경로 매개변수

파이썬 포맷 문자열이 사용하는 동일한 문법으로 "매개변수" 또는 "변수"를 경로에 선언할 수 있습니다:

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

경로 매개변수 `item_id` 의 값은 함수의 `item_id` 인자로 전달됩니다.

그래서 이 예제를 실행하고 http://127.0.0.1:8000/items/foo로 이동하면, 다음 응답을 볼 수 있습니다:

```
{"item_id":"foo"}
```

타입이 있는 매개변수

파이썬 표준 타입 어노테이션을 사용하여 함수에 있는 경로 매개변수의 타입을 선언할 수 있습니다:

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

지금과 같은 경우, `item_id` 는 `int` 로 선언 되었습니다.

!!! check "확인" 이 기능은 함수 내에서 오류 검사, 자동완성 등을 편집기를 지원합니다

데이터 변환

이 예제를 실행하고 http://127.0.0.1:8000/items/3을 열면, 다음 응답을 볼 수 있습니다:

```
{"item_id":3}
```

!!! check "확인" 함수가 받은(반환도 하는) 값은 문자열 `"3"` 이 아니라 파이썬 `int` 형인 `3` 입니다.

즉, 타입 선언을 하면 **FastAPI**는 자동으로 요청을 <abbr title="HTTP 요청에서 전달되는 문자열을 파이썬 데이터로 변환">"파싱"</abbr>합니다.

데이터 검증

하지만 브라우저에서 http://127.0.0.1:8000/items/foo로 이동하면, 멋진 HTTP 오류를 볼 수 있습니다:

```
{
    "detail": [
        {
            "loc": [
                "path",
                "item_id"
            ],
            "msg": "value is not a valid integer",
            "type": "type_error.integer"
        }
```

```
    ]
}
```

경로 매개변수 `item_id` 는 `int` 가 아닌 `"foo"` 값이기 때문입니다.

`int` 대신 `float` 을 전달하면 동일한 오류가 나타납니다: http://127.0.0.1:8000/items/4.2

!!! check "확인" 즉, 파이썬 타입 선언을 하면 **FastAPI**는 데이터 검증을 합니다.

```
오류는 검증을 통과하지 못한 지점도 정확하게 명시합니다.

이는 API와 상호 작용하는 코드를 개발하고 디버깅하는 데 매우 유용합니다.
```

문서화

그리고 브라우저에서 http://127.0.0.1:8000/docs를 열면, 다음과 같이 자동 대화식 API 문서를 볼 수 있습니다:

!!! check "확인" 다시 한번, 그저 파이썬 타입 선언을 하기만 하면 **FastAPI**는 자동 대화식 API 문서(Swagger UI 통합)를 제공합니다.

```
경로 매개변수는 정수형으로 선언됐음을 주목하세요.
```

표준 기반의 이점, 대체 문서화

그리고 생성된 스키마는 OpenAPI 표준에서 나온 것이기 때문에 호환되는 도구가 많이 있습니다.

이 덕분에 **FastAPI**는 http://127.0.0.1:8000/redoc로 접속할 수 있는 (ReDoc을 사용하는) 대체 API 문서를 제공합니다:

이와 마찬가지로 호환되는 도구가 많이 있습니다. 다양한 언어에 대한 코드 생성 도구를 포함합니다.

Pydantic

모든 데이터 검증은 Pydantic에 의해 내부적으로 수행되므로 이로 인한 모든 이점을 얻을 수 있습니다. 여러분은 관리를 잘 받고 있음을 느낄 수 있습니다.

`str` , `float` , `bool` 과 다른 복잡한 데이터 타입 선언을 할 수 있습니다.

이 중 몇 가지는 자습서의 다음 장에서 살펴봅니다.

순서 문제

*경로 동작*을 만들때 고정 경로를 갖고 있는 상황들을 맞닦뜨릴 수 있습니다.

`/users/me` 처럼, 현재 사용자의 데이터를 가져온다고 합시다.

사용자 ID를 이용해 특정 사용자의 정보를 가져오는 경로 `/users/{user_id}` 도 있습니다.

*경로 동작*은 순차적으로 평가되기 때문에 `/users/{user_id}` 이전에 `/users/me` 를 먼저 선언해야 합니다:

```
{!../../../docs_src/path_params/tutorial003.py!}
```

그렇지 않으면 `/users/{user_id}` 는 매개변수 `user_id` 의 값을 `"me"` 라고 "생각하여" `/users/me` 도 연결합니다.

사전정의 값

만약 *경로 매개변수*를 받는 *경로 동작*이 있지만, 유효하고 미리 정의할 수 있는 *경로 매개변수* 값을 원한다면 파이썬 표준 `Enum` 을 사용할 수 있습니다.

