Hanbit eBook

Realtime 76

자바스크립트로 시작하는 Vert.x

김대희 지음

한빛미디어 Hanbit Media, Inc.

자바스크립트로 시작하는

Vert.x

자바스크립트로 시작하는 Vert.x

초판발행 2014년 7월 24일

지은이 김대희 / **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 978-89-6848-672-2 15000 / **정가** 11,000원

책임편집 배용석 / **기획·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

저자 소개

지은이_김대희

2003년부터 프로그래밍을 시작하였고, 일본에서 Python, Ruby, Java 등 다양한 언어로 프로젝트를 수행하였다. 언어를 가리지 않으나 주로 스크립트 언어를 좋아하는 편이다. 웹 기술을 포함한 프런트엔드 기술에도 관심이 있으며, 최근 가장 큰 관심사는 JVM에서 동작하는 동적 언어를 이용한 서버사이드 개발이다. 현재 클라우드 관련 회사에서 일하고 있다.

저자 서문

웹 환경이 변화하면서 사용자와의 상호작용이 웹 개발에서 큰 비중을 차지하게 되었습니다. 하지만 상호작용하는 웹 서비스는 복잡하고 구현하기가 어려웠습니다. 이를 가능하게 한 것이 HTML5의 새로운 기능인 WebSocket입니다. HTML5의 WebSocket 기술이 등장하면서 이를 활용하여 전보다 쉽게 실시간으로 상호작용하는 복잡한 웹 서비스를 구현할 수 있게 되었습니다. WebSocket은 지원하는 브라우저의 제한 탓에 초창기에 관심을 받지 못하였지만, Node.js의 Socket.IO가 인기를 얻으며 관련 기술로 다시 주목받고 있습니다. 이제는 브라우저에서 서버 푸시Push를 이용한 실시간 이벤트를 받아 다양한 작업도 할 수 있게 되었습니다.

이 책은 Vert.x라는 프레임워크의 개념과 다양한 예제를 통해 리얼타임 기술(WebSocket) 구축뿐만 아니라 일반적인 웹 개발에도 도움이 되도록 구성하였습니다.

필자는 Vert.x의 설치부터 간단한 웹 서버 구현, 확장 및 공개 모듈을 사용한 예제를 작성해 보면서 Vert.x의 이벤트 버스 시스템이 얼마나 강력한지 알 수 있었습니다. 특히 웹 애플리케이션을 모듈 구조로 개발하여 재활용할 수 있다는 점이 가장 인상적입니다. 이 책에서는 언급되지 않았지만, Vert.x는 웹 시스템뿐만 아니라 임베디드 형태로 어디서든 쓰일 수 있습니다.

가볍게 시작하는 마음으로 Vert.x라는 무거운 주제를 자바스크립트로 집필하였지만, 집필 도중 어려움도 있었습니다. 디버깅이나 자바스크립트의 콘솔 출력 기능이 훌륭하지 못하거나 필자의 자바스크립트 실력이 미약해서 느끼는 어려움 등이었습니다.

분명 자바스크립트는 굉장히 유연한 언어이면서 뛰어난 표현력을 가지고 있습니다. 자바스크립트뿐만 아니라 다른 언어로 구현해 본다면 분명 또 다른 재미를 느낄 수 있을 겁니다. 하지만 자바스크립트로 자바의 API를 사용하면 상대적으로 적은 양의 코딩으로 서비스를 구현할 수 있습니다. 독자들이 이런 재미를 느낄 수 있길 바랍니다.

대상 독자 및 참고사항

초급 **초중급** 중급 중고급 고급

이 책은 Vert.x를 이용해서 웹 프로그래밍에 리얼타임 기술을 구현하는 방법을 소개하는 책입니다. 자바스크립트를 알고 있고 자바의 기초적인 문법과 개념을 알면 누구나 읽을 수 있습니다.

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

- Java 7 이상 설치된 개발 환경

이 도서의 예제 소스는 다음 웹 사이트에서 내려받을 수 있습니다.

- https://github.com/cloud9ine/starting_vertx

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 Vert.x와 이벤트 기반 프로그래밍

전통적으로 동시성을 다루는 스레드 기반 네트워크 프로그래밍은 하나의 스레드를 각 I/O 리소스에 배분한다. 이것은 각 연결Connection에 하나의 스레드를 배정하는 것을 뜻하고, CPU가 하나의 스레드를 처리하기 위해 CPU의 자원과 메모리를 소비하는 동안에 다른 스레드는 대기 상태로 Blocking이 된다. 이 방식은 웹 서버에 한 번에 수천 개의 연결이 이루어지고 수많은 스레드가 CPU와 메모리에 부담을 줘서 때로는 서버 다운으로 이어지기도 한다.

반면에 이벤트 루프Event Loop 기반의 Non-blocking 네트워크 프로그래밍은 하나의 스레드로 수많은 연결을 처리할 수 있다. 이벤트 루프 스레드는 이벤트가 발생할 때마다 해당하는 처리(콜백 함수 호출)를 하고 다시 새 이벤트가 발생할 때까지 대기 상태로 들어간다. 이처럼 비동기 프로그래밍 모델은 적은 스레드로 많은 자원을 효과적으로 다룰 수 있다.

