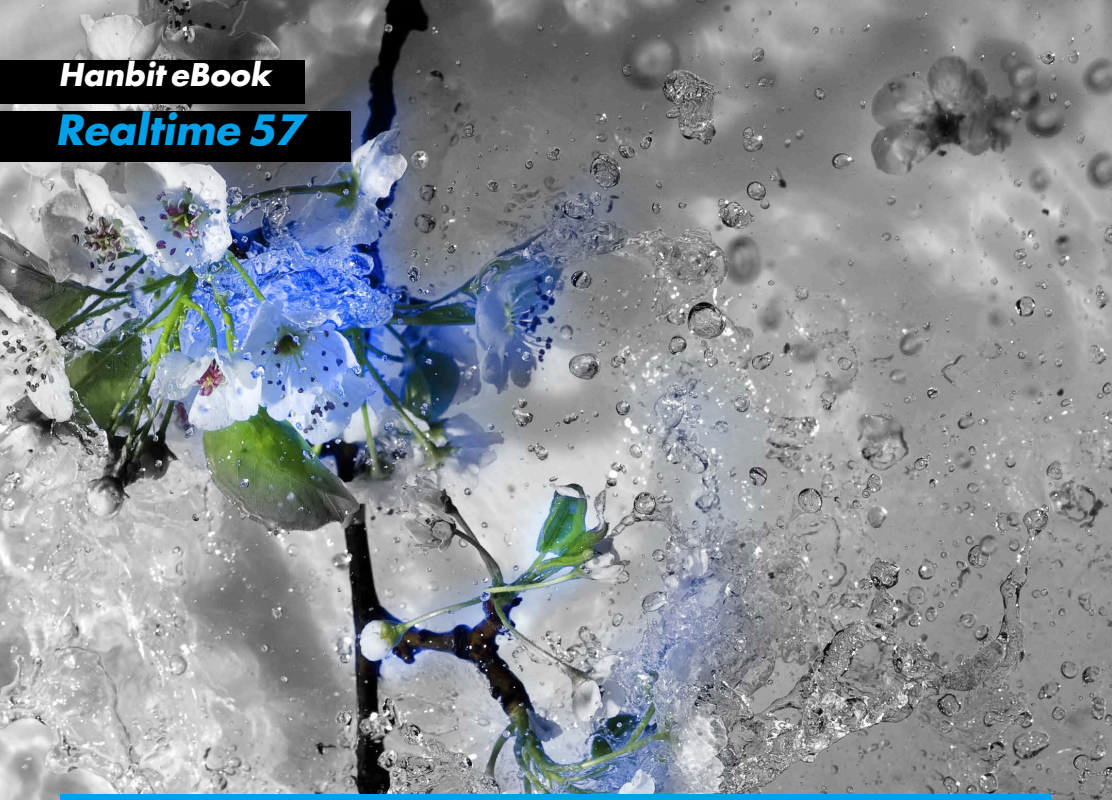


Hanbit eBook

Realtime 57

스프링을 이용한 RESTful 웹 서비스 구축하기

실전 예제로 배우는 REST 방식의 스프링 웹 서비스

김강우 지음

한빛미디어 Hanbit Media, Inc.

스프링을 이용한 RESTful 웹 서비스 구축하기

실전 예제로 배우는 REST 방식의 스프링 웹 서비스

스프링을 이용한 RESTful 웹 서비스 구축하기 실전 예제로 배우는 REST 방식의 스프링 웹 서비스

초판발행 2014년 02월 27일

지은이 김강우 / **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 978-89-6848-689-0 15000 / **정가** 12,000원

책임편집 배용석 / **기획** 이중민 / **편집** 정지연
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 최송실
영업 김형진, 김진불, 조유미 / **마케팅** 박상용, 서은옥, 김옥현

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

Published by HANBIT Media, Inc. Printed in Korea

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

저자 소개

지은이_ **김강우**

“프로그래밍은 기술이 아니라 예술이다”라고 외치며 방랑하는 떠돌이 개발자다. 오픈 데이터 플랫폼에 관심이 많으며, 서로 간의 소통을 통해서 가치를 창출하고 진화해가는 것을 좋아한다. 지난 십여 년을 개발자로 일해왔으나, 지금은 시대의 흐름을 느끼기 위해 잠시 방황하고 있다. 아득히 들려오는 빗소리를 벗 삼아 비움의 자세를 견지해 나가려고 무던히 노력 중인 바보 중의 바보다.

저자 서문

시대 변화에 발맞추어 오픈 API의 열풍이 불고 있습니다. 많은 곳에서 오픈 API를 공개하고 있고 사용자들에게 다양한 방식의 활용을 기대하고 있습니다. 이 오픈 API를 만드는 데 가장 많이 사용되는 기술이 바로 REST일 것입니다. REST의 단순성과 웹의 특성을 이용한다는 장점 때문에 구글, 아마존 같은 해외의 유수 사이트들도 REST 방식의 오픈 API를 제공하기도 합니다.

이 책은 이러한 REST 기반의 서비스를 구축하는 데 조금이나마 도움이 되고자 만들었습니다. 물론, 여기에 나와 있는 방법이 정답은 아닙니다. 다른 방식으로 구현해도 되고, Jersey나 Apache CXF 같은 다른 프레임워크를 사용해도 됩니다. 프로그래밍에 정답은 없겠죠. 때에 따라서 길을 선택하는 것일 뿐. 단지 아쉬운 점은 REST에 대한 좀 더 깊은 이야기를 하지 못한 것인데, 이 또한 REST의 자유성을 존중하는 의미에서는 괜찮을지도 모르겠다는 생각이 듭니다.

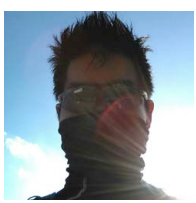
끝으로, 이 책이 세상의 빛을 볼 수 있도록 많은 도움을 주신 한빛미디어 김창수 님, 정지연 님, 이중민 님께 감사의 말을 전합니다.

이 책을 만드는 데 도움 주신 분들

베타 리더_ 김광남

전 세계 사람들이 무료로 사용하는 프로그램을 개발하는 것이 꿈인 프로그래머다. 초기에는 게임 개발을 하다가 지금은 노래방 회사에서 웹/스마트폰/TV용 프로그램을 개발하고 있다. Java와 C++를 주로 사용한다. 책 읽기를 즐기며 사랑하는 아내와 아들과 함께 살고 있다.

베타 리더_ 김지헌



'개발은 취미 생활을 즐기기 위한 부업 활동'이라는 건방진 소리를 쉬이 내뱉는 프로그래머. 스쿠버 다이빙, 스포츠 클라이밍, 로드 라이딩을 즐기는 괴상한 프로그래머로 개발과 관련된 새로운 기술에 대해서 많은 관심이 있다. 귀찮은 걸 싫어하는 게으름뱅이치고는 부지런하게 여행 다니고 레저를 즐기고 개발 관련 콘퍼런스에 참가하고 개발자들과 만나는 것을 즐긴다. '넓고 얕은 지식' 체계를 선호하는 제너럴리스트를 지향하는 일반인이다.

