Hanbit eBook

Realtime 89

자바 개발자를 위한

Vert.x 애플리케이션 개발

이연복 지음

한빛미디어 Hanbit Media, Inc.

자바 개발자를 위한

Vert.x
애플리케이션 개발

자바 개발자를 위한 **Vert.x 애플리케이션 개발**

초판발행 2015년 2월 5일

지은이 이연복 / **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 978-89-6848-728-6 15000 / **정가** 14,000원

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

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_이연복

“들은 것은 잊어버리지만 본 것은 기억한다. 하지만 행동해야 진정으로 이해할 수 있다.”라는 말에 깊이 공감하고, 선택의 문제가 산재한 삶 속에 적용하기 위해 고민하는 개발자다. 그 결과, “하지 않고 후회하는 것보다 하고 나서 후회하는 것이 낫다.”라는 신념을 지니게 되었고 이 신념에 따라 2011년 SW 마에스트로 과정을 마친 직후 ‘helloworld’라는 벤처기업을 시작했다. 현재까지 ‘helloworld’에서 Java를 사용하여 다양한 서비스 개발에 참여하고 있다. 주로 서버 사이드 개발에 관심이 많다.

저자 서문

바야흐로 모바일 시대를 넘어 IoT[Internet of Things] 시대가 성큼 다가왔습니다. 많은 기기가 다양한 센서를 통해 외부 세계를 인지하고 정보를 수집하며, 수집된 정보는 기기가 연결된 네트워크를 따라 다른 기기 또는 인터넷을 통해 그것을 처리할 서버로 보내집니다. IoT 기기에 네트워크 기능이 없다면 이들 기기의 활용도는 매우 제한적일 수밖에 없습니다. 센서로 수집한 날씨, 속도, 사용자 생체 정보 등은 다른 기기 또는 인터넷과의 연동으로 좀 더 가치 있고 쓸모 있는 정보로 취급될 수 있기 때문입니다. 따라서 IoT 시대에서 빼놓을 수 없는 중요한 특징 중 하나가 바로 네트워크 기술입니다.

이에 따라 수많은 IoT 기기와 연결되고, 이들 기기가 보내오는 정보를 효과적으로 처리하기 위해서는 새로운 형태의 애플리케이션 개발 방법이 필요합니다. 전통적으로 이런 요구 사항을 반영하는 애플리케이션은 다중 스레드 방식으로 개발됐습니다. 그러나 다중 스레드 방식은 연결된 기기가 많아질수록 시스템 리소스가 과도하게 소모된다는 단점이 있는데, Vert.x는 이런 단점을 해결하고 효과적으로 다수의 클라이언트를 동시에 처리할 수 있는 강력한 방법을 제공합니다. Vert.x에서는 이 방법을 Multireactor라고 정의합니다.

최근 주목받는 또 다른 기술로 WebSocket을 꼽을 수 있습니다. HTML5에 포함된 WebSocket은 웹 브라우저와 웹 서버 간의 양방향 통신을 지원하는 차세대 표준으로, 웹 애플리케이션의 반응성을 극적으로 향상하여 기존의 Ajax에 기반을 둔 대화형 웹 애플리케이션을 뛰어넘는 훨씬 효율적이고 역동적인 웹 애플리케이션 개발을 가능하게 합니다. Vert.x에서는 WebSocket을 훌륭하게 지원하고 있으며, 이에 더해 WebSocket 에뮬레이터 중 하나인 SockJS로 좀 더 편리하게 실시간 웹 애플리케이션을 개발할 수 있도록 API를 제공하고 있습니다.

특히 SockJS Event Bus Bridge라는 확장 기술은 기존 웹 애플리케이션의 한계를 벗어나 좀 더 자유롭게 애플리케이션을 설계하는 데 도움을 주는 아주 강력한 도구입니다.

마지막으로, Vert.x는 요즘 주목받는 폴리글랏polyglot 패러다임을 지원합니다. 폴리글랏이란 여러 개의 언어를 동시에 사용할 수 있는 특징을 말하는데, 이 책에서 Vert.x 예제를 작성하기 위해 사용된 Java 외에도 JavaScript, Ruby, Python, Groovy, Clojure, Scala 등 다양한 언어를 사용할 수 있습니다. 물론 동일한 프로젝트에서 2개 이상의 언어를 혼용해 사용하는 것도 가능합니다.

Vert.x는 앞에서 언급한 내용 외에도 주목할 만한 몇 가지 특징과 함께 빠르게 성장하고 있는 오픈소스 플랫폼입니다. Vert.x가 시작된 지 이제 겨우 3년이 지났지만, 최근 JAX Innovation awards 2014에서 Most Innovative Java Technology 부문을 수상하며[01] 더욱 그 가치를 높여가고 있는 만큼 Vert.x는 이제 전 세계 수많은 개발자가 주목하는 이벤트 기반 비동기 프로그래밍 모델의 대표 플랫폼이라 할 수 있습니다.

이 책은 이처럼 빠르게 성장하고 있는 Vert.x의 중요한 개념과 철학, 기본적인 응용 방법을 설명하고, 이를 통해 다양한 요구 조건이 있는 실무 프로젝트에서 Vert.x가 활용될 수 있기를 바라며 집필하였습니다. Java로 작성된 이 책의 예제들을 JavaScript나 Scala로 다시 작성해 보는 것도 좋은 학습 방법이 될 것입니다. 특히, 최근 발표된 Java 8과 함께 도입된 람다식을 적용해보는 것도 아주 흥미로운 주제가 될 것 같습니다.

---

01 https://jax.de/awards2014/

Vert.x를 사용하며 막히는 부분은 Vert.x Google Groups[02]를 참고하거나 가능하다면 직접 Vert.x 구현 소소 코드를 살펴볼 것을 권장합니다. 개인적으로 필자에게 연락을 주신다면 필자의 능력 내에서 최대한 답변하겠습니다.

끝으로, 이 책을 집필할 기회를 제공해준 김창수 팀장님, 정지연 과장님을 비롯한 한빛미디어 관계자 여러분께 감사드리며, 항상 곁에서 응원해주는 사랑하는 가족과 여자친구 인희에게도 감사의 말을 전합니다.