인터넷 대중화부터 최근 모바일과 소셜 네트워크의 발전까지 네트워크 사용자는 점점 늘어나고 있고, 웹 애플리케이션은 이전보다 더 많은 접속을 처리해야 한다. 이런 이유로 이벤트 기반의 비동기 방식으로 작동하는 새로운 아키텍처가 주목받고 있다.

이벤트 기반의 Non-blocking 방식으로 작동하는 프로그램은 Node.js, Redis, Nginx 등이 있다. 2012년 세계적인 인맥 사이트 LinkedIn에서 모바일의 백 엔드 서비스로 사용하던 RORRuby On Rails을 Node.js로 변경한 이유도 더 많은 트래픽을 처리하기 위해서다.

Vert.x는 이러한 시대의 요구에 부응하여 2011년 팀 폭스Tim Fox가 시작한 프로젝트로, 성능상으로는 Node.js를 압도한다.[01] 다음은 Node.js 와 Vert.x를 간단히 비교한 내용이다.

[표 1-1] Node.js와 Vert.x 비교

구분	Node.js	Vert.x
기반 언어	C	Java
내부 엔진	V8 JavaScript Engine	Netty
개발 지원 언어	JavaScript	JavaScript, Java, Ruby, Python, CoffeeScript, Groovy 등
모듈 저장소	상당히 많은 모듈이 존재한다.	120여 개
클러스터링	별도로 인스턴스를 실행해야 한다. 인스턴스 간 메시지 공유가 불가능하다.	노드 간 인스턴스 참조 공유가 가능하다.
에코 시스템	풍부하다.	자바 에코 시스템을 그대로 사용한다.
모듈 관리	npm cli 기반 명령어를 제공한다.	vertx cli 기반 명령어를 제공한다.
설치 및 작동 환경	시스템에 따라 빌드 시 종속 라이브러리가 존재한다.	자바 가상 머신만 필요하다.

전체적으로 Vert.x와 Node.js는 닮은 점이 많아 보인다. 하지만 Vert.x는 다양한 언어로 구현할 수 있어서 자바를 몰라도 접근할 수 있다. 현재 사용할 수 있는 모듈 수는 적지만, 자바의 라이브러리를 그대로 사용할 수 있고 모듈로 만들어 사용할 수 있으므로 문제가 되지 않는다. 무엇보다 멀티코어를 지원하는 덕에 Node.js처럼 다수의 인스턴스를 실행시킬 필요가 없을 뿐만 아니라 복잡한 설정도 필요하지 않다.

NOTE

- 현재 Vert.x를 사용하는 곳
  http://game-gorilla.com/
  http://gvmtool.net
  NHN은 차기 RTCS(Real Time Communication System)로 사용할 플랫폼에 Vert.x를 고려 중

01 http://vertxproject.wordpress.com/2012/05/09/vert-x-vs-node-js-simple-http-benchmarks/

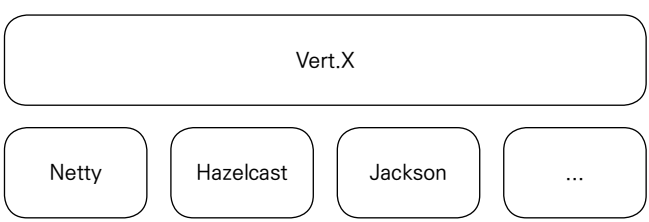
이 책에서는 Vert.x의 기본 개념과 아키텍처를 설명하고 이벤트 기반 애플리케이션을 쉽게 작성할 수 있도록 다양한 예제를 제공한다. 모든 예제는 자바스크립트로 구현되고 자바로 작성된 전체 소스를 제공한다. 자바스크립트를 선택한 이유는 사용자에게 친숙하고 작성하기 편하며 이해하기 쉽다는 점이다.[02]

1.2 Vert.x의 특징

Vert.x는 이벤트 기반의 범용 자바 플랫폼으로 비동기 처리를 쉽게 개발하기 위해 설계되었다. 물론 자바에도 비동기 처리를 위한 nio나 Netty라는 근사한 프레임워크가 존재하지만, 학습하는 데 많은 시간이 필요하다. 그러나 Vert.x는 자바의 에코 시스템을 최대한 이용할 수 있도록 심플한 API를 제공하여 배우기가 쉽다.

Vert.x는 [그림 1-1]과 같은 기반 시스템을 사용한다. 이벤트 기반 프로그래밍을 쉽게 할 수 있도록 고레벨의 이벤트 모델을 제공하고, 저레벨에서는 Netty와 Hazelcast를 사용해서 클러스터링을 구성한다.

[그림 1-1] Vert.x의 기반 시스템



02 다른 언어를 선호한다면 해당 언어의 API 문서를 보고 접근할 수 있다.

NOTE

- Netty: 네트워크 프로그래밍을 위한 비동기 처리 프레임워크
- Hazelcast: 확장 가능한 클러스터링을 구성하기 위한 데이터 분산 레이어
- Jackson: 고성능의 JSON 프로세서

Vert.x는 Node.js와 다른 다음과 같은 고유한 특징이 있다.

