뉴스 크롤러 서비스 테스트 케이스 설계서

개요

뉴스 크롤러 서비스의 API 기능을 검증하기 위한 종합적인 테스트 케이스를 정의합니다.

테스트 환경

- **테스트 서버**: http://localhost:8083
- **테스트 도구**: Postman, curl, JUnit
- **데이터베이스**: 테스트용 DB (별도 구성)
- **FTP 서버**: 테스트용 FTP 서버

---

1. 크롤링 관리 API 테스트

1.1 크롤링 시작 API 테스트

TC-001: 정상 크롤링 시작

테스트 목적: 크롤링 시작 API가 정상적으로 동작하는지 확인

전제 조건:

- 크롤러 서비스가 정상 실행 중
- 데이터베이스 연결 정상

테스트 단계:

1. `POST /api/crawler/start` 요청 전송
2. 응답 상태 코드 확인
3. 응답 본문 검증

예상 결과:

- HTTP 상태 코드: 200
- 응답 본문:

```
{
  "status": "success",
  "message": "배포 환경 최적화 크롤링이 시작되었습니다.",
  "timestamp": "2025-01-20T10:30:00"
}
```

테스트 데이터:

```
curl -X POST http://localhost:8083/api/crawler/start \
  -H "Content-Type: application/json" \
  -d '{}'
```

TC-002: 서비스 장애 시 크롤링 시작

테스트 목적: 서비스 장애 시 적절한 에러 응답 확인

전제 조건:

- 크롤러 서비스에 장애 상황 시뮬레이션

테스트 단계:

1. 서비스 장애 상황 생성
2. `POST /api/crawler/start` 요청 전송
3. 에러 응답 확인

예상 결과:

- HTTP 상태 코드: 500
- 응답 본문:

```
{
  "status": "error",
  "message": "크롤링 시작 실패: [에러 메시지]",
  "timestamp": "2025-01-20T10:30:00"
}
```

1.2 파일서버 데이터 DB 저장 API 테스트

TC-003: 정상 파일서버 데이터 저장

테스트 목적: 파일서버 데이터 DB 저장이 정상 동작하는지 확인

전제 조건:

- 파일서버에 테스트 데이터 존재
- 데이터베이스 연결 정상

테스트 단계:

1. `POST /api/crawler/save-fileserver` 요청 전송
2. 응답 상태 코드 확인
3. 데이터베이스 저장 결과 확인

예상 결과:

- HTTP 상태 코드: 200
- 응답 본문:

```
{
  "status": "success",
  "message": "파일서버 데이터 DB 저장이 시작되었습니다.",
  "timestamp": "2025-01-20T10:30:00"
}
```

테스트 데이터:

```
curl -X POST http://localhost:8083/api/crawler/save-fileserver \
  -H "Content-Type: application/json" \
  -d '{}'
```

TC-004: 파일서버 연결 실패 시 저장

테스트 목적: 파일서버 연결 실패 시 적절한 에러 처리 확인

전제 조건:

- 파일서버 연결 불가 상황

테스트 단계:

1. 파일서버 연결 차단
2. `POST /api/crawler/save-fileserver` 요청 전송
3. 에러 응답 확인

예상 결과:

- HTTP 상태 코드: 500
- 에러 메시지에 파일서버 연결 실패 내용 포함

1.3 크롤링 상태 확인 API 테스트

TC-005: 정상 상태 확인

테스트 목적: 크롤러 서비스 상태 확인 API 동작 검증

전제 조건:

- 크롤러 서비스 정상 실행 중

테스트 단계:

1. `GET /api/crawler/status` 요청 전송

2. 응답 상태 코드 확인
3. 응답 본문 필드 검증

예상 결과:

- HTTP 상태 코드: 200
- 응답 본문:

```
{
  "status": "running",
  "message": "크롤러 서비스가 실행 중입니다.",
  "timestamp": "2025-01-20T10:30:00",
  "service": "crawler-service",
  "port": "8083",
  "deployment-optimized": true
}
```

테스트 데이터:

```
curl -X GET http://localhost:8083/api/crawler/status
```

1.4 크롤러 설정 조회 API 테스트

TC-006: 설정 정보 조회

테스트 목적: 크롤러 설정 정보 조회 API 동작 검증

전제 조건:

- 크롤러 서비스 정상 실행 중

테스트 단계:

1. `GET /api/crawler/config` 요청 전송
2. 응답 상태 코드 확인
3. 설정 값들 검증

예상 결과:

- HTTP 상태 코드: 200
- 응답 본문에 모든 설정 필드 포함:
  - targetCount: 100
  - batchSize: 10
  - maxConcurrentRequests: 5
  - retryAttempts: 3
  - categories 배열 (9개 카테고리)
  - deployment-optimized: true

테스트 데이터:

```
curl -X GET http://localhost:8083/api/crawler/config
```

1.5 헬스 체크 API 테스트

TC-007: 헬스 체크 정상 동작

테스트 목적: 헬스 체크 API가 정상 동작하는지 확인

전제 조건:

- 크롤러 서비스 정상 실행 중

테스트 단계:

