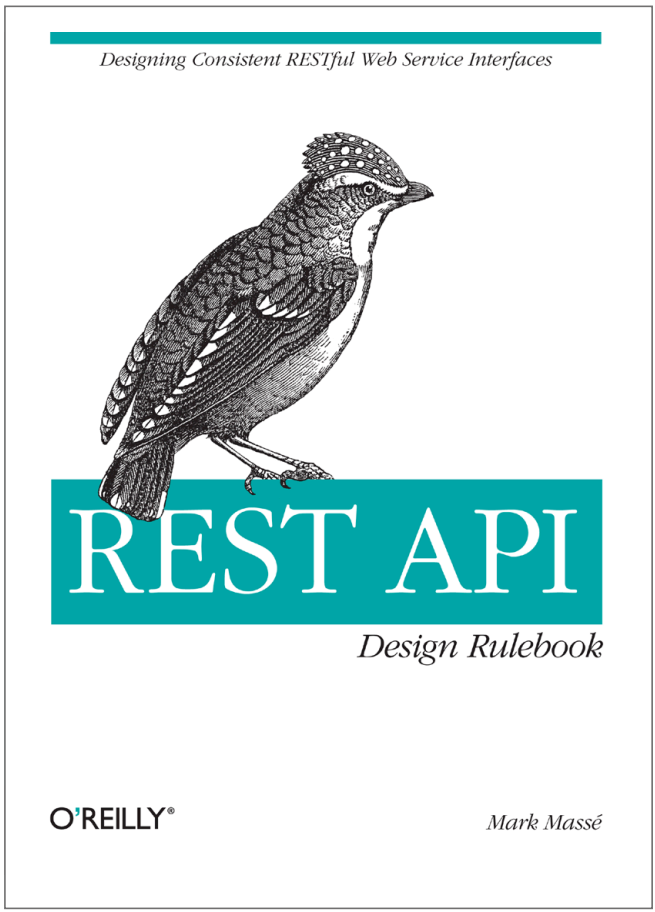

Hanbit eBook
Realtime 06
일관성 있는 웹 서비스 인터페이스 설계를 위한
REST API
디자인 규칙
REST API Design Rulebook
마크 마세 지음 / 김관래, 권원상 옮김
O'REILLY®
한빛미디어
Hanbit Media, Inc.



이 도서는 O'REILLY의
REST API Design Rulebook의
번역서입니다.

일관성 있는 웹 서비스 인터페이스 설계를 위한

REST API 디자인 규칙

일관성 있는 웹 서비스 인터페이스 설계를 위한 **REST API 디자인 규칙**

초판발행 2012년 10월 18일

지은이 마크 마세 / **옮긴이** 김관래, 권원상 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-7914-945-6 15560 / **정가** 11,000원

책임편집 배용석 / **기획** 김창수 / **편집** 김병희, 안선화
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
영업 김형진, 김진불, 조유미 / **마케팅** 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanb.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 소개

저자 _**마크 마세**

마크 마세는 ESPN의 엔지니어링 담당 시니어 엔지니어 디렉터로, 시애틀에 살고 있다. 월트디즈니사에서 14년 동안 엔지니어링, 관리, 아키텍처 관련 일을 하였으며 ESPN 스포츠 존, NFL.com, NASCAR 온라인 등에서 사용되는 인터랙티브 자바 애플릿을 개발하는 스타웨이브 사에서 경력을 쌓았다. 마크는 ESPN.com, ABC.com, Disney.com을 포함한 디즈니 웹 사이트에서 사용되는 콘텐츠 관리 시스템(CMS)을 설계하고 개발하였다. 2008년에는 웹 페이지를 생성하는 데 사용되는 데이터 모델을 결정하기 위한 시스템 및 방법을 개발하여 디즈니 발명상Disney Inventor Award을 수상하였다.

역자 소개

역자_ **김관래**

삼성전자에서 웹 관련 프로젝트를 수행하였다. 서버단 REST API 디자인을 하였고, 프론트단 프로토타이핑과 테스트를 진행하였다. KT로 이직하여 ucloud open API 테스트와 SDK 제작에 참여하였다.

