생성형 AI 기반 RAG 개발자 과정

질의응답 API 구현

Session 22 / 33 — Day 3

업로드 문서 대상 질의 — POST /ask

박수현 · (사)한국정보공학기술사회 · 2026

LLM 이론 · LangChain · ChromaDB CRUD · Agent — 4일 핸즈온

출처: Day 3 질의응답 API

이번 시간을 마치면

💬 질의응답 체인

- **Retriever** 로 근거 검색
- **Context** 조립
- **Prompt + LLM** 으로 답변

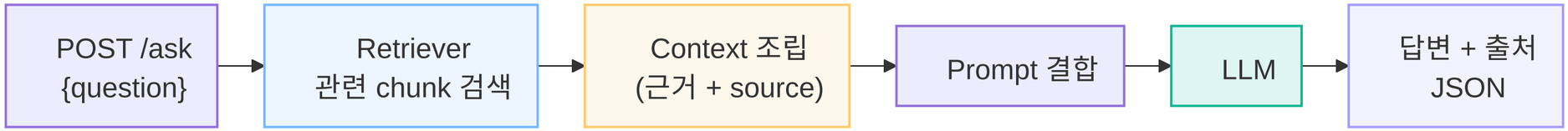
📎 신뢰 장치

- **출처(source)** 함께 반환
- 점수(score) 활용
- 근거 없을 시 "모른다" 응답

> 🎯 목표: 질문 한 줄로 **근거 기반 답변 + 출처**를 JSON으로 반환한다.

출처: Day 3 질의응답 API

질문에서 답변까지의 처리 흐름



> 💡 검색 → 근거 주입 → 생성. RAG 3단계를 그대로 엔드포인트로 구성.

출처: Day 3 질의응답 API

근거와 점수를 함께 반환 (스니펫)

```
def search_with_score(question, k=5, sources=None):
    flt = {"source": {"$in": sources}} if sources else None
    hits = store.similarity_search_with_score(
        question, k=k, filter=flt)
    return [(doc, score) for doc, score in hits]
```

- `similarity_search_with_score` — chunk + **유사도 점수**
- `sources` 지정 시 **특정 문서 내부로 한정**한 검색 (where 필터)

> 📁 참고: `vectorstore.py` 의 `search_with_score`

근거를 프롬프트에 주입해 답변 생성 (스니펫)

```
def answer_question(question, sources=None):
    hits = search_with_score(question, sources=sources)
    context = "\n\n".join(d.page_content for d, _ in hits)

    prompt = f"""아래 자료에 근거해서만 답하세요.
없으면 "자료에 없습니다".
[자료]\n{context}\n[질문] {question}"""

    answer = llm.invoke(prompt).content
    used = [{"file": d.metadata["source"],
             "page": d.metadata.get("page", 0) + 1,
             "score": round((1 - s) * 100, 1)} for d, s in hits]
    return {"answer": answer, "sources": used}
```

> 🔑 답변과 함께 **출처(파일·페이지·점수) 목록**을 반환해 화면에 출처 칩으로 표시.

라우트는 서비스 호출만 (스니펫)

```
@app.post("/ask")
def ask():
    body = request.get_json()
    q = body.get("question", "").strip()
    if not q:
        return jsonify({"error": "질문이 비었습니다"}), 400
    return jsonify(answer_question(q, body.get("sources")))
```

응답 예 (프런트와의 JSON 계약):

```
{"answer": "환불은 7일 이내 가능합니다.",
 "sources": [{"file": "faq.pdf", "page": 2, "score": 87.3}]}
```

> 🔑 프런트는 `answer`를 말풍선에, `sources`를 **파일·페이지·점수 칩**으로 표시.

토큰 단위 점진 전송 (스니펫)

긴 답변을 전부 생성한 뒤 한 번에 반환하면 대기 시간이 길어진다. **생성되는 즉시 토큰을 전송**하면 체감 응답 속도가 크게 향상된다.

```
@app.post("/ask/stream")
def ask_stream():
    q = (request.get_json() or {}).get("question", "")
    def gen():
        for chunk in llm.stream(build_prompt(q)):
            yield f"data: {chunk.content}\n\n"   # SSE 포맷
    return Response(gen(), mimetype="text/event-stream")
```

`llm.stream()`으로 토큰 제너레이터를 받아 **SSE**(`text/event-stream`)로 전송한다. 출처는 스트림 종료 후 마지막 이벤트로 별도 전송.

> 💡 답변은 스트리밍, **출처는 완료 후 일괄 전송**. 프런트는 말풍선을 실시간 갱신.

출처: Day 3 질의응답 API

핵심 6가지

1. ✅ `POST /ask` = RAG 3단계(**검색→근거주입→생성**)의 엔드포인트
2. ✅ `similarity_search_with_score` 로 **근거 + 점수**
3. ✅ `sources` 필터로 **특정 문서만** 질의 가능
4. ✅ 답변과 함께 **출처(파일·페이지·점수)** 를 JSON 으로 반환
5. ✅ 근거 없으면 **"자료에 없습니다"** — 환각 억제
6. ✅ 긴 답변은 **SSE 스트리밍**, 출처는 완료 후 한 번