02 https://groups.google.com/forum/#!forum/vertx

대상 독자 및 참고사항

초급 **초중급** 중급 중고급 고급

이 책은 Vert.x를 이용하여 실시간 채팅 서비스를 개발하는 과정을 담고 있습니다. Vert.x를 어느 정도 알고 있다면 좀 더 쉽게 이 책의 내용을 이해할 수 있지만, 꼭 Vert.x에 관한 사전 지식이 있을 필요는 없습니다. 그러나 이 책의 예제 소스는 대부분 Java로 작성되었기 때문에 Java의 기초적인 문법과 개념을 알고 있어야 합니다.

이 책의 예제 코드를 실행하려면 다음 환경이 갖춰져 있어야 합니다.

- JDK 7 이상 설치된 개발 환경
- Vert.x 2.1 이상 설치된 개발 환경

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

- https://github.com/devop84/vertx-examples

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 | 들어가며

Vert.x의 간단한 역사와 핵심원리, 특징을 살펴본 후 이클립스에서 메이븐 기반으로 Vert.x 애플리케이션을 개발하는 방법을 알아본다. 이 장을 마칠 때는 언제나 그렇듯 'hello, world'를 출력하는 아주 간단한 Vert.x 애플리케이션을 볼 수 있다.

1.1 Vert.x의 소개

Vert.x는 2011년 팀 폭스Tim Fox에 의해 시작된 후로 VMware의 지원 아래 발전하다가 팀 폭스의 거취 문제로 우여곡절을 겪은 끝에 현재는 Eclipse Foundation에서 관리하고 있다.[01] 최근 JAX Innovation awards 2014에서 Most Innovative Java Technology 부문을 수상하며[02] 더욱 그 가치를 높여가고 있는 만큼 Vert.x는 이제 전 세계 수많은 개발자가 주목하는 이벤트 기반 비동기 프로그래밍EAP[03] 모델의 대표 플랫폼이라 할 수 있다.

Vert.x와 유사한 개발 모델을 제공하는 또 다른 유명한 플랫폼으로 Node.js를 언급하지 않을 수 없다. Node.js는 라이언 달Ryan Dahl에 의해 2009년부터 시작된 프로젝트로, Vert.x보다 앞선 시기에 이미 많은 이슈와 관심을 불러일으켰고, 그 결과 오늘날 Vert.x보다 훨씬 풍부한 에코 시스템을 보유하고 있다.

Vert.x는 Node.js에서 영감을 얻어 시작된 프로젝트인 만큼 Node.js를 접해본 개발자라면 아주 쉽고 빠르게 Vert.x에 익숙해질 수 있을 정도로 근본 원리는

01 http://www.infoq.com/news/2013/01/vertx-eclipse

02 http://jax.de/awards2014

03 편집자 주_ Event-based Asynchronous Pattern과 혼동될 수 있으나 이 책에서는 Event-based Asynchronous Programming의 약자로 사용한다.

Node.js의 그것과 크게 차이가 없지만, 한편으로는 Node.js의 단점을 극복하고 더욱 발전된 구조와 성능을 보여준다.[04] 또한, Vert.x가 JVM(Java Virtual Machine)이라는 오랜 시간 동안 충분히 검증된 훌륭한 공학적 산물 위에서 동작하는 만큼 JVM의 안정성과 성능 등의 이점을 그대로 안고 갈 수 있다. 그리고 Node.js와는 비교할 수 없을 정도로 방대한 Java 레퍼런스 및 서드 파티 라이브러리(Third Party Library)를 그대로 이용할 수 있다는 점은 Java 개발자뿐만 아니라 EAP 플랫폼을 주목하는 모든 개발자에게 아주 매력적인 장점이다.

그런데 최근에 Vert.x나 Node.js와 같은 EAP 플랫폼이 주목받는 이유는 무엇일까? 어떤 대단한 알고리즘이나 신기술이 있어 개발자로 하여금 풀기 어려운 문제를 극복할 수 있게 하는가? EAP 플랫폼에서 제공하는 동시 처리(Concurrency) 모델의 장단점과 주요 특징을 알아보며 이 질문에 대한 해답을 생각해 보자.

1.2 왜 Vert.x인가

결론부터 말하자면 여기에 그 어떤 새로운 알고리즘이나 신기술은 없다. EAP 플랫폼에서 제공하는 동시 처리 모델은 10년도 더 된 오래된 기술이다.[05] 네트워크 프로그래밍 경험이 있는 독자라면 잘 알고 있는 `select()`, `poll()`, `WaitForMultipleObjects()`와 같은 I/O 디멀티플렉서(Demultiplexer)에 기초한 Reactor 모델이 바로 EAP 플랫폼의 핵심 원리라 할 수 있다.

---

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

05 오래된 기술을 사용하고 있다고 해서 아무런 의미가 없는 것은 아니다. 본래 Reactor 모델은 프로그래밍 복잡도가 높고 디버깅하기가 어렵다. Vert.x에서는 이러한 Reactor 모델의 복잡도를 낮추고 일반 개발자들이 좀 더 쉽게 접근하고 사용할 수 있도록 간결하고 일관성 있는 API를 제공한다.

NOTE_

리눅스 커널 2.6 이상의 시스템에서 Java 6 이후 버전을 사용한다면 기존의 `select()`, `poll()`보다 더 나은 I/O 처리 성능을 보여주는 `epoll()`이라는 고성능 디멀티플렉서를 기본으로 사용하게 된다.[06] Windows 시스템에서는 Reactor 모델의 효율적인 구현이 어렵다. 그러나 Windows 시스템에서 제공하는 IOCP는 `epoll()`을 기반으로 구현한 Reactor 모델에 절대 뒤떨어지지 않는 성능을 보여준다. IOCP는 Reactor 모델과 마찬가지로 최대한 적은 수의 스레드로 동시 처리를 가능하게 하는 또 다른 방식으로 Proactor 모델이라 한다.

1.2.1 Vert.x의 동시 처리 모델

Reactor 동시 처리 모델

많은 작업을 동시에 수행해야 하는 애플리케이션은 다중 스레드(운영체제나 하드웨어에 따라 스레드가 아닌 프로세스를 사용할 수도 있음)를 사용해 작업을 처리하도록 디자인하는 것이 일반적인 방법이다. 이 방법은 요청된 작업을 처리하는 데 필요한 시간이 일정하지 않은, 특히 장시간 처리가 필요할 때 적합한 방법이다.

