🔧 API 라우터 연결 트러블슈팅 가이드

> **작성일:** 2025-11-07
> **소요 시간:** 약 8시간
> **최종 결과:** 모든 API 엔드포인트 정상 작동 및 전체 분류 파이프라인 완성

---

1. 문제 상황 분석

1.1 발생 오류

```
curl -X POST "http://localhost:8000/api/onboarding/step2?
user_id=test_user&keywords=work,family,health"
{"detail":"Not Found"}
```

HTTP 404 Not Found - 엔드포인트를 찾을 수 없는 상황

1.2 근본 원인

FastAPI 라우터 등록 불일치:

- ✅ 라우터 파일 `onboarding_routes.py` 존재
- ❌ `main.py`에서 라우터를 `include_router()`로 등록하지 않음
- ❌ 쿼리 파라미터가 `Query()` 로 명시되지 않아 body로 인식

```
# ❌ 문제 상황 분석

main.py 상태:
├─ classifier_router 등록 (✅)
├─ api_router 등록 (✅)
└─ onboarding_router 미등록 (❌) ← 라우터가 "보이지 않음"

onboarding_routes.py 상태:
├─ APIRouter 생성 (✅)
├─ step2 엔드포인트 정의 (✅)
└─ Query 파라미터 미명시 (❌) ← 파라미터가 "인식 안 됨"

결과: 404 Not Found
```

1.3 시도한 실패 방법들

시도 1: 라우터 prefix 중복 설정 (실패) ❌

```
    # onboarding_routes.py
    router = APIRouter(prefix="/api/onboarding", tags=["onboarding"])
```

```
# main.py
app.include_router(onboarding_router, prefix="/api/onboarding")

# 결과: /api/onboarding/api/onboarding/step2 (경로 중복)
```

시도 2: Query 파라미터 명시 생략 (실패) ❌

```
# ❌ 틀린 방법
@router.post("/step2")
async def onboarding_step2(
    user_id: str,           # FastAPI가 어디서 받을지 불명확
    keywords: str           # body? query? path?
):

# 결과: 쿼리 문자열 무시, body 요청으로 변환
# → HTTP 422 Unprocessable Entity (또는 404)
```

---

2. 핵심 솔루션 (세 가지 수정 사항)

2.1 Step 1: onboarding_routes.py 수정

```
# backend/routes/onboarding_routes.py

from fastapi import APIRouter, Query, HTTPException  # ← Query import 필수!

# ✅ prefix 제거 (main.py에서 설정할 것)
router = APIRouter(tags=["onboarding"])

@router.post("/step1")
async def onboarding_step1(input_data: Step1Input):
    """Step 1: 사용자 직업 입력"""
    try:
        user_id = str(uuid.uuid4())
        # ... 로직
        return {
            "status": "success",
            "user_id": user_id,
            "occupation": input_data.occupation
        }
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))

# ✅ 핵심: Query() 명시!
@router.post("/step2")
async def onboarding_step2(
    user_id: str = Query(...),      # ← Query(...) 필수!
```

```
    keywords: str = Query(...)      # ← Query(...) 필수!
):
    """Step 2: 사용자 키워드 저장"""
    try:
        keyword_list = keywords.split(",")
        return {
            "status": "success",
            "user_id": user_id,
            "keywords": keyword_list
        }
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

2.2 Step 2: main.py 라우터 등록

```
# backend/main.py

from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
import logging

# ✅ 라우터 import
from backend.routes.api_routes import router as api_router
from backend.routes.classifier_routes import router as
classifier_router
from backend.routes.onboarding_routes import router as
onboarding_router

logger = logging.getLogger(__name__)

app = FastAPI()

# CORS 설정
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# ✅ 라우터 등록 (일관된 prefix 방식)
app.include_router(api_router)
logger.info("✅ api_router 등록 완료")

app.include_router(classifier_router, prefix="/api/classify")
logger.info("✅ classifier_router 등록 완료")

# ✅ 핵심: onboarding_router 등록!
app.include_router(onboarding_router, prefix="/api/onboarding")
logger.info("✅ onboarding_router 등록 완료")
```

