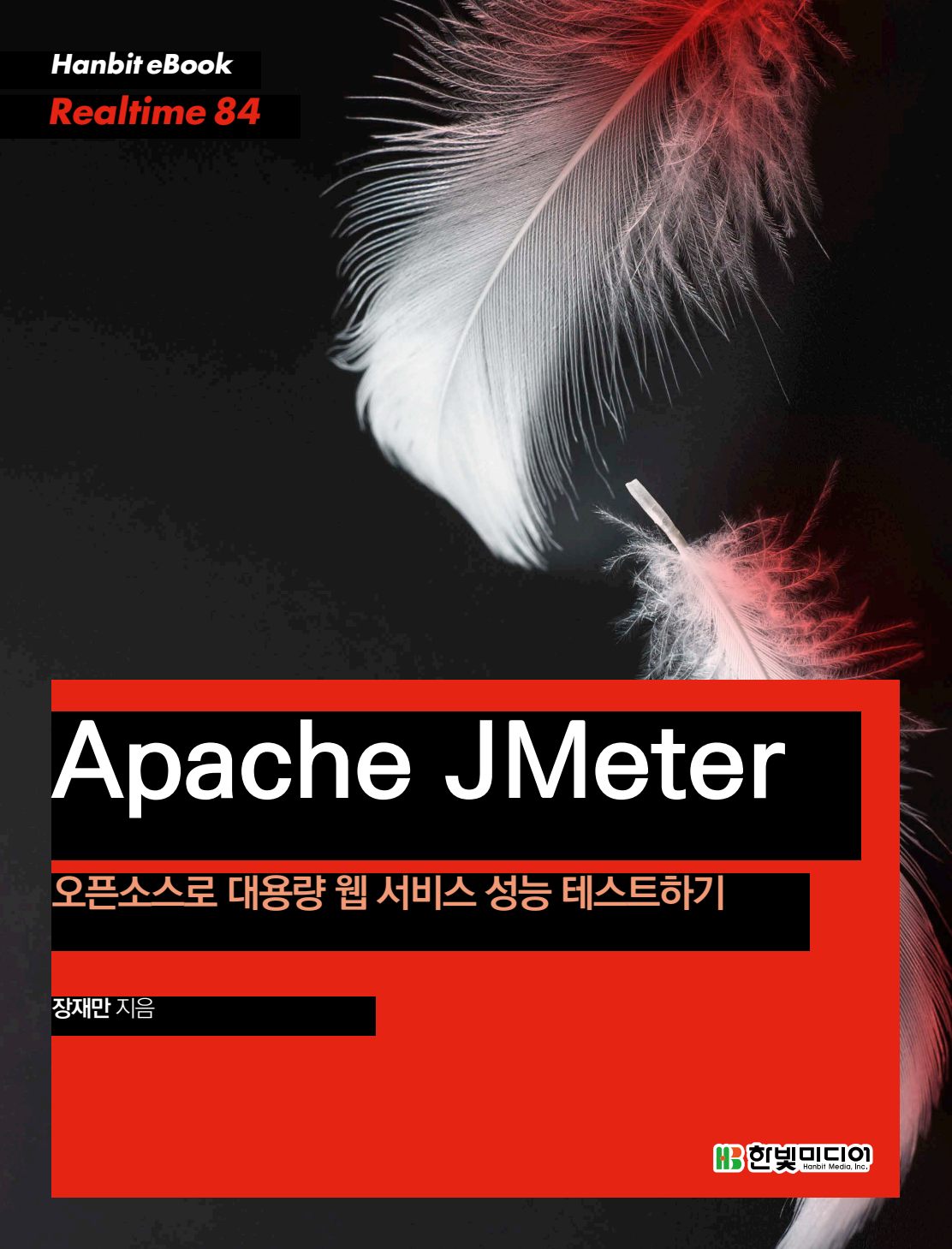

Hanbit eBook
Realtime 84
Apache JMeter
오픈소스로 대용량 웹 서비스 성능 테스트하기
장재만 지음
한빛미디어
Hanbit Media, Inc.