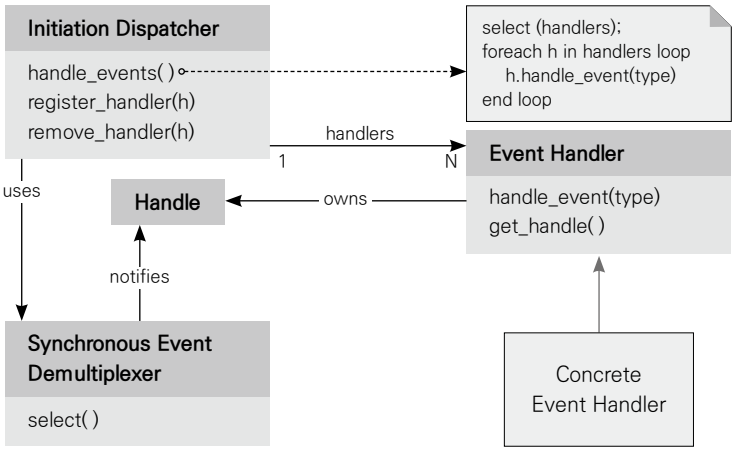
그러나 동시에 실행되는 2개 이상의 스레드 사이의 상호 작용과 공유 자원 동시 접근 문제 등을 다루려면 세마포어Semaphore나 뮤텍스Mutex 같은 동기화 장치를 반드시 사용해야 하는데, 이는 곧 데드락Deadlock 같은 잠재적인 문제를 내재하고 있음을 의미한다. 또한, 스레드의 생성과 관리는 증가하는 스레드에 비례하여 스레드 스택 메모리 유지와 문맥 전환Context Switching에 필요한 비용이 증가하고, 이는 곧 애플리케이션의 응답성이 나빠짐을 의미한다.

그렇다면 다중 스레드를 사용하지 않고 단일 스레드 상에서 많은 작업을 동시에 수행할 수 없을까? 이런 마법 같은 일을 수행해내는 것이 바로 Reactor 동시 처리 모델이다. Reactor의 구성 요소와 동작 방식을 [그림 1-1]에서 알아보자.

---

06 http://docs.oracle.com/javase/7/docs/technotes/guides/io/enhancements.html

[그림 1-1] Reactor OMT 다이어그램



출처: http://blog.csdn.net/chexlong/article/details/22416775

굵은 글씨 부분이 Reactor의 주요 구성 요소로, 이 그림에서 Synchronous Event Demultiplexer는 운영체제에서 제공하는 디멀티플렉서다(select( ) 외에도 poll( ), WaitForMultipleObjects( ), epoll( ) 등을 사용할 수 있다).

Initiation Dispatcher에는 이벤트 핸들러의 구체 클래스(Concrete Event Handler)를 등록, 해제, 호출하기 위한 인터페이스가 있어 블록된 상태를 유지하다가 디멀티플렉서를 통해 처리해야 할 한 개 이상의 이벤트를 감지하면 각 이벤트에 대응되는 이벤트 핸들러를 차례로 호출한다. Initiation Dispatcher는 단일 스레드에서 동작하므로 이벤트 핸들러의 동시 실행을 걱정할 필요는 없다.

이벤트 핸들러는 Initiation Dispatcher에서 필요한 추상 인터페이스를 정의하고 있어서 특정 애플리케이션에 비의존적인 메커니즘을 제공할 수 있다. 이와 같은 Initiation Dispatcher와 이벤트 핸들러의 느슨한 관계는 개발자가 집중해야 할 비즈니스 로직 부분을 이벤트 핸들링 부분과 분리해 주어 이벤트 주도Event Driven 상황이 필요한 어떤 곳이라도 Reactor 동시 처리 모델을 쉽게 적용할 수 있다.

앞서 언급한 것처럼 Initiation Dispatcher는 단일 스레드에서 동작하므로 스레드 동기화 문제에서 벗어날 수 있지만, 본질적으로는 정교한 시분할 기법으로 각각의 작업을 순차적으로 처리하는 것에 지나지 않는다(순차적 처리가 매우 빨라서 사용자에게는 마치 동시에 작업이 처리되는 것처럼 보인다). 블록될 가능성이 있는 API를 호출하거나 CPU 연산 집약적인 처리로 이벤트 처리가 지연된다면 이는 곧 애플리케이션 전체에 심각한 병목 현상을 가져올 수 있음을 의미한다. 또한, 여러 개의 CPU를 가지고 이용할 수 있는 병렬 처리에 대한 이점을 얻을 수 없다는 단점이 있다.

실제로 이러한 단점은 Reactor 동시 처리 모델에 기반을 둔 Vert.x나 Node.js에서 그대로 나타나지만, Vert.x는 처음부터 이러한 단점을 보완하기 위한 구조를 갖추고 있어서 Node.js보다는 유리한 위치에 있다고 할 수 있다(Vert.x의 이런 독특한 구조를 Multireactor라 한다. Multireactor에 대한 내용은 뒤에서 자세히 설명하겠다). 그러나 Vert.x나 Node.js 관계없이 이벤트 기반 비동기 애플리케이션을 개발할 때 반드시 지켜야 할 규칙은 Initiation Dispatcher가 실행되는 스레드(Vert.x나 Node.js에서는 이벤트 루프 스레드Event Loop Thread라고 함)를 블록시키거나 시간이 오래 걸리는 작업을 실행하느라 이벤트 처리가 지연되지 않게끔 해야 한다는 것이다. 이 한 가지 규칙만 잘 지킨다면 스레드 동기화와 같은 어려운 문제에서 벗어나 단일 스레드만으로도 충분한 성능을 발휘하는 애플리케이션을 개발할 수 있다.[07]

이와 같은 Reactor 동시 처리 모델에 기반한 애플리케이션 개발은 요즘처럼 웹 관련 기술이 빠르게 발전하고 모바일과 IoT 기기가 폭 넓게 사용되고 있는 시점에 큰 의미를 지닌다. 다음 두 예를 살펴보자.

HTML5에 포함된 WebSocket은 웹 브라우저와 웹 서버 간의 양방향 통신을 지원하기 위

07 Vert.x가 실제로 단일 스레드로 동작하는 것은 아니다. 블록 API나 시간이 오래 걸리는 작업 처리하기 위해 별도의 스레드 풀(Thread Pool)을 제공한다.

한 차세대 표준으로, 웹 애플리케이션의 반응성을 극적으로 향상시켜 기존의 Ajax에 기반을 둔 대화형 웹 애플리케이션을 뛰어넘는 훨씬 효율적이고 역동적인 웹 애플리케이션 개발을 가능하게 한다.[08] Gmail, Google Docs, Facebook, 웹 채팅 서비스 등이 바로 그것으로, 이러한 종류의 웹 애플리케이션은 사용자에게 필요한 정보를 효과적이고 빠르게 전달해 준다는 공통점이 있다.

