REST API 기반의 통합 아키텍처 연구 사례

Research practices on Integrated Architecture based on REST API

저자 (Authors)
안현식, 박보경, 김영철, 이재협
Hyun Sik An, Bo Kyung Park, R. Young Chul Kim, Jaehyub Lee

출처 (Source)
JOURNAL OF PLATFORM TECHNOLOGY 7(1), 2019.3, 3-9(7 pages)

발행처 (Publisher)
ICT플랫폼학회
Information Communication Technology Platform

URL
http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07997600

APA Style
안현식, 박보경, 김영철, 이재협 (2019). REST API 기반의 통합 아키텍처 연구 사례. JOURNAL OF PLATFORM TECHNOLOGY, 7(1), 3-9

REST API 기반의 통합 아키텍처 연구 사례

[1]안현식, [2]박보경, [*3]김영철, [4]이재협
[1 제 1 저자]홍익대학교 소프트웨어 공학 연구실, ahn@selab.hongik.ac.kr
[2]홍익대학교 소프트웨어 공학 연구실, park@selab.hongik.ac.kr
[*3 교신저자] 홍익대학교 소프트웨어 공학 연구실, bob@hongik.ac.kr
[4] 한국기술교육대학교 컴퓨터공학부, jae@koreatech.ac.kr

Research practices on Integrated Architecture based on REST API

[1]Hyun Sik An, [2]Bo Kyung Park, [3]R. Young Chul Kim, [4]Jaehyub Lee
[12,*3] SELab. Hongik University, {[1]ahn, [2]park, [*3]bob}@selab.hongik.ac.kr
[4]School of Computer Science and Engineering, Koreatech, jae@koreatech.ac.kr

요 약

국내 과학기술 지식 인프라는 과학기술 정보 수집, 가공 활용을 위한 포털의 형태로 제공되고 있으나 거버넌스를 위한 사이언스 아키텍처 적용을 통한 개선이 필요하다. 현행 S&T 지식 정보 인프라는 개별 정보 서비스로 인한 서비스 결과의 제한적 전달이 발생하여 연구 활동 관리에 필요한 서비스 연동을 통한 신규 지식 획득과 융합연구 지원의 제약점을 갖게 된다. 이를 위해서는 성공적인 시스템 통합 전략과 추진 방법을 제시하고, 각 조직의 시스템이 시행착오를 겪지 않고 효과적인 통합 방안이 제시되어야 한다. 본 연구에서는 REST API 방법을 통해 기존 시스템의 서비스와 컴포넌트를 활용하여 신규 통합 서비스를 효과적으로 개발하기 위한 방법을 제안한다.

Abstract

The domestic science and technology knowledge infrastructure is provided in the form of a portal for collecting and processing science and technology information. Therefore improvement through the application of open science architecture for governance is necessary. Current S & T's knowledge information infrastructure will generate limited delivery of service results by individual information services, and will have new knowledge acquisition through integration of services necessary for managing research activities and limitations of fusion research support become. To that end, a successful system integration strategy and promotion method should be presented, and the system of each organization should be presented with an effective integration proposal without trial and error. In this study, we provide a method of effectively developing new integrated services by utilizing the services and components of existing systems using REST API Mechanism.

***Keywords**: REST API, Service Architecture, Functional Component, Micro Service, Software Architecture Style*

Ⅰ. 서론

최근 디지털 기술의 확산으로 연구의 성과와 과정이 개방화되고 있으며, 이러한 추세를 '오픈 사이언스'라 지칭한다. 이 오픈 사이언스를 위한 플랫폼 개발이 활발하게 이루어지고 있다.

* Corresponding Author
Received: Mar. 19, 2019, Revised: Apr. 02, 2019, Accepted: Apr. 05, 2019

오픈 사이언스의 증가로 온라인을 통한 학술 정보 열람이 증가하고 연구데이터의 공개, 공유 및 온라인 연구협력과 같은 연구협력을 지원하는 디지털 인프라 및 서비스가 확대되고 있다 [1].

현행 S&T 지식정보 인프라는 개별 정보서비스를 통해 서비스 결과 전달이 사용자에게 제한적으로 제공됨으로써 정보섬(information island) 문제를 가질 수 있다. 이를 해결하기 위해선 각 조직의 시스템을 융합 할 수 있는 시스템 통합 방안이 제시되어야 한다.