베타 리더_ 김태경

함수형 언어와 오픈 소스에 관심이 많은 제주도민이다. 최근까지 ETL 플랫폼과 AngularJS, Spring을 이용하여 플랫폼 운영 툴을 만들었다. 최근에는 로그와 데이터분석에 관심이 있다.

베타 리더_ 박범진

전자결재업무를 담당하는 자바와 자바스크립트 프로그래머다. 실력은 많이 부족하지만, 핸디소프트에서 가늘고 길게 직장생활을 하고 있다. 재미없는 업무에 지친 마음을 미드 감상이나 SNS, IT 커뮤니티 행사 참석 등으로 달래고 있다.

베타 리더_ 박범진

이것저것 깨작거리기를 좋아하고 웹이나 튜닝, 보안 등을 관심 있으며 정보보안기사 준비하고 있다. 요즘은 다이어트와 곧 태어날 말똥말똥(태명)이가 자라는 배를 만지며 살고 있다.

베타 리더_ 송기용

우아한 프로그래밍을 할 때 도파민이 분출되는 개발자다. 소박한 현업주부의 삶 뒤에 화려한 일렉 기타리스트의 꿈에 도전 중이다.

베타 리더_ 송영준

(요청으로 따로 소개는 기재하지 않았습니다.)

베타 리더_ 홍성민

현재 OSS(오픈 소스 소프트웨어) 기술 검증 및 OSS 기반 분산 확장 아키텍처링 관련 업무를 하고 있다. 역서로는 『데브옵스: 개발자, QA, 관리자가 함께 보는 리눅스 서버 트러블슈팅 기법(2013년, 위키북스)』과 저서로는 『파이썬 웹 프로그래밍: 플라스크를 이용한 쉽고 빠른 웹 개발(2014년, 위키북스)』이 있다.

웹이 거대한 하나의 플랫폼으로 자리매김하는 시점에 자바 개발 프레임워크에서 일종의 대세이자 표준으로 취급되고 있는 스프링을 기반으로 RESTful 서비스를 구축하는 방법을 설명한 이 책은 많은 독자가 RESTful 서비스를 개발할 때 궁금해하는 점을 요약해서 잘 설명하고 있습니다. 모든 빈Bean 관계를 설정 파일이 아닌 자바 어노테이션을 이용하여 설명한 점도 눈에 띕니다. 스프링을 살짝 맛보았거나 기초에 대해 알고 계신 분들이 RESTful 서비스를 빠르게 구축하고자 할 때 많은 도움을 얻을 수 있을 것이라 생각합니다.

대상 독자 및 참고사항

초급 | 초중급 | **중급** | 중고급 | 고급

이 책은 Spring 3.2를 이용하여 REST 기반의 웹 서비스를 만드는 방법을 소개합니다. 자바라는 언어를 알고 있고, 스프링과 REST에 관심이 있는 분이라면 누구나 읽을 수 있습니다.

이 책의 예제 코드를 실행하려면 다음과 같은 환경이 갖춰져 있어야 합니다.

- Java 6 이상이 설치된 개발 환경

이 도서의 예제 소스 코드는 다음 웹 사이트에서 내려받을 수 있습니다.

- http://www.hanbit.co.kr/exam/2689

이 책은 실무 예제를 중심으로 REST 기반의 웹 서비스를 만드는 방법을 소개하고 있습니다. 이론적인 부분을 참고하고 싶다면 『일관성 있는 웹 서비스 인터페이스 설계를 위한 REST API 디자인 규칙(2013, 한빛미디어)』를 함께 보시길 추천합니다.

한빛 eBook 리얼타임

한빛 eBook 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 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 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 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위해 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 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 한 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 | 들어가기

1.1 개요

스마트폰 시대가 되면서 데이터의 흐름에 많은 변화가 생겼다. 이러한 변화 속에서 서로 다른 시스템 또는 애플리케이션 간의 소통을 위해서 웹 서비스Web Service 기술이 주목받게 되었다.

기업들과 기관들은 오픈 API라는 이름으로 자신들의 데이터를 일반 사용자에게 공개하기도 하고, 어떤 기업들은 서버에 있는 데이터를 스마트폰 애플리케이션에 전달하고 소통하기 위해서 웹 서비스를 만들기도 한다. 이러한 웹 서비스를 만드는 방법 중에서 가장 손쉬운 것이 바로 REST 기반 웹 서비스다.

자바 플랫폼에서는 JSR-311(JAX-RS: The Java™ API for RESTful Web Services) 스펙을 제공함으로써 자바 어노테이션을 이용한 RESTful 웹 서비스를 구현할 수 있게 도와주고 있다. Apache CXF나 Jersey, Restlet 같은 구현체들을 사용하면 REST 기반의 웹 서비스를 만들 수 있다.

Spring Web MVC는 JSR-311을 따르지는 않지만, REST 기반의 웹 서비스 개발에 필요한 기능 대부분을 구현하고 있다. 그뿐만 아니라, 간단하게 하나의 자원을 여러 개의 Representation(JSON/XML/ATOM/RSS 등)으로 표현할 수 있고, 브라우저에서 지원하지 않는 PUT/POST 요청을 처리할 수 있는 등 여러 가지 기능을 가지고 있다. 무엇보다도 Spring이 지원하는 강력한 기능들을 사용할 수 있다는 장점이 있다.

이 책에서는 Spring Web MVC를 사용해서 REST 웹 서비스를 만드는 방법을 소개한다.

1.2 REST

1.2.1 REST

REST는 네트워크 구조 원리의 모음으로, 리소스를 정의하고 자원에 대한 주소를 지정하는 방법에 대한 조건들을 의미한다. 즉, 도메인 지향 데이터를 HTTP 위에서 부가적인 전송 레이어 없이 전송하기 위한 간단한 구조를 정의한 것이다.

2000년도에 로이 필딩Roy Fielding은 자신의 박사학위 논문에서 REST(Representational State Transfer, 표현 상태 전이)라고 이름을 붙인, 웹의 구조적 스타일Web's architectural state에 대한 제약조건들을 설명하였다. 그 조건들은 다음과 같다.

- 클라이언트/서버(Client/Server): 웹의 일관된 인터페이스를 따른다는 전제하에 클라이언트와 서버는 독립적으로 구현되어야 한다.
- 균일한 인터페이스(Uniform Interface): 자원 식별, 표현을 통한 자원 처리, 자기 서술적 메시지, HATEOASHypermedia as the Engine of Application State 같은 인터페이스 제약에 따라 서로 일관성 있게 상호 운영되어야 한다.
- 계층 시스템(Layered System): 웹의 일관된 인터페이스를 사용해서 프락시 또는 게이트웨이 같은 네트워크 기반의 중간매체를 사용할 수 있어야 한다.
- 캐시 처리(Cacheable): 웹 서버가 응답 데이터마다 캐시 여부를 선언할 수 있어야 한다.
- 무상태(Stateless): 웹 서버가 클라이언트의 상태를 관리할 필요가 없어야 한다.
- 주문형 코드(Code-on-demand): 선택사항으로 스크립트나 플러그인 같은 실행 가능한 프로그램을 클라이언트에 전송하여 클라이언트가 실행할 수 있도록 해야 한다.

이러한 REST 원리를 충실히 따르면 'RESTful' 하다고 할 수 있다.

