상호운영성을 고려한 CMS 구축을 위한 기술적 기반 마련 방안

최미리*, 유동근**
*(주)지오인프라
**(주)지오인프라
e-mail:mlchoi@gioinfra.co.kr

A Technical Framework for Building Interoperable Content Management Systems (CMS)

Mi-Ri Choi*, Dong-Geun Yoo**
*Gioinfra Co., Ltd.
**Gioinfra Co., Ltd.

요 약

본 논문에서는 콘텐츠관리시스템(Content Management System, CMS)의 상호운영성 확보를 위한 기술적 기반을 분석하고, 이를 해결하기 위한 차세대 설계방안을 제안한다. 기존 CMS가 지닌 구조적 한계 -XML 기반 설정, 동기식 호출 방식, DBMS 종속성. 지속적인 통합 및 배포의 어려움 등-을 분석하고, React 기반의 비동기 구조, Spring Boot 설정 방식, CI/CD 자동화, Docker 기반 환경 일치화 등의 기술적 해결책을 모색하였다. 본 연구는 다양한 시스템과의 연계성을 강화, 확장성 확보로 효율적인 CMS 구축을 위한 실무적 가이드라인을 제공한다.

1. 서론

콘텐츠관리시스템(Content Management System, CMS)은 웹사이트 및 디지털 콘텐츠의 생성, 수정, 관리에 필수적인 도구로 자리 잡았다. 그러나 다양한 시스템과 플랫폼 간 데이터 및 서비스의 원활한 연동, 즉 상호운영성(interoperability)은 여전히 중요한 과제로 남아 있다. 상호운영성의 부재는 데이터 중복, 관리 효율성 저하, 사용자 경험 악화 등 여러 문제를 야기하며, CMS 구축 시 반드시 해결해야 할 핵심 요소로 인식되고 있다.

최근 클라우드 컴퓨팅, API 표준화, 마이크로서비스 아키텍처 등의 기술 발전은 CMS의 상호운영성을 강화할 수 있는 새로운 기회를 제공하고 있다. 하지만 이러한 기술들을 효과적으로 통합하고 적용하기 위한 기술적 기반 마련은 아직 초기 단계에 머물러 있으며, 관련 연구와 실무 적용 사례가 부족한 실정이다.

본 논문에서는 상호운영성을 고려한 CMS 구축을 위해 필요한 기술적 요소들을 분석하고, 이를 통합할 수 있는 체계적인 방안을 제시한다. 특히, 표준 기반의 데이터 교환, 모듈화된 시스템 설계, API 관리 및 보안 체계 구축을 중심으로 기술적 기반을 마련하는 전략을 탐구한다. 이를 통해 다양한 시스템 간 유연한 연동과 확장성을 확보하고, 효율적인 콘텐츠 관리를 지원하고자 한다.

2. 관련연구

2.1 기존 CMS 구축 분석

CMS의 상호운영성 문제는 다양한 연구에서 지속적으로 다루어져 왔다. Aicontentfy(2023)는 XML 및 JSON 기반의 데이터 표준화가 콘텐츠의 구조적 재사용 성과 변환 유연성을 높여 CMS 간 데이터 교환을 촉진한다고 보고하였다. 또한, Contentstacck(2023)은 RESTful API 설계가 시스템 간 연동성을 향상시키는 데 효과적이며, 동기식 호출 방식이 초기에는 단순한 구현과 명확한 흐름이라는 장점을 지녔으나, 클라우드 기반 환경에서는 응답 지연과 에러 처리 복잡성으로 인해 성능 저하를 초래할 수 있다고 지적하였다. 이와 함께, 특정 DBMS 벤더에 종속된 영속성 계층 구조는 성능 최적화에는 기여하였지만, 클라우드 환경에서의 유연성과 확장성 확보에는 제약을 발생시킨다는 비판도 제기되었다(Journal or Cloud Computing, 2016). 실제로 기존 CMS는 다음과 같은 기술적 문제를 내포하고 있다.