비동기 처리Asynchronous

Vert.x는 서버와 클라이언트, 파일 처리, 데이터베이스 처리 프로그램 등 모든 API를 비동기 처리로 작성할 수 있다. 이 때문에 I/O 부하가 심한 대규모 서비스를 개발하기에 적합하다.

JVMJava Virtual Machine

자바는 수년간 안정성을 검증받은 언어로, 자바 가상 머신 위에서 작동한다. Vert.x도 마찬가지로 자바 가상 머신 위에서 작동하며 JVM의 이점을 그대로 이어받는다. 특히 멀티코어를 지원하기 때문에 고성능 프로그램을 작성할 수 있다.

분산 이벤트 버스Distributed Event Bus

이벤트 버스는 Vert.x의 신경계 시스템으로 서로 다른 Verticle 사이의 메시지 교환을 가능하게 한다. 이벤트 버스는 쉽게 확장할 수 있고, 로컬 머신 뿐만 아니라 다른 머신에 있는 Verticle과도 통신할 수 있다. 또한, 클라이언트와도 SockJS를 통해 통신이 가능하다

다양한 언어 지원Polyglot

폴리글랏Polyglot은 사전적 의미로 '여러 언어에 능통한'이라는 뜻이다. 즉, Vert.x가 특정 언어를 위한 플랫폼이 아니며 다양한 언어로 작성할 수 있다는 것을 뜻한

다. 현재 JavaScript, Java, Ruby, Python, CoffeeScript, Groovy로 작성할 수 있으며, 조만간 Clojure와 Scala 그리고 PHP도 지원할 예정이다.

모듈 시스템Module System

Vert.x에는 이미 존재하는 여러 자바 API를 사용하기 위한 모듈이 존재한다. 메이븐Maven과 Bintray에 산재된 모듈을 자신만의 모듈로 통합할 수 있고, 모듈 저장소[03]에도 공개할 수 있다.

1.3 Vert.x의 주요 용어

Vert.x에서는 고유 용어를 새로 정의하거나, 일반적인 용어를 다시 정의해서 사용하기도 한다. Vert.x에서 사용하는 주요 용어는 다음과 같다.

Verticle

Vert.x에서 실행되는 모든 코드는 Verticle이라고 할 수 있다. 즉, 자바의 main 메소드를 가지고 있는 클래스라고 생각하면 쉽다. Verticle은 각기 다른 언어로 작성할 수 있고, 하나 또는 복수의 Verticle을 하나의 애플리케이션으로 구성할 수도 있다. 각 Verticle은 자신의 유일한 클래스로더를 가지고 있어서 서로 다른 Verticle이 정적Staic 변수나 전역Global 변수 등에 접근할 수 없도록 한다.[04] 따라서 Verticle끼리는 직접 통신할 수 없으므로 이벤트 버스를 통해서 메시지를 주고받는다.

Vert.x Instance

모든 Verticle은 Vert.x 인스턴스 내부에서 실행된다. 하나의 Vert.x 인스턴스에서 여러 개의 Verticle이 동시에 실행될 수 있다.

---

03 http://modulereg.vertx.io/

04 사실 Vert.x 런타임 때문에 불가능하다.

Event Loop

Vert.x 인스턴스는 내부적으로 스레드 풀을 관리하고 CPU 코어 개수만큼 스레드 풀을 가질 수 있다. 또한, 오직 1개의 이벤트 루프를 가진 Node.js와 달리 각 스레드는 각자의 이벤트 루프를 실행하고 이벤트 루프는 끊임없이 루프를 순회하면서 발생하는 이벤트를 감지한다.

Event Bus

Verticle 간 메시지를 주고받는 일종의 통로라고 볼 수 있다.

Message Passing

Verticle 간 메시지로는 JSON, 문자열, 숫자, 오브젝트, 바이트 버퍼 등을 사용할 수 있다. 이 책에서는 주로 JSON을 사용하며, 주된 개발 언어가 자바스크립트일 때 JSON은 더욱 다루기 쉽다.

2장에서는 본격적으로 Vert.x를 작동시키기 전 사전 준비와 간단한 예제를 통해 폴리글랏의 의미를 살펴본다.

2 | Vert.x 설치 및 개발 환경

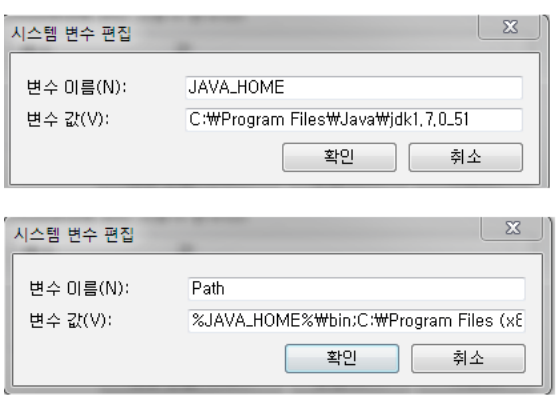
2.1 사전 준비

Vert.x는 JVM 위에서 작동하므로 반드시 Java를 설치해야 한다. 유의할 점은 Java 7부터 invokedynamic이라는 새로운 명령어 셋이 추가되어 동적으로 바이트 코드를 생성할 수 있다. 따라서 폴리글랏에 해당하는 다른 동적 언어로 코드를 작성하는 Vert.x가 작동하려면 Java 7 버전 이상으로 설치해야 한다. 여기서는 JDK 7을 사용하므로 JDK 다운로드 페이지[01]에서 JDK 7 파일을 내려받아서 설치하고 운영체제에 맞는 환경 변수를 설정한다.

