스트에 정상적으로 작동하는지 확인을 해야합니다.

만약에 모든 것이 정상 작동하거나 필요한 부분을 변경하고, 모든 검사를 통과한다면, 새로운 버전의 `fastapi` 를 표시할 수 있습니다.

Starlette에 대해

`starlette` 의 버전은 표시할 수 없습니다.

서로다른 버전의 **FastAPI**가 구체적이고 새로운 버전의 Starlette을 사용할 것입니다.

그러므로 **FastAPI**가 알맞은 Starlette 버전을 사용하도록 하십시오.

Pydantic에 대해

Pydantic은 **FastAPI** 를 위한 검사를 포함하고 있습니다. 따라서, 새로운 버전의 Pydantic(`1.0.0` 이상)은 항상 FastAPI와 호환됩니다.

작업을 하고 있는 `1.0.0` 이상의 모든 버전과 `2.0.0` 이하의 Pydantic 버전을 표시할 수 있습니다.

예를 들어 다음과 같습니다:

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