첫째, 설정 방식이 XML 기반으로 구성되어 있어 가독성과 유

지보수 측면에서 비효율적이며, 현대의 JSON 기반 API 구조와의 호환성에도 한계가 존재한다.
둘째, 시스템 구조가 모놀리식(monolithic) 형태로 설계되어 있어 기능 확장 및 변경이 어려우며, 프로젝트 설정이 수동으로 이루어져 반복적인 오류 발생 시 각 프로젝트마다 수정을 진행해야 하는 번거로움이 발생한다.
셋째, 동기식 데이터 호출 방식은 요청에 따라 서버 부하를 증가시키고, 불필요한 리소스나 데이터의 반복되는 호출 및 복잡한 예외 처리 로직을 구현함으로써 성능 및 안정성 측면에서 문제를 야기한다.
넷째, 사용자와 관리자 애플리케이션이 분리되어 있지 않아, 접근 권한 통제가 미흡하고 보안 취약점이 존재한다. 이와 같이 기존 CMS는 과거에는 각 기술 요소의 효율성에 기반하여 구축되었으나, 현대의 분산형 아키텍처 및 상호운영성 요구가 증대되는 환경에서는 여러 한계점이 노출되고 있으며, 이에 대한 통합적 대한 마련이 필요한 상황이다.

2.2 상호운영성 확보를 위한 기술 요소 분석

콘텐츠 관리 시스템(Content Management System, CMS)의 상호운영성 확보를 위해 다양한 기술 요소들이 도입되고 있다. 본 절에서는 이러한 기술 요소들의 발전 배경과 적용 사례를 분석하여, 향후 CMS 설계 방안의 기초 자료로 활용하고자 한다.

2.2.1 RESTful API를 통한 시스템 간 연동성 강화

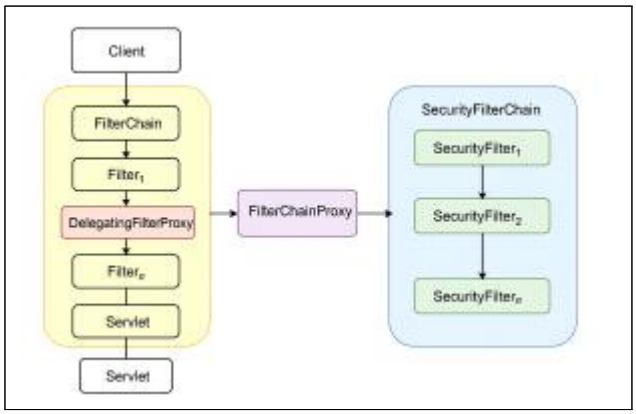
RESTful API는 완성된 HTML 문서 대신 필요한 데이터만 교환함으로써 네트워크 트래픽을 절감하며, 클라이언트–서버 분리를 통해 UI 수정 및 기능 확장 시 유연한 대응을 가능케 한다. 또한, 동일 API의 재사용을 통해 중복 개발을 감소시키고 유지보수를 단순화한다. 이러한 구조적 장점은 서비스 확장성과 다양한 플랫폼 지원에 유리한 기반을 제공한다.

2.2.2 RESTful API를 통한 시스템 간 연동성 강화

개발 환경과 운영 환경의 분리는 시스템의 안정성과 보안성을 확보하는 데 중요하다. 이를 위해 Docker와 Kubernetes와 같은 컨테이너 기술이 도입되어, 일관된 환경에서의 개발과 배포를 가능하게 한다. 또한, GitHub Actions와 같은 CI/CD 도구를 활용하여 테스트 및 배포 과정을 자동화함으로써, 개발 효율성과 시스템의 신뢰성을 향상시키고 있다.

2.2.3 보안 강화를 위한 인증 및 권한 관리

CMS의 보안성을 강화하기 위해, Spring Security와 같은 프레임워크를 활용하여 인증 및 권한 관리를 체계화하고 있다[1].



[그림 1] Spring Security 흐름도

이를 통해 사용자와 관리자 애플리케이션의 분리를 구현하고, 역할 기반의 접근 제어를 통해 시스템의 보안성을 확보하고 있다.
이러한 기술 요소들은 CMS의 상호운영성을 확보하고, 시스템의 확장성과 개발 편의성을 향상시키는 데 기여하고 있다.