Flume 등 오픈 소스 관련한 스터디를 진행하고 있으며, 최근에는 R 언어 등을 이용한 데이터 프로세싱/비주얼라이제이션을 공부하고 있다. 아두이노와 같은 오픈 소스 하드웨어 플랫폼과 소프트웨어의 적절한 조합에도 관심이 많다.

지속가능한 엔지니어링은, 문제를 해결함에 있어 '디자인+디벨롭먼트'의 조화로운 균형이라 생각하고 있다.

역자_ **권원상**

배우고 알게 된 것을 다른 사람과 나누는 것을 좋아한다. 지난 십 수년간 교육용 CD-ROM 타이틀에서 웹 사이트, 디지털 아카이브 시스템, 홈 네트워크까지 여러 분야에서 개발자로 일 해왔다. 최근 몇 년간은 파워포인트로 일하는 시간이 많아서, 스스로에게 아쉬워할 때가 많다. 컴퓨터에서 발생하는 오류보다 사람 사이에 발생하는 오해가 훨씬 더 해결하기 어려울 때가 많다는 것을 매일매일 확인하며 살고 있다.

역자 서문

소프트웨어와 관련된 일을 하면서 알게 된 중요한 사실은 소프트웨어 개발도 결국은 '소통의 문제'로 귀결된다는 것이다. 요구 사항을 수집하고, 설계를 공유하고, 코딩하고, 테스트하는 과정에서 컴퓨터와 사람, 사람과 사람의 소통이 일어난다. 그리고 소통이 제대로 될 때 잘 정리된 소프트웨어를 만들 수 있다.

이 책은 '웹 서비스 개발자' - '웹 서비스' - '웹 서비스 클라이언트' 사이에 소통을 위해서, REST API를 어떻게 이해하고 쓰면 좋을지를 설명한다. REST 스타일의 API를 '사용'하는 것은 이미 많은 개발자에게는 익숙한 일이다. 그러나 REST 스타일의 API를 다른 사람이 쓸 수 있도록 '정의'하는 일에 익숙한 개발자는 많지 않을 것이다. 그냥 가져다 쓸 때보다 고려해야 할 것이 많은, 이 또한 소통의 문제이기 때문이다.

REST API 정의에 익숙하지 않은 사람들에게 이 책은 REST API를 디자인할 때 고려해야 할 것들을 짧은 문장의 규칙으로 정리해 둔 가이드북이다. 여행 가이드북은 방안에 앉아서 쭉 읽어 보는 것도 좋지만, 아무래도 그곳에 가서 직접 가이드북을 보면서 여행할 때 더 재미가 있다. 이 책에 나온 규칙들을 REST API로 하나씩 만들고 정의해 보면서 읽을 때 더 도움이 될 것이다.

마지막으로 책을 번역하면서 한빛의 스마트 미디어팀과 소통은 잘 되었는지를 생각했다. 이북으로 출판하는 것도 익숙치 않았고 여러 가지 문제들도 있었는데, 모든 것을 꾹 참고 소통을 시도해 준 김병희님께 감사드린다. 뒤에서 배후 조종하셨을 것으로 추정되는 김창수 팀장님께도 감사드린다. (다음번에는 '사람과 사람' 사이의 소통에 대한 이북도 한 번 기획해주세요)

무엇보다 회사 다니면서, 바쁜 시간을 쪼개서 책을 번역하고 정리한 우리 스스로 그동안 수고했다고 말해 주고 싶다. 수고했다. (남과의 소통만큼 자신과의 소통도 중요하다^^)

번역을 마치고 **김관래, 권원상**

한빛 리얼타임

한빛 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1 | REST 소개

1.1 Hello World Wild Web

웹은 스위스 제네바의 유럽입자물리연구소CERN 산하 '데이터 수집과 제어' 그룹에 근무하던 컴퓨터 프로그래머가 새로운 소프트웨어 프로젝트를 진행하기 위해 낸 기발한 아이디어에서 시작되었다.

