

- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: <https://smith.langchain.com/hub/teddynote/summary-stuff-documents>

▼ HWP (한글)

- **HWP**
 - 한글(**HWP**) = 한글과컴퓨터에서 개발한 워드프로세서 = 한국의 대표적인 문서 작성 프로그램
 - 파일 확장자 = **.hwp** 사용 → 기업, 학교, 정부 기관 등에서 널리 활용
 - **LangChain**에는 아직 **integration** 이 되지 않아 직접 구현한 **HWPLoader**를 사용해야 함

▼ 1) Mac 사용자 추천 - [Rust HWP Loader](#)

▼ 개요

- [libhwp](#)
 - 라이브러리 **RustHwpLoader**를 사용 → **HWP** 파일을 로드하는 **Python** 클래스
 - 모든 **OS**에서 작동
- **Document**
 - **HWP** 파일에서 모든 문단과 표를 추출 → 객체 목록으로 반환
 - 각 **Document** 객체는 문단 또는 표의 내용과 관련 메타데이터를 포함
 - 첫 번째 객체는 **Document** 각 표의 텍스트를 포함하여 **HWP** 파일의 모든 문단을 포함
 - 두 번째 객체는 **Documents** 각 표의 문단
 - 안타깝게도 이 로더는 표의 행과 열을 구분하지 않음
 - 각 속성 **metadata**: **Document**파일 경로 (소스 키 아래), 페이지 유형 (텍스트 또는 테이블) 포함
 - 다른 **HWP** 로더가 더 뛰어난 기능을 제공하지만, **RustHwpLoader**외부 **HWP** 로더 서버 or **Windows** 전용 **HWP** 프로그램이 필요하지 않기 때문에 **MacOS** 및 **Linux** 사용자에게 좋은 옵션



용법

- 초기화

- a. **HWP** 파일 경로 초기화

```
loader = RustHwpLoader("/path/to/hwp/file")
```

- b. **libhwp** 라이브러리가 설치되지 않은 경우: **pip** 사용해서 **Import Error** 설치하려는 메시지와 함께 오류 발생

```
pip install libhwp          # 터미널에 먼저 설치
pip install ragchain
```

- 문서 로딩: **HWP** 파일에서 객체를 **RustHwpLoader** 로드하는 두 가지 방법을 제공

- a. **load()**: **Documents** → 모든 항목의 목록을 반환

```
documents = loader.load()
```

- b. **lazy_load()**:

```
for document in loader.lazy_load():          # process document
```

- Documents** → 객체를 느리게 생성하는 생성기
- 목록에 모두 로드하면 많은 메모리를 소모할 수 있는 **대용량 HWP 파일**로 작업할 때 유용

- 각각은 문자열과 사전 **Document** 을 포함 **load** or **lazy_load** 생성

- 여기에는 **source** 키(파일 경로), **page_type** 키(**text** 또는 **table**)가 포함
- page_content**, **metadata**, **metadata**



2) 개발환경 충돌

- 사용하는 환경 **Python = 3.13.**
- 참고하는 패키지 및 이미 만들어진 로더들(`libhwp`, `pyhwp`, `llama-index-hwp` 등)이 대부분 **Python = 3.11**까지 지원

