생성형 AI 기반 RAG 개발자 과정

Document Loader
TXT · PDF · DOCX

Session 11 / 33 — Day 2

RAG 의 입구 — LangChain 으로 문서 읽기

박수현 · (사)한국정보공학기술사회 · 2026

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

이번 시간을 마치면

📄 형식별 로딩

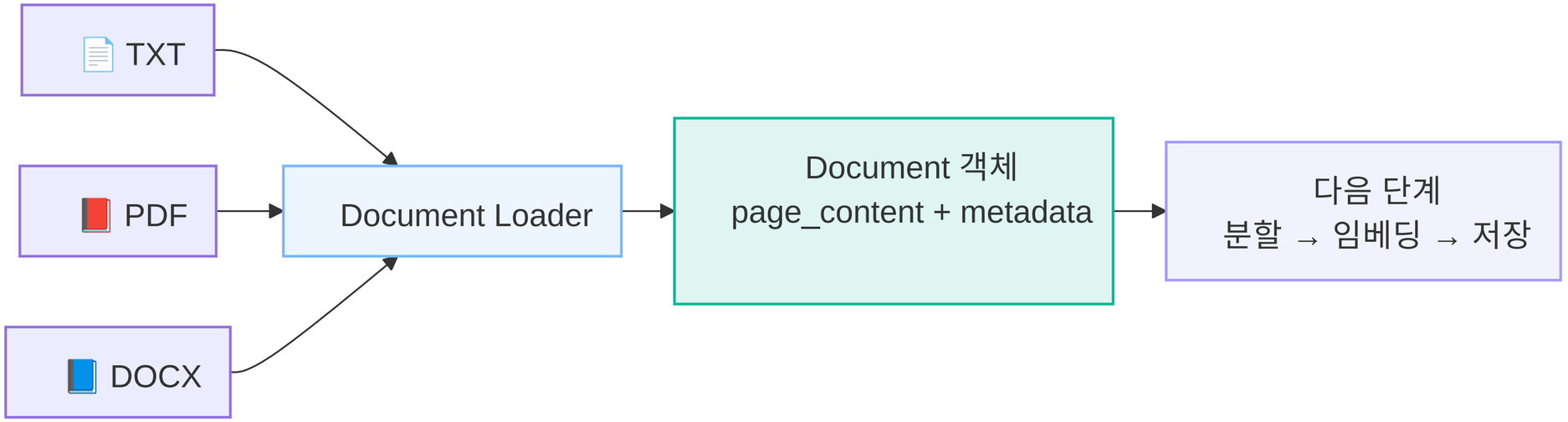
- **TextLoader** — TXT
- **PyPDFLoader** — PDF (페이지 단위)
- **DOCX** 로더 — 워드 문서

🏷️ 공통 산출물

- `Document(page_content, metadata)`
- **자동 메타데이터** (source·page)
- **출처 추적**에 필요한 기준 정보