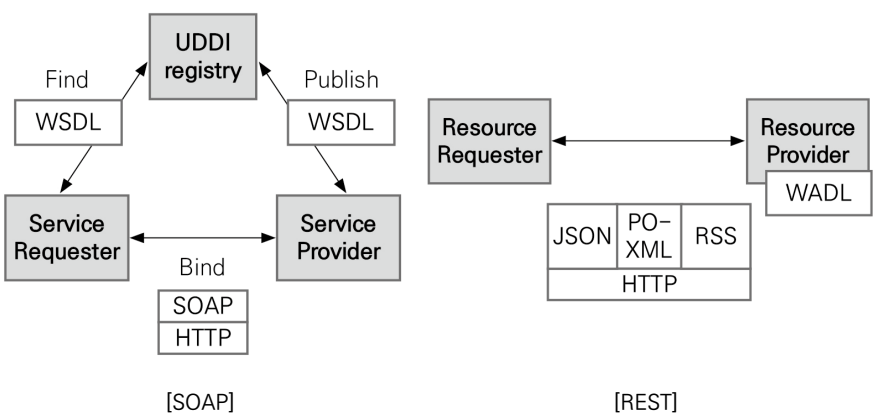
1.2.2 REST API

웹 서비스는 네트워크상에서 서로 다른 시스템 간의 상호 작용을 위한 기술이다. 이는 주고 받는 데이터 형식에 대한 표준을 정의함으로써 플랫폼과 프로그램 언어와는 독립된 방법으로 서로 연동할 수 있다.

간단히 말하면, 클라이언트가 웹 서버에서 제공하는 API를 이용하여 데이터와 기능을 제공받을 수 있다.

예전에는 SOAP(Simple Object Access Protocol)을 기반으로 웹 서비스를 많이 구현했다. 하지만 SOAP 처리의 오버헤드 및 복잡성 때문에 요즘에는 REST 구조 스타일을 사용한 웹 서비스를 많이 사용하고 있다. 이런 REST 구조 스타일에 적합한 API를 REST API라고 한다.

[그림 1-1] SOAP 기반과 REST 기반 웹 서비스의 차이(출처: 전자통신동향분석, 제25권 제2호, 한국전자통신연구원, 2010)



■ Resource

REST API는 URI(Uniform Resource Identifier) 경로를 사용해서 자원을 나타내고, 포워드 슬래시(/)로 경로 구문을 나눈다. 예를 들어 bookstore 사이트의 1번 책은

http://www.bookstore.com/books/1로 표현할 수 있다. 이 경로 구문은 자원 계층에서 유일한 자원을 나타낸다.

그렇다면 해당 자원에 대한 행위는 어떻게 나타내야 할까? REST API에서는 CRUD (Create-생성, Read-읽기, Update-수정, Delete-삭제) 기능을 수행할 때는 URI에 나타내지 않는다. URI는 자원을 식별할 때만 사용하고, CRUD 기능을 수행할 때에는 HTTP Request Method를 사용한다. 즉 GET, POST PUT, DELETE 메소드를 이용하여 처리한다.

[표 1-1] 자원 관리 방법

HTTP Method	의미	CRUD	예제
POST	새로운 자원을 생성한다	Create	POST /books
GET	자원을 조회한다	Read	GET /books/1
PUT	기존에 존재하는 자원을 변경한다	Update	PUT /books/1
DELETE	기존에 존재하는 자원을 삭제한다	Delete	DELETE /books/1

1.3 Spring Web MVC

1.3.1 Model 1 VS Model 2(MVC)

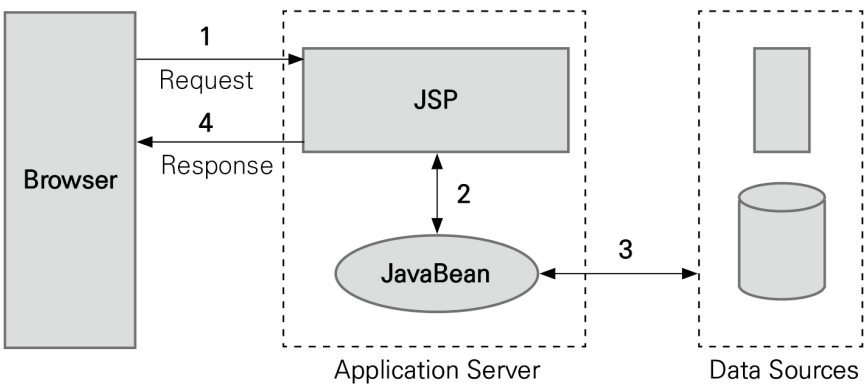
애플리케이션을 개발하는 방법에는 여러 가지가 있다. 프레젠테이션 로직과 비즈니스 로직을 어떻게 구현하고 어디에 위치시킬 것인지, 필요한 데이터를 어떻게 유지하고 공유할 것인가에 대해서 정의하고 때에 따라서는 계층으로 나누기도 한다. 간단히 말하면 하나의 자바 클래스에 모든 것을 담아서 처리할 수도 있고, 로직의 성격에 따라 계층화하여 여러 클래스로 나누어서 처리할 수도 있다. 어느 방법을 사용하든지 그것이 옳다 그르다고 단정 지어 얘기할 수는 없다. 정답이라는 것은 존재하지 않을지 모른다. 하지만 수많은 시행착오를 통해서 좀 더 효율적인 방법을 경험하게 된다. 이러한 경험적 방법들을 체계화시키고 정리한 것을 우리는 패턴 Pattern이라고 부른다.

자바 웹 애플리케이션은 일반적으로 Model 1 방식과 Model 2 방식의 두 가지 구조 패턴Architecture Pattern으로 분류한다.

■ Model 1

Model 1은 예전에 많이 사용했던 방식으로, 사용자의 요청을 서블릿이나 JSP가 받아서 필요한 자바빈JavaBean을 호출한 다음 결과값을 출력한다.

[그림 1-2] JSP Model 1 Architecture

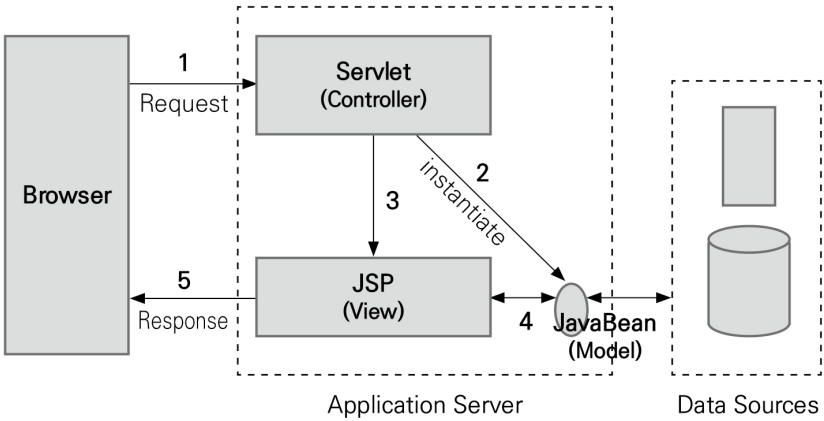


Model 1 방식은 개발 속도가 빠르고, 초보자도 쉽게 배우며 빠르게 적용할 수 있다는 장점이 있다. 하지만 프레젠테이션 로직과 비즈니스 로직이 섞여 있어서 개발과 화면 디자인의 영역 구분이 어렵고, 코드가 복잡해지는 단점이 있다.