Apache JMeter

오픈소스로 대용량 웹 서비스 성능 테스트하기

Apache JMeter 오픈소스로 대용량 웹 서비스 성능 테스트하기

초판발행 2015년 1월 16일

지은이 장재만 / **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 978-89-6848-723-1 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_ 장재만

포털과 CDN 서비스 업체를 거쳐 현재는 (주)에임투지의 플랫폼연구소 팀장으로 근무하고 있으며 대용량 트래픽 제어 솔루션(NetFUNNEL)을 개발하고 있다. Apache JMeter를 이용하여 다수의 성능 테스트를 한 경험이 있으며 컨설팅 및 오픈소스에 많은 관심이 있다.

취미로 밴드에서 베이스를 연주하며 '안전제일'이라는 밴드에서 활동하며 2장의 음반을 발표했다.

- 홈페이지: www.jacojang.com

저자 서문

시간이 갈수록 웹 사이트의 QoSQuality of Service가 중요해지고 있습니다. 비슷한 많은 서비스가 서로 경쟁을 하다 보니 웹 서비스에 방문한 사용자들은 응답을 기다리며 시간을 낭비하는 것을 점점 더 싫어하게 되었습니다. 많은 연구기관에서는 이러한 사용자의 반응에 기반을 둔 QoS와 매출 간의 연관성에 관한 자료를 쏟아내고 있습니다. 하지만 서비스 운영자/개발자 입장에서는 점점 더 복잡해져 가는 IT 인프라와 다양해진 접속 환경을 모두 만족하게 하기란 쉽진 않습니다.

QoS를 향상시키는 가장 좋은 방법은 성능 테스트를 통해서 QoS에 악영향을 미치는 병목 지점을 하나씩 제거해 나가는 것입니다. 성능 테스트를 위해서 ab(Apache HTTP 서버 벤치마킹 툴)와 같은 단순한 커맨드라인 툴부터 Load Runner와 같은 복잡한 상용 툴까지 매우 많은 종류의 툴이 있지만, 이 책에서 제가 소개하려는 툴은 Apache JMeter입니다. 10년 이상 된 오픈소스 프로젝트로, 현재도 활발히 개발 중입니다.

처음 성능 테스트를 진행하면서 Apache JMeter를 선택한 이유는 비용 때문이었습니다. 상용 툴을 사용하고 싶지만, 비용 문제로 원하는 테스트를 하기 힘든 경우가 많아서 어쩔 수 없이 JMeter와 같은 오프소스 툴을 이용했습니다. 그러나 지금은 비용뿐 아니라 성능과 기능 면에서 충분히 활용할 수 있는 수준이라고 생각합니다.

이 책에는 JMeter를 이용하면서 제가 찾아낸 성능 테스트 노하우와 주의사항이 정리되어 있습니다. JMeter를 이용해서 성능 테스트를 하고자 하는 분들에게 많은 도움이 되었으면 합니다.

마지막으로, 책을 집필하는 동안 많은 도움을 주신 회사 동료들과 한빛미디어의 정지연 님, 무엇보다 큰 힘이 되어준 아내 조혜영에게 감사의 말을 전합니다.

대상 독자

초급 | 초중급 | **중급** | 중고급 | 고급

이 책은 Apache JMeter를 이용해서 웹 서비스의 성능 테스트를 하려는 시스템 운영자와 웹 개발자를 위한 책입니다. 자바와 웹 서비스 인프라, 웹 프로토콜에 대한 기초적인 지식을 가지고 있는 분이라면 쉽게 읽을 수 있습니다.

기존의 상용 성능 테스트 툴에서 오픈소스 성능 테스트 툴로 변경하고 싶은 테스트 담당자에게도 좋은 자료가 될 것입니다.

- 자료실: http://www.jacojang.com/jmeter/

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

들어가기

JMeter를 이용하여 실제 성능 테스트를 실행하기 전에 JMeter와 성능 테스트의 기본 개념, 그리고 JMeter를 사용하기 위한 주요 용어와 그 개념을 알아본다.

1 | JMeter와 성능 테스트

1.1 JMeter란

Apache JMeter는 웹 애플리케이션처럼 클라이언트-서버 구조로 된 소프트웨어의 성능 테스트를 위해서 만들어진 100% 순수 자바 프로그램이다. 스테파노 마조끼Stefano Mazzocchi가 개발했으며, 이는 현재 톰캣Tomcat으로 이름이 바뀐 Apache JServ의 테스트를 위한 코드에서 시작됐다. 이후 이 코드에 GUI와 기능을 추가하여 JMeter가 만들어졌다.