1990년 12월, 지식을 쉽게 공유하고자 월드와이드웹WorldWideWeb[01]이라는 비영리 소프트웨어 프로젝트를 시작한 팀 버너스리Tim Berners-Lee는 약 1년간 작업한 후 다음과 같은 내용을 고안하고 구현했다.

- URIUniform Resource Identifier : 모든 웹 도큐먼트에 할당된 유일한 주소
- HTTPHyper Text Transfer Protocol : 인터넷을 통해 컴퓨터가 통신하기 위한 메시지 기반 언어[02]
- HTMLHyper Text Mark-up Language : 정보를 제공하는 도큐먼트를 표현하기 위한 하이퍼텍스트 마크업 언어. 도큐먼트는 관련된 다른 도큐먼트에 대한 링크를 포함할 수 있음
- 최초의 웹 서버[03]
- WorldWideWeb과 Nexus : 버너스리는 최초의 웹 브라우저를 'WorldWideWeb'이라 불렀으나 이후에 웹 자체와의 혼동을 피하기 위해 Nexus라고 이름 붙임
- 웹 브라우저에 탑재된 최초의 위지윅(WYSIWYG)[04] HTML 에디터

1991년 8월 6일, 버너스리는 최초의 웹 페이지에 다음과 같이 썼다.

---

01 WorldWideWeb 프로젝트는 후에 공백을 삽입하여 'World Wide Web'으로 개명되었다.

02 Berners-Lee, Tim. The Original HTTP as defined in 1991, W3C, 1991 (http://www.w3.org/Protocols/HTTP/AsImplemented.html).

03 최초의 웹 서버는 지금도 운영하고 있다. (http://info.cern.ch)

04 WYSIWYG은 What You See Is What You Get(보는 대로 얻는다)의 약어다.

WorldWideWeb(W3)은 전 세계에 있는 모든 도큐먼트[05]에 접근하는 것을 목표로 하는 광범위한 하이퍼미디어 정보검색의 시작이다.

이후 웹은 폭발적으로 성장하여 5년간 사용자 수가 4천만 명으로 급등했으며, 어느 시점에서는 사용자 수가 두 달간 두 배 증가하기도 했다. 버너스리가 사용했던 '문서의 우주'는 그렇게 팽창했다.

그렇지만 너무 빨리, 너무 크게 성장했던 웹은 붕괴의 조짐을 보이기 시작했다. 웹 트래픽이 인터넷 인프라 용량을 넘어설 정도로 빨리 늘어났고, 웹의 핵심 프로토콜은 일관성 있게 구현되지 않아서 캐시나 다른 중간매체의 기능을 지원하지 못했다. 이같은 급격한 팽창과 증가하는 요구에 맞춰 웹을 확장할 수 있을 것인지에 대한 명확한 답 역시 없었다.

1.2 웹 아키텍처

1993년 말에 공동으로 '아파치 HTTP 서버 프로젝트'[06]를 시작한 로이 필딩Roy Fielding은 웹 확장성에 관심이 있었다. 필딩은 웹 분석을 통해 웹 확장성은 몇 가지 주요 제약점에 따라 좌우된다는 것을 알게 되었고, 동료와 함께 실용적인 접근법으로 웹 구현 방법을 개선하기 시작했다. 그들의 목표는 제약점을 모두 만족시켜서, 웹이 지속적으로 확장하도록 하는 것이었다.

필딩은 제약점들을 여섯 가지 범주로 묶어 웹의 구조적 스타일Web's architectural style이라고 불렀다. 언급한 웹의 구조적 스타일은 다음과 같다.

1. 클라이언트/서버Client/Server
2. 균일한 인터페이스Uniform Interface

---

05 Berners-Lee, Tim. World Wide Web, W3C, 1991 (http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html).

06 http://httpd.apache.org

3. 계층 시스템Layered System
4. 캐시Cache
5. 상태 없음Stateless
6. 주문형 코드Code-on-demand

클라이언트/서버

웹은 클라이언트/서버 기반 시스템으로, 클라이언트/서버 규약의 핵심 사항은 관심사의 분리다. 그래서 시스템 내에서 클라이언트와 서버의 역활은 분명하게 나뉜다. 웹의 일관된 인터페이스를 따른다는 전제하에, 클라이언트와 서버는 각자의 언어 및 기술을 사용하여 독립적으로 구현되고 배포될 수 있다.