```
    @app.get("/health")
    async def health():
        return {"status": "healthy", "timestamp":
datetime.now().isoformat()}
```

2.3 동작 메커니즘

```
    FastAPI 라우팅 프로세스:
    ──────────────────────────

    1️⃣ 요청 도착
    POST http://localhost:8000/api/onboarding/step2?
user_id=test_user&keywords=work,family,health

    2️⃣ main.py에서 prefix 매칭
    include_router(onboarding_router, prefix="/api/onboarding")
        ↓
    /api/onboarding 으로 시작하는 요청 찾음 ✅

    3️⃣ onboarding_routes.py에서 경로 매칭
    @router.post("/step2")
        ↓
    /step2 엔드포인트 찾음 ✅
    (최종 경로: /api/onboarding + /step2 = /api/onboarding/step2)

    4️⃣ Query 파라미터 파싱
    @router.post("/step2")
    async def onboarding_step2(
        user_id: str = Query(...),
        keywords: str = Query(...)
    )
        ↓
    쿼리 문자열에서 user_id, keywords 추출 ✅

    5️⃣ 함수 실행 및 응답 반환
    {
        "status": "success",
        "user_id": "test_user",
        "keywords": ["work", "family", "health"]
    }
```

---

3. 실전 적용

3.1 파일 구조 확인

```
backend/
├── routes/
│   ├── __init__.py
│   ├── api_routes.py           (기존 라우터)
│   ├── classifier_routes.py    (분류 라우터)
│   └── onboarding_routes.py    ✅ (신규 라우터)
├── main.py                     ✅ (수정 필요)
└── ...
```

3.2 핵심 수정 체크리스트

항목	상태	확인 사항
onboarding_routes.py	🔧 수정	`from fastapi import Query` 추가
		`router = APIRouter(tags=[...])` prefix 제거
		`@router.post()` 함수에 `Query(...)` 명시
main.py	🔧 수정	`onboarding_router` import 추가
		`app.include_router(onboarding_router, prefix="/api/onboarding")` 추가
테스트	✅ 확인	`curl` 또는 Postman으로 검증

3.3 완벽한 구현 코드 (통합)

```python
# backend/routes/onboarding_routes.py
from fastapi import APIRouter, Query, HTTPException
from pydantic import BaseModel
from datetime import datetime
import uuid
import json
import os
from backend.data_manager import DataManager

router = APIRouter(tags=["onboarding"])  # ← prefix 없음!
data_manager = DataManager()

class Step1Input(BaseModel):
    occupation: str
    name: str = "Anonymous"

class OnboardingStatus(BaseModel):
    user_id: str
    occupation: str
    areas: list[str]
```

```
    is_completed: bool

@router.post("/step1")
async def onboarding_step1(input_data: Step1Input):
    """Step 1: 사용자 직업 정보 수집"""
    try:
        user_id = str(uuid.uuid4())

        # 데이터 저장
        data_manager.save_user_profile({
            "user_id": user_id,
            "occupation": input_data.occupation,
            "name": input_data.name,
            "created_at": datetime.now().isoformat()
        })

        return {
            "status": "success",
            "user_id": user_id,
            "occupation": input_data.occupation,
            "message": "Step 1 완료. Step 2로 진행하세요."
        }
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))

@router.post("/step2")
async def onboarding_step2(
    user_id: str = Query(...),          # ✅ Query 필수!
    keywords: str = Query(...)          # ✅ Query 필수!
):
    """Step 2: 사용자 관심사 키워드 저장"""
    try:
        keyword_list = [k.strip() for k in keywords.split(",")]

        # 데이터 저장
        data_manager.save_user_keywords({
            "user_id": user_id,
            "keywords": keyword_list,
            "saved_at": datetime.now().isoformat()
        })

        return {
            "status": "success",
            "user_id": user_id,
            "keywords": keyword_list,
            "message": "온보딩 완료!"
        }
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))

@router.get("/status/{user_id}")
async def get_onboarding_status(user_id: str):
    """사용자 온보딩 상태 조회"""
    try:
```