JMeter는 단위/성능/스트레스 테스트 등 많은 곳에서 활용할 수 있다. 프로토콜Protocol도 계속 추가되어 TCP, HTTP(S), FTP, JDBC, LDAP, SMTP, SOAP/XML RPC 등 현재 범용으로 사용되는 프로토콜 대부분을 지원한다.

JMeter는 통신 프로토콜 단계에서만 동작하고 웹 브라우저에서는 동작하지 않는다. 즉, 통신규약에 맞도록 클라이언트와 서버 간 메시지만 송수신할 뿐이고 클라이언트 자체에서 행해지는 연산 동작은 하지 않는다. 가장 대표적인 예가 ActiveX를 이용하여 암호화나 연산 작업을 하는 사이트다. 이러한 사이트는 ActiveX 로직을 모두 이해하고 자바를 이용하여 별도로 JMeter 내부 모듈을 구현하지 않는 이상 테스트가 불가능하다.

JMeter는 2001년에 1.0을 발표한 후 10여 년 동안 꾸준히 기능과 성능을 향상하여 2011년에는 Apache Software Foundation에서 Top Level Apache Project에 선정되기도 하였다. 또한, 자바 가상 머신JVM, Java Virtual Machine과 H/W의 성능이 향상되면서 초기에 문제가 되던 성능이나 기능 면에서 부족함이 많이 해소되어 현재는 웬만한 성능 테스트에서 핵심으로 활약하기에 부족함이 없는 상태에 도달한 것으로 보인다.

기능이 다양하고 성능이 좋음에도 JMeter는 오픈소스라는 이유로 성능 테스트 현장에서 많이 활용되지 못하고 있는 것이 현실이다. LoadRunner와 같은 외산 상용 솔루션이 가장 많이 사용되지만 라이선스 비용 문제로 현장에서 충분히 활용되지 못하며, 국내 몇몇 솔루션은 엔지니어 층이 얇거나 제대로 검증되지 않아서 신뢰성이 떨어지는 경우가 많다.

필자가 JMeter를 이용해서 많은 사이트의 성능 테스트를 진행하면서 도달한 결론은 불과 몇 년 전만 해도 다소 부족했지만, 현재는 JMeter로도 대부분 사이트에서 원하는 테스트를 수행할 수 있으며 성능이나 기능 면에서 부족함을 느끼기 어렵다는 것이다.

JMeter는 IT 업계에서 종사하는 엔지니어가 사용하기에 그리 복잡하지 않은 프로그램이다. 하지만 이 프로그램이 테스트 환경과 결합하면 많은 변수를 고려해야 하는 상황이 발생한다. 그 변수가 JMeter 자체 문제인지 네트워크 환경 문제인지 그리고 애플리케이션 서버 구성 또는 애플리케이션 자체 문제인지를 적절하게 구별하고 JMeter가 보여주는 결과 수치로 얼마나 의미 있는 값을 찾아내는가가 중요하다.

이 책에서는 JMeter를 사용하는 기초적인 방법부터 실제 성능 테스트 과정에서 일어날 수 있는 문제점에 대한 해결 방안과 JMeter로 대용량 테스트 환경을 구축하기 위한 최적의 방법까지 설명한다.

1.2 성능 테스트

JMeter를 시작하기 전에 성능 테스트에 대한 기본적인 지식을 알고 있는 것이 좋다. 이는 테스트를 계획하고 결과를 분석하는 데 중요한 역할을 한다.

이 책에서 말하는 '성능 테스트'란 서비스 및 서비스 시스템의 성능을 확인하기 위해서 실제 사용 환경과 비슷한 환경에서 테스트를 진행하는 것을 말한다. 이를 통

해서 응답시간Response Time과 처리량Throughput, 병목구간 등을 확인할 수 있고, 성능 테스트로 얻은 정보로 서비스나 서비스 시스템의 문제점을 확인하고 이를 개선Tuning하여 보완할 수 있다.

성능 테스트는 쓰임에 따라 다음과 같이 나뉜다.