본 논문에서는 REST API 방법을 통해 기존 시스템의 서비스와 컴포넌트를 활용하여 신규통합서비스를 효과적으로 개발하기 위한 방법을 제안한다. 이 과정에서 컴포넌트간의 유연한 상호연동성과 범용적인 인터페이스, 컴포넌트의 독립적인 배포를 설계하고 레거시 시스템을 인캡슐레이션하는 중간 컴포넌트 역할을 수행한다.

본 논문은 다음과 같은 순서로 이루어진다. 2 장에서는 관련 연구로 REST API 에 대해서 설명한다. 3 장에서는 과학기술 지식 인프라의 조직들에게 어떻게 REST API 를 적용시킬지 방안에 대해서 제안한다. 4 장은 결론 및 향후 연구에 대한 내용이다.

II. 관련 연구

2.1 REST API

REST 는 Representational State Transfer 의 약자로, 월드와이드웹과 같은 분산 하이퍼미디어 시스템에서 운영되는 소프트웨어 아키텍처 스타일이다. 이는 웹(HTTP)의 공동 창시자인 Roy Thomas Fielding 이 처음 제안한 방법이다. 기존에 다양한 웹 아키텍처 스타일이 정의되어 있지만, 현대 웹 아키텍처의 특성을 보다 잘 반영하기 위한 네트워크 기반의 아키텍처이다.

REST 는 주로 HTTP/1.0 사양과 초기 HTTP/1.1 을 기반으로 개발하면서 웹 개념을 전달하는 수단으로 사용됐다. 가장 큰 특징은 모든 자원을 리소스로 표현하는 ROA(Resource Oriented Architecture)를 기반으로 함으로써 Hypertext Transfer Protocol(HTTP)의 Uniform Resource Identifiers(URI)와 메소드를 간단하게 사용할 수 있다. 즉, 웹상의 리소스들을 HTTP 에서 SOAP 이나 쿠키를 통한 세션 트래킹 등의 별도 전송 계층 없이 전송할 수 있는 간단한 인터페이스이다. 또한 여러 네트워크 기반 아키텍처 스타일에서 파생된 하이브리드 스타일로, 균일한 커넥터 인터페이스를 정의하는 제약 조건을 제공한다 [2].

2.2 REST API 의 설계목표

데이터 요청 및 전송을 위해 다양한 기술들이 등장했다. 이 방식들이 복잡하고 어려웠던 문제를 해결하기 위해서 웹을 활용해보려는 시도에서 REST 가 등장하게 되었다. REST 는 HTTP 를 그대로 적용해보자하는 시도를 통해 개발되었고 REST 의 핵심설계 목표는 다음과 같다.

- 컴포넌트간의 유연한 상호연동성

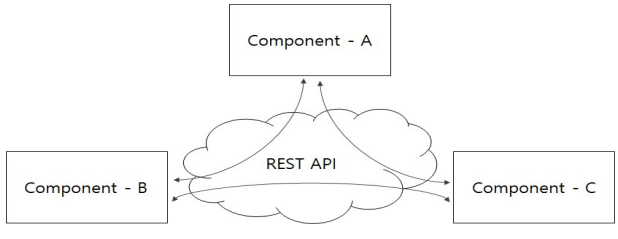

Figure 1. The interoperability using REST API

<그림 1>은 REST API 를 통한 컴포넌트 간의 상호 연동성을 보여준다. 이는 서로 다른 두 개 이상의 컴포넌트를 결합함으로써 더 효율적인 서비스를 제공하거나 작업을 수행하는데 목적이 있다.

- 범용 인터페이스

표준으로 사용되고 있는 HTTP 와 URI 를 기준으로 하므로, 어디서든지 사용 가능한 범용 인터페이스를 제공한다.

- 컴포넌트의 독립적 배포

컴포넌트들은 REST 를 기반으로 연동되어 사용되고 있지만, 이들은 완전히 독립된 프로그램으로 독립적인 배포 및 개발이 가능하다.