```
            profile = data_manager.get_user_profile(user_id)
            return {
                "status": "success",
                "data": profile
            }
        except Exception as e:
            raise HTTPException(status_code=404, detail="사용자 정보 없음")
```

---

4. 핵심 교훈

4.1 FastAPI 라우팅 원칙

원칙	설명	결과
Prefix 관리	라우터 생성 시 prefix 안 함, main.py에서 관리	중복 방지 ✅
Parameter 명시	Query/Body/Path 명시적 표기	파라미터 정확히 인식 ✅
로거 활용	라우터 등록 시 로그 남기기	디버깅 용이 ✅
모듈화	각 기능별 별도 라우터 파일	유지보수성 ⬆️

4.2 기술적 인사이트

- **FastAPI의 prefix 동작**: `prefix` + 라우터 내 경로 = 최종 URL
- **Query 파라미터**: `Query(...)` 또는 기본값 필수
- **Dependency Injection**: 각 엔드포인트는 독립적으로 작동

---

5. 성공 지표

```
✅ HTTP 200 OK 응답
✅ /api/onboarding/step1 작동
✅ /api/onboarding/step2 작동 (쿼리 파라미터 정상)
✅ /api/classify/para 작동
✅ /api/classify/keywords 작동
✅ /health 작동
✅ 전체 분류 파이프라인 완성
✅ JSON 응답 형식 일관성
```

---

6. 디버깅 팁

6.1 404 Not Found 해결 순서

```
# 1️⃣ 라우터가 main.py에 등록되었는지 확인
# main.py에서:
app.include_router(onboarding_router, prefix="/api/onboarding")
```

```
# 2️⃣ 함수 데코레이터 확인
# onboarding_routes.py에서:
@router.post("/step2")

# 3️⃣ Query 파라미터 확인
# onboarding_routes.py에서:
async def onboarding_step2(user_id: str = Query(...), ...):

# 4️⃣ URL 경로 확인
# 예상: /api/onboarding/step2
# 실제: curl -X POST "http://localhost:8000/api/onboarding/step2?..."
```

6.2 파라미터 인식 확인

```
# ❌ 틀린 요청 (body 방식)
curl -X POST "http://localhost:8000/api/onboarding/step2" \
-H "Content-Type: application/json" \
-d '{"user_id": "test", "keywords": "work,family"}'

# ✅ 올바른 요청 (쿼리 파라미터 방식)
curl -X POST "http://localhost:8000/api/onboarding/step2?
user_id=test&keywords=work,family"

# ✅ 올바른 요청 (body 방식 - Pydantic 모델 사용)
@router.post("/step2")
async def onboarding_step2(input_data: Step2Input):
    # 이 경우:
    user_id = input_data.user_id
    keywords = input_data.keywords
```

6.3 서버 로그 분석

```
# 라우터 등록 확인
(myenv) ➜  python backend/main.py
✅ api_router 등록 완료
✅ classifier_router 등록 완료
✅ onboarding_router 등록 완료    ← 이 라인이 있어야 함!
INFO:     Uvicorn running on http://0.0.0.0:8000

# 요청 로그
INFO:     127.0.0.1:12345 - "POST /api/onboarding/step2?
user_id=test_user&keywords=work,family,health HTTP/1.1" 200 OK

↑

200 = 성공! (404가 아님)
```

7. 전체 API 엔드포인트 정리

HTTP Method	경로	설명	Query Params	Body
GET	`/health`	서버 상태 확인	-	-
POST	`/api/onboarding/step1`	직업 정보 수집	-	`{occupation, name}`
POST	`/api/onboarding/step2`	키워드 저장	`user_id`, `keywords`	-
GET	`/api/onboarding/status/{user_id}`	온보딩 상태 조회	-	-
POST	`/api/classify/para`	문단 분류	-	`{text, user_id}`
POST	`/api/classify/keywords`	키워드 분류	-	`{text, user_id}`

8. 참고사항

- **FastAPI 버전**: 0.104.0 이상
- **호환성**: Python 3.8+ 모든 버전
- **성능 영향**: 거의 없음 (라우팅은 프레임워크 내부)
- **유지보수**: 매우 간단하고 명확함

🎯 **핵심 메시지:** 라우터 등록과 파라미터 명시 두 가지만 정확히 하면, 문제 해결 가능! 💙 ✨