✓ ① TeddyNote에서 HWPLoader의 핵심 기능만 추출해보기

- `custom_hwp_loader.py`
- `custom_hwp_loader.py` 코드

```
import os
from typing import Iterator, List
from langchain_core.documents import Document
from langchain_core.document_loaders.base import BaseLoader

class CustomHWPLoader(BaseLoader):
    """개선된 HWP 로더 - Python 3.13 호환"""

    def __init__(self, file_path: str):
        self.file_path = file_path

    def lazy_load(self) -> Iterator[Document]:
        """HWP 파일을 로드하여 Document로 변환"""
        try:
            # 방법 1: olefile로 시도
            text_content = self._extract_with_olefile()

            if not text_content or "추출 실패" in text_content:
                # 방법 2: zipfile로 시도 (HWP 5.0+는 ZIP 기반)
                text_content = self._extract_with_zipfile()

            if not text_content or "추출 실패" in text_content:
                # 방법 3: 바이너리 직접 읽기
                text_content = self._extract_with_binary()

            metadata = {
                "source": self.file_path,
                "file_type": "hwp",
                "extraction_method": "custom_parser"
            }

            yield Document(
                page_content=text_content,
                metadata=metadata
            )
```

```

    )

except Exception as e:
    print(f"전체 HWP 로딩 오류: {e}")
    yield Document(
        page_content=f"HWP 파일 로딩 실패: {e}",
        metadata={"source": self.file_path, "error": str(e)}
    )

def _extract_with_olefile(self) -> str:
    """olefile을 사용한 추출"""
    try:
        import olefile

        if not olefile.isOleFile(self.file_path):
            return "olefile: HWP 형식이 아님"

        with olefile.OleFileIO(self.file_path) as ole:
            sections = []

            # 모든 스트림 탐색
            for stream_name in ole.listdir():
                stream_path = '/'.join(stream_name)

                # BodyText 관련 스트림 찾기
                if 'BodyText' in stream_path or 'bodytext' in stream_path.lower():
                    try:
                        data = ole.open(stream_name).read()

                        # 여러 인코딩 시도
                        for encoding in ['utf-16le', 'utf-16', 'cp949', 'euc-kr']:
                            try:
                                text = data.decode(encoding, errors='ignore')
                                if text and len(text.strip()) > 10: # 의미있을 것 같음
                                    sections.append(f"[{encoding}] {text[:50]}")
                                    break
                            except:
                                continue

                    except Exception as e:
                        print(f"스트림 읽기 오류 {stream_path}: {e}")
                        continue

            result = '\n'.join(sections) if sections else "olefile: 텍스트 섹션 없음"
            return result

    except ImportError:
        return "olefile 라이브러리가 설치되지 않음"

```

```

except Exception as e:
    return f"olefile 추출 실패: {e}"

def _extract_with_zipfile(self) -> str:
    """zipfile을 사용한 추출 (HWP 5.0+)"""
    try:
        import zipfile
        import xml.etree.ElementTree as ET

        with zipfile.ZipFile(self.file_path, 'r') as hwp_zip:
            sections = []

            # ZIP 내부 파일 목록 확인
            file_list = hwp_zip.namelist()
            print(f"ZIP 내부 파일들: {file_list}")

            # XML 파일들 확인
            for file_name in file_list:
                if file_name.endswith('.xml') or 'content' in file_name.lower():
                    try:
                        content = hwp_zip.read(file_name)

                        # XML 파싱 시도
                        try:
                            root = ET.fromstring(content)
                            # 텍스트 노드 추출
                            for elem in root.iter():
                                if elem.text and elem.text.strip():
                                    sections.append(elem.text.strip())
                        except:
                            # XML이 아니면 직접 텍스트 추출
                            for encoding in ['utf-8', 'cp949', 'euc-kr']:
                                try:
                                    text = content.decode(encoding, errors='replace')
                                    if text and len(text.strip()) > 10:
                                        sections.append(f"[{file_name}] {text.strip()}")
                                    break
                                except:
                                    continue

                    except Exception as e:
                        print(f"ZIP 파일 읽기 오류 {file_name}: {e}")
                        continue

            return '\n'.join(sections) if sections else "zipfile: 텍스트를 찾을 수 없습니다"

    except zipfile.BadZipFile:
        return "zipfile: ZIP 형식이 아님"

```

```

except Exception as e:
    return f"zipfile 추출 실패: {e}"

def _extract_with_binary(self) -> str:
    """바이너리 직접 읽기로 텍스트 추출"""
    try:
        with open(self.file_path, 'rb') as f:
            data = f.read()

        # 한글 문자열 패턴 찾기
        text_parts = []

        # 여러 인코딩으로 시도
        for encoding in ['utf-16le', 'cp949', 'euc-kr', 'utf-8']:
            try:
                decoded = data.decode(encoding, errors='ignore')

                # 한글이 포함된 문장 찾기
                import re
                korean_sentences = re.findall(r'[가-힣\s]{10,}', decoded)

                if korean_sentences:
                    text_parts.extend([f"[{encoding}] {sent}" for sent in korean_sentences])

            except:
                continue

        if text_parts:
            return '\n'.join(text_parts)
        else:
            return f"바이너리 추출: 한글 텍스트를 찾을 수 없음 (파일 크기: {len(data)})"

    except Exception as e:
        return f"바이너리 추출 실패: {e}"

if __name__ == "__main__":
    # 절대 경로로 수정
    import os

    # 현재 파일의 디렉토리 기준으로 경로 설정
    current_dir = os.path.dirname(os.path.abspath(__file__))
    hwp_file_path = os.path.join(current_dir, "data", "디지털 정부혁신 추진계획.hwp")

    print(f"파일 경로: {hwp_file_path}")
    print(f"파일 존재 여부: {os.path.exists(hwp_file_path)}")

    if os.path.exists(hwp_file_path):
        # olefile 설치 확인

```

```

try:
    import olefile
    print("✅ olefile 라이브러리 사용 가능")
except ImportError:
    print("⚠️ olefile 설치 필요: pip install olefile")

loader = CustomHWPLoader(hwp_file_path)
docs = loader.load()

print(f"\n🎉 성공! 로드된 문서: {len(docs)}")
print(f"📄 내용 미리보기:")
print("-" * 50)
print(docs[0].page_content[:1500]) # 더 많이 출력
print("-" * 50)
print(f"📁 메타데이터: {docs[0].metadata}")
else:
    print("❌ 파일을 찾을 수 없습니다!")

