DongriGo Data Integrity Spec

Phase 1 Baseline

Version	[e.g., v1.0]
Date	2025-12-28
Author	Donghun Kim
Repository	https://github.com/lincoln-cmd/DongriGo
Production URL	https://dongrigo.onrender.com/
Related Commands	Country / Post / PostImage, seed_prod, check_integrity, ops_check

Table of Contents

1. 목적

운영 환경에서 다음 유형의 사고를 예방한다.

- 국가 ISO 코드 형식 오류(길이 불일치 등)로 인한 시딩/검증 실패
- Slug 변경, 중복으로 인한 링크 깨짐 및 URL 충돌
- PostImage 토큰 불일치로 인한 이미지 미출력, 깨짐
- 시딩을 반복 실행했을 때, 데이터 중복 삽입 또는 예기치 않은 덮어쓰기기

2. 적용 범위

- Country / Post / PostImage
- 시딩(seed_prod)
- 정합성 검사(check_integrity)

3. 정합성 규칙(Validation Rules)

3.1 Country

- ‘iso_a2’는 2자리여야 한다.
- ‘iso_a3’는 3자리여야 한다.
- ‘slug’는 비어 있으면 안된다 (필요 시 자동 생성).
- ‘aliases’는 쉽표와 공백 기준으로 정규화될 수 있다.

3.2 Post

- ‘slug’는 비어있으면 title 기반으로 자동 생성될 수 있다.
- 동일 country, category 범위에서 ‘slug’ 충돌이 없어야 한다.
- ‘is_published=True’인 경우, ‘published_at’는 설정되는 것이 권장된다. (경고 대상)

3.3 PostImage

- ‘order’는 표시 순서이며, 저장 시 자동으로 10 단위 간격으로 재정렬될 수 있다.
- 본문 이미지 삽입은 ‘[[img:<PostImage.id>]]’ 토큰을 사용한다.
- 토큰은 해당 Post에 연결된 PostImage만 치환 대상으로 인정한다.

4. 자동 정규화/보수 정책(Fix Policy)

정합성 검사는 아래 자동 보수(fix)할 수 있다.

- ISO 값의 trim, upper 같은 안전한 정규화
- aliases 포맷 정규화 (중복 제거, 구분자 통일)
- PostImage order 재정렬 (표시 안정성)
- 누락, 불일치 토큰 탐지 (수정 여부는 정책에 따라)

※ 어떤 항목을 자동으로 고칠지, 경고만 할지는 'check_integrity' 구현에 따른다.

5. 검사 커맨드 표준 실행(Standard Commands)

5.1 check_integrity

권장 실행:

- 로컬/운영 공통
  - python manage.py check_integrity –fix –json

목적:

- 데이터 정합성 검사
- 가능한 범위에서 자동 보수
- JSON 출력으로 로그, 증빙 저장

5.2 출력 포맷(JSON) 정의

목표: 정합성 검사, 보수 결과를 자동 기록 및 비교 가능하게 만들기

권장 JSON 구조(필수 필드):

- fix_applied (bool): 자동 보수 적용 여부
- summary (object): 전체 요약 (총 개수, 수정 개수, 오브젝트 타입별)
- counts (object): 경고, 변경, 누락 토큰 등의 카운트
- warnings_sample (array): 경고 샘플 N개 (개수 제한)
- changes_sample (array): 변경 샘플 N개 (개수 제한)
- missing_image_tokens_sample (array): 누락 토큰 샘플 N개

6. 운영 점검과의 관계

- 'ops_check'는 환경, 설정, 정적, DB, 마이그레이션, 시드 메타를 점검한다.
- 'check_integrity'는 데이터 내용의 정합성을 점검한다.
- 운영에서는 배포 직후 아래 순서가 권장된다.
  1) python manage.py migrate
  2) python manage.py ops_check –json
  3) python manage.py seed_prod
  4) python manage.py check_integrity –fix –json

7. 변경 관리(Change Control)

원칙: 데이터 규칙 변경은 운영 영향이 크므로 변경을 기록하고, 되돌릴 수 있어야 한다.

7.1 변경 유형

- 규칙 변경(Validation Rule Change)
  - ISO 길이 정책, aliases 파싱 규칙, slug 중복 처리 방식 등
- 자동 보수 정책 변경(Fix Policy Change)
  - 기존엔 경고만 하던 것을 자동 수정으로 전환
- 출력 포맷 변경(JSON Schema Change)
  - JSON 필드명, 구조 변경(로그 파서, 자동화 영향)

7.2 변경 절차

1. 변경 제안(이유, 영향 범위, 리스크) 작성
2. 로컬에서 fixture, seed, check_integrity, ops_check 전부 재검증
3. 운영 반영 전, 롤백 계획 준비
   - 코드 롤백(이전 커밋)
   - 데이터 롤백(운영 DB 백업 또는 최소한의 복구 절차)
4. 운영 배포 후 점검 실행

   migrate → ops_check –json → seed_prod → check_integrity –fix –json

5. 변경 로그 기록 (문서 버전, 커밋 해시, 적용 일시)

7.3 운영 영향 주의사항

- 모델 필드 제약 변경 (iso_a2 max_length 변경 등)은 마이그레이션이 동반되며, 기존 데이터와의 호환성을 사전에 확인해야 한다.
- slug 정책 변경은 기존 링크 보존(redirect)과 충돌 가능성이 있으므로, PostSlugHistory 정책과 함께 검토한다.
- JSON 포맷 변경은 자동 기록, 비교 도구가 있다면 함께 수정해야 한다.