- 지연 감소, 보안강화, 레거시 시스템을 인캡슐레이션하는 중간 컴포넌트 역할

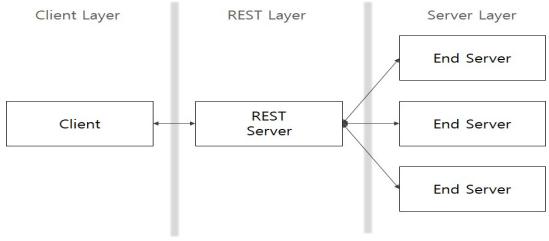

Figure 2. The intermediary REST layer between client and server

REST 서버가 클라이언트와 엔드 서버 중간에서 중간 컴포넌트의 역할을 수행하므로 클라이언트는 엔드 서버에 직접 연결할 필요 없이 서비스를 이용할 수 있다. <그림 2>는 클라이언트와 서버의 중계자 역할을 하는 REST 서버를 나타낸다. 이 구조를 통해서 로드 밸런싱, 공유 메모리(캐쉬)등을 이용할 수 있고, 가용성, 확장성, 성능을 향상시키고 보안 정책을 적용하기도 간단하고, 대표적으로 오픈소스 솔루션 NGINX, WSO2 가 있다 [3,4].

III. REST API 기반 통합 서비스 적용 방안

마이크로서비스 아키텍처를 사용하여 소프트웨어 응용 프로그램을 개발하거나 기존 응용 프로그램/서비스를 마이크로서비스로 변환 할 수 있다. 어떤 경우든 마이크로서비스의 크기, 범위 및 기능을 적절히 결정하는 것이 중요하고, 이를 통해 서비스 개발 및 제공의 민첩성을 충족시킬 수 있다. 고려 사항은 다음과 같다 [5]. 고려 사항은 다음과 같다.

- 서비스들의 경계를 찾고 이를 비즈니스 역량과 비교
- 마이크로 서비스 설계가 민첩하고 독립적인 개발 및 서비스를 보장하는지 확인
- 크기 및 범위는 주어진 비즈니스 기능을 용이하게 하는 것이 목적
- SOA 와 달리 마이크로서비스는 매우 적은 기능과 단순한 메시지 포맷을 가짐
- 광범위한 서비스에서 시작하여 작은 요구사항으로 리팩토링

3.1 마이크로서비스의 메시징

모놀리식 응용 프로그램에서는 여러 프로세서/구성요소의 비즈니스 기능이 함수 호출이나 언어 수준 메소드 호출을 통해 서비스 간의 메시징이 이루어진다. REST 기반의 융합 서비스에서는 좀 더 느슨하게 결합된 웹 서비스 수준 메시징으로 변환되었고 HTTP 프로토콜을 기반으로 한다.

- 동기식 메시징 – REST
  : 클라이언트가 서비스의 응답을 기다린다. 이는 모든 자원이 리소스로 표시되는 REST API 를 기반으로 HTTP 요청-응답으로 구현되는 간단한 메시징 스타일을 제공한다. <그림 3>과 각 서비스의 REST API 를 개발해야 한다.

Figure 3. Micro services based on synchronous messaging

3.2 마이크로서비스의 통합

마이크로서비스 아키텍처에서 소프트웨어 응용 프로그램은 독립적인 서비스로 구성된다. 따라서 비즈니스 유스 케이스를 실현하려면 다양한 마이크로서비스/프로세스 간에 통신 구조가 있어야 한다.

- API-Gateway style
  : 모든 클라이언트를 위한 기본 진입점으로 <그림 4>와 같이 간단한 메시지 게이트웨이를 사용하고 이 수준에서 일반적인 비기능 요구사항을 구현한다.

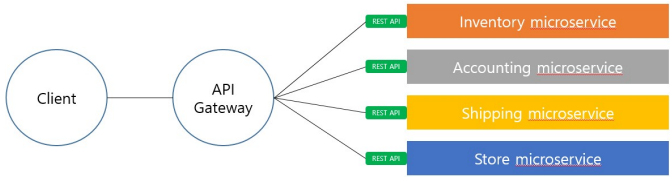

Figure 4. Micro service integration based on API-Gateway style

이를 통해서 REST/HTTP 를 통해 관리되는 API 를 사용할 수 있고 다음과 같은 장점을 제공한다.

- 기존 마이크로서비스의 게이트웨이 레벨에서 필요한 추상화 제공
- 경량화된 메시지 라우팅 및 변환
- 보안, 모니터링과 같은 비기능적 요구사항 적용 가능
- 마이크로 서비스 경량화

3.3 분산 데이터 관리

마이크로서비스 아키텍처에서는 기능이 분산되어 있으므로 동일한 중앙 데이터베이스를 사용하면 각 서비스가 독립적이지 않은 구조를 갖게 된다. 따라서 각 마이크로서비스들은 자체 데이터베이스를 갖는다. 분산 데이터 관리의 핵심 요소는 다음과 같다.