균일한 인터페이스

웹을 구성하는 클라이언트, 서버, 네트워크 등의 매체 간 인터페이스는 각 인터페이스의 일관성에 기반한다. 이러한 웹 컴포넌트들 중 하나라도 기존에 구축된 표준에서 벗어나면, 웹 커뮤니케이션 체계는 붕괴된다.

웹 컴포넌트들은 필딩이 구분한 다음 네 가지의 인터페이스 제약에 따라 서로 일관성 있게 상호 운영된다.

1. 리소스 식별
2. 표현을 통한 리소스 처리
3. 자기-서술적 메시지Self-descriptive messages
4. 애플리케이션 상태 엔진으로서의 하이퍼미디어(HATEOAS[07])

■ 리소스 식별

리소스는 웹 상에서 서로 구별되는 어떤 개념을 의미하며, URI와 같은 고유 식별

07 Hypermedia as the Engine of Application State의 약어다.

자로 나타낼 수 있다. 예를 들어, "http://www.hanb.co.kr"는 홈페이지 URI인데, 특정 웹사이트의 루트 리소스를 표현하는 유일한 방법이다.

■ 표현을 통한 리소스 처리

클라이언트는 리소스 표현을 처리한다. 같은 리소스라도 다른 클라이언트에서는 다른 방식으로 표현될 수 있다. 예를 들어, 하나의 도큐먼트가 웹 브라우저에서 HTML로 특정 프로그램에서는 JSON으로 표현될 수도 있다. 여기서 중요한 점은 표현은 리소스와 상호작용하는 방법이지 리소스 그 자체는 아니라는 것이다. 이러한 개념적 구별에 의해 리소스는 다른 방식과 형식으로 표현되지만 식별자 자체는 변하지 않는다.

■ 자기-서술적 메시지

리소스의 요청 상태는 클라이언트의 요청 메시지로 표현된다. 리소스의 현재 상태는 그 요청에 응답하는 서버의 응답 메시지로 표현되어, 클라이언트에 전달된다. 예를 들어, 위키피디아의 웹 페이지를 수정하는 클라이언트는 서버가 관리하는 리소스인 웹 페이지를 새로운 상태로 업데이트하도록 제안하는 표현을 전달할 때, 요청 메시지를 사용할 수 있다. 클라이언트의 요청을 받아들일 것인가 여부는 전적으로 서버에 달려 있다.

자기-서술적 메시지는 메타데이터를 포함하는데, 이것은 리소스의 상태, 표현 형식과 크기, 메시지 자신에 대한 추가 정보를 전달할 수 있다. HTTP 메시지는 일정한 형태로 여러 종류의 메타데이터 정보를 넣을 수 있는 헤더를 제공한다.

■ 애플리케이션 상태 엔진으로서의 하이퍼미디어

리소스의 상태 표현은 관련 리소스의 링크를 포함한다. 링크가 실타래처럼 웹을 하나로 엮기 때문에 사용자들은 정보와 애플리케이션들을 직접적인 방식으로 이리저리 훑어볼 수 있다. 페이지 내 링크의 존재 여부는 리소스의 현재 상태에서 중요하다.

계층 시스템

계층 시스템이라는 제약조건은 프락시 또는 게이트웨이 같은 네트워크 기반의 중간매체를 사용할 수 있게 한다. 웹의 일관된 인터페이스를 사용하면 중간매체를 클라이언트와 서버 사이에 마치 없는 것처럼 배치할 수 있다.

일반적으로 네트워크 기반의 중간매체는 특별한 목적을 위해 클라이언트와 서버 간 통신을 가로챌 수 있다. 네트워크 기반의 중간매체는 보통 보안을 강화하거나, 응답을 캐싱하거나, 부하를 분산하는 용도로 사용한다.