모바일 기기의 킬러 앱이 WhatsApp, KakaoTalk, Line 등과 같은 메신저 서비스라는 것은 의심할 여지가 없다. 모바일 기기의 메신저 서비스는 대부분 메시지를 전달하기 위해 푸시 알림PUSH Notification 방식을 사용한다. 푸시 알림 방식의 메시지 전달은 서버와 TCP 연결을 생성하고, 일정 주기로 Ping을 주고받으며 연결을 유지하다가 전달할 메시지가 발생하면 해당 TCP 연결을 통해 메시지를 전송받는 방식이다. 이 방식은 주기적으로 메시지를 확인하는 폴링Polling 방법보다 배터리와 네트워크 트래픽 소모량이 적다는 장점이 있다.

두 예에서 확인할 수 있는 사실은 HTML5의 WebSocket과 푸시 알림 방식 모두 높은 반응성과 모바일 기기의 효율성을 위해 TCP와 같은 연결 지향 기술을 사용한다는 점이다. 다중 스레드 동시 처리 모델은 연결되는 TCP 채널이 많아질수록 스레드 수도 많아지고 스레드 스택 메모리 유지와 문맥 전환에 필요한 비용도 함께 커져서 시스템 유지가 어려워지는 결과를 가져온다. 그러나 Reactor 동시 처리 모델은 연결되는 TCP 채널의 개수와 관계없이 항상 일정한 스레드 개수가 유지되어 불필요한 오버헤드가 없다.

또한, 일반적인 Reactor 동시 처리 모델은 단일 스레드에서 동작하여 병렬 처리의 이점을 얻을 수 없다는 단점이 있지만, Vert.x에서는 이러한 단점을 극복했다. Vert.x에서는 이것을 Multireactor이라 한다. Multireactor에 대해 좀 더 자세히 알아보자.

---

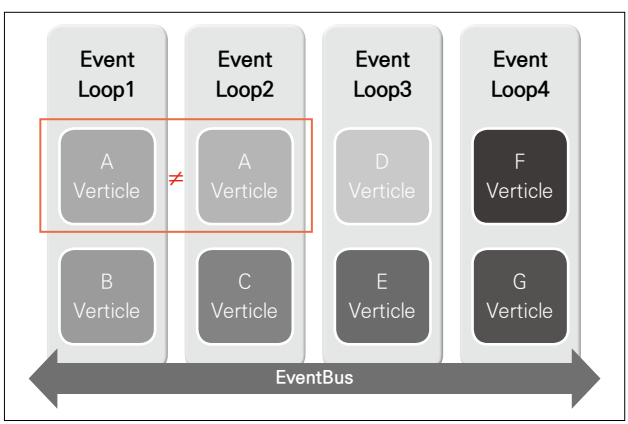
08 2005년 Ajax의 등장으로 전통적인 페이지 기반 웹 애플리케이션 개발의 한계를 벗어나 비동기 통신을 이용한 대화형 웹 애플리케이션 개발이 대중화되었다.

Multireactor

Multireactor는 Node.js와는 차별화되는 Vert.x만의 특징으로, 여러 개의 CPU를 이용할 수 있는 병렬 처리의 이점을 활용할 수 있게 한다. 이것은 시스템의 CPU 개수만큼 이벤트 루프 스레드를 생성해서 이벤트를 처리하는데, 각각의 이벤트 루프 스레드는 서로 다른 클래스로더ClassLoader로 만들어진 완전히 독립적인 코드를 실행하므로 스레드 동기화 문제는 걱정하지 않아도 된다. 또한, 이렇게 특정 이벤트 루프 스레드에서 한 번 실행된 코드는 절대 다른 이벤트 루프 스레드에서 실행되지 않음을 보장한다.

[그림 1-2]는 4개의 CPU를 사용할 수 있는 시스템에서 Vert.x를 실행했을 때의 모습이다.

[그림 1-2] Vert.x의 Multireactor



4개의 이벤트 루프 스레드가 생성되고 각 이벤트 루프 스레드는 서로독립적인 Verticle[09]을 실행한다. 각 Verticle은 서로 다른 클래스로더로 만들어지므로

09 Vert.x의 이벤트 루프 스레드에서 실행되는 코드 단위를 Verticle이라고 한다.

A라는 Verticle이 이벤트 루프 스레드 1과 이벤트 루프 스레드 2에서 동시에 실행되고 있어도 서로 완전히 독립된 상태를 유지한다. 즉, Verticle 내에 static으로 선언한 변수를 포함한 그 어떤 값도 공유되지 않는다. 그리고 D~G Verticle은 이벤트 루프 스레드 3과 4에 할당되어 있는데, 이 Verticle은 최초 할당된 이벤트 루프 스레드에서만 실행됨을 보장하며 그 외 이벤트 루프 스레드에서는 절대 실행되지 않는다.

1.2.2 Vert.x의 주요 특징

일반적으로 Reactor 동시 처리 모델에 기반을 둔 Vert.x, Node.js, Meteror, EventMachine 같은 구현체들은[10] 모두 비슷한 특징을 공유하므로 서로를 대체할 수 있다. 그러나 Vert.x는 Multireactor라는 고유의 독특한 구조 덕에 다른 구현체와 비교해 좀 더 나은 성능을 보여줄 수 있다.[11] 그럼 이번에는 다른 Reactor 동시 처리 모델 구현체와 차별화되는 Vert.x의 주요 특징을 알아보자.

단순함

Vert.x에서 제공하는 Core API는 단순하고 일관적이며 사용하기 쉽다. 또한, 앞서 설명한 것처럼 스레드 간 동기화 문제를 생각하지 않아도 되므로 개발자가 좀 더 생산적인 일에 집중할 수 있다.

다양한 언어 지원

Vert.x는 Java뿐만 아니라 다양한 언어를 지원한다. 현재 Vert.x에 지원하는 언어는 Java 외에도 JavaScript, Ruby, Python, Groovy, Clojure, Scala가 있으며 해당 언어의 샘플과 매뉴얼은 Vert.x 공식 홈페이지[12]에서 제공한다. 물론

10 http://en.wikipedia.org/wiki/Reactor_pattern#Implementations

11 그렇다고 무조건 Vert.x만을 고집하는 것은 바람직하지 않다. Reactor 동시 처리 모델에 기반을 둔 각 구현체의 장단점을 파악하고, 개발하려는 서비스의 성격에 맞는 것을 선택하는 것이 중요하다.

