

PromptTemplate 중괄호 이스케이프 문제 해결 가이드

작성일: 2025-10-31

소요 시간: 약 2시간

최종 결과: JSON 형식 프롬프트와 PromptTemplate 변수의 완벽한 호환성 확보

1. 문제 상황 분석

1.1 발생 오류

```
'Input to PromptTemplate is missing variables {\n\n  "category"\n}'.  
Expected: [\n\n  "category", \n  'text'\n]  
Received: [\n  'text']
```

1.2 근본 원인

PromptTemplate의 중괄호 인식 충돌:

- PromptTemplate의 변수 표현: `{text}` (템플릿 변수)
- JSON 형식의 중괄호: `{ "key": "value" }` (리터럴 텍스트)
- PromptTemplate이 JSON의 `{...}` 를 변수로 잘못 인식

❌ 원본 프롬프트 파일 (문제 상황)
당신은 분류 전문가입니다.

텍스트: {text}

JSON 형식으로 응답하세요:

```
{  
  "category": "Projects",  
  "confidence": 0.95  
}
```

↓ PromptTemplate 해석

→ {text} = 변수 (맞음)

→ { "category": ... } = 변수로 인식 (문제!)

1.3 시도한 실패 방법들

시도 1: 정규표현식 기반 (실패) ❌

```
import re
```

```
content = re.sub(r'```\s\S]*?```
```

```
# 문제: 백틱 외부의 JSON 처리 안 됨
# 결과: 여전히 단일 } 남음 → 에러
```

시도 2: 복잡한 라인 처리 (실패) ❌

```
def escape_json_braces_complete(content: str) -> str:
    """복잡한 라인 기반 이스케이프"""
    lines = []
    in_code = False

    for line in content.split('\n'):
        if line.strip().startswith('```'):
            in_code = not in_code
        elif not in_code and re.match(r'^\s*\{', line):
            line = re.sub(r'\{\s', '{{ ', line)
            # ...더 많은 정규표현식

    return '\n'.join(lines)

# 문제: 엣지 케이스 많음, 단일 } 여전히 발생
```

2. 핵심 솔루션 (간단함이 정답!)

2.1 성공 코드

```
def get_para_classification_prompt() -> str:
    """프롬프트 파일 읽기 + 간단한 이스케이프"""

    prompt_path = CLASSIFIER_DIR / "prompts" /
    "para_classification_prompt.txt"

    with open(prompt_path, "r", encoding="utf-8") as f:
        content = f.read()

    # ✅ 핵심 로직: 라인 단위 이스케이프 + 변수 보호
    lines = []
    for line in content.split('\n'):
        # {text}는 건드리지 말기 (변수이니까!)
        if '{text}' in line:
            lines.append(line)
        else:
            # 나머지 모든 { } 는 이스케이프
            line = line.replace('{', '{{').replace('}', '}}')
            # {text}는 원상복구 (변수 유지)
            line = line.replace('{{text}}', '{text}')
            lines.append(line)

    return '\n'.join(lines)
```

2.2 동작 메커니즘

입력 (원본 프롬프트):

당신은 분류가입니다.

텍스트: {text}

응답: { "category": "..."} }

처리 과정:

[라인 1] "당신은 분류가입니다."

→ {text} 없음 → replace('{', '{{'), replace('}', '}}')

→ "당신은 분류가입니다." (변화 없음 - 종괄호가 없으니까)

[라인 2] "텍스트: {text}"

→ {text} 포함 → 그대로 유지!

→ "텍스트: {text}"

[라인 3] "응답: { \"category\": \"...\" }"

→ {text} 없음 → replace('{', '{{'), replace('}', '}}')

→ "응답: {{ \"category\": \"...\" }}"

→ replace('{{text}}', '{text}') (해당 없음)

→ "응답: {{ \"category\": \"...\" }}"

최종 출력:

당신은 분류가입니다.