■ Model 2(MVC)

Model 2 방식은 MVC 패턴을 이용한다. MVC 패턴은 Model(모델), View(뷰), Controller(컨트롤러)의 세 영역으로 나누어지는 구조로 애플리케이션을 설계한다.

[그림 1-3] JSP Model 2 Architecture



Model 영역

애플리케이션에서 사용하는 데이터를 다루거나 비즈니스 로직을 다루는 영역이다.

View 영역

클라이언트(사용자)에게 보이는 표현 영역으로 프레젠테이션 로직을 처리한다. 일반적인 웹 애플리케이션의 경우 View 영역을 통해서 사용자가 보는 웹 페이지(HTML)를 생성한다.

Controller 영역

Model과 View 영역 간의 흐름을 제어하는 역할을 한다. 클라이언트의 요청을 받아서 이를 처리하기 위한 Model(비즈니스 로직)을 호출하여 처리하고 그 결과를 알맞은 View에게 전달한다.

MVC 패턴의 구성 요소인 Model, View, Controller는 서로 연관된 작업을 수행하지만, 각각의 역할 구분은 명확하게 나누어져 있다. 즉, Model은 오직 비즈니스와 관련한 부분만 처리하면 되고, 사용자에게 보일 화면이나 흐름 제어에 대해서는 처

리할 필요가 없다. View는 사용자에게 알맞은 화면을 보여주는 역할을 하면 되고, 비즈니스 로직이나 흐름 제어에 대해서는 처리할 필요가 없다. 또한, Controller는 흐름을 제어하는 역할을 하여 사용자 요청에 알맞은 모델을 사용하고, 사용자에게 보여줄 View를 선택하기만 하면 된다.

이렇게 서로에 대한 역할 구분이 명확하기 때문에 각각의 영역에 대한 독립성을 보장한다. 그래서 MVC 패턴으로 잘 설계한 애플리케이션은 핵심 부분을 건드리지 않고 애플리케이션을 재설계할 수 있다. 예를 들어 HTML 기반의 웹 애플리케이션을 개발하다가 모바일 앱 개발을 위한 REST API 기반으로 요구사항이 변경되었을 경우 View 영역만 변경하면 손쉽게 전환할 수 있다.

1.3.2 Spring Web MVC

Spring Web MVC는 Spring에서 제공하는 MVC 프레임워크다. Spring이 제공하는 풍부한 기능들을 그대로 사용하면서 MVC 패턴에 기반을 둔 웹 애플리케이션을 개발할 수 있도록 도와준다.

Spring Web MVC는 클라이언트로부터 요청을 받아서 DispatchServlet이라는 Front Controller를 이용해서 요청을 처리할 Controller에게 전달해준다. @Controller 어노테이션과 @RequestMapping 어노테이션만으로 손쉽게 구현할 수 있고, 스프링 3.x부터 지원하는 @PathVariable, @RequestBody, @ResponseBody 등의 어노테이션을 이용해서 REST API를 개발할 수 있다.

Spring Web MVC는 REST 서비스를 생성하기 위해 두 가지 방법을 제공한다. MVC의 ModelAndView를 사용하는 방법과 HTTPMessageConverter를 사용하는 방법이다. 두 가지 방법은 5장에서 간단히 소개하고, HTTPMessageConverter를 위주로 REST 서비스를 생성해 보겠다.

2 | Spring 3.2 와 REST

이 장에서는 예제 소스를 만들기 위한 요구사항 정의 및 전반적인 개발 환경에 대하여 알아보도록 하겠다. 도서 정보를 처리하는 간단한 REST API 서비스를 만드는 요구사항을 정의하고, 개발에 필요한 기본적인 구성 요소를 살펴 본 후, Spring Web MVC에 대해서 좀 더 알아본다.

2.1 요구사항 정의

도서 정보를 처리하는 REST API 서비스를 만들어 보자. 간단하게 조회, 등록, 수정, 삭제 기능을 구현할 것이다.

- 도서 정보 목록을 조회할 수 있어야 한다.
- 도서 상세 정보를 조회할 수 있어야 한다.
- 도서 정보를 등록할 수 있어야 한다.
- 도서 정보를 수정할 수 있어야 한다.
- 도서 정보를 삭제할 수 있어야 한다.

2.2 개발 환경

도서 정보를 처리하는 REST API 서비스를 만들기 위한 개발 환경은 다음과 같다.

- Java 6 이상
- Maven 3.0.x
- Logback / SLF4J
- Spring 3.2
- MyBatis 3.2

예제 파일들은 Java 6 환경에서 만들어졌고, 프로젝트 관리 도구로서 메이븐Maven을 사용하였다. 개발 프레임워크는 Spring 3.2버전이고, 로깅 라이브러리는 SLF4J와 Logback을 사용했다.

자바 설치 방법과 메이븐 설치 방법은 잘 알 것으로 생각하고 생략한다. IDE는 STSSpring Tool Suite를 사용했으나 별도의 기능을 사용하지 않으므로 자신의 취향에 맞는 것을 사용하면 된다. STS는 기본적으로 이클립스와 동일하다. 이클립스 기반에 유용한 확장 플러그인과 스프링 개발 시 도움을 주는 기능들이 포함되어 있다. 물론 이클립스의 메이븐 플러그인인 m2e 플러그인도 포함되어 있다. STS는 Spring STS 사이트[01]에서 내려받을 수 있다.

2.2.1 메이븐

메이븐은 3.0.x 버전을 사용한다. 주사용 기능은 의존성 추가를 통한 자바 라이브러리 자동 내려받기와 테스트 케이스 실행이다. 꼭 메이븐을 사용할 필요는 없고, Gradle, Apache Ivy 같은 다른 도구를 사용해도 된다. 간단하게 이클립스의 WTP를 사용해도 된다.[02] 이클립스를 사용한다면 m2e라는 메이븐 플러그인을 설치해서 간편하게 사용할 수 있다.

메이븐은 프로젝트 객체 모델Project Object Model이라는 개념을 바탕으로 의존성 관리, 생명 주기 관리 기능 등을 제공하는 프로젝트 관리 도구다. 컴파일과 동시에 빌드를 수행할 수 있고, 테스트를 실행할 수도 있으며, 서버로 디플로이할 수 있는 환경도 제공한다. 개발자에게 가장 큰 장점은 의존성 관리인데, 의존성을 추가하면 해당 라이브러리를 저장소에서 자동으로 내려받는다. 즉, 필요한 라이브러리를 직접 내려받아서 프로젝트에 추가할 필요 없이 의존성 선언만 하면 된다.

---

01 http://spring.io/tools

02 필요한 라이브러리는 직접 내려받아야 한다.

2.2.2 Logback / SLF4J

애플리케이션 개발 시 가장 중요한 것은 뭐니뭐니해도 로그를 잘 남기는 것이다. 그래야 에러가 발생했을 때 손쉽게 대응할 수 있다. 에러가 발생했는데 발생한 원인을 찾을 수 없다면 지옥이 따로 없다. 이럴 때 필요한 것이 바로 에러 로그를 관리하는 로깅 라이브러리다.