12 http://vertx.io/docs.html

2개 이상의 언어를 동시에 사용하여 애플리케이션을 개발할 수도 있다.

JVM

Vert.x는 JVM 위에서 동작한다. 이것은 오랜 시간 동안 연구되고 발전한 JVM의 안정성, 성능 등의 이점을 Vert.x에서도 그대로 안고 갈 수 있다는 것을 의미한다. 또한, 방대한 Java 레퍼런스 및 서드 파티 라이브러리Third Party Library를 Vert.x에서 그대로 이용할 수 있다는 것은 분명 큰 장점이다. 그러나 기존 Java 라이브러리를 가져와서 사용할 때 한 가지 주의해야 할 점이 있는데, 표준 JDBC API와 같이 스레드를 블록시킬 수 있는 API를 이벤트 루프 스레드 내에서 사용하면 안 된다는 것이다. 나중에 설명하겠지만, 이러한 API는 Worker Verticle이라는 이벤트 루프 스레드와는 별도로 관리되는 Worker 스레드 풀을 통해 실행해야 한다.

범용성

Vert.x는 웹 애플리케이션은 물론 P2P, 로드 밸런서, 프락시 서버 등 이벤트 주도가 필요한 어떠한 상황에도 적용할 수 있는 범용성이 있다. 심지어 임베디드 방식을 사용한다면 기존 Java 애플리케이션에서도 Vert.x의 기능들을 활용할 수 있다.[13]

확장성

Vert.x에서는 최소한의 노력으로 수평 확장이 가능한 애플리케이션을 구성할 수 있다. 네트워크상의 서로 다른 JVM에서 동작하는 Vert.x를 클러스터로 구성할 수 있으며, 동일한 클러스터에 묶인 Vert.x끼리 이벤트 버스Event Bus를 통해 String, JSON, Byte Buffer 형태의 메시지를 주고받을 수 있다. 심지어 Vert.x에서 제공하는 SockJS Event Bus Bridge를 사용하면 클라이언트 사이드의 웹 브라우저까지도 대규모 클러스터에 참여시킬 수 있다.

---

13 임베디드 방식을 사용하려면 Vert.x에서 제공하는 *.jar 형태의 라이브러리를 기존 애플리케이션에 추가하면 된다. 그러나 이 책에서는 임베디드 방식의 Vert.x 사용은 다루지 않으므로 자세한 정보는 Vert.x 홈페이지에서 제공하는 매뉴얼을 참고하기 바란다.

1.3 Vert.x 설치 및 개발 환경 설정

지금까지 Vert.x의 동시 처리 모델의 장점과 Vert.x의 특징을 알아보았다. 이어서 Vert.x를 실제 프로젝트에 사용하기 위한 이클립스와 메이븐 기반의 개발 환경 설정 방법을 알아보자.

1.3.1 JDK 설치

Vert.x는 JDK가 필요하므로 먼저 JDK를 설치해야 한다. Vert.x는 JDK7 이상 버전에서만 사용할 수 있으므로 JDK6 이하 버전을 사용 중이라면 JDK7을 새로 설치해야 한다.

1) 사용하는 운영체제에 맞게 JDK7 이상 버전을 내려받는다.

2) 환경 변수에 JAVA_HOME을 추가하고 JDK를 설치한 경로를 입력한다.

3) $JAVA_HOME\bin을 $PATH에 추가한다.

4) 콘솔에서 `java - version` 명령을 입력하고 결과가 정상적으로 출력되는지 확인한다.

다음은 Windows 8 PowerShell에서 `java - version` 명령을 실행한 결과다.

```
PS C:\Users\yeon> java -version
java version "1.8.0_05"
Java(TM) SE Runtime Environment (build 1.8.0_05-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.5-b02, mixed mode)
```

1.3.2 Vert.x 설치

JDK 설치를 마쳤으면 이제 Vert.x 배포본을 설치해 보자. Vert.x의 최신 배포본은 Vert.x 홈페이지[14]에서 받을 수 있다.

14 http://vertx.io/downloads.html

1) Vert.x 최신 배포본을 내려받는다(현재 최신 버전은 2.1.5다).

2) 적당한 위치에 내려받은 파일의 압축을 푼다.

3) 환경 변수에 VERTX_HOME을 추가하고, Vert.x의 압축을 푼 경로를 입력한다.

4) $VERTX_HOME\bin을 $PATH에 추가한다.

5) 콘솔에 `vertx version` 명령을 입력하고 결과가 정상적으로 출력되는지 확인한다.

다음은 Windows 8 PowerShell에서 `vertx version` 명령을 실행한 결과다.

```
PS C:\Users\yeon> vertx version
2.1.1 (built 2014-06-18 14:11:03)
```

1.3.3 환경 설정

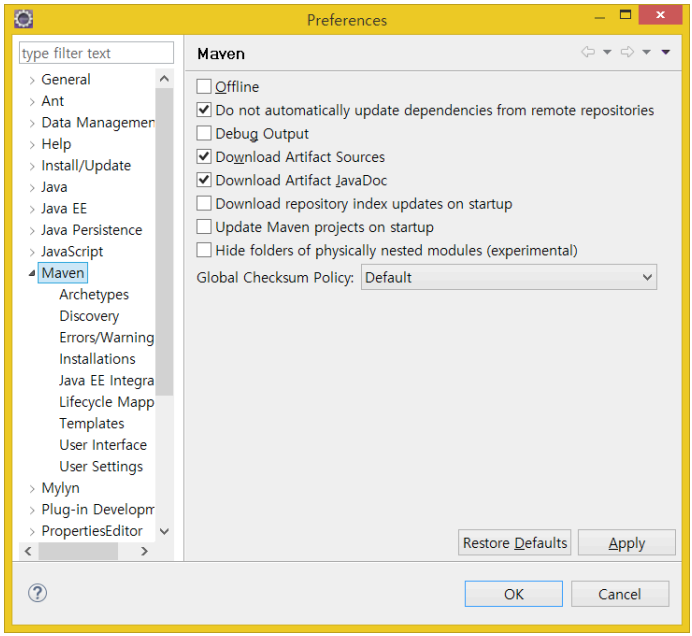
JDK와 Vert.x 설치가 모두 끝났다. 이제 이클립스에서 메이븐을 사용해 Vert.x 애플리케이션을 개발하기 위한 설정을 하면 된다.