```

- 출력 결과

```

HWP 로딩 오류: [Errno 2] No such file or directory: '.././././06 Document Loader/data/디지털 정부혁신 추진계획.hwp'
성공! 로드된 문서: 1
내용:
{'source': '.././././06 Document Loader/data/디지털 정부혁신 추진계획.hwp', 'error': None}
HWP 로딩 오류: [Errno 2] No such file or directory: '.././././06 Document Loader/data/디지털 정부혁신 추진계획.hwp'
성공! 로드된 문서: 1
내용:
{'source': '.././././06 Document Loader/data/디지털 정부혁신 추진계획.hwp', 'error': None}

```

- 올바르지 않은 경로 실행

- 1_터미널: 올바른 경로에서 실행 → 문제 그대로
- 2_Jupyter notebook에서 테스트 (아래 셀 참고) → 문제 그대로

- 로직 다시 수정 결정

올바른 경로에서 확인하기 - 두번째 방법

```

import os
import sys

```

```

# 현재 경로 확인
print(f"현재 작업 디렉토리: {os.getcwd()}")

```

```

# 파일 경로들 확인

```

```
possible_paths = [
    "./data/디지털 정부혁신 추진계획.hwp",
    "../06_Document_Loader/data/디지털 정부혁신 추진계획.hwp",
    "/Users/jay/Projects/20250727-langchain-note/06_Document_Loader/data/디지털 정부
]

for path in possible_paths:
    if os.path.exists(path):
        print(f"✅ 파일 발견: {path}")
        hwp_path = path
        break
    else:
        print(f"❌ 파일을 찾을 수 없습니다!")

# data 폴더 내용 확인
data_dir = "./data"
if os.path.exists(data_dir):
    print(f"\n📁 {data_dir} 폴더 내용:")
    for file in os.listdir(data_dir):
        print(f"  - {file}")
```

- 셀 출력

현재 작업 디렉토리: /Users/jay/Projects/20250727-langchain-note/06_Document_Loader

✅ 파일 발견: ./data/디지털 정부혁신 추진계획.hwp

📁 ./data 폴더 내용:

- 디지털 정부혁신 추진계획.hwp
- client.html
- appendix-keywords-CP949.txt
- reference.txt
- 디지털_정부혁신_추진계획.pdf
- appendix-keywords-EUCKR.txt
- SPRI_AI_Brief_2023년12월호_F.pdf
- audio_utils.py
- chain-of-density.txt
- titanic.csv
- sample-word-document.docx
- titanic.xlsx
- sample-ppt.pptx
- appendix-keywords.txt
- people.json
- appendix-keywords-utf8.txt