로깅 라이브러리로는 JUL[java.util.logging], Log4j, Logback 등이 있다. 자바 1.4부터 JUL 패키지가 포함되어 로깅을 자체적으로 지원하고 있지만, 뒤늦게 출현한 덕분인지 아니면 모양새가 마음에 안 들어서 그런지 잘 사용되지 않고 있다. 아마 대부분은 Log4j를 가장 많이 사용할 것이다. Log4j가 훌륭한 로깅 라이브러리임이 틀림없고 지금도 충분히 그 역할을 다 할 수 있지만, 시대의 흐름 속에서 Logback[03]이라는 새로운 로깅 라이브러리가 등장했다.

Logback은 Log4j 보다 많은 발전을 이루었다. 성능이 향상되었고 설정 파일의 자동 리로딩, I/O 실패의 복구, 로그 파일 압축, 오래된 로그 파일 삭제 등 다양한 기능을 제공한다. 그래서 개인적으로 Logback 사용을 추천한다.

SLF4J[Simple Logging Facade for Java]는 로깅 파사드 라이브러리[Logging Facade Library]로 로깅 요청을 기존에 존재하는 다양한 로깅 라이브러리로 전달하는 역할을 한다. 간단하게 기존 로깅 라이브러리를 한 번 감싸서 사용하는 것이라고 할 수 있다. SLF4J를 사용하고 구현체를 Logback으로 사용한다면 로그 출력 코드는 SLF4J 라이브러를 사용하지만, 실제 출력은 Logback을 통해 이루어진다.

애플리케이션을 Log4j로 개발했다고 가정해 보자. 그런데 어느 날 Logback이라는 좋은 라이브러리가 나타났다는 소문을 듣고 Logback으로 전환하려면 Log4j가 들어가 있는 모든 코드를 수정해야 한다. 하지만 SLF4J를 사용하고 구현체를

03 Logback은 Log4j의 개발자가 만들었다.

Log4j를 사용했다면 구현체만 Logback으로 변경하면 되고, 소스 코드는 변경할 필요가 없다.

그리고 애플리케이션을 개발할 때는 많은 라이브러리를 가져다 쓰게 된다. 그런데 라이브러리마다 로깅 라이브러리를 다르게 쓴다면 관리하기가 매우 힘들 것이다. 이럴 때 하나의 일관된 창구 기능을 제공하기도 한다.

이외에 로깅 파사드 라이브러리로는 JCL(Jakarta Common Logging)이 있다.[04]

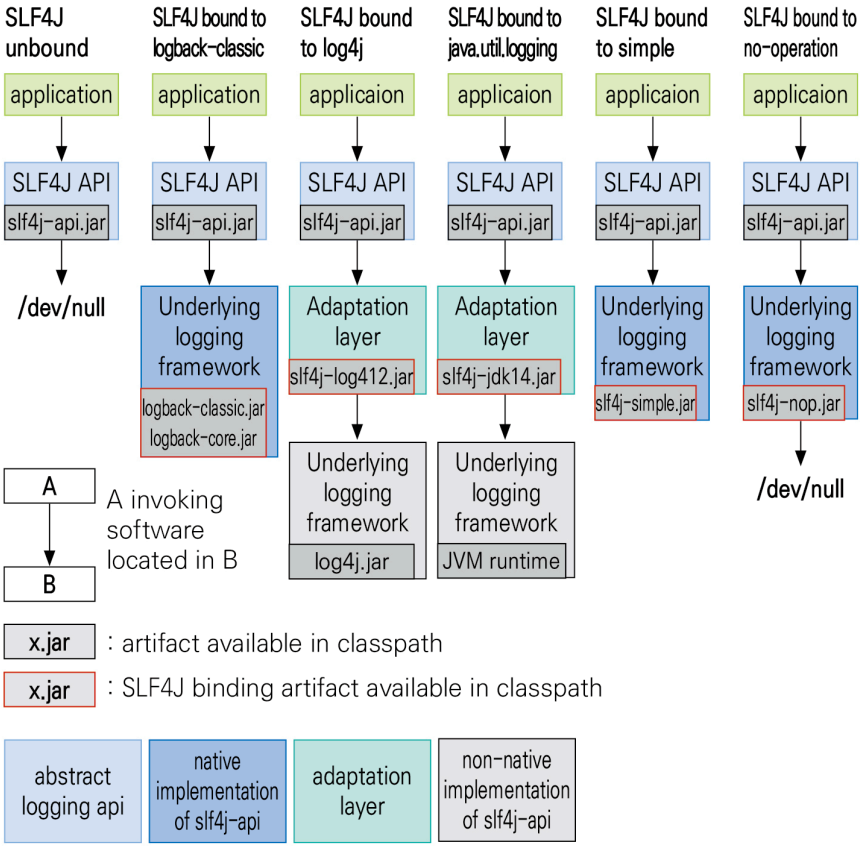
SLF4J

SLF4J는 다중 클래스로더(ClassLoader)를 사용하는 환경에서 발생할 수 있는 JCL의 문제점을 해결하기 위해서 등장하였다.

SLF4J는 그림 2-1과 같이 Logback, Log4j, JUL과 같은 로깅 라이브러리를 지원하고 있으며, 자체적으로 Simple이라는 구현체를 가지고 있다.