캐시

캐시는 웹 구조의 중요 제약조건 중 하나다. 캐시라는 제약조건에 의해 웹 서버가 응답 데이터마다 캐시 여부를 선언한다. 캐싱 응답 데이터는 클라이언트가 느끼는 지연을 감소시키고 애플리케이션의 전체적인 이용 가능성과 안정성을 향상시키며, 웹 서버의 부하를 제어한다. 다시 말해, 캐싱[08]은 웹의 전체적인 비용을 낮출 수 있는 중요한 방법이다. 캐시는 클라이언트와 서버 사이의 네트워크 경로라면 어디에라도 위치할 수 있다. 웹 서버단, 콘텐츠 전송망CDN단, 또는 클라이언트 내에도 가능하다.

상태 없음

상태 없음 제약조건은 웹 서버가 클라이언트의 상태를 관리할 필요가 없다는 의미다. 따라서 각 클라이언트는 웹 서버와 상호작용하는 관련 상황 정보를 직접 관리해야 한다. 웹 서버는 웹 애플리케이션과의 복잡한 커뮤니케이션을 위해 필요한 상태 관리를 클라이언트에 맡김으로써 더 많은 클라이언트에 서비스할 수 있다. 이러한 트레이드오프는 웹 구조적인 스타일의 확장성에 이바지하는 주요 요소다.

---

08 명령어와 데이터를 캐시 기억 장치 또는 디스크 캐시에 일시적으로 저장하는 것을 의미한다.

주문형 코드

웹은 주문형 코드Code-On-Demand를 아주 많이 사용한다. 이 제약조건은 스크립트나 플러그인 같은 실행 가능한 프로그램을 일시적으로 클라이언트에 전송하여, 클라이언트가 실행할 수 있도록 한다.

주문형 코드는 웹 서버와 클라이언트 사이에 기술적 결합을 만들어내기도 하는데, 클라이언트는 필요할 때마다 서버에서 내려받은 실행 코드를 이해해야 하기 때문이다. 이러한 이유로 주문형 코드는 웹의 구조적 스타일에서 유일한 선택사항이다. 주문형 코드 제약사항의 예로 자바 애플릿, 자바스크립트, 플래시 같은 웹 브라우저 기반의 기술들이 있다.

1.3 웹 표준

필딩은 팀 버너스리를 비롯한 다른 개발자들과 함께 웹 확장성을 높이는 작업을 했다. 이런 작업을 통해 표준 설계안을 만들었고, HTTP의 새로운 버전인 HTTP/1.1[09] 스펙을 작성하였다. 또한, URI 문법 형식도 RFC 3986[10]으로 규정하였다. 이렇게 정리된 표준은 웹을 통해 급속히 확산되었고, 웹이 지속적으로 성장하는 기틀을 마련했다.

1.4 REST

2000년 웹의 확장과 관련된 위기를 극복한 후, 필딩은 자신의 박사학위 논문에서 '웹의 구조적 스타일'이라고 이름 붙인 이 문제에 대해 설명하였다.[11] 필딩이

09 Fielding, Roy T., Tim Berners-Lee, et al. HTTP/1.1, RFC 2616, RFC Editor, 1999 (http://www.rfc-editor.org/rfc/rfc2616.txt).

10 Berners-Lee, Tim, Roy T. Fielding, et al. Uniform Resource Identifier (URI): Generic Syntax, RFC 3986, RFC Editor, 2005 (http://www.rfc-editor.org/rfc/rfc3986.txt).

11 Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm).

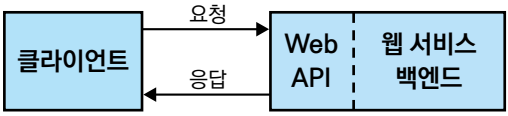
'Representational State Transfer(표현 상태 전이, REST)'이라는 이름을 붙인, 웹의 구조적 스타일에 대한 설명은 제약조건들로 이루어져 있다.