- **custom_hwp_loader2.py**

- 로직 수정하기

- [한컴오피스](#)에서 공식적으로 발표한 **HWP** 파싱 방법 적용하기
- `custom_hwp_loader2.py`의 코드

```
# 06_Document_Loader/custom_hwp_loader.py - 한컴 공식 방법 적용
import os
import olefile
import zlib
import struct
from typing import Iterator, List
from langchain_core.documents import Document
from langchain_core.document_loaders.base import BaseLoader

class CustomHWPLoader(BaseLoader):
    """한컴 공식 방법 기반 HWP 로더"""

    def __init__(self, file_path: str):
        self.file_path = file_path

    def lazy_load(self) -> Iterator[Document]:
        """HWP 파일을 로드하여 Document로 변환"""
        try:
            text_content = self._extract_hwp_text()

            metadata = {
                "source": self.file_path,
                "file_type": "hwp",
                "extraction_method": "hancom_official"
            }

            yield Document(
                page_content=text_content,
                metadata=metadata
            )

        except Exception as e:
            print(f"HWP 로딩 오류: {e}")
            yield Document(
                page_content=f"HWP 파일 로딩 실패: {e}",
                metadata={"source": self.file_path, "error": str(e)}
            )

    def _extract_hwp_text(self) -> str:
        """한컴 공식 방법으로 HWP 텍스트 추출"""
        try:
            f = olefile.OleFileIO(self.file_path)
            dirs = f.listdir()
```

```

print(f"📁 HWP 내부 구조:")
for d in dirs:
    print(f"  - {'/'.join(d) if isinstance(d, list) else d}")

# HWP 파일 검증
if ["FileHeader"] not in dirs or ["\\x05HwpSummaryInformation"] not
    print("⚠️ HWP 파일 형식이 아닙니다")
    return self._try_prvtext_method(f)

# 문서 포맷 압축 여부 확인
header = f.openstream("FileHeader")
header_data = header.read()
is_compressed = (header_data & 1) == 1
print(f"🔍 압축 여부: {is_compressed}")

# Body Sections 찾기
nums = []
for d in dirs:
    if isinstance(d, list) and len(d) >= 2 and d == "BodyText":
        try:
            section_num = int(d[21][len("Section"):])
            nums.append(section_num)
        except:
            continue

if not nums:
    print("📁 BodyText 섹션을 찾을 수 없음, PrvText 방법 시도")
    return self._try_prvtext_method(f)

sections = [f"BodyText/Section{x}" for x in sorted(nums)]
print(f"📁 발견된 섹션: {sections}")

# 전체 텍스트 추출
text = ""
for section in sections:
    try:
        print(f"🔍 섹션 처리 중: {section}")
        bodytext = f.openstream(section)
        data = bodytext.read()

        if is_compressed:
            unpacked_data = zlib.decompress(data, -15)
        else:
            unpacked_data = data

        # 각 섹션 내 텍스트 추출
        section_text = self._extract_section_text(unpacked_data)
    
```

```

        text += section_text + "\n"

    except Exception as e:
        print(f"❌ 섹션 {section} 처리 오류: {e}")
        continue

    return text if text.strip() else self._try_prvtext_method(f)

except Exception as e:
    print(f"❌ HWP 추출 오류: {e}")
    return f"HWP 추출 실패: {e}"
finally:
    if 'f' in locals():
        f.close()

def _extract_section_text(self, unpacked_data: bytes) -> str:
    """섹션 데이터에서 텍스트 추출"""
    section_text = ""
    i = 0
    size = len(unpacked_data)

    try:
        while i < size:
            if i + 4 > size:
                break

            header = struct.unpack_from("<I", unpacked_data, i)
            rec_type = header & 0x3ff
            rec_len = (header >> 20) & 0xfff

            # 텍스트 레코드 타입 (67번)
            #if rec_type in :
            if rec_type == 67:
                if i + 4 + rec_len <= size:
                    rec_data = unpacked_data[i + 4:i + 4 + rec_len]
                    try:
                        text = rec_data.decode('utf-16le')
                        section_text += text + "\n"
                    except:
                        try:
                            text = rec_data.decode('utf-16')
                            section_text += text + "\n"
                        except:
                            pass

                    i += 4 + rec_len

    except Exception as e:

```

```

        print(f"⚠️ 섹션 텍스트 추출 중 오류: {e}")

    return section_text

def _try_prvtext_method(self, f) -> str:
    """PrvText 방법으로 텍스트 추출 (백업 방법)"""
    try:
        print("🔄 PrvText 방법으로 시도 중...")
        encoded_text = f.openstream('PrvText').read()
        decoded_text = encoded_text.decode('UTF-16le')
        print("✅ PrvText 방법 성공!")
        return decoded_text
    except Exception as e:
        print(f"❌ PrvText 방법도 실패: {e}")
        return f"모든 추출 방법 실패: PrvText 오류 - {e}"

if __name__ == "__main__":
    import os

    # 현재 파일의 디렉토리 기준으로 경로 설정
    current_dir = os.path.dirname(os.path.abspath(__file__))
    hwp_file_path = os.path.join(current_dir, "data", "디지털 정부혁신 추진계획.hwp")

    print(f"📁 파일 경로: {hwp_file_path}")
    print(f"✅ 파일 존재 여부: {os.path.exists(hwp_file_path)}")

    if os.path.exists(hwp_file_path):
        print("\n🚀 HWP 파싱 시작...")
        loader = CustomHWPLoader(hwp_file_path)
        docs = loader.load() # 이걸 리스트를 반환함

        print(f"\n🎉 성공! 로드된 문서: {len(docs)}")

        if docs: # 문서가 존재하는지 확인
            first_doc = docs[0] # 첫 번째 문서 가져오기
            print(f"📄 내용 길이: {len(first_doc.page_content)} 문자")
            print(f"\n📄 내용 미리보기:")
            print("=" * 60)
            print(first_doc.page_content[:2000]) # 2000자까지 출력
            print("=" * 60)
            print(f"\n📄 메타데이터: {first_doc.metadata}")
        else:
            print("❌ 문서를 로드할 수 없습니다!")
    else:
        print("❌ 파일을 찾을 수 없습니다!")