- **Load 테스트** : 시스템의 성능을 벤치 마크하기 위한 테스트를 의미한다. 이 테스트는 부하Load를 순차적으로 증가시키면서 응답시간이 급격히 증가하거나 더는 처리량이 증가하지 않거나 시스템의 CPU와 Memory 등이 기준값 이상으로 증가하는 등 비정상 상태가 발생하는 임계점을 찾아내고 이를 바탕으로 성능 이슈에 대한 튜닝과 테스트를 반복한다.
- **Stress 테스트** : 임계값 이상의 요청이나 비정상적인 요청을 보내 비정상적인 상황의 처리 상태를 확인하고 시스템의 최고 성능 한계를 측정하기 위한 테스트를 의미한다.
- **Spike 테스트** : 이 테스트는 예를 들어 빌딩에 화재 경보가 발생했을 때 빌딩에 있는 직원들이 동시에 안전한 장소를 향해서 이동할 경우 시간이 얼마나 걸리며 어떤 문제가 발생하는지를 테스트하는 것과 같다. 즉, 갑자기 사용자가 몰렸을 때 요청이 정상적으로 처리되는지 그리고 그 업무 부하Workload가 줄어들 때 정상적으로 반응하는지를 확인하기 위한 테스트를 의미한다.
- **Stability 테스트 / Soak 테스트** : 긴 시간 동안 테스트를 진행해서 테스트 시간에 따른 시스템의 메모리 증가, 성능 정보의 변화 등을 확인하는 테스트를 의미한다. 짧게는 한두 시간부터 길게는 며칠 동안 진행하기도 한다.

성능 테스트 프로세스는 [표 1-1]과 같이 진행되며 단계별로 담당자가 바뀐다.

[표 1-1] 성능 테스트 프로세스

단계	프로세스	내용
1	요구사항 분석	• 테스트 목적과 범위를 정하는 단계로, 효율적인 테스트를 위해서는 목적을 정확히 설정해야 한다. • 구 시스템과 신규 시스템의 비교 테스트, 신규 시스템 오픈 전 사전 임계치 테스트, 장애 발생을 대비한 Failover-Failback 테스트 등 테스트 범위와 우선순위를 결정해야 한다. 모든 서비스를 테스트하면 좋겠지만, 보통 서비스 개발이나 유지 보수 때는 테스트를 위한 시간이 그다지 충분하지 않다. 그러므로 중요도와 테스트 목적에 맞는 우선순위와 그 범위를 정한다. • 범위가 정해지면 해당 시스템의 소프트웨어적인 구조와 하드웨어적인 구조를 분석한다.
2	테스트 계획	• 언제, 누가, 어떤 방법으로, 어디서 테스트할 것인지 정하는 단계다. • 테스트 수행에는 많은 내/외부 인력이 필요하며 많은 준비사항이 있으므로 테스트 계획이 필수다. • 테스트에 필요한 인력과 역할은 [표 1-2]를 참고한다.
3	테스트 환경 구축	• 테스트 단계 내에서 테스트 환경 구축을 언제 수행할지는 그리 중요하지 않을 수 있다. 자주 테스트를 수행하는 곳에서는 테스트 전용 서버팜(Serverfarm)이 이미 구성되어 있기 때문이다. 요즘은 클라우드 환경의 테스트 팜을 구성하는 경우도 있다. • 테스트 환경을 구축할 때 가장 중요한 것은 부하발생기와 테스트 대상 서버 사이의 네트워크가 최단 구간 안에 존재하게 하는 것이다. 중간에 많은 보안 장비와 스위치/라우터(Switch/Router) 등을 거치면 예상하지 못한 결과가 발생할 수 있다.
4	테스트 설계	• 테스트 절차 및 테스트 시나리오를 작성하고 테스트 케이스 작성 및 스크립트를 구현하며 테스트에 필요한 데이터 셋(Dataset)을 준비하는 단계다. • 테스트에 필요한 데이터 셋을 준비하는 과정은 상당히 중요하다. 테스트에 필요한 충분히 많은 데이터가 준비되지 않는다면 의도하지 않은 결과가 나올 가능성이 높기 때문이다. 예를 들어, 어떤 시스템에서 실제로는 DB의 I/O에 의해서 성능 저하가 발생한다고 가정했을 때, DB의 테스트 데이터가 충분히 입력되지 않고 접근 데이터가 고르지 않으면 실제 서비스 때보다 성능이 좋게 나올 가능성이 높다. 즉, 실제 환경과 비슷한 수준의 많은 데이터를 준비할수록 테스트 결과가 좀 더 실제 값과 비슷해진다.