3. 제안한 차세대 CMS 설계 방안

본 장에서는 상호운영성을 기반으로, 기존 CMS의 구조적 한계를 극복할 수 있는 차세대 콘텐츠 관리 시스템(Content Management System, CMS)의 설계 및 개발 방법론을 제안한다. 제한하는 시스템은 기술적 모듈화, 비동기식 통신, 자동화된 배포 체계, 보안 강화, 환경 일치성 확보를 핵심 원칙으로 하여, 다양한 서비스와의 유연한 연동과 안정적인 운영을 지향한다.

3.1 모듈 기반의 설계

기존 CMS의 단일 구조(Monolithic Architecture)는 기능을 추가하거나 유지보수를 진행할 때 시스템 전체에 영향을 주기에 대규모 기능 확장 시 복잡도가 기하급수적으로 증가한다. 또한, 모듈 간 분리도가 낮아 특정 기능에 대한 수정이 전체 애플리케이션의 성능과 안정성에 직결되는 문제가 발생한다. 이에 대한 대안으로 제안된 모듈 기반 설계는 확장성과 유지보수 편의성을 동시에 확보하기 위한 방법론으로 주목받고 있다.

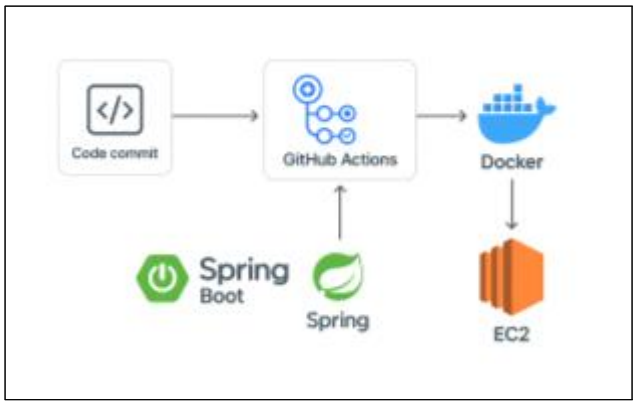
3.2 비동기 통신 구조

기존의 서버 렌더링 방식은 클라이언트가 페이지 요청 시마다 서버로부터 전체 HTML 문서를 반복적으로 수신해야 하므로, 네트워크 리소스의 낭비와 서버 부하를 유발하는 단점이 존재한다. 이는 빈번한 페이지 전환 과정에서 사용자 경험을 저하시킬 뿐 아니라 트래픽 관련 비용 증가라는 문제로 이어진다. 이러한 한계를 극복하기 위한 대안으로 SPA(Single

Page Application) 기반의 비동기 통신 구조가 등장하였으며, 이는 변경이 필요한 부분만 선택적으로 재요청하여 화면을 동적으로 갱신하는 방식으로 문제 해결에 기여한다.

3.3 코드 기반 설정 및 자동화된 배포 환경

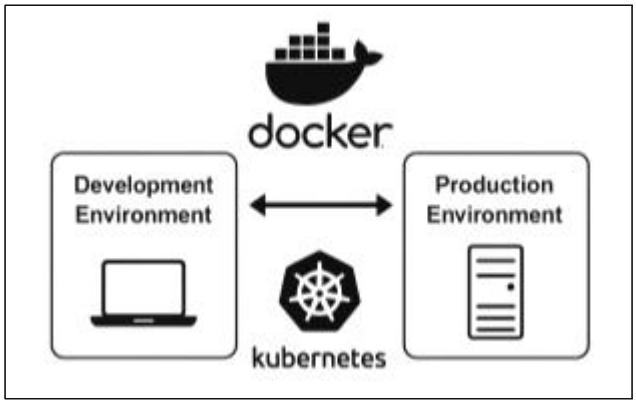
설정 방식은 XML 기반의 정적 파일에서 Java 코드 기반의 명시적 설정 방식으로 전환하며, Spring Boot 프레임워크를 활용하여 설정의 명확성과 오류 추적성을 높였다. 또한, GitHub Actions를 활용한 CI/CD(지속적 통합 및 배포) 파이프라인을 구축하여[3], Code Commit 시 자동으로 테스트 및 배포가 이루어지도록 구성하였다. 이러한 자동화는 배포와 형상 관리 과정에서 일어나는 실수나 오류를 줄이고 운영 안정성을 확보하는 데 효과적이다.