1.5 REST API

웹 서비스는 특정한 목적을 위해 만들어진 웹 서버로, 다른 사이트나 다른 애플리케이션이 필요로 하는 것을 제공한다. 클라이언트 프로그램은 웹 서버에서 제공하는 API를 이용하여 웹 서비스와 통신한다. 보통 API는 데이터와 기능의 집합을 제공하여 컴퓨터 프로그램 간 상호작용을 촉진하며, 서로 정보를 교환할 수 있게 해준다. 그림 1-1에 표현된 것처럼, Web API는 웹 서비스의 얼굴이며 클라이언트의 요청을 직접적으로 듣고 응답한다.

REST 구조 스타일은 최근 웹 서비스를 위한 API 설계에 많이 적용되고 있다. REST 구조 스타일에 적합한 Web API를 REST API라고 한다.

그림 1-1 Web API



REST API를 제공하는 웹 서비스를 'RESTful'하다고 할 수 있다. REST API는 상호 연결된 리소스의 결합체며, 이 리소스 집합을 일컬어 REST API의 리소스 모델이라고 한다. 잘 설계된 REST API는 웹 서비스를 사용할 클라이언트 개발자들을 불러모은다. 최근 오픈 마켓은 웹 서비스로 관심을 끌기 위해 치열하게 경쟁하고 있는데, 잘 만들어진 REST API 설계는 사용자들의 주목을 받기 위해서 반드시 갖추어야 할 특징이 되었다.

1.6 REST API 설계

REST API 설계는 때론 과학이 아닌 예술과 같을 때가 있다. REST API 설계의 몇 가지 모범 사례는 HTTP 표준에 포함되어 있지만, 지난 몇 년 동안 표준과 같은 몇 가지 접근 방법이 새롭게 부각되었다. 여전히 우리는 다음과 같은 질문들에 대한 답을 찾아야 한다.

- URI 경로path 세그먼트는 언제 복수로 써야 하는가?
- 리소스의 상태를 업데이트하려면, 어떤 메서드를 사용해야 하는가?
- CRUD[12]가 아닌 연산을 어떻게 URL에 매핑하는가?
- 특정한 시나리오에 가장 적합한 HTTP 응답은 무엇인가?
- 리소스 상태 표현의 버전은 어떻게 관리할 수 있는가?
- JSON에 포함된 하이퍼링크는 어떻게 구조화하는가?

규칙

이 책의 목표는 REST API 설계 규칙을 제공하여, 위에 나열한 것과 같은 반복되는 질문에 명확히 답하는 것이다. 책에서 제안하는 규칙은 REST API를 일관된 방식으로 설계하는 데 도움을 주며, 클라이언트에서 API 활용을 촉진할 것이다. 이 규칙들은 완벽한 세트 또는 개별적으로 제공된다. 규칙들에 대해 이의를 제기할 수도 있겠지만, 각 규칙은 오랫동안 심사숙고한 결과라는 것은 분명하게 말할 수 있다.

설계 규칙의 많은 부분은 사실상의 표준이 된 모범 사례에서 가져왔다. REST API를 설계한 경험이 있다면, URI 설계를 다룬 2장과 HTTP 사용을 다룬 3장의 관련 규칙들에 익숙할 것이다. 반면, 4장과 5장에 나오는 규칙들은 미디어 타입과 표현 형태를 다루는 것으로 논란의 여지가 있을 수 있는 주관적인 방식들이다.

12 Create, Retrieve, Update, Delete의 약어다.

NOTE_ 규칙에 대해 설명할 때 사용한 "해야 한다", "해서는 안 된다", "필요하다", "가능한 한 추천한다", "할 수 있다", "선택적"이라는 말의 의미에 대해서는 RFC 2119에 설명한 바를 따를 것이다.

WRML

REST API 설계와 구현에 도움을 줄 수 있는 WRML[Web Resource Modeling Language](워믈이라고 읽는다)이라는 프레임워크를 고안했는데, 이 프레임워크는 리소스 모델을 다이어그램으로 그리는 기술을 기반으로 개발되었다. 이 기술은 기본 도형을 사용하여 리소스 원형을 그리는데 리소스 원형 부분에서 설명하였다. 4장의 '미디어 타입 설계'에서 설명할, 확장 가능한 포맷과 스키마를 포함하는 application/wrml 미디어 타입[13]으로 WRML의 범위를 확장할 수 있다. 책 후반부에서는 주로, WRML의 아이디어를 사용하여 현재의 모범 사례와 일반적인 상황에서 이상적인 사례에 대해 설명할 것이다.