단계	프로세스	내용
5	테스트 수행 및 결과 수집	작성된 스크립트로 실제 테스트를 수행하는 단계다. 테스트 수행은 크게 두 부분으로 나누어진다. • Pre-Test: Main-Test 전에 스크립트가 제대로 작성되었는지, 테스트 환경(서버/네트워크/보안 시스템/외부 연동 등)과 준비된 데이터 셋에 문제가 없는지 확인하기 위한 테스트다. Main-Test에는 많은 인력이 투입되므로 Pre-Test가 정상적으로 이루어지지 않으면 많은 인력이 불필요하게 대기하는 상황이 발생할 수 있으니 사전에 꼭 수행해야 한다. • Main-Test: 실제 테스트를 수행하는 단계로, 테스트 분석에 필요한 시스템 성능 자료를 수집한다. 통합 SMS 솔루션이 있으면 OS의 CPU, Memory, I/O 등의 정보는 쉽게 수집할 수 있다. 그렇지 않다면 별도의 시스템 정보 수집 스크립트를 이용해서 수집해야 한다. AP(Application Server)는 APM(Application Performance Management)과 같은 전용 모니터링 도구를 이용하면 많은 정보를 편리하게 수집할 수 있다.
6	테스트 분석	테스트 결과 자료와 시스템 성능 자료를 모아서 테스트 결과를 분석한다. 분석된 자료를 통해서 성능에 영향을 미치는 문제점을 찾는다.
7	문제점 수정 및 재테스트	테스트 분석에서 발견된 문제점을 개발팀(Development Team)이나 시스템 운영팀(System Engineering Team)에 전달하여 문제점을 수정하고 다시 한 번 테스트를 수행한다.
8	결과 리포트 작성	테스트 리포트는 테스트 목적에 따라 해당 목적을 가장 잘 표현할 수 있는 방식으로 작성하는 것이 좋다. 요약 리포트와 상세 리포트를 분리해서 상세 결과를 필요로 하는 부서와 요약 결과만을 필요로 하는 부서에 별도로 리포트를 제출하는 것도 좋은 방법이다.

[표 1-2] 테스트 인력 및 역할

테스트 인력	역할
Test Leader(PM)	전체적인 테스트 계획, 목적 수립, 시나리오 작성, 일정 관리와 인력 배치를 담당한다.
Test Scripter(Designer)	정의된 테스트 목적 및 범위에 따라 작성된 시나리오로 사이트(서비스)를 분석하고 이를 바탕으로 상세 케이스를 작성한다.
Test Operator	작성된 테스트 스크립트를 이용해서 실제 테스트를 수행한다.
Development Team	테스트에 필요한 데이터를 준비하고 애플리케이션을 모니터링하며 발견된 문제점과 개선점을 찾아낸다.
System Engineer(SE) Team	테스트 환경을 구축하고 시스템(OS, 네트워크, 스토리지 등)을 모니터링하여 발견된 문제점과 개선점을 찾아낸다.

테스트 인력	역할
외부 업체 지원 인력	시스템 운영팀(System Engineer Team)에서 모든 정보를 수집하고 분석하면 좋겠지만, 특정 솔루션이나 시스템은 외부 엔지니어의 도움을 받아야 하는 경우가 생긴다. 외부 엔지니어는 해당 솔루션(또는 시스템)의 전문가이므로 테스트 결과 분석에 많은 도움이 되지만 외부 인력이므로 일정 관리를 잘해야 한다.

1.3 대용량 성능 테스트

이 책에서 의미하는 '대용량 성능 테스트'란 매우 많은 가상 사용자Virtual User, Thread가 필요한 테스트나 매우 높은 웹 트랜잭션 처리량을 테스트하는 것을 의미한다. 즉, 대규모 장비가 필요한 테스트다. 대용량 성능 테스트를 진행하다 보면 결과에 많은 영향을 주는 요소들이 발생한다. 이것은 현재 수행한 테스트 결과가 믿을만한지, 어떤 요소가 한계점에 다다라서 결과가 왜곡되지 않았는지에 대한 고민에 빠지게 한다. 예를 들어, 여러 대의 부하발생기가 연결되면 부하발생기의 네트워크 구성에 따라 결과가 달라질 수 있으며, 애플리케이션 서버의 능력보다 네트워크 스위치의 성능이나 OS 설정값에 의해 결과가 매우 달라질 수 있다.