먼저 이클립스 최신 배포본을 http://www.eclipse.org/downloads에서 내려받는다. 메이븐을 사용하므로 가능하면 Maven Integration for Eclipse Plug-In이 포함된 Java EE Developers나 Java Developers 버전을 선택한다.[15]

이클립스에서 'Preferences → Maven'을 선택하여 [그림 1-3]과 같은 화면이 보이면 메이븐을 사용할 준비가 모두 끝난 것이다(체크된 항목은 다를 수 있다).

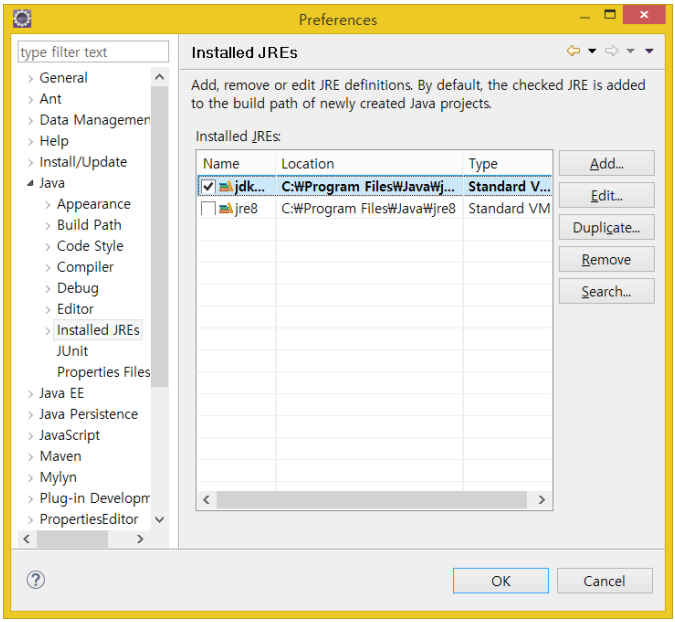
15 Maven Integration for Eclipse Plug-In이 포함되어 있지 않다면 Eclipse Marketplace에서 m2eclipse를 검색해서 사용 중인 이클립스에 맞는 최신 버전의 Plug-In을 설치한다.

[그림 1-3] Maven Integration for Eclipse Plug-In 확인



메이븐이 준비되었다면 이클립스에 JDK 설치 경로를 설정해 보자. 'Preferences → Java → Installed JREs'를 선택하여 확인하면 기본적으로 시스템에 설치된 JRE가 설정되어 있다. Vert.x 애플리케이션을 개발하려면 JRE가 아닌 JDK가 필요하다. [그림 1-4]의 화면에서 우측의 Add 버튼을 클릭하고 JDK 설치 경로를 추가한 후 해당 JDK를 기본값으로 사용하도록 체크한다.

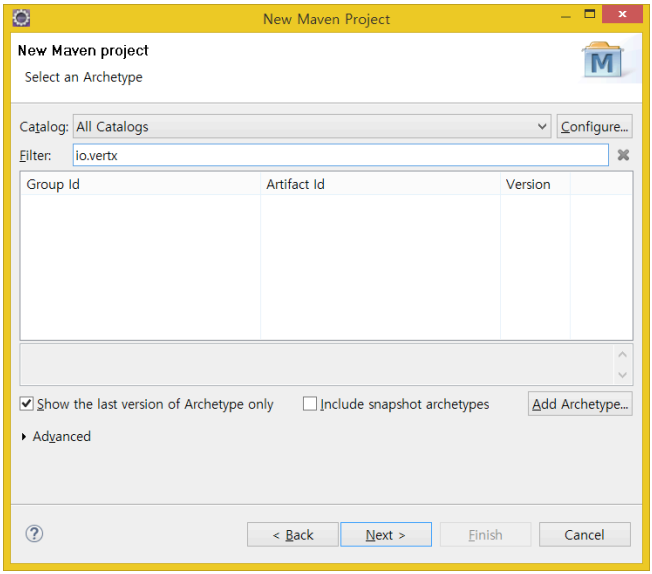
[그림 1-4] Installed JREs 설정



다음으로 메이븐의 Vert.x Archetype을 사용해 프로젝트를 생성해 보자. 'File → New → Others'를 선택한 후, 'Maven → Maven Project'를 선택한다. Select project name and location 부분에서 바로 Next를 선택하면 [그림 1-5]와 같이 Archetype을 지정하는 화면을 볼 수 있다. Filter에 io.vertx를 입력한다. 검색 결과가 아무것도 없다면 Vert.x Archetype을 설치해야 한다.[16]

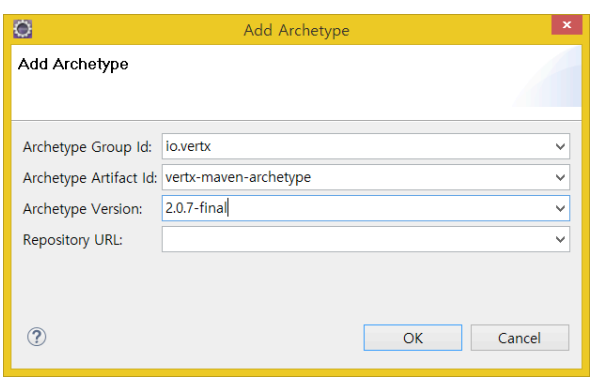
16 Vert.x Archetype이 이미 추가되어 있다면 다음 내용은 건너뛰어도 된다

[그림 1-5] 메이븐 Archetype 입력화면



http://search.maven.org/에서 vertx-maven-archetype를 검색한다. 가장 최신 버전의 GroupId, ArtifactId, Version을 확인한 후 [그림 1-5] 우측 하단의 Add Archetype 버튼을 클릭해 [그림 1-6]처럼 입력한다(현재 최신 버전은 2.0.11-final이다).

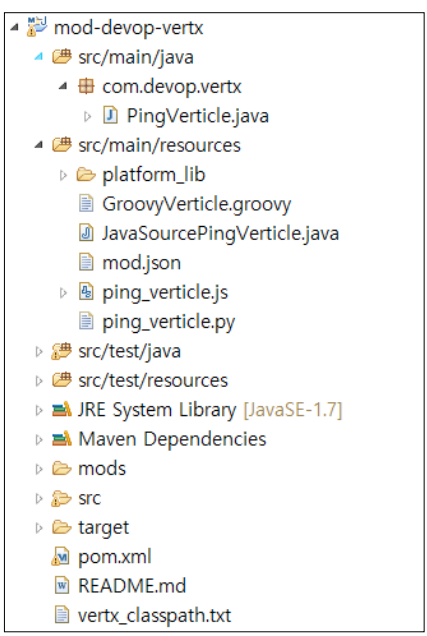
[그림 1-6] Vert.x Archetype 추가



완료되면 [그림 1-5]처럼 비어있는 검색 결과 화면이 아닌 Vert.x Archetype이 추가된 화면을 볼 수 있다. 해당 Archetype을 선택하고 Next를 클릭한다. 적절한 GroupId와 ArtifactId를 입력한 후 Finish를 클릭하면 프로젝트 생성이 완료된다(예시에서 프로젝트의 GroupId와 ArtifactId는 'com.devop.vertx'와 'mod-devop-vertx'를 입력했다).