1. `GET /api/crawler/health` 요청 전송
2. 응답 상태 코드 확인
3. 헬스 상태 검증

예상 결과:

- HTTP 상태 코드: 200
- 응답 본문:

```
{
  "status": "UP",
  "service": "crawler-service",
  "deployment-optimized": true,
  "timestamp": "2025-01-20T10:30:00"
}
```

테스트 데이터:

```
curl -X GET http://localhost:8083/api/crawler/health
```

---

2. FTP 업로드 API 테스트

2.1 CSV 파일 업로드 (JSON) 테스트

TC-008: 정상 CSV 업로드

테스트 목적: JSON 방식 CSV 업로드가 정상 동작하는지 확인

전제 조건:

- FTP 서버 연결 가능
- 유효한 CSV 데이터 준비

테스트 단계:

1. 유효한 CSV 데이터로 요청 전송
2. 응답 상태 코드 확인
3. FTP 서버에 파일 업로드 확인

테스트 데이터:

```
{
  "path": "pm/2025-01-20_pm/list",
  "filename": "politics_list_2025-01-20-15-26.csv",
  "content": "제목,내용,카테고리,날짜\n정치뉴스1,정치내용1,POLITICS,2025-01-20\n경제뉴스1,경제내용1,ECONOMY,2025-01-20"
}
```

예상 결과:

- HTTP 상태 코드: 200
- 응답: "업로드 성공"
- FTP 서버에 파일 존재 확인

테스트 명령:

```
curl -X POST http://localhost:8083/api/ftp/upload \
  -H "Content-Type: application/json" \
  -d '{
    "path": "pm/2025-01-20_pm/list",
    "filename": "politics_list_2025-01-20-15-26.csv",
    "content": "제목,내용,카테고리,날짜\n정치뉴스1,정치내용1,POLITICS,2025-01-20\n경제뉴스1,경제내용1,ECONOMY,2025-01-20"
  }'
```

TC-009: 필수 필드 누락 시 업로드

테스트 목적: 필수 필드 누락 시 적절한 에러 처리 확인

테스트 단계:

1. 필수 필드(path, filename, content) 중 하나 누락
2. 요청 전송
3. 에러 응답 확인

테스트 데이터:

```
{
  "path": "pm/2025-01-20_pm/list",
  "filename": "politics_list_2025-01-20-15-26.csv"
  // content 필드 누락
}
```

예상 결과:

- HTTP 상태 코드: 500
- 에러 메시지에 필드 누락 내용 포함

TC-010: FTP 서버 연결 실패 시 업로드

테스트 목적: FTP 서버 연결 실패 시 적절한 에러 처리 확인

전제 조건:

- FTP 서버 연결 불가 상황

테스트 단계:

1. FTP 서버 연결 차단
2. 정상 CSV 데이터로 요청 전송
3. 에러 응답 확인

예상 결과:

- HTTP 상태 코드: 500
- 응답: "업로드 실패" 또는 "업로드 오류: [에러 메시지]"

2.2 파일 업로드 (MultipartFile) 테스트

TC-011: 정상 파일 업로드

테스트 목적: MultipartFile 방식 파일 업로드가 정상 동작하는지 확인

전제 조건:

- FTP 서버 연결 가능
- 테스트용 CSV 파일 준비

테스트 단계:

1. CSV 파일과 경로 정보로 요청 전송
2. 응답 상태 코드 확인
3. FTP 서버에 파일 업로드 확인

테스트 데이터:

- 파일: test_news_data.csv
- 경로: "pm/2025-01-20_pm/list"

예상 결과:

- HTTP 상태 코드: 200
- 응답: "파일 업로드 성공"
- FTP 서버에 파일 존재 확인

테스트 명령:

```
curl -X POST http://localhost:8083/api/ftp/upload-file \
  -F "file=@test_news_data.csv" \
  -F "path=pm/2025-01-20_pm/list"
```

TC-012: 파일 크기 초과 시 업로드

테스트 목적: 대용량 파일 업로드 시 적절한 처리 확인

전제 조건:

- 대용량 CSV 파일 준비 (예: 100MB 이상)

테스트 단계:

1. 대용량 파일로 요청 전송
2. 응답 확인
3. 업로드 성공 여부 확인

예상 결과:

- HTTP 상태 코드: 200 또는 500 (파일 크기에 따라)
- 적절한 처리 시간 확인

TC-013: 잘못된 파일 형식 업로드

테스트 목적: CSV가 아닌 파일 업로드 시 처리 확인

전제 조건:

- CSV가 아닌 파일 준비 (예: .txt, .jpg)

테스트 단계:

1. 잘못된 형식 파일로 요청 전송
2. 응답 확인
3. 업로드 결과 확인

예상 결과:

- HTTP 상태 코드: 200 또는 500
- 적절한 처리 확인

---

3. 통합 테스트

3.1 전체 크롤링 워크플로우 테스트

TC-014: 완전한 크롤링 프로세스

테스트 목적: 크롤링부터 FTP 업로드까지 전체 프로세스 검증

전제 조건:

- 모든 서비스 정상 실행
- 테스트 환경 완전 구성

테스트 단계:

1. 크롤링 시작 API 호출
2. 크롤링 상태 확인
3. 파일서버 데이터 저장 API 호출
4. FTP 업로드 API 호출
5. 전체 결과 검증

예상 결과:

- 모든 API 호출 성공
- 데이터베이스에 뉴스 데이터 저장
- FTP 서버에 파일 업로드
- 로그에 정상 처리 기록

3.2 부하 테스트

TC-015: 동시 요청 처리

테스트 목적: 여러 동시 요청 처리 능력 확인

테스트 단계:

1. 10개 동시 크롤링 시작 요청
2. 10개 동시 FTP 업로드 요청
3. 응답 시간 및 성공률 측정

예상 결과:

- 모든 요청 처리 완료
- 적절한 응답 시간 (5초 이내)
- 100% 성공률

TC-016: 대용량 데이터 처리

테스트 목적: 대용량 뉴스 데이터 처리 능력 확인

테스트 단계:

1. 1000개 뉴스 데이터로 크롤링
2. 대용량 CSV 파일 업로드
3. 처리 시간 및 메모리 사용량 측정

예상 결과:

- 모든 데이터 처리 완료
- 메모리 사용량 적정 수준
- 처리 시간 합리적 범위

---

4. 보안 테스트

4.1 인증 및 권한 테스트

TC-017: 인증 없는 접근

테스트 목적: 인증 없이 API 접근 시 처리 확인

테스트 단계:

1. 인증 헤더 없이 API 호출
2. 응답 확인

예상 결과:

- 적절한 인증 에러 응답
- 보안 로그 기록

TC-018: 잘못된 인증 정보

테스트 목적: 잘못된 인증 정보로 접근 시 처리 확인

테스트 단계:

1. 잘못된 인증 헤더로 API 호출
2. 응답 확인

예상 결과:

- 인증 실패 에러 응답
- 보안 로그 기록

---

5. 에러 처리 테스트

5.1 네트워크 에러 테스트

TC-019: 네트워크 타임아웃

테스트 목적: 네트워크 타임아웃 시 적절한 에러 처리 확인

테스트 단계:

1. 네트워크 지연 상황 시뮬레이션
2. API 호출
3. 타임아웃 처리 확인

예상 결과:

- 적절한 타임아웃 에러 응답
- 재시도 로직 동작 확인

TC-020: 서비스 다운 시 복구

테스트 목적: 서비스 다운 후 복구 시 동작 확인

테스트 단계:

1. 서비스 중단
2. API 호출 시도
3. 서비스 재시작
4. 정상 동작 확인

예상 결과:

- 서비스 다운 시 적절한 에러 응답
- 서비스 복구 후 정상 동작

---

6. 성능 테스트

6.1 응답 시간 테스트

TC-021: API 응답 시간 측정

테스트 목적: 각 API의 응답 시간 측정

테스트 단계:

1. 각 API 100회 호출
2. 응답 시간 측정
3. 평균, 최대, 최소 시간 계산

예상 결과:

- GET API: 100ms 이내

- POST API: 500ms 이내
- 파일 업로드: 파일 크기에 비례

6.2 메모리 사용량 테스트

TC-022: 메모리 누수 확인

테스트 목적: 장시간 운영 시 메모리 누수 확인

테스트 단계:

1. 24시간 동안 주기적 API 호출
2. 메모리 사용량 모니터링
3. 메모리 누수 여부 확인

예상 결과:

- 메모리 사용량 안정적 유지
- 메모리 누수 없음

---

7. 테스트 실행 계획

7.1 테스트 우선순위

1. **높음**: TC-001, TC-005, TC-007, TC-008, TC-011
2. **중간**: TC-003, TC-006, TC-009, TC-012, TC-014
3. **낮음**: TC-002, TC-004, TC-010, TC-013, TC-015~TC-022

7.2 테스트 환경 구성

```
# 테스트 서버 실행
docker-compose up -d crawler-service

# 테스트 데이터 준비
./scripts/prepare-test-data.sh

# 테스트 실행
./scripts/run-tests.sh
```

7.3 테스트 결과 보고서

- 테스트 실행 일시
- 테스트 케이스별 결과 (성공/실패)
- 성능 지표 (응답 시간, 처리량)
- 발견된 이슈 및 개선사항
- 권장사항

---

8. 자동화 테스트 스크립트

8.1 Postman Collection

```
{
  "info": {
    "name": "Crawler Service API Tests",
    "schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
  },
  "item": [
    {
      "name": "Crawler Management",
      "item": [
        {
          "name": "Start Crawling",
          "request": {
            "method": "POST",
            "url": "http://localhost:8083/api/crawler/start"
          }
        }
      ]
    }
  ]
}
```

8.2 JUnit 테스트 클래스

```
@SpringBootTest
@AutoConfigureTestDatabase
class CrawlerControllerTest {

    @Test
    void testStartCrawling() {
        // TC-001 구현
    }

    @Test
    void testGetStatus() {
        // TC-005 구현
    }
}
```

이 테스트 케이스 설계서를 통해 뉴스 크롤러 서비스의 모든 기능을 체계적으로 검증할 수 있습니다.