텍스트: {text}

응답: {{ "category": "..."} }}

← 변수로 유지

← JSON의 종괄호 이스케이프

2.3 PromptTemplate 해석

이스케이프된 프롬프트를 PromptTemplate에 전달

```
prompt = PromptTemplate(
    input_variables=["text"],
    template=processed_content
)
```

 PromptTemplate 해석

→ {text} = 변수 (맞음!) ← 유일한 변수로 인식

→ {{ "category": ... }} = 리터럴 텍스트 (정확함!)

→ 이중 종괄호는 단일로 렌더링됨

3. 실전 적용

3.1 완전한 구현 코드

```

from pathlib import Path
from langchain_core.prompts import PromptTemplate

class ParaClassifier:
    def __init__(self, classifier_dir: Path):
        self.classifier_dir = classifier_dir

    def load_and_escape_prompt(self) -> str:
        """프롬프트 파일 로드 및 이스케이프"""
        prompt_path = self.classifier_dir / "prompts" /
        "para_classification_prompt.txt"

        with open(prompt_path, "r", encoding="utf-8") as f:
            content = f.read()

        # 이스케이프 처리
        lines = []
        for line in content.split('\n'):
            if '{text}' in line:
                lines.append(line)
            else:
                line = line.replace('{', '{{').replace('}', '}}')
                line = line.replace('{{text}}', '{text}')
                lines.append(line)

        return '\n'.join(lines)

    def create_prompt_template(self) -> PromptTemplate:
        """PromptTemplate 생성"""
        prompt_content = self.load_and_escape_prompt()

        return PromptTemplate(
            input_variables=["text"],
            template=prompt_content
        )

```

3.2 테스트 및 검증

```

# 이스케이프 확인
python -c "
from pathlib import Path

content =
Path('backend/classifier/prompts/para_classification_prompt.txt').read_text()



print(f'{{ 개수: {content.count("\{\\")}}}')
print(f'}} 개수: {content.count("\}\\")}')
print(f'{{text}} 개수: {content.count("\{text}\\")}')

```

```
# 예상 결과:
# { 개수: 0 (모두 이스케이프됨)
# } 개수: 0 (모두 이스케이프됨)
# {text} 개수: 1 (변수 유지)
"
```

4. 핵심 교훈






4.1 문제 해결 원칙

원칙	설명
Occam's Razor	복잡한 정규표현식보다 단순한 논리 
명확한 의도	보호할 변수 명시, 나머지 처리
테스트 우선	각 단계 검증하며 진행
성능	간단 = 유지보수성 

4.2 기술적 인사이트

- **PromptTemplate의 동작:** `{{ }}` → `{ }` 로 자동 변환
- **이중 중괄호의 역할:** 리터럴 중괄호를 보존하는 표준 방법
- **라인 단위 처리:** 개별 분석으로 정확성 확보

5. 성공 지표

-  API 호출 성공 (HTTP 200 OK)
-  JSON 파싱 완벽 (유효한 JSON 구조)
-  분류 결과 정확 (Projects/Areas/Resources/Archives)
-  한국어 텍스트 지원 (한글 자연 처리)
-  메타데이터 호환 (선택적 입력 처리)

6. 디버깅 팁

6.1 문제 진단

```
# 프롬프트가 제대로 이스케이프되었는지 확인
def check_prompt_escaping(template_str: str):
    print(f"원본 중괄호: {template_str.count('{')}")
    print(f"이스케이프된 중괄호: {template_str.count('{{}}')}")
    print(f"변수: {template_str.count('{text}')}")
```

6.2 역진단

```
# PromptTemplate 생성 전 검증
try:
    prompt = PromptTemplate(
        input_variables=["text"],
        template=my_template
    )
    print("✅ 프롬프트 유효")
except ValueError as e:
    print(f"❌ 오류: {e}")
# → 중괄호 불일치 확인
```

참고사항

- 적용 범위: LangChain PromptTemplate 모든 사용
- 호환성: LangChain 1.0.0 이상 모든 버전
- 성능 영향: 거의 없음 (문자열 처리만)
- 유지보수: 매우 간단하고 명확함

🔗 **핵심 메시지:** 문제의 원인을 정확히 파악하면, 복잡해 보이는 것도 간단한 해결책으로 해결 가능!