Vert.x Archteype을 통해 자동 생성된 Vert.x 애플리케이션 프로젝트의 구조는 [그림 1-7]과 같다. 이 애플리케이션은 이벤트 버스로 메시지를 수신하면 'pong!'이라는 문자를 출력하는 PingVerticle을 다양한 언어(Java[17], Groovy JavaScript, Python)로 구현한 아주 간단한 프로그램이다.

[그림 1-7] Vert.x Archetype 샘플 프로젝트 구조



17 컴파일된 *.class와 원본 소스인 *.java 모두 Vert.x에서 사용할 수 있다.

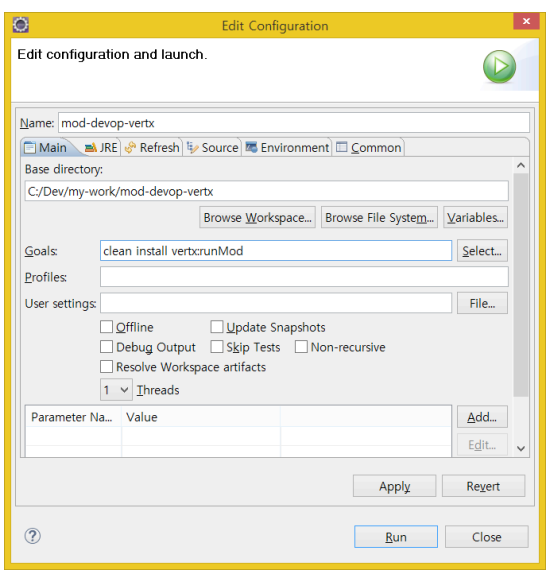
프로젝트를 구성하는 각 항목은 다음과 같다.

[표 1-1] Vert.x Archetype 샘플 프로젝트 구성

디렉터리		설명
src/main	java	해당 디렉터리를 포함한 모든 하위 디렉터리 내의 *.java 파일들은 컴파일된 후 *.class 형태로 Vert.x에서 사용된다. 즉, *.class 형태의 Verticle을 사용할 계획이라면 *.java 소스 코드를 이 디렉터리 내부에 두어야 한다.
	resources	Vert.x에서 지원하는 다양한 언어로 구현된 Verticle들을 모아둔다. *.java 형태의 Verticle도 이 디렉터리 내부에 둘 수 있는데, 이런 경우 *.java는 컴파일되지 않는다. mod.json이라는 Vert.x 모듈 설정 파일도 여기에 둔다.
src/test	java	*.class Verticle 테스트 케이스를 여기에 둔다.
	resources	Vert.x에서 지원하는 다양한 언어로 구현된 Verticle 테스트 케이스를 여기에 둔다.

이제 Vert.x 애플리케이션을 빌드하고 실행해 보자. 이클립스에서 해당 프로젝트를 선택하고 '오른쪽 버튼 클릭 → Run As → Maven Build'(단축키는 Alt+Shift+X, M이다)를 선택하면 [그림 1-8]과 같은 화면이 나온다.

[그림 1-8] Vert.x 애플리케이션 실행



Main 탭의 Goals에 'clean install vertx:runMod'를 입력한 후[18], 두 번째 JRE 탭에서 Runtime JRE가 JDK 7 이상 버전으로 지정되어 있는지 확인한다. JDK 6 이하 버전이라면 JDK 7 이상으로 변경해야 한다. 마지막으로 Common 탭에서 Encoding을 UTF-8로 설정한 후 실행한다. 필요한 모듈을 내려받고 src/test의 테스트 케이스가 성공적으로 실행되면 최종적으로 Vert.x 애플리케이션이 실행되는 것을 확인할 수 있다('PingVerticle started'라는 문자가 출력된다).

NOTE_ 오류 메시지 처리 방법

1. Vert.x 애플리케이션 실행이 실패하고, 'Fatal error compiling: invalid target release: 1.7'이라는 오류 메시지가 나온다면 JDK 7 이상 버전을 사용하지 않아서 발생한 오류다. 이때는 프로젝트에서 '오른쪽 버튼 클릭 → Run As → Run Configurations'를 선택한 다음 [그림 1-8] 화면의 JRE 탭에서 JDK 7 이상 버전을 사용하도록 설정하면 정상적으로 실행된다.
2. Vert.x 애플리케이션 실행이 실패하고, 'No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK?' 라는 오류 메시지가 나온다면 [그림 1-4]와 [그림 1-8]의 JRE 탭에서 JRE가 아닌 JDK로 수정해서 Vert.x 애플리케이션을 실행한다
3. Maven pom.xml 파일에서 'Plugin execution not covered by lifecycle configuration'이라는 오류가 발생할 수 있다. 해당 오류를 무시해도 Vert.x 애플리케이션을 빌드하고 실행하는 데에는 아무 문제 없다. 하지만 오류가 신경 쓰인다면 https://www.eclipse.org/m2e/documentation/m2e-execution-not-covered.html 내용을 참고하여 pom.xml 파일의 〈build〉〈/build〉 부분에 다음 내용을 추가한다.

```
<pluginManagement>
  <plugins>
     <plugin>
        <groupId>org.eclipse.m2e</groupId>
        <artifactId>lifecycle-mapping</artifactId>
         <version>1.0.0</version>
```

18 https://github.com/vert-x/vertx-maven에서 더 많은 정보를 확인할 수 있다.

```
            <configuration>
                <lifecycleMappingMetadata>
                <pluginExecutions>
                    <pluginExecution>
                        <pluginExecutionFilter>
                            <groupId>org.apache.maven.plugins</groupId>
                            <artifactId>maven-dependency-plugin</artifactId>
                            <versionRange>[1.0.0,)</versionRange>
                            <goals>
                                <goal>copy-dependencies</goal>
                            </goals>
                        </pluginExecutionFilter>
                            <action>
                                <ignore />
                            </action>
                        </pluginExecution>
                    </pluginExecutions>
                </lifecycleMappingMetadata>
            </configuration>
        </plugin>
    </plugins>
</pluginManagement>
```

1.4 간단한 Vert.x 애플리케이션

Vert.x가 설치된 것을 확인하기 위해 'hello, world'를 출력하는 아주 간단한 Vert.x 애플리케이션을 만들어 보면서 Vert.x의 주요 용어를 정리해 보자.