윈도우

'컴퓨터 속성 → 시스템 → 고급 시스템 설정 → 고급 → 환경 변수'에서 JAVA_HOME을 설정한다.

[그림 2-1] 자바 환경 변수 설정



01 http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

리눅스 또는 OS X

/etc/profile이나 ~./bash_profile을 편집한다.

```
export JAVA_HOME=/usr/local/jdk7
export PATH=$PATH:$JAVA_HOME/bin
```

수정 후 환경 변수에 적용하기 위해(/etc/profile을 수정했다면) 다음 명령어를 실행하여 시스템 환경 변수를 갱신한다.

```
$ source /etc/profile
```

환경 변수 설정이 끝나면 자바가 제대로 작동하는지 확인한다. 다음 명령어를 실행하면 정상적으로 설치되었는지 확인할 수 있다.

```
> java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```

2.2 Vert.x 설치

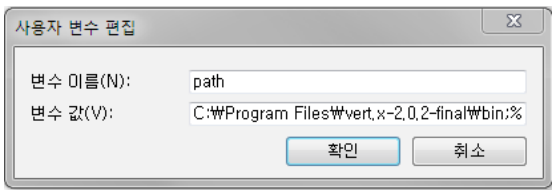
준비 과정을 마쳤다면 이제 Vert.x를 설치한다. Vert.x 홈페이지[02]에서 최신 버전[03]을 내려받아서 압축을 해제하고 경로를 설정한다.

02 http://vertx.io

03 이 책에서는 2.0.2-final 버전을 사용한다.

윈도우

[그림 2-2] Vert.x 환경 변수 설정



리눅스 또는 OS X

마찬가지로 /etc/profile이나 ~./bash_profile을 편집하고 source [파일 이름]으로 환경 변수를 갱신한다.

```
export PATH=$PATH:$JAVA_HOME/bin:/usr/local/vert.x-2.0.2-final/bin
```

Vert.x 작동 확인

설치가 끝나면 다음과 같이 작동을 확인한다.

```
> vertx version
2.0.2-final (built …….
```

2.3 개발 환경 및 IDE 사용하기

여기서는 이클립스Eclipse와 메이븐Maven을 사용하여 Vert.x의 개발 환경을 구축하는 방법을 살펴보겠다.

2.3.1 개발 환경

텍스트 에디터

자바스크립트를 작성하기에 편한 텍스트 에디터라면 어느 것이든 가능하다.

웹 브라우저

웹 브라우저는 콘솔 모드를 이용하는 데 필요하다. 크롬에서 제공하는 콘솔을 이용하면 특히 이벤트 버스를 이용하는 간단한 테스트 등에 아주 유용하다. 이 책에서는 크롬 브라우저를 사용하지만, 콘솔 기능을 제공하는 어떤 브라우저를 사용해도 관계없다. 크롬에서 콘솔 모드를 실행하려면 브라우저를 실행한 후 'Chrome 맞춤 설정 및 제어 → 도구 → 자바 스크립트 콘솔'을 선택하거나 단축키 'Ctrl+Shift+J' 또는 'F12'를 눌러서 콘솔 탭을 사용한다.

IDE

개발 통합 환경을 원한다면 이클립스나 IntelliJ, Netbeans 등을 사용할 수 있다. 그러나 자바로 개발할 때는 모두 사용할 수 있지만, 자바스크립트가 메인 언어일 때는 아직 디버깅 환경 등이 아직 갖추어지지 않는 상태다.

기타

Http 프로토콜을 테스트하려면 curl이나 wget과 같은 툴을 사용하면 좋다. 사용하려는 브라우저가 플러그인을 지원하면 Rest Client를 사용해보기 바란다. 참고로 크롬에서 Rest Client를 설치하려면 메뉴 → 도구 → 확장 프로그램에서 크롬 웹 스토어로 가면 내려받을 수 있다.

2.3.2 메이븐

메이븐 프로젝트 홈페이지[04]에서 설치 파일을 내려받는다.[05]

---

04 http://maven.apache.org/download.cgi

05 이 책에서는 3.2 버전을 사용한다.

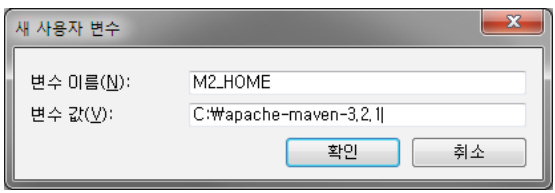
설정

메이븐의 CLI 명령어를 사용하려면 환경 변수에 메이븐 경로를 HOME이라는 변수에 추가한다.

윈도우

M2_HOME이란 환경 변수를 설정하고 PATH 변수에도 추가한다.

[그림 2-3] 메이븐 환경 변수 설정



리눅스 또는 OS X

/etc/profile이나 ~./bash_profile을 다음과 같이 수정한다.