04 SLF4J는 로깅 API 구현체와의 매핑이 static하게 이루어지는 데 반해서, JCL은 매핑이 dynamic하게 이루어진다.

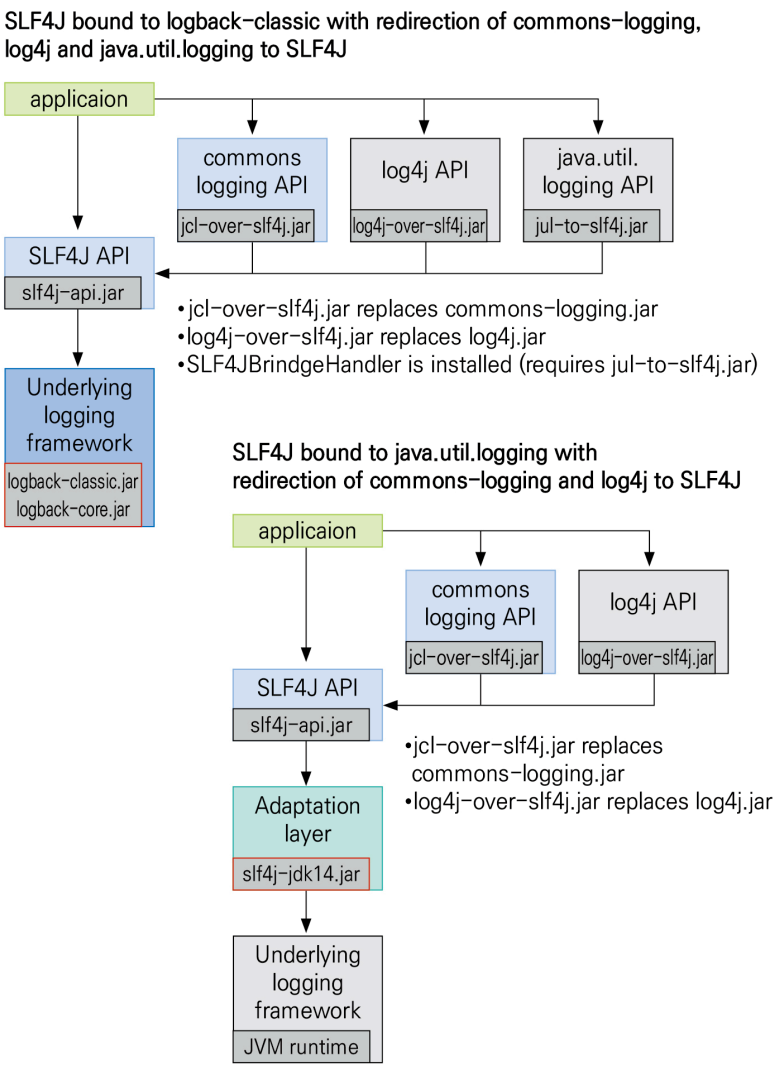
[그림 2-1] concrete bindings



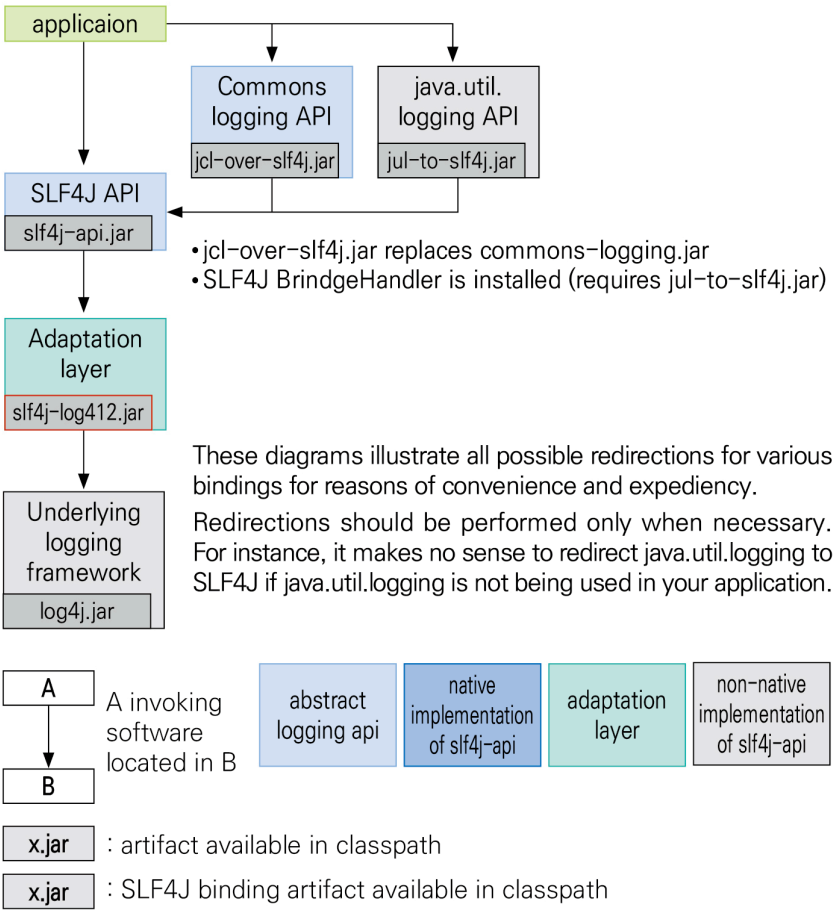
출처: http://www.slf4j.org/manual.html

SLF4J는 Bridging을 지원하며 JCL, Log4j, JUL의 로깅 출력 결과를 SLF4J로 전달하는 가교 역할을 한다.

[그림 2-2] Bridging legacy APIs



SLF4J bound to log4j with redirection of commons-logging and java.util.logging to SLF4J



출처: http://www.slf4j.org/legacy.html

Logback

Logback은 Log4j의 개발자인 'Ceki Gülcü'가 만들었다. 오랫동안 검증된 Log4j의 아키텍처를 기반으로 재작성하여 성능이 10배 정도로 빨라졌고, 메모리 점유율도 낮아졌다. 기본 설정 파일은 XML로 작성됐고, Groovy 문법도 지원한다. Logback 웹 사이트에서는 Log4j 설정 파일을 Logback 설정 파일로 변경해주는 툴[05]을 제공하고 있다.

Logback의 가장 멋진 기능은 설정 파일 자동 재로딩 기능이다. 일반적으로 운영 서버에서는 로그 레벨을 WARN 또는 ERROR로 설정한다. 그런데 만약 운영 중에 좀 더 상세한 로그를 보길 원한다면 어떻게 될까? 기존의 Log4j라면 서버를 중지시키고, 설정 파일을 변경한 다음 서버를 재시작해야 한다. 하지만 Logback은 설정 파일만 변경하면 알아서 변경된 내용이 반영된다.

이외에 저장된 로그 파일 자동 압축, 오래된 로그 파일 자동 삭제 등의 기능도 제공하고, 다수의 JVM에서 하나의 파일에 로그를 출력할 때 쓰는 Log Aggregation과 현재 발생하는 로그 이벤트에 대한 정보를 볼 수 있도록 지원하는 Lilith 등 유용한 기능을 제공한다.

2.2.3 Spring 3.2

스프링 프레임워크는 3.2 버전을 사용한다. 스프링 프레임워크 3.1 이하 버전에서도 REST API 서비스를 개발할 수 있으나, 3.2 버전부터 지원되는 ControllerAdvice를 이용한 예외 처리와 MVC Test 프레임워크 등을 사용할 수 없는 불편함이 있다.

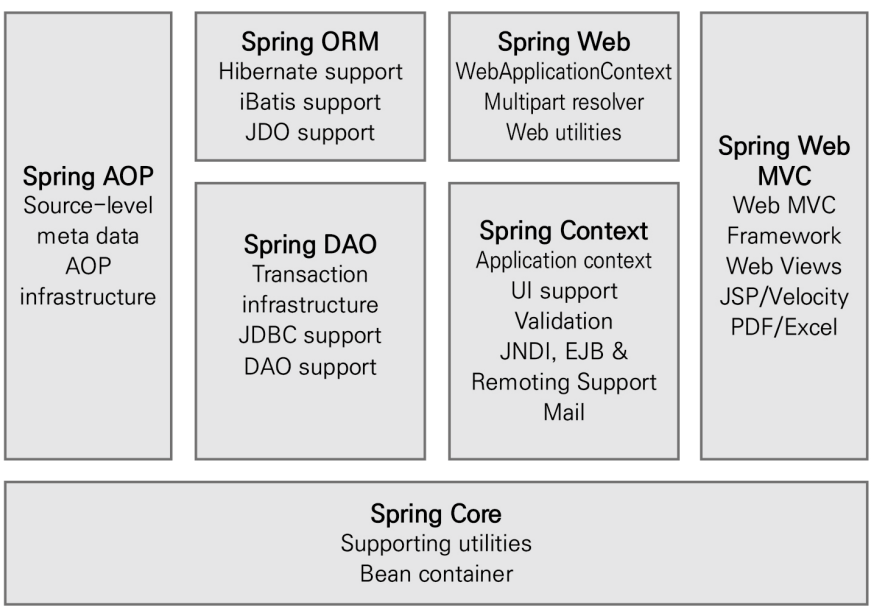
스프링 프레임워크는 간단히 말하면 IoC(Inversion of Control: 제어의 역전) 컨테이너라고 할 수 있다. IoC는 스프링 프레임워크가 가진 핵심적인 기능으로, 자바의 객