- 각 마이크로서비스에는 개별 데이터베이스에만 액세스 할 수 있고 타 데이터베이스에는 액세스 불가
- 일부 비즈니스 시나리오에서는 단일 트랜잭션에 대해 여러 데이터베이스를 갱신해야 할 수도 있으므로, 이러한 경우 API 를 통해서 업데이트 가능함

3.4 분산 거버넌스

마이크로서비스 아키텍처에서는 다양한 기술과 플랫폼으로 완벽하게 독립적이고 분리된 서비스가 구축된다. 따라서 다음과 같이 분산된 거버넌스는 다음과 같은 특징을 갖는다:

- 마이크로서비스 아키텍처에서는 중앙 집중식 Design-time 관리를 요구하지 않음
- 마이크로서비스는 설계 및 구현에 대한 자체 결정 가능
- 마이크로 서비스 아키텍처는 공통/재사용 가능한 서비스의 공유 촉진
- 모니터링, 보안 요구사항, 서비스 검색과 같은 Run-time 관리 기능 중 일부는 API Gateway 수준에서 구현 가능

3.5 서비스 레지스트리 및 서비스

마이크로서비스 아키텍처에서 서비스의 수는 상당히 많고 신속하고 민첩한 개발/배치 특성으로 인해 위치가 동적으로 변경된다. 따라서 런타임 중 마이크로서비스 위치를 찾아야 한다. 이 문제에 대한 해결책은 서비스 레지스트리와 서비스 발견 기능을 사용하는 것이다.

① 서비스 레지스트리
: 마이크로서비스 인스턴스는 시작할 때 서비스 레지스트리에 등록되고, 종료 시 등록이 취소된다. 소비자는 레지스트리를 통해 사용 가능한 마이크로서비스와 그 위치를 찾는다.

② 서비스 발견
: 사용 가능한 마이크로서비스와 그 위치를 찾기 위한 서비스 발견 메커니즘이 필요하고 클라이언트 발견과 서버 발견 두 가지 유형이 존재한다.

③ 클라이언트 발견
: <그림 5>와 같이 클라이언트 또는 API Gateway 는 서비스 레지스트리에 질의하여 서비스 인스턴스의 위치 데이터를 전달받는다.

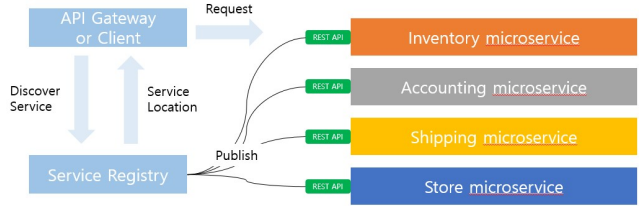

Figure 5. Service discovery on the client

① 서버 발견
: <그림 6>과 같이 클라이언트 또는 API Gateway 가 전달받은 서비스 인스턴스 위치로 요청을 보낸다.

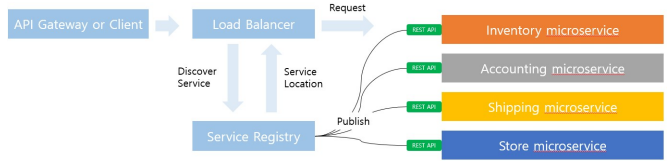

Figure 6. Service discovery on the server

3.6 전개

마이크로서비스 아키텍처에서 각 마이크로서비스는 독립적 배포, 서비스 수준 확장, 신속한 구축이 가능하고 특정 마이크로서비스의 오류가 타 서비스에 영향을 주지 않아야 한다. 이 요구사항을 해결하여 서비스를 배포하기 위해 컨테이너를 적용한다. 대표적인 컨테이너는 Docker 로 이를 사용하는 주요 단계는 다음과 같다.

① 마이크로서비스를 Docker 컨테이너 이미지로 패키징
② 각 서비스 인스턴스를 컨테이너로 배포
③ 확장은 컨테이너 인스턴스 수를 변경하여 수행
④ 빠른 개발, 배포, 실행 지원

3.7 보안

마이크로서비스 아키텍처에서는 OAuth2 및 OpenID Connect 와 같이 널리 사용되는 API 보안 표준을 활용할 수 있다.

- OAuth2
  : 액세스 위임 프로토콜로 클라이언트 권한을 인증 받고 '액세스 토큰'을 확보한다. 이는 사용자/클라이언트에 대한 정보가 포함되어 있지 않고 인증 서버에서만 검색 가능한 참조만을 갖는다. 공공 네트워크/인터넷에서도 안전하게 적용된다 [6].
- OpneID Connect
  : OAuth 와 유사하지만 액세스 토큰 외에도 인증 서버가 사용자/클라이언트 정보가 포함된 'ID 토큰'을 발행한다. 이는 주로 JWT(JSON Web Token)에 의해 구현되며 인증서버에 의해 서명된다 [7].