```
export M2_HOME=/usr/local/apache-maven-3.2.1
export PATH=$PATH:$M2_HOME/bin
```

프로젝트 생성

CLI 명령어를 사용하여 프로젝트를 생성한다.[06]

```
> mvn archetype:generate -DarchetypeGroupId=io.vertx
-DarchetypeArtifactId=vertx-maven-archetype -DarchetypeVersion=2.0.5-final
-DgroupId=local.vertx -Dartifa
```

06 이 책을 쓰는 시점에는 아키타입(Archetype)이 2.0.5-final 버전이었다. 만약 잘못된 버전을 사용해도 메이븐에서 사용할 수 있는 아키타입을 제시한다.

```
ctId=hello -Dversion=0.0.1-SNAPSHOT
[INFO] Scanning for projects...
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/
maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom
Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/
maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom (4 KB at 4.0 KB/sec)
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/
maven-plugins/22/maven-plugins-22.pom
Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/plugins/
maven-plugins/22/maven-plugins-22.pom (13 KB at 35.5 KB/sec)
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/
maven-parent/21/maven-parent-21.pom
Downloaded:
http://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/21/
maven-parent-21.pom (26 KB at 50.0 KB/sec)
Downloading:
http://repo.maven.apache.org/maven2/org/apache/apache/10/apache-10.pom

중략

Downloaded:
http://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy/1.8.3/groovy-
1.8.3.jar (5394 KB at 171.2 KB/sec)
[INFO] Generating project in Interactive mode
[INFO] Archetype repository missing. Using the one from
[io.vertx:vertx-maven-archetype:2.0.5-final] found in catalog remote
[INFO] Using property: groupId = local.vertx
[INFO] Using property: artifactId = hello
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = local.vertx
Confirm properties configuration:
groupId: local.vertx
```

```
artifactId: hello
version: 0.0.1-SNAPSHOT
package: local.vertx
 Y:: //여기에서 Y를 입력한다
[INFO] ------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: vertx-
maven-archetype:2.0.5-final
[INFO] ------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: local.vertx
[INFO] Parameter: artifactId, Value: hello
[INFO] Parameter: version, Value: 0.0.1-SNAPSHOT
[INFO] Parameter: package, Value: local.vertx
[INFO] Parameter: packageInPathFormat, Value: local/vertx
[INFO] Parameter: package, Value: local.vertx
[INFO] Parameter: version, Value: 0.0.1-SNAPSHOT
[INFO] Parameter: groupId, Value: local.vertx
[INFO] Parameter: artifactId, Value: hello
[WARNING] Don't override file
C:\hello\src\main\resources\platform_lib\README.txt
[INFO] project created from Archetype in dir: C:\hello
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 03:57 min
[INFO] Finished at: 2014-05-01T11:04:06+09:00
[INFO] Final Memory: 12M/122M
[INFO] ------------------------------------------------------------------------
```

2.3.3 이클립스와 연동하기

이클립스 프로젝트로 변경

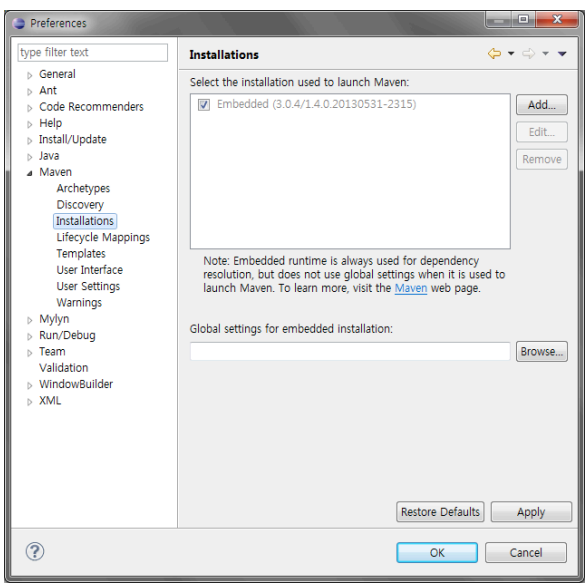
생성한 프로젝트를 이클립스에서 사용하려면 다음과 같이 입력한다.