`Enum` 클래스 생성

`Enum` 을 임포트하고 `str` 과 `Enum` 을 상속하는 서브 클래스를 만듭니다.

`str` 을 상속함으로써 API 문서는 값이 `string` 형이어야 하는 것을 알게 되고 제대로 렌더링 할 수 있게 됩니다.

고정값으로 사용할 수 있는 유효한 클래스 어트리뷰트를 만듭니다:

```
{!../../../docs_src/path_params/tutorial005.py!}
```

!!! info "정보" 열거형(또는 enums)은 파이썬 버전 3.4 이후로 사용가능합니다.

!!! tip "팁" 혹시 헷갈린다면, "AlexNet", "ResNet", 그리고 "LeNet"은 그저 기계 학습 모델들의 이름입니다.

경로 매개변수 선언

생성한 열거형 클래스(`ModelName`)를 사용하는 타입 어노테이션으로 *경로 매개변수*를 만듭니다:

```
{!../../../docs_src/path_params/tutorial005.py!}
```

문서 확인

*경로 매개변수*에 사용할 수 있는 값은 미리 정의되어 있으므로 대화형 문서에서 멋지게 표시됩니다:

파이썬 *열거형*으로 작업하기

*경로 매개변수*의 값은 *열거형 멤버*가 됩니다.

열거형 멤버 비교

열거체 `ModelName` 의 *열거형 멤버*를 비교할 수 있습니다:

```
{!../../../docs_src/path_params/tutorial005.py!}
```

열거형 값 가져오기

`model_name.value` 또는 일반적으로 `your_enum_member.value` 를 이용하여 실제값(지금의 경우 `str`)을 가져올 수 있습니다:

```
{!../../../docs_src/path_params/tutorial005.py!}
```

!!! tip "팁" `ModelName.lenet.value` 로도 값 `"lenet"` 에 접근할 수 있습니다.

열거형 멤버 반환

*경로 동작*에서 중첩 JSON 본문(예: `dict`) 역시 *열거형 멤버*를 반환할 수 있습니다.

클라이언트에 반환하기 전에 해당 값(이 경우 문자열)으로 변환됩니다:

```
{!../../../docs_src/path_params/tutorial005.py!}
```

클라이언트는 아래의 JSON 응답을 얻습니다:

```
{
  "model_name": "alexnet",
  "message": "Deep Learning FTW!"
}
```

경로를 포함하는 경로 매개변수

`/files/{file_path}` 가 있는 *경로 동작*이 있다고 해봅시다.

그런데 여러분은 `home/johndoe/myfile.txt` 처럼 *path*에 들어있는 `file_path` 자체가 필요합니다.

따라서 해당 파일의 URL은 다음처럼 됩니다: `/files/home/johndoe/myfile.txt` .

OpenAPI 지원

테스트와 정의가 어려운 시나리오로 이어질 수 있으므로 OpenAPI는 *경로*를 포함하는 *경로 매개변수*를 내부에 선언하는 방법을 지원하지 않습니다.

그럼에도 Starlette의 내부 도구중 하나를 사용하여 **FastAPI**에서는 할 수 있습니다.

매개변수에 경로가 포함되어야 한다는 문서를 추가하지 않아도 문서는 계속 작동합니다.

경로 변환기

Starlette에서 직접 옵션을 사용하면 다음과 같은 URL을 사용하여 *path*를 포함하는 *경로 매개변수*를 선언 할 수 있습니다:

```
/files/{file_path:path}
```

이러한 경우 매개변수의 이름은 `file_path` 이고 마지막 부분 `:path` 는 매개변수가 *경로*와 일치해야함을 알려줍니다.

그러므로 다음과 같이 사용할 수 있습니다:

```
{!../../../docs_src/path_params/tutorial004.py!}
```

!!! tip "팁" 매개변수가 `/home/johndoe/myfile.txt` 를 갖고 있어 슬래시로 시작(`/`)해야 할 수 있습니다.

이 경우 URL은: `/files//home/johndoe/myfile.txt`이며 `files`과 `home` 사이에 이중 슬래시(`//`)가 생깁니다.

요약

FastAPI과 함께라면 짧고 직관적인 표준 파이썬 타입 선언을 사용하여 다음을 얻을 수 있습니다:

- 편집기 지원: 오류 검사, 자동완성 등
- 데이터 "파싱"
- 데이터 검증
- API 주석(Annotation)과 자동 문서

위 사항들을 그저 한번에 선언하면 됩니다.

이는 (원래 성능과는 별개로) 대체 프레임워크와 비교했을 때 **FastAPI**의 주요 가시적 장점일 것입니다.