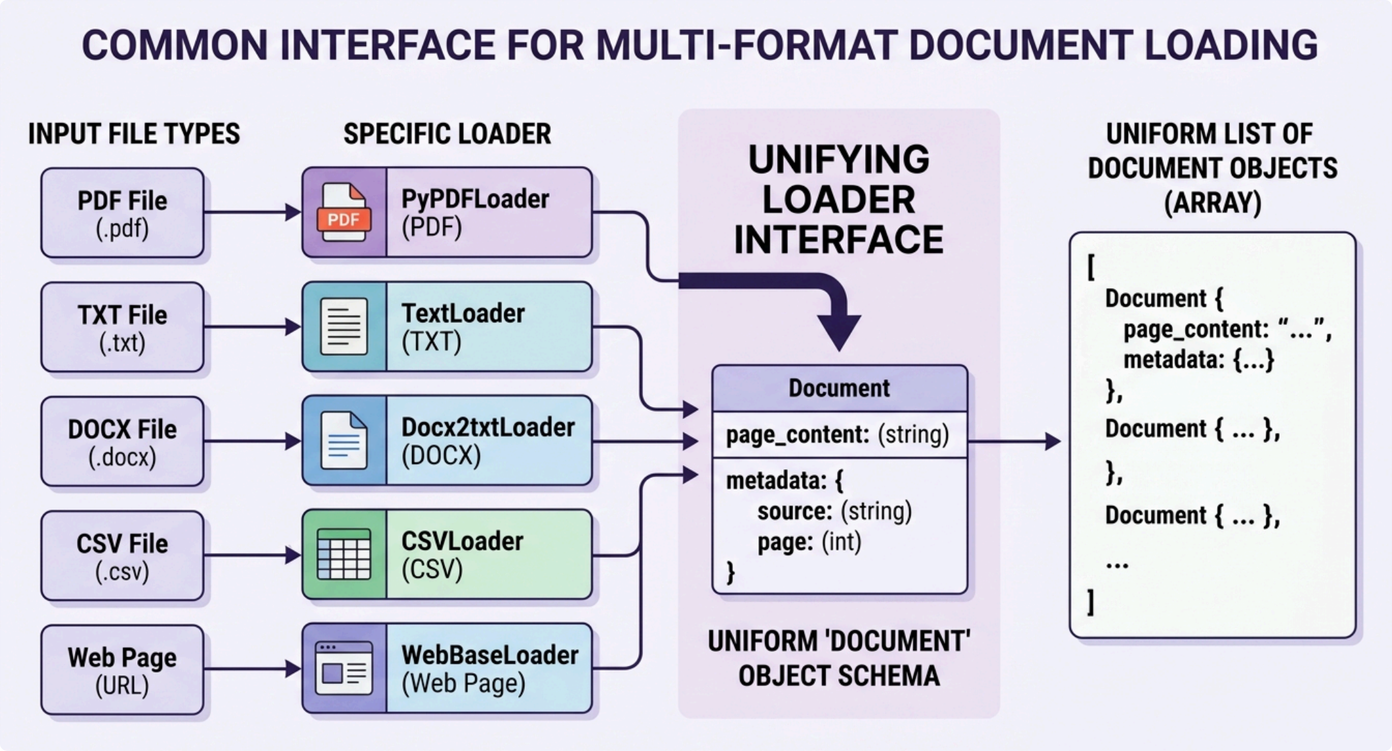
🎯 목표: 어떤 형식이든 **동일한** `Document` **객체**로 통일해 다음 단계(분할·임베딩)에 전달한다.

입력 형식은 다양해도 출력은 Document 하나



💡 핵심: Loader 는 **형식별 추출 차이를 흡수**해 후속 단계가 입력 형식에 의존하지 않게 한다.

입력 형식은 다양해도 출력은 Document 하나



🖼️ Document Loader · PyPDFLoader·TextLoader·Docx2txtLoader·WebBaseLoader가 다양한 포맷을 Document(page_content, metadata)로 통합

`Document` = 본문 + 출처 정보

두 개의 필드

- `page_content` — 추출된 텍스트(문자열)
- `metadata` — 출처 정보(dict)
- `source` (파일 경로)
- `page` (PDF 쪽 번호)
- 작성자·날짜 등 직접 추가 가능

```
Document(
  page_content="환불은 7일 이내…",
  metadata={"source": "faq.pdf",
            "page": 3}
)
```

🔑 `metadata` 는 이후 **출처 표시·문서 삭제**의 키로 쓰인다 (Day3 CRUD).

최소 구성 로더 (스니펫)

```
from langchain_community.document_loaders import TextLoader

docs = TextLoader("faq.txt", encoding="utf-8").load()
print(len(docs))               # 1  (파일 1개 = Document 1개)
print(docs[0].page_content[:40])
print(docs[0].metadata)        # {'source': 'faq.txt'}
```

- TXT 1개 → `Document` 1개 (통째로)
- `source` 메타데이터 **자동** 부착

> 📁 전체 예제: `2.langchain/7.RAG/2.loaders/2.1_text_loader.py`

페이지 단위 로딩 — PDF (스니펫)

```
from langchain_community.document_loaders import PyPDFLoader

docs = PyPDFLoader("manual.pdf").load()
print(len(docs))               # 페이지 수 만큼 Document
print(docs[0].metadata)        # {'source': 'manual.pdf', 'page': 0}
```

- PDF **1쪽 = Document 1개** → `page` 메타데이터 자동
- ⚠️ Loader 는 **청킹하지 않는다** — 긴 페이지도 단일 Document, 분할은 다음 Splitter 단계의 몫
- 표·이미지가 많은 PDF 는 추출 품질 주의 (레이아웃 깨짐)
- 스캔 이미지 PDF → 별도 **OCR** 필요

> 📁 전체 예제: `2.langchain/7.RAG/2.loaders/2.2_pdf_loader.py`

워드 문서 로딩 — DOCX (스니펫)