```
> cd hello
> mvn eclipse:eclipse
```

메이븐 설정 정보 변경

이제 이클립스를 실행하고 메이븐 설정 정보를 변경한다. Window → Preferences → Maven → Installations에서 Embedded 체크를 해제하고 Global settings for embedded installation에서 이미 설치된 메이븐의 설정 파일을 찾아서 사용한다.

[그림 2-4] 메이븐 설정 정보 변경

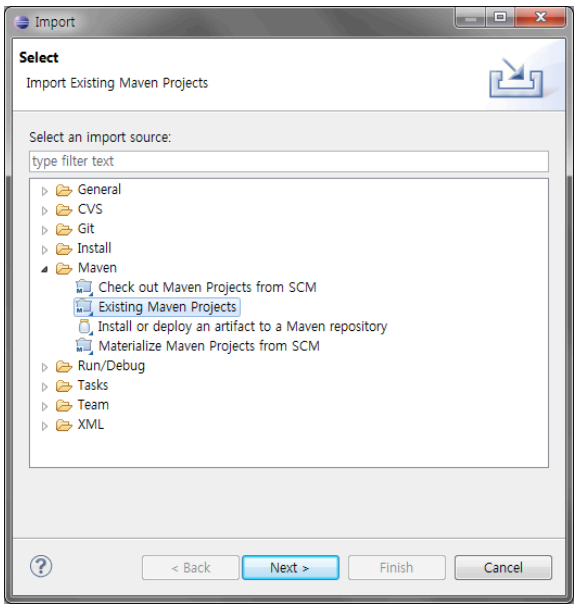


C:\Users\[사용자계정]\.m2\archetype-catalog.xml라는 파일을 선택한다.

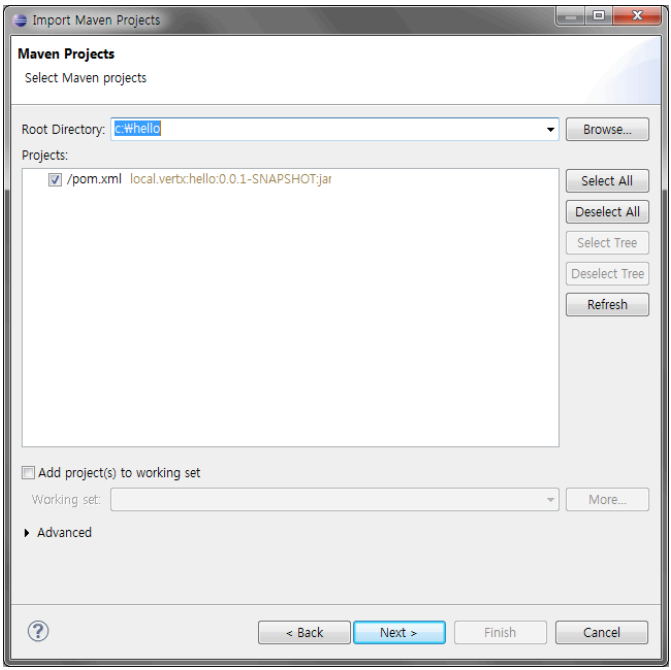
이클립스 프로젝트 Import

이제 메이븐 CLI로 생성한 프로젝트를 Import한다. 이클립스에서 File → Import를 실행하고 Maven 폴더의 Existing Maven Projects를 선택한다.

[그림 2-5] 프로젝트 Import



[그림 2-6] 프로젝트 선택



원격 디버깅을 위한 예제 작성

원격Remote 디버깅을 위한 간단한 예제를 작성해 보자. 콘솔에서 서버를 실행하고 텔넷으로 서버에 접속하는 예제다.

[TcpServer.java]

```
package local.net;

import org.vertx.java.core.Handler;
import org.vertx.java.core.net.NetSocket;
import org.vertx.java.platform.Verticle;
```

```
public class TcpServer extends Verticle{
     @Override
     public void start() {

          vertx.createNetServer().connectHandler(new Handler<NetSocket>() {

               @Override
               public void handle(NetSocket socket) {

                    String addr = socket.remoteAddress().getHostName();