5장과 6장에서 설명하는 규칙들에는 JSON[Javascript Object Notation]을 사용하는 예제가 포함될 것이다. JSON[14]은 장점이 많은 중요한 포맷으로 자바스크립트가 지원되고, 다양한 형태로 지원되며, 익숙한 문법이라는 장점이 있다. 그러나 JSON 포맷 자체만으로 중요한 REST API 개념-링크, 링크 관계, 스키마 등을 위한 일관된 구조를 제공할 수는 없다. 5장의 '하이퍼미디어 표현'에 있는 규칙과 '스키마 표현'에 있는 규칙들은 WRML을 사용해서 이러한 주요 구성을 JSON 포맷 표현 형식으로 시연한다.

마지막으로 7장에서는 API 디자인의 일관성이 교육적인 추구에만 있지 않음을 보여준다. 개발자가 REST API를 설계하고 개발할 때 도움이 되는 풍부한 개발 툴과 프레임워크를 사용함으로써 개발 환경이 향상될 것이다.

13 http://www.wrml.org
14 http://www.json.org

1.7 정리

1장에서는 웹의 탄생과 안정화 과정에 대해서 다뤘다. 이런 웹의 발전사를 통해 이 책의 규칙-지향적인 설명과 균일한 REST API 설계 방법론을 확산시키고자 하는 개념적인 프레임워크인 WRML을 소개하였다. 이러한 내용들은 다음 장부터 설명할 REST를 잘 활용할 수 있는 API 설계 과정을 이해하는 데 도움이 될 것이다. 이 장에서 소개된 용어들을 표 1-1에서 정리하였다.

표 1-1 용어 리뷰

용 어	설 명
HATEOAS	REST의 "애플리케이션 상태의 엔진으로서의 하이퍼미디어Hypermedia as the Engine of Application State" 단일 인터페이스 제약을 나타내는 두문자어로, 리소스의 가능한 액션을 지정할 수 있는 연결에 대한 상태-인지state-aware 목록을 제공하는 관습을 지칭하는 말이다.
HTTP/1.1	로이 필딩, 팀 버너스리, 그들의 동료들이 기여한 통신 프로토콜의 가장 최근 버전에 대한 표준화다.
아키텍처 스타일	로이 필딩이 박사 학위 논문에서 시스템 내의 상호 연결된 컴포넌트의 동작을 제한하는 일련의 조건을 설명하기 위해 사용한 말이다.
캐시	웹 서버에서 클라이언트의 요청을 만족할 수 있도록 자원 상태 표현을 유지하는 네트워크 기반의 중간 매개체에 대한 REST 제약이다.
클라이언트/서버	컴포넌트의 영역을 두 개로 구분하여, 각각의 구현이 독립적으로 발전할 수 있도록 한 REST 제약이다.
주문형 코드	웹 서버에서 클라이언트의 필요에 따라 선택적으로 요청되는 실행 가능한 프로그램을 전송할 수 있도록 하는 REST 제약이다.
엔티티 바디 (Entity Body)	콘텐츠를 포함하도록 정의된 HTTP 메시지의 부분으로 없을 수도 있으며, 대개는 리소스 표현이다.
엔티티 헤더 (Entity Header)	HTTP 메시지의 한 부분으로, 리소스와 이에 대한 표현을 나타내는 메타 정보를 주고받을 수 있다.
애플리케이션 프로그래밍 인터페이스	컴퓨터 프로그램 간의 상호작용을 촉진하기 위한 데이터와 기능의 집합을 노출하는 것이다.
하이퍼미디어	멀티미디어 정보 네트워크를 설계할 수 있도록 다양한 형식을 조합하고 연결하는 것을 가능하게 하는 하이퍼텍스트의 확장이다.