HelloVerticle.java 파일을 만들어서 [코드 1-1]의 내용을 에디터로 작성한다. 콘솔을 열고 '`vertx run HelloVerticle.java`'를 입력해 HelloVerticle을 실행하면 콘솔에 'hello, world'라는 문자가 출력된다.

NOTE_

앞서 살펴본 이클립스와 메이븐 기반의 개발 환경은 기본적으로 Vert.x 모듈을 만드는 데 최적화되어 있다(Vert.x 모듈에 대해서는 4장에서 살펴본다). 물론 패키지 자동 import나 인텔리센스(IntelliSense) 같은 IDE 기능을 활용하면 좀 더 편리하게 HelloVerticle.java 같은 간단한 Verticle을 작성할 수 있다. 4장 이후부터는 Vert.x 모듈을 개발하는 데 이클립스와 메이븐 개발 환경을 적극적으로 활용한다.

[코드 1-1] Vert.x hello world 프로그램(HelloVerticle.java)

```
import org.vertx.java.core.logging.Logger;
import org.vertx.java.platform.Verticle;

public class HelloVerticle extends Verticle {
      private Logger logger;
      @Override
      public void start() {
            logger = container.logger();
            logger.info("hello, world");
      }
}
```

무슨 일이 벌어진 것일까? 하나씩 살펴보자. 먼저 Verticle이 의미하는 것이 무엇이지 확인할 필요가 있다. Verticle은 Vert.x의 가장 기본이 되는 구성 요소로, 실행과 배포Deploy가 가능한 애플리케이션 코드라 할 수 있다. Vert.x 애플리케이션은 HelloVerticle처럼 1개의 Verticle로 구성될 수 있고, 2개 이상의 Verticle로 구성될 수도 있다. 각각의 Verticle은 서로 다른 클래스로더로 만들어져 이벤트 루프 스레드에 할당되므로 Verticle 내부의 그 어떤 상태 변수도 공유되지 않으며, 최초 할당된 이벤트 루프 스레드에서만 실행된다는 특징이 있다. 이 때문에 Verticle을 작성할 때 스레드 동기화 문제를 걱정하지 않아도 된다.

org.vertx.java.platform.Verticle의 코드를 보자.

[코드 1-2] org.vertx.java.platform.Verticle

```
package org.vertx.java.platform;

import org.vertx.java.core.Future;
import org.vertx.java.core.Vertx;

public abstract class Verticle {
      protected Vertx vertx;
      protected Container container;

      public Container getContainer() {
            return container;
      }
      public void setContainer(Container container) {
            this.container = container;
      }
      public Vertx getVertx() {
            return vertx;
      }
      public void setVertx(Vertx vertx) {
            this.vertx = vertx;
      }
      public void start() {
      }
      public void start(Future<Void> startedResult) {
            start();
            startedResult.setResult(null);
      }
      public void stop() {
      }
}
```

org.vertx.java.platform.Verticle에서는 2개의 핵심 객체와 몇 개의 중요한 메서드를 제공하고 있으며, Verticle을 작성할 때 이 클래스를 상속받아서 필요한 메서드를 오버라이드Override하면 된다.

[표 1-2] Verticle에서 제공하는 주요 객체 및 메서드

구분		설명
객체	vertx	실행 중인 Vert.x 애플리케이션 Runtime 객체다.
	container	실행 중인 Vert.x 애플리케이션의 환경 정보와 설정 정보를 가지고 있다. Logging 객체도 여기에 포함되어 있다. 또한, 이 객체를 통해 Verticle이나 모듈을 Deploy/Undeploy할 수 있다.
Method	start( )	Verticle이 Deploy되었을 때 실행되는 메서드다.
	stop( )	Verticle이 Undeploy되었을 때 실행되는 메서드다.

Verticle을 실행하려면 콘솔을 열고 '`vertx run YourVerticleFilePath`' 명령을 입력하면 되는데, 명령이 실행되면 새로운 Vert.x Instance가 만들어지고 Instance 내부에 Verticle이 배포된다.

Vert.x Instance는 JVM 위에서 실행되는 프로세스로, 하나의 Vert.x Instance에는 1개 이상의 Verticle이 포함될 수 있다. 동일한 Vert.x Instance 범위 안에 있는 Verticle들은 별도의 설정 없이도 이벤트 버스를 통해 서로 메시지를 주고받을 수 있다. 물론 네트워크의 서로 다른 JVM 에서 동작하는 Vert.x Instance들을 클러스터로 구성할 수 있으며, 이때 클러스터에 참여한 Vert.x Instance의 Verticle 사이에서는 이벤트 버스를 통해 메시지를 주고받을 수 있다. Verticle은 서로 독립적이며 어떠한 상태 변수도 공유하지 않으므로 이 이벤트 버스가 Verticle 간 데이터를 공유할 수 있는 유일한 수단이다.[19] [그림 1-2]를 다시 확인해 보면 4개의 이벤트 루프 스레드를 포함하는 Vert.x Instance에 총 8개의 Verticle이 배포된 형태임을 알 수 있다.

19 지금까지 따로 언급하지는 않았지만, 하나의 예외가 존재한다. Vert.x Instance 내부의 Verticle 사이에서는 SharedData를 통해 Immutable Type 데이터를 공유할 수 있다. SharedData 클러스터는 지원하지 않는다.

1.5 요약

지금까지 Vert.x의 주요 개념과 특징, 이클립스와 메이븐을 사용해 Vert.x 애플리케이션을 구현하는 방법까지 알아보았다. 시작이 반이라는 말처럼 개발자가 새로운 어떤 것을 배우는 데 가장 어렵고 시간을 들이는 부분이 바로 개발 환경을 설정하는 부분이다. 이제 힘든 부분은 지났으니 가벼운 마음으로 2장으로 넘어가 보자. 2장부터는 Vert.x로 채팅 서비스를 개발해 보고, Vert.x에서 제공하는 TCP, SockJS와 같은 기능을 자세히 알아본다.