05 http://logback.qos.ch/translator/

체 생성 및 의존 관계를 관리하는 기능을 제공한다. 원래 자바 프로그래밍은 자바 코드 안에서 다른 객체를 생성하고 의존 관계를 설정하는 방법을 사용한다. 하지만 스프링 프레임워크를 사용하면 설정 파일을 통해 객체를 생성하고, 의존 관계를 설정할 수 있다. 즉, DI(Dependency Injection: 의존성 주입) 패턴을 지원한다. 이 때문에 객체 간의 느슨한 결합을 유지할 수 있다.

이러한 특징들 때문에 스프링은 AOP(Aspect Oriented Programming: 관점 지향 프로그래밍)이 가능하다. AOP는 문제를 해결하는 핵심 사항과 전체 적용하는 공통 사항을 분리해서 프로그래밍함으로써 공통 모듈을 여러 코드에 쉽게 적용할 수 있다. 즉, 트랜잭션이나 보안, 로깅 등과 같이 여러 모듈에서 공통으로 사용하는 기능들을 분리해서 각 모듈에 적용할 수 있다.

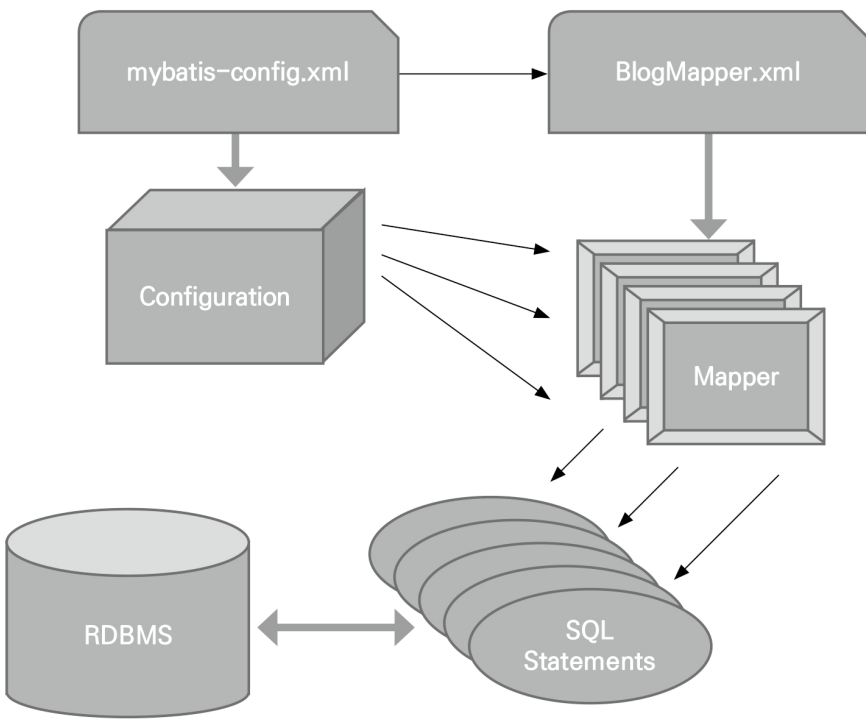
[그림 2-3] Spring Framework



2.2.4 MyBatis

데이터베이스의 자원을 사용하려면 JDBC[Java Database Connectivity]를 이용해야 하고, 자바에서는 JDBC를 이용하여 데이터 처리 작업을 한다. 이 작업을 좀 더 간편하게 하는 프레임워크를 퍼시스턴스 프레임워크라고 한다. MyBatis[06]는 퍼시스턴스 프레임워크로, 개발자가 SQL, 저장 프로시저 등을 좀 더 쉽게 사용할 수 있게 해준다. 간결함과 쉬운 접근성 때문에 국내에서 많이 사용되고 있다.

[그림 2-4] MyBatis 구성도



06 MyBatis는 iBatis의 새로운 버전이다. 아파치 소프트웨어 재단이 구글보다 변화에 늦게 반응한다고 느낀 팀원들은 Apache iBatis 프로젝트를 2010년 6월 16일에 종료하고, Google Code로 자리를옮겨서 MyBatis라는 프로젝트로 다시 시작했다.

MyBatis는 크게 두 부분으로 이루어진다. MyBatis 설정을 다루는 Configuration과 Mapper다.

Configuration은 MyBatis에 대한 전반적인 설정 정보를 가지고 있으며, XML 파일이나 Configuration 클래스를 이용해서 설정할 수 있다.

Mapper는 보통 Mapper XML 파일과 자바 인터페이스인 Mapper Interface로 이루어진다.[07] Mapper XML 파일은 MyBatis의 가장 큰 특징으로 SQL Statement를 XML에 정의하는 것이다. 이렇게 함으로써 데이터베이스에서 사용하는 SQL과 자바 코드를 분리하도록 도와준다. Mapper Interface는 Mapper XML에 정의한 SQL들을 자바에서 손쉽게 사용할 수 있도록 도와주는 역할을 한다. 그뿐 아니라 어노테이션을 이용해서 Mapper Interface에 직접 SQL을 정의하는 기능도 제공한다.

2.3 개발 환경 구축하기

Spring Web MVC를 사용하기 위한 기본적인 개발 환경을 구축해 본다. 메이븐으로 웹 애플리케이션 프로젝트를 생성하고, 필요한 의존성을 추가한 다음 스프링을 사용하기 위한 설정 작업을 한다.

2.3.1 메이븐으로 프로젝트 생성하기

웹 애플리케이션 기반의 메이븐 프로젝트를 생성한다. 메이븐을 사용하려면 Apache Maven Proejct[08]에서 내려받아서 설치해야 한다.

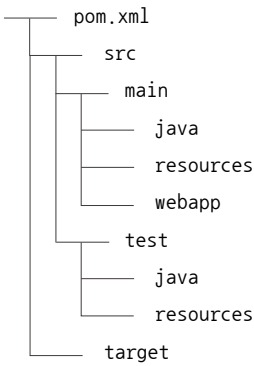
■ 디렉터리 구조

메이븐은 정규화된 디렉터리 구조를 제공한다. 소스 파일들은 /src 디렉터리 밑에

07 정확하게 말하면 XML 파일과 인터페이스 둘 중 하나를 사용해도 되고, 둘 다 사용해도 된다.
08 http://maven.apache.org/

위치하고, 빌드된 결과물은 /target 디렉터리 밑에 위치한다. 프로젝트가 성공적으로 생성되었다면, artifactId에 입력한 값과 동일한 이름의 디렉터리가 생긴다. archetype을 maven-archetype-webapp으로 생성한 디렉터리 구조는 다음과 같다.