용 어	설 명
하이퍼텍스트	탐색할 수 있는 정보 그물망을 만들 수 있는 연결 링크와 관련 도큐먼트 정보가 포함된 텍스트 기반의 도큐먼트다.
하이퍼텍스트 마크업 언어	팀 버너스리가 만든 것으로, 웹 리소스의 상태 정보와 관계를 표현할 수 있다.
하이퍼텍스트 전송 프로토콜	팀 버너스리가 개발한 것으로, 인터넷으로 통신할 때 사용할 수 있는 메시지 기반의 언어다.
아키텍처 제약	일관성을 강화하고 기대하는 특성을 만족할 수 있도록 하는 시스템 컴포넌트 동작에 대한 제한이다.
자바스크립트	웹 개발에 많이 사용되는 강력한 스크립트 언어다.
자바스크립트 객체 표현(JSON)	자바스크립트에서 파생된 구조적 데이터 교환에 사용되는 표준화된 텍스트 형식이다.
계층 시스템	단일 인터페이스 제약을 수정하지 않고, 클라이언트와 서버 사이에 네트워크 기반의 매개체가 가능하게 하는 REST 제약이다.
미디어 타입	콘텐츠의 형태를 기술하기 위한 문법이다.
메시지	자원의 상태에 대한 표현 정보를 전송하기 위해 사용되는 자기-서술적인 봉함이다.
표현 (Representation)	컴포넌트 간에 이동하는 메시지를 통해 전송될 수 있는 자원의 형식화된 상태다.
REST	로이 필딩이 웹의 구조적 스타일로부터 유도한 것이다.
요청 메시지	URI로 지시하는 웹 리소스와 상호작용하기 위해 클라이언트에서 송신되는 것으로, 자원 상태를 나타내는 표현이 포함될 수 있다.
리소스	유일한 식별자로 참조할 수 있고, 단일 인터페이스로 조작할 수 있는 모든 웹 기반 개념이다.
리소스 식별자	특정한 웹 기반 개념의 유일한 ID다.
리소스 모델	상호 연결된 웹 기반 개념의 집합이다.
자원 상태 표현	웹 서버에서 소유한 자원의 표현 상태로, 애플리케이션의 클라이언트 서버 간 전송되는 정보다.
응답 메시지	클라이언트 요청에 대한 결과를 가리키는 서버의 응답. 리소스 상태를 전달하는 표현이 포함될 수 있다.
REST API	웹의 구조적 스타일을 따르는 웹 서비스 인터페이스다.
확장성	증가되는 부하를 부드럽게 처리할 수 있는 능력이다.

용 어	설 명
상태 없음	더 많은 클라이언트를 지원할 수 있도록 웹 서버에서 클라이언트의 특정 상태 정보를 유지하는 것을 제한하는 REST 제약이다.
단일 인터페이스	웹 기반 컴포넌트 간의 통신을 표준화한 네 개의 REST 제약 집합이다.
URI	모든 웹 리소스에 유일한 ID를 부여하기 위해서 팀 버너스리가 고안한 문법이다.
Web API	웹 서비스와 상호작용하기 위해 클라이언트에서 사용하는 것이다.
웹 브라우저	일반적인 형태의 웹 클라이언트다. 팀 버너스리는 HTML 도큐먼트를 보거나 편집할 수 있는 최초의 웹 브라우저를 개발하였다.
웹 클라이언트	자원 상태 표현을 받거나 서버로 전송하기 위해 REST 단일 인터페이스에 따르는 클라이언트 프로그램이다.
웹 컴포넌트 (컴포넌트)	REST 단일 인터페이스를 따르는 클라이언트, 네트워크 기반의 매개체나 서버다.
WRML	일정한 REST API로 설계하고 구현하는 과정에 도움이 되는 개념적인 프레임워크다.
웹 서버	자원 상태 표현을 받거나 클라이언트로 전송하기 위해서 REST의 단일 인터페이스 제약을 따르는 컴퓨터 프로그램이다.
웹 서비스	특정한 그리고 종종 재사용할 수 있는 기능으로 프로그램된 웹 서버다.