[그림 2] GitHub Actions 구성도

3.4 개발-운영 환경 일치성

Docker 및 Kubernetes를 통해 개발 환경을 운영 환경과 동일하게 구성함으로써, 테스트 환경과 실제 서비스 환경 간의 차이를 최소화한다[2]. 이는 예외 상황을 사전에 탐지하고, 운영 시 발생 가능한 문제를 미연에 방지하는 데 기여한다.



[그림 3] Docker-Kubernetes 환경 구성도

3.5 Caching을 활용한 성능 최적화

접근 빈도가 높은 데이터(최근 게시글, 웹사이트 메타 데이터 등)에 캐싱 방식을 적용하면 데이터베이스 조회 및 웹 애플리케이션의 트랜잭션에서 발생하는 오버헤드를 최소화할 수 있다. 이는 사용자가 경험하는 응답 속도를 향상시키고 서버의 부하를 효과적으로 감소시키는 장점이 있다.

3.6 보안 및 유지보수성 향상

사용자와 관리자 시스템을 완전히 분리하고, Spring Security 기반의 인증 및 권한 체계를 구축하여 접근 제어를 강화하였다. 또한, 개발 단계에서 언어 혹은 프레임워크가 제공하는 Assertion 기반의 예외 검증 로직을 삽입하여 비즈니스 로직에 대한 명확한 요구 조건을 표현하고, 코드 구현 로직에서 사전 조건 실패 시 즉시 예외 처리를 진행함으로 인해 의도하지 않은 데이터 흐름이나 조건 누락을 조기에 차단하도록 하였다[4].

4. 결론

본 논문에서는 콘텐츠 관리 시스템(Content Management System, CMS)의 상호운영성 확보를 중심 과제로 삼고, 기존 CMS 구축 방식의 기술적 한계를 분석한 후, 이를 보완할 수 있는 차세대 CMS 설계 방안을 제시하였다. 먼저, XML 기반 설정, 동기식 호출 방식, 단일 애플리케이션 구조, DBMS 종속성 등 기존 시스템의 문제점을 분석하였으며, 이는 유지보수의 어려움, 확장성 저하, 에러나 버그의 높은 발생 빈도 등 실질적인 운영상의 한계로 이어짐을 확인하였다. 이러한 문제를 극복하기 위해, 본 연구에서는 비동기식 호출 구조, 코드 기반 설정 방식, 모듈러 모놀리식(Modular Monolithic) 구조, 지속적인 테스트 및 배포(CI/CD), 환경 일치성 확보(Docker/Kubernetes), 인증 및 권한 관리(Spring Security) 등 다양한 최신 기술 요소를 통합한 차세대 CMS 설계를 제안하였다. 제안된 구조는 상호운영성 뿐 아니라 보안성, 확장성, 운영 효율성이 측면에서도 개선된 성능을 기대할 수 있다. 특히, 본 논문은 기존 기술들을 개별적으로 적용하는 것이 아닌, 상호 연계되는 구조로 통합하여 설계함으로써, 복잡한 CMS 운영 환경에서의 실질적인 대안을 제시했다는 점에서 학술적 의의가 있다. 향후 연구에서는 제안된 시스템을 기반으로 한 실증적 구현 및 성능 비교, 사용자 관점의 경험 분석 등을 통해 보다 구체적인 평가가 이루어질 필요가 있다.

참고문헌

[1] Mazharul Islam 외, “Coding Practices and Recommendations of Spring Security for Enterprise Applications,” arXiv:2007.14319, 2020.

[2] Varnsi Krishna Yepuri 외, “Containerization of a Polyglot Microsrvice Application Using Docker and Kubernetes,” arXiv:2305.00600, 2023.
[3] Sk Golam Saroar 외, “Developers’ Perception of GitHub Actions: A Survey Analysis,” arXiv:2303.04084, 2023.
[4] Valerio Terragni 외, “GAssert: A Fully Automated Tool to Improve Assertion Oracles,” arXiv:2103.02901, 2021.