<그림 7>과 같이 이러한 API 보안 표준은 마이크로서비스 보안을 구현하는 주요한 요소이다. OAuth 와 OpenID Connect 서버에 인증을 남겨놓으면 마이크로서비스가 누군가의 데이터를 사용할 권한을 부여받으면 성공적으로 액세스가 가능하다. 이 경우 모든 클라이언트 요청에 대해 단일 진입점이 있는 API Gateway 스타일을 사용한다. 클라이언트는 권한 서버에 연결하여 액세스 토큰을 얻고 이와 함께 API Gateway 로 요청 메시지를 보낸다. API Gateway 에서는 토큰과 요청을 마이크로서비스에 전달한다.

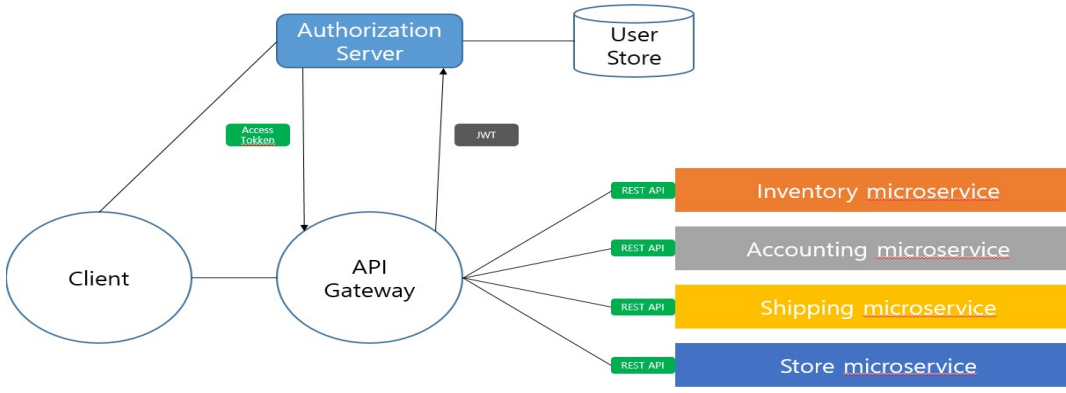

Figure 7. Microservice security mechanism

IV. 결론

REST API 는 월드와이드웹과 같은 분산 하이퍼미디어 시스템에서 운영되는 소프트웨어 아키텍처 스타일이다[8]. 현대 웹 아키텍처의 특성을 보다 잘 반영하기 위하여 설계단계에서 컴포넌트간의 유연한 상호연동성, 범용적인 인터페이스, 컴포넌트의 독립적 배포가 가능하도록 설계하였고 이를 위해 API-Gateway Style 을 제안하였다. 이로인해 기존 마이크로서비스의 게이트웨이 레벨에서 필요한 추상화를 제공하고, 경량화된 메시지 라우팅 및 변환이 가능하다.

V. 감사의 글

본 논문은 한국과학기술정보연구원이 지원하는 2018 년도 위탁연구과제(No. P18014)와 2017 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2017R1D1A3B03035421)의 연구수행으로 인한 결과물 임을 밝힙니다.

VI. 참고문헌

[1] Open Science Definition, https://www.fosteropenscience.eu/foster-taxonomy/open-science-definition
[2] Roy T. Fielding, Richard N. Taylr, Justin R. Erenkrantz, Michael M. Gorlick, Jim Whitehead, Rohit Khare, Peyman Oreizy, “Reflection on the REST architectural style and ‘Principled design of the modern web architecture’”, Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp. 4-14, 2017.
[3] NGINX, Chris Richardson of Eventuate, Inc., Building Microservices : Using an API Gateway, May 19, 2015., https://www.nginx.com/blog/introduction-to-microservices/
[4] WSO2, Kasun Indrasiri, Microservices in Practice-Key Architectural Concepts of an MSA, 08, 2016.
[5] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty, Mike Amundsen, O’REILLY, “Microservice Architecture”
[6] OAuth2, https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2
[7] OpenID Connect, https://openid.net/connect/
[8] REST API, https://www.mulesoft.com/resources/api/what-is-rest-api-design