                    socket.write("Welcome to the MyServer from " + addr);
               }
          }).listen(23);
     }
}
```

서버를 실행하기 전에 Vert.x의 실행 파일 vertx.bat 또는 vertx(셸 스크립트)에 옵션을 설정한다.

[vertx.bat]

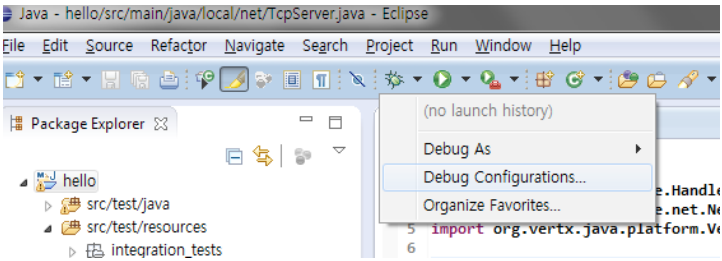
```
set
JVM_OPTS=-agentlib:jdwp=transport=dt_socket,address=8000,server=y,suspend=n
```

[실행 결과]

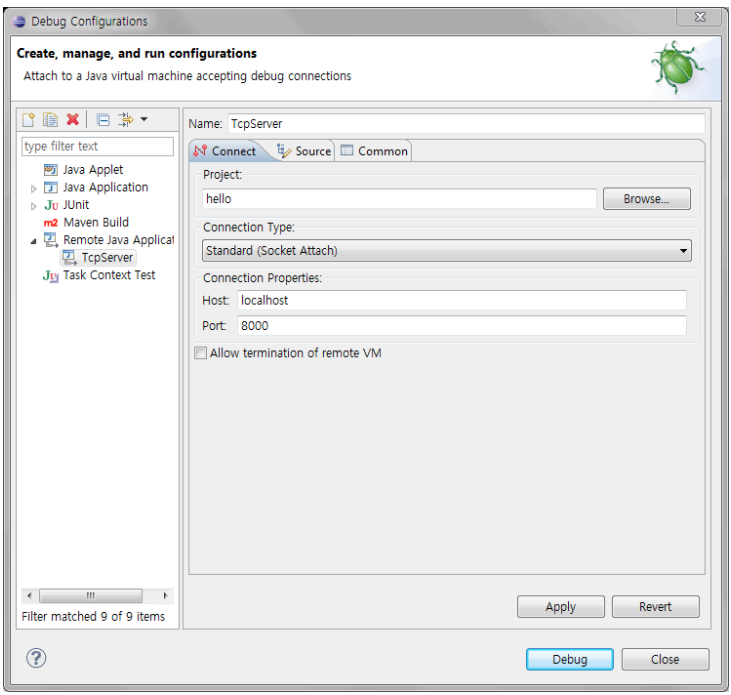
```
> vertx run local/net/TcpServer.java
Listening for transport dt_socket at address: 8000
```

실행하면 8000 포트를 통해서 원격으로 디버깅할 수 있다. 이제 이클립스에서 디버그 환경([그림 2-6]과 [그림 2-7] 참고)에 대한 상세한 정보를 설정한다.

[그림 2-7] 디버그 환경 설정 1



[그림 2-8] 디버그 환경 설정 2

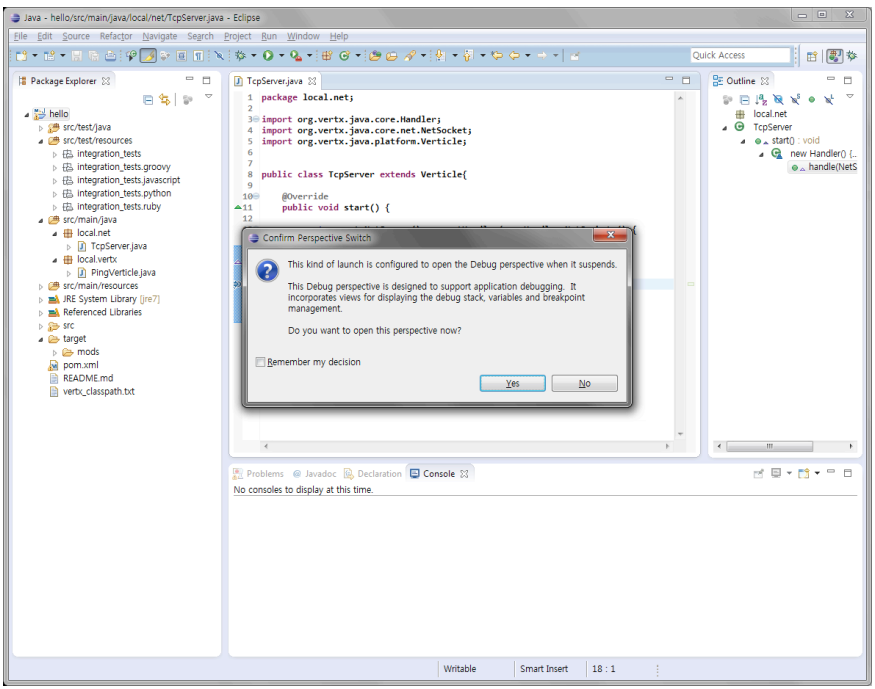


[텔넷으로 접속]

텔넷으로 접속하자마자 이클립스의 디버그 퍼스펙티브Perspective가 활성화된다.

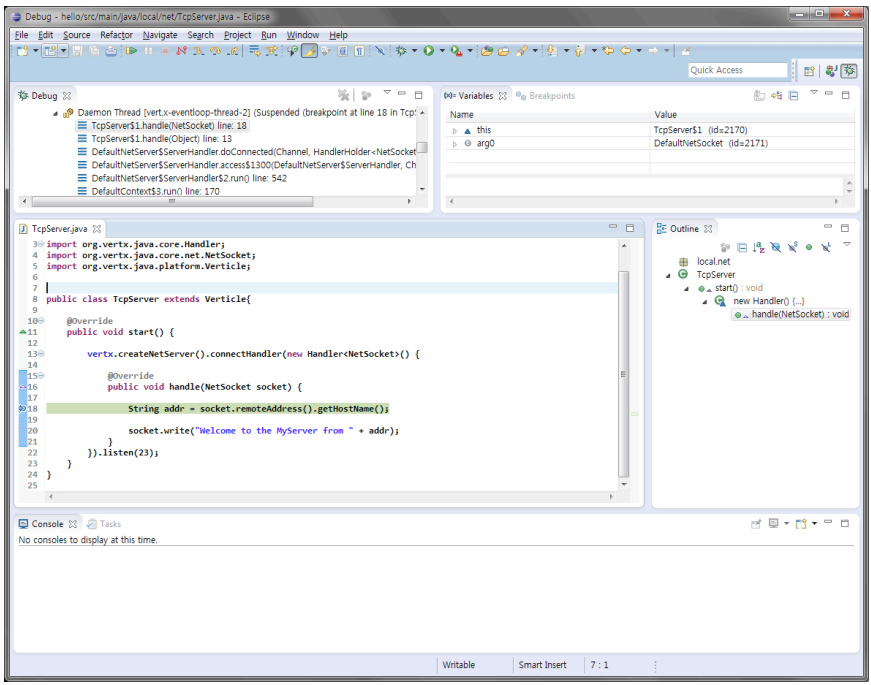
```
> telent localhost
```

[그림 2-9] 디버그 퍼스펙티브 활성화



다음과 같이 원격 디버깅이 실행된다.

[그림 2-10] 원격 디버깅 실행 화면



2.4 간단한 확인 예제

'Hello World'라는 메시지를 출력하는 간단한 웹 서버를 작성하여 설치 결과를 테스트해 보자. 여기서는 Vert.X의 Ployglot 특성을 보여주기 위해 예제를 언어별로 작성하였다.

JavaScript로 작성