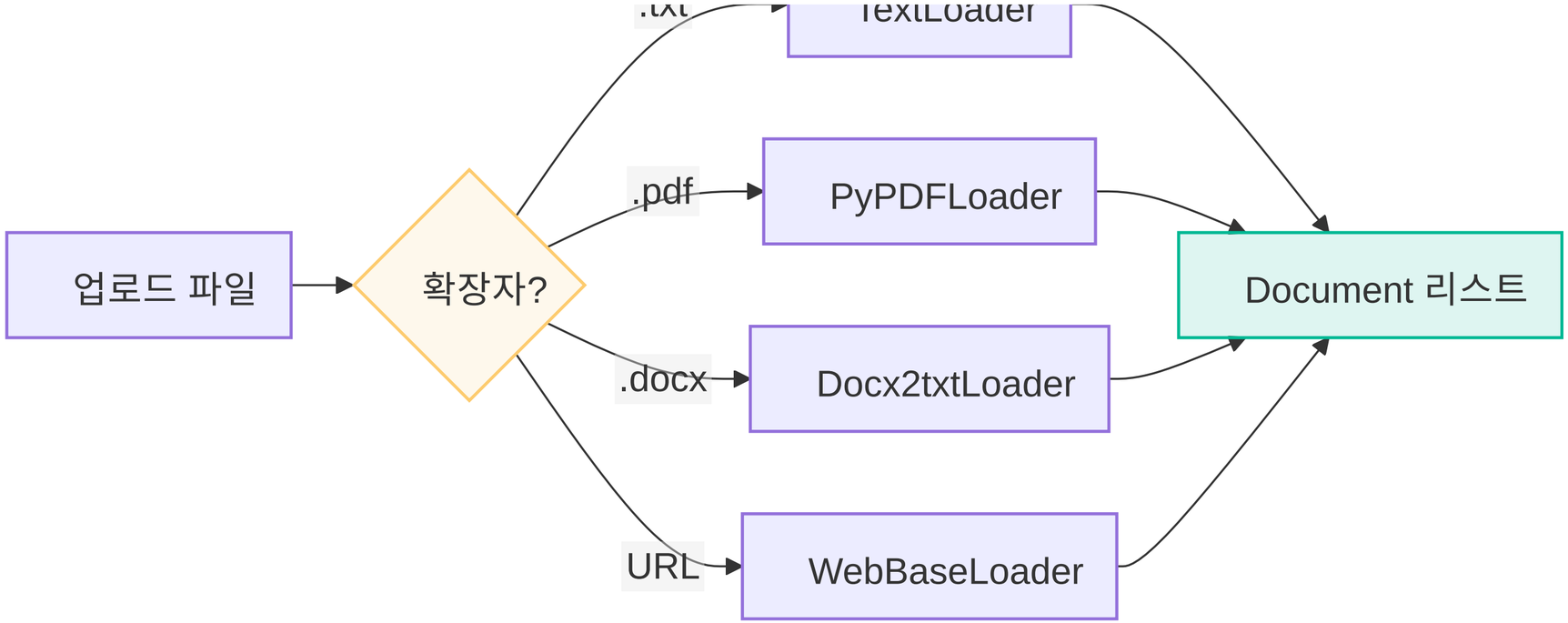
```
# pip install docx2txt
from langchain_community.document_loaders import Docx2txtLoader

docs = Docx2txtLoader("report.docx").load()
print(docs[0].metadata)        # {'source': 'report.docx'}
```

- DOCX → 단락 텍스트 추출 (`Document` 1개)
- 표·머리글은 형식 손실 가능 → 핵심 본문 위주
- 🌐 웹 문서는 **WebBaseLoader** — URL 을 입력받아 본문을 `Document` 로 변환 (파일 외 입력원)

> 💡 Day3 웹서비스는 업로드 확장자에 따라 **로더를 자동 선택**한다 (`.pdf → PyPDF` , `.docx → Docx2txt`).

Day3 서비스 라우팅 미리보기



🔑 이 분기 로직은 Day3 `POST /api/index` 의 첫 단계에 해당한다 (Session 19).

출처 추적을 위한 메타데이터 보강

보강이 필요한 경우

- 자동 `source` · `page` 만으로 부족할 때
- **문서 종류·작성일·부서** 등으로 필터링할 때
- Day2 Retriever 의 **metadata filter** 와 연계할 때

```
for d in docs:
    d.metadata["doc_type"] = "규정"
    d.metadata["dept"] = "HR"
```

> 🔑 메타데이터는 **검색 필터·출처 표시·선택 삭제**를 잇는 공통 키다.

핵심 5가지

1. ✅ Loader = 형식별 추출 차이를 `Document` **로 통일**
2. ✅ `Document` = **page_content(내용) + metadata(출처)**
3. ✅ **TXT/PDF/DOCX** = TextLoader · PyPDFLoader · Docx2txtLoader
4. ✅ PDF 는 **페이지 단위**, `page` 메타 자동
5. ✅ 메타데이터는 **검색 필터·출처·선택 삭제**를 잇는 공통 키