```

- 터미널 실행 결과

📁 파일 경로: </Users/jay/Projects/20250727-langchain-note/06 Document Loader/data/>

✅ 파일 존재 여부: True

🚀 HWP 파싱 시작...

📁 HWP 내부 구조:

- HwpSummaryInformation
- BinData/BIN0001.jpg
- BinData/BIN0002.bmp
- BinData/BIN0003.bmp
- BinData/BIN0004.bmp
- BinData/BIN0005.bmp
- BinData/BIN0006.bmp
- BinData/BIN0007.bmp
- BinData/BIN0008.bmp
- BodyText/Section0
- DocInfo
- DocOptions/_LinkDoc
- FileHeader
- PrvImage
- PrvText
- Scripts/DefaultJScript
- Scripts/JScriptVersion

⚠️ HWP 파일 형식이 아닙니다

🔄 PrvText 방법으로 시도 중...

✅ PrvText 방법 성공!

🎉 성공! 로드된 문서: 1

📄 내용 길이: 1022 문자

📖 내용 미리보기:

=====

<디지털 정부혁신 추진계획>

2019. 10. 29.

<><관계부처 합동>

<><순 서><><I. 개요 1 II. 디지털 정부혁신 추진계획 2 1. 우선 추진과제 2 ① 선제적·

<I. 개 요>

□ 추진 배경

- 우리나라는 국가적 초고속 정보통신망 투자와 적극적인 공공정보화 사업 추진에 힘입어 세계 최고수준의 전자정부
* UN전자정부평가에서 2010·12·14년 1위, 16·18년 3위, UN공공행정상 13회 수상
- 그러나, 인공지능·클라우드 중심의 디지털 전환(Digital Transformation) 시대가 도래함에 따라 기존 제
- 축적된 행정데이터에도 불구하고 기관간 연계·활용 미흡, 부처 단위로 단절된 서비스, 신기술 활용을 위한 제도
- 디지털 전환을 위한 컨트롤타워가 없고, 구체적 전략도 부재
- 이에, '19.3월부터 공공부문 ICT 활용현황 및 문제점 검토에 착수하여 공공분야 디지털 전환을 위한 추진계


* 관계부처 협의 21회(행안, 과기정통, 기재, 복지, 권익위, 국정원 등), 민간전문가 의견청취 10회

□ 문제점 진단 및 평가

- (서비스) 국민과 최종 이용자 관점에서 서비스 혁신 미흡
 - 자격이 있어도 자신이 받을 수 있는 공공서비스를 파악하기 어려워 사각지대가 발생하고, 온라인 신청 가능한
- (데이터) 기관별로 축적·보유한 데이터의 연계와 활용 부족
 - A기관에서 서류를 발급받아 B기관에 제출하는

=====

 메타데이터: {'source': '/Users/jay/Projects/20250727-langchain-note/06 Document

- 터미널 실행 결과 화면 캡처  custom_hwp_loader2.py 결과
- 한컴오피스의 공식 방법 활용
 - 완전한 텍스트 추출 ○
 - 구조 분석 ○: 이미지 (BIN0001.jpg ~ BIN0008.bmp), BodyText, PrvText 등 모두 확인
 - Python 3.13 호환 버전 HWP 파서 ○
 - LangChain 통합: Document 객체로 변환 ○
- 각 단계별 디버깅 적용
- 필요한 패키지들 설치 필요(터미널):

```
# 외부 모듈 (설치 필요)
pip install olefile
pip install langchain-core
```

OLE 파일 처리
LangChain 문서 객체

- Python 내장 모듈: zlib (압축/해체), struct (바이너리 데이터 처리), os (운영체제 인터페이스), zipfile (ZIP 파일 처리)
 - a. zlib 설치 확인 방법

```
python -c "import zlib; print('✅ zlib 사용 가능!')"
```

** b. struct = 역시 Python 내장 모듈 (별도 설치 불필요)

```
# 필요한 모든 모듈 확인
python -c "
import zlib
import struct
import olefile
print('✅ 모든 모듈 사용 가능!')
"
```

-
- *next:* **CSV**
-