```
var vertx = require('vertx');
```

```
vertx.createHttpServer().requestHandler(function(req) {
     req.response.end("Hello World");
}).listen(8080);
```

CoffeeScript로 작성

```
vertx = require 'vertx'
vertx.createHttpServer().requestHandler( (req) -> req.response.end("Hello World")
).listen 8080
```

Python으로 작성

```
Import vertx
server = vertx.create_http_server()
@server.request_handler
def handle(req):
     req.response.end("Hello World")
server.listen(8080)
```

Ruby로 작성

```
require 'vertx'
include Vertx
HttpServer.new.request_handler do |req|
     req.response.end("Hello World")
end.listen(8080)
```

Java로 작성

```
import org.vertx.java.platform.Verticle;

public class HelloWorld extends Verticle {
      public void start() {
            vertx.createHttpServer().requestHandler(new
Handler<HttpServerRequest>() {
                        public void handle(HttpServerRequest req) {
                              req.response().end("Hello World");
                        }
      }).listen(8080);
}
```

Groovy로 작성

```
vertx.createHttpServer().requestHandler {
      req -> req.response.end "Hello World"
}.listen(8080)
```

예제를 실행하려면 명령어 프롬프트 또는 셸에서 다음과 같이 입력한다.

```
vertx run 파일 이름
```

브라우저에서 http://localhost:8080으로 접속하면 인사말을 볼 수 있다. 서버를 멈추려면 'Ctrl+C'를 누른다.

예를 들어, 자바스크립트로 작성된 예제를 실행하면 해당 언어의 확장자에 해당하는 Polyglot 구현체를 내려받아서 설치한다. 지원하는 언어별 구현체는 Vert.x가

설치된 conf 디렉터리 내부의 langs.properties에서 확인할 수 있다.

[langs.properties]

```
rhino=io.vertx~lang-rhino~2.0.0-final:org.vertx.java.platform.impl.RhinoVerti
cleFactory
......
.js=rhino
.coffee=rhino
......
```

자바스크립트(CoffeeScript 포함)는 라이노 스크립트 엔진에서 실행되는 것을 알 수 있다. 해당 구현체는 Vert.x가 설치된 경로의 sys-mods 아래에 io.vertx~lang-rhino~2.0.0~final이라는 디렉터리 이름으로 설치된다.

여기서는 라이노 스크립트 엔진을 사용하고 있지만, 다른 스크립트 엔진으로도 교체할 수 있다. 예를 들어, dyn.js나 Java 8에서 도입되는 Nashorn 등도 가능하다. dyn.js는 초기 버전이지만, 공개 모듈 저장소에서 모듈로 제공되고 있다. 교체 방법은 langs.properties를 참조하여 구현체 엔진의 클래스 이름만 적으면 간단히 교체할 수 있다.

라이노 스크립트 엔진

라이노 스크립트Rhino Script는 모질라 재단에서 만든 오픈 소스 자바스크립트 엔진으로, 엔진 자체는 자바로 작성되었고 ECMAScript 5(ECMA-262)와 호환된다. 인터프리터 모드로 작동하거나 .js 파일을 자바 클래스로 컴파일할 수 있으며, 자바의 API를 사용할 수 있다.

현재 라이노 스크립트 엔진에서 구현된 파일의 상세 설명은 다음과 같다.

[표 2-1] 라이노 스크립트 파일

파일명	설명
buffer.js	버퍼를 다루는 모듈
console.js	콘솔, 로그 관련 모듈
container.js	Verticle 설치, 삭제 관련 모듈
event_bus.js	이벤트 버스 모듈
file_system.js	파일 시스템 모듈
http.js	HTTP 서버와 클라이언트를 구성하는 모듈
multi_map.js	키와 값을 가지는 맵 유틸리티
net.js	비동기 서버와 클라이언트를 구성하는 모듈
pump.js	펌프처럼 특정 스트림을 읽어서 다른 스트림으로 전달하는 모듈
read_stream.js	데이터 스트림 읽기 관련 모듈
shared_data.js	Verticle 간 데이터를 공유하게 하는 모듈
sockjs.js	HTTP 서버를 SockJS 서버로 사용할 수 있게 하는 모듈
ssl_support.js	SSL을 이용한 보안 모듈
timer.js	타이머 관련 모듈
write_stream.js	데이터 스트림 쓰기 관련 모듈

이 장에서는 Vert.x를 설치하고 기본적인 개발 환경을 설정했다. 3장부터는 자바스크립트를 이용하여 본격적으로 Vert.x를 구성하고 웹 애플리케이션을 구현하는 과정을 진행해 본다.