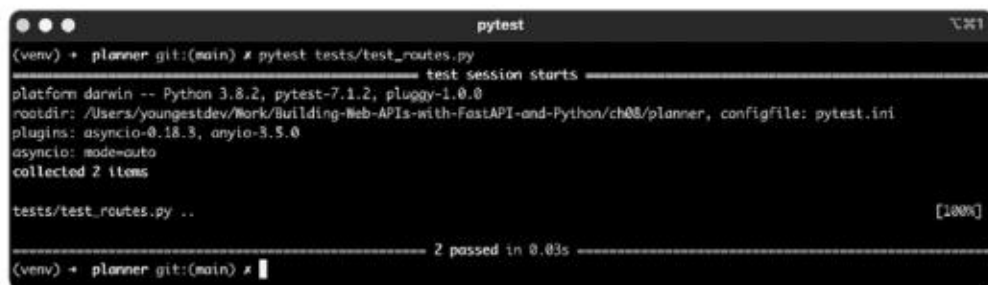


```
assert response.status_code == 200
assert response.json()["creator"] == mock_event.creator
assert response.json()["_id"] == str(mock_event.id)
```

이 코드는 단일 이벤트를 추출하는 데 사용되는 라우트를 테스트한다. 전달된 이벤트 ID는 `mock_event` 픽스처에서 추출되며 실행한 요청의 결과를 `mock_event` 픽스처에 저장된 데이터와 비교한다. 다음 명령을 사용해 테스트를 실행해보자.

```
(venv)$ pytest tests/test_routes.py
```

실행 결과는 [그림 8-7]과 같다.



```
pytest
(venv) + planner git:(main) x pytest tests/test_routes.py
test session starts
platform darwin -- Python 3.8.2, pytest-7.1.2, pluggy-1.0.0
rootdir: /Users/youngestdev/Work/Building-Web-APIs-with-FastAPI-and-Python/ch08/planner, configfile: pytest.ini
plugins: asyncio-0.18.3, anyio-3.5.0
asyncio: mode=auto
collected 2 items

tests/test_routes.py ..                                     [100%]

----- 2 passed in 0.03s -----
(venv) + planner git:(main) x
```

그림 8-7 성공한 단일 이벤트 추출 테스트

이어서 신규 이벤트 생성 테스트를 작성해보자.

## 생성 라우트 테스트

앞서 만든 픽스처를 사용해 접속 토큰을 추출하고 테스트 함수를 정의한다. 이 함수는 서버로 전송될 요청 페이로드를 생성한다. 요청 페이로드에는 콘텐츠 유형과 인증 헤더가 포함된다. 테스트 응답도 정의되는데 요청이 실행되면 실제 결과와 응답이 비교된다. 다음과 같이 코드를 추가하자.

```
@pytest.mark.asyncio
async def test_post_event(default_client: httpx.AsyncClient, access_token: str) ->
    None:
```