[그림 2-5] 웹 애플리케이션의 메이븐 디렉터리 구조



기본적으로 생성되는 주요 디렉터리는 다음과 같다.

- pom.xml: 프로젝트에 대한 전반적인 정보를 가지고 있다.
- src/main/java: 자바 소스 파일이 위치한다.
- src/main/resource: 배포할 리소스 파일(XML, properties 등)이 위치한다.
- src/main/webapp: 웹 애플리케이션 관련 파일(web.xml 등)이 위치한다.
- src/test/java: 테스트 케이스 자바 소스 파일이 위치한다.
- src/test/resources: 테스트 케이스에 사용할 리소스 파일이 위치한다.
- target: 빌드된 결과물

src 디렉터리를 main과 test로 나누는 이유는 실제 개발에 필요한 코드와 테스트 코드를 분리하기 위해서다.

■ 프로젝트 생성

메이븐의 프로젝트 생성 명령어는 다음과 같다.

```
mvn archetype:generate
```

이 명령어를 실행하면 메이븐 프로젝트를 생성하는 데 필요한 정보를 입력하라는 메시지가 단계적으로 나온다. 항목별로 알맞은 값을 입력하면 프로젝트가 생성된다. 또는 아래처럼 필요한 정보를 명령어에 추가하여 생성할 수 있다.

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

프로젝트 생성 시 필요한 입력 항목

- groupId: 프로젝트가 속하는 그룹 식별 값으로, 단체나 회사를 뜻하는 값을 사용하며 패키지 형식으로 표현한다.
- artifactId: 프로젝트 결과물의 식별 값으로, 프로젝트나 모듈을 뜻하는 값을 사용한다.
- version: 프로젝트 결과물의 버전을 의미한다. 입력하지 않을 경우 기본값인 1.0-SNAPSHOT을 사용한다.
- package: 기본으로 생성할 패키지를 의미한다. 입력하지 않을 경우 groupId와 동일한 구조의 패키지를 생성한다.
- archetype: 생성할 프로젝트의 타입을 지정해준다. 웹 애플리케이션 기반의 프로젝트를 생성하려면 maven-archetype-webapp 타입으로 생성하면 된다.

다음과 같이 콘솔에서 메이븐 명령어를 실행하면 웹 애플리케이션 기반의 프로젝트를 생성할 수 있다.

```
mvn archetype:generate -DgroupId=devfun.bookstore -DartifactId=restapp
-DarchetypeArtifactId= maven-archetype-webapp -DinteractiveMode=false
```

이클립스를 사용하고 있다면 새 프로젝트를 생성하고 메이븐 프로젝트를 생성한 후 archetype을 maven-archetype-webapp으로 선택하면 된다. 만약 기본적으로 생성되지 않은 디렉터리가 있다면 직접 생성한다.

■ 컴파일 / 테스트 / 패키지

소스 코드를 컴파일하려면 다음 명령어를 실행한다.

```
mvn compile
```

컴파일된 클래스 파일은 target/classes 디렉터리에 생성된다.

테스트 클래스를 실행하려면 다음 명령어를 사용한다.

```
mvn test
```

[출력 결과]

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building restapp Maven Webapp 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ restapp ---
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

```
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.730s
[INFO] Finished at: Mon Dec 02 13:26:04 KST 2013
[INFO] Final Memory: 5M/244M
[INFO] ------------------------------------------------------------------------
```

명령어를 실행하면 테스트 코드를 컴파일한 후 테스트 코드를 실행하고, 테스트 성공 여부를 화면에 출력해 준다. 컴파일된 테스트 클래스들은 target/test-classes 디렉터리에 생성되며, 테스트 결과 리포트는 target/surefire-reports 디렉터리에 저장된다.

컴파일도 정상적으로 되고 테스트도 통과하면, 배포 가능한 war 파일을 만든다. 다음 명령어를 실행하면 프로젝트를 패키징해서 결과물을 생성한다.

```
mvn package
```

[출력 결과]

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building restapp Maven Webapp 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ restapp
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ restapp ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @
```

```
restapp ---
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:testCompile (default-testCompile) @
restapp ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ restapp ---
[INFO]
[INFO] --- maven-war-plugin:2.2:war (default-war) @ restapp ---
[INFO] Packaging webapp
[INFO] Assembling webapp [restapp] in [C:\Users\kangwoo\Documents\workspace-
sts restapp\target\restapp]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\Users\kangwoo\Documents\workspace-sts\
restapp\src\main\webapp]
[INFO] Webapp assembled in [22 msecs]
[INFO] Building war: C:\Users\kangwoo\Documents\workspace-sts\restapp\target\
restapp.war
[INFO] WEB-INF\web.xml already added, skipping
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.017s
[INFO] Finished at: Mon Dec 02 13:34:00 KST 2013
[INFO] Final Memory: 6M/244M
[INFO] ------------------------------------------------------------------------
```

프로젝트 패키징이 성공적으로 완료되면, target 디렉터리에 프로젝트 이름과 동일한 war 파일이 생성된다. target 디렉터리를 조회해보면 restapp.war 파일이 생성된 것을 확인할 수 있다.

■ POM 파일

메이븐 프로젝트를 생성하면 pom.xml 파일이 프로젝트 루트 디렉터리에 생성된다. 이 파일은 Project Object Model 정보를 담고 있는데, 주요 설정 정보는 다음과 같다.

- 기본 정보: 패키징 방법, 의존 프로젝트, 부모 프로젝트, 의존 관리, 하위 모듈, 속성값 등을 기술한다.
- 빌드 설정: 소스, 리소스 및 플러그인 등 빌드에 관한 설정과 리포팅에 관한 설정을 기술한다.
- 상세 프로젝트 정보: 프로젝트 이름, 설명, URL, 라이선스, 개발자 정보 등을 기술한다.
- 환경 설정: 이슈 관리, CI 관리, SCM, 저장소, 플러그인 저장소, Profile 등을 기술한다.

archetype을 maven-archetype-webapp으로 선택한 경우 생성되는 pom.xml 파일은 다음과 같다.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>devfun.bookstore</groupId>
  <artifactId>restapp</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>restapp Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
```

```
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>restapp</finalName>
  </build>
</project>
```

pom.xml 파일에서 기술한 항목은 다음과 같다.

- <groupId>: 프로젝트 그룹의 식별값을 의미한다.
- <artifactId>: 프로젝트의 식별값을 의미한다.
- <version>: 버전을 의미한다.
- <packaging>: 패키징 타입을 의미한다. 예제의 경우 프로젝트의 결과물을 war 파일로 생성한다. war뿐만 아니라, jar, ear 등의 패키징 타입이 존재한다.
- <name>: 프로젝트 이름을 의미한다.
- <url>: 프로젝트 사이트의 주소를 의미한다.
- <properties>: 속성값으로 사용할 하위 태그들을 기술한다.
- <dependencies>: 프로젝트에서 의존하는 다른 프로젝트의 정보를 기술한다.
- <dependency>: 의존하는 다른 프로젝트의 POM 정보를 기술한다.
- <groupId>: 의존하는 프로젝트의 그룹 groupId다.
- <artifactId>: 의존하는 프로젝트의 artifactId다.
- <version>: 의존하는 프로젝트의 버전이다.
- <scope>: 의존 범위를 설정한다.