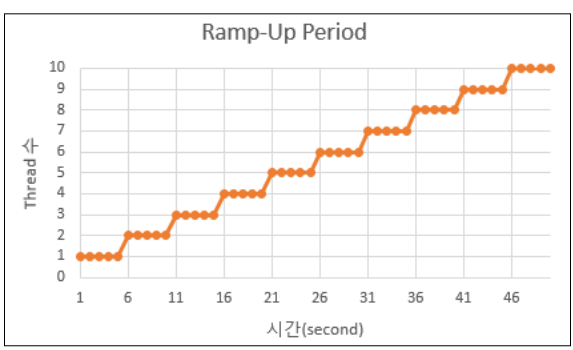
그러므로 좀 더 신뢰성 높은 결과값을 얻기 위해서는 결과에 영향을 미치는 요소들을 최대한 제거해야만 한다. 이 책에서는 기본적인 JMeter의 사용법과 함께 대용량 성능 테스트를 수행할 때 발생할 수 있는 문제점과 그에 대한 해결 방법을 실무에서 겪은 다양한 경험과 시행착오를 바탕으로 테스트 담당자들이 이런 시행착오를 겪지 않도록 하기 위한 팁과 노하우를 다룬다. 또한, 이를 바탕으로 테스트의 신뢰성을 높이고 테스트 결과값에서 의미 있는 내용을 찾는 방법을 설명한다.

1.4 주요 용어 및 개념

성능 테스트와 관련하여 자주 사용되는 용어와 그 개념을 간략하게 정리해 보자.

- **Active User** : 실제 서버에 연결된 상태로 요청을 처리 중인 사용자를 말한다.
- **InActive User** : 웹 브라우저에 결과 화면이 출력된 상태에서 화면의 내용을 읽거나 정보를 입력하고 있는 사용자다. 서버와의 세션Session정보를 가지고 있지만 직접 접속하여 요청을 주고받는 상태가 아닌 사용자를 의미한다.
- **Concurrent User(Active User + InActive User)** : 보통 '동시 접속 사용자 수'라고 표현한다. 일반적으로 사용자 수의 많고 적음을 표현하는 값으로, 성능 테스트에서 가상 사용자 수를 결정하는 기준이 된다. 서비스 유형과 시간에 따라 그 비율이 달라지긴 하지만, 일반적으로 Active User와 InActive User 비율이 1:10 정도다.
- **Virtual User** : 가상 사용자 수로, JMeter에서는 Thread 수로 표현하기도 한다.
- **Ramp-Up Period** : Thread(Virtual User) 생성에 걸리는 시간을 의미한다. Ramp-Up Period 동안 차례대로 Thread를 생성한다. [그림 1-1]은 Ramp-Up Period를 이해하기 쉽도록 작성한 그래프다.

[그림 1-1] Ramp-Up Period에 따른 시간별 Thread 수 변화

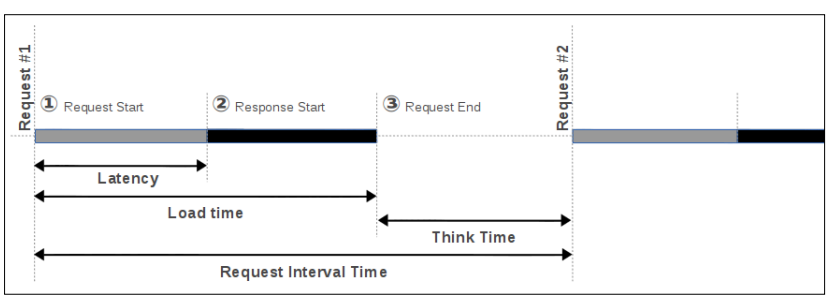


Thread 수 = 10(개) / Ramp-Up Period = 50(초)

이 그래프는 '10개의 Thread를 50초 동안 차례대로 생성하라'는 의미다. 즉, 5초(50초/10개)마다 Thread를 하나씩 생성하는 것과 같은 의미다.

- **Throughput** : 단위 시간당 대상 서버(웹서버, WAS, DB 등)에서 처리되는 요청의 수를 말한다. JMeter에서는 시간 단위를 보통 TPS(Transaction Per Second)로 표현한다.
- **Response Time/Load Time** : 응답시간 또는 처리시간이라고 표현한다. 요청을 보낸 후 응답이 완료되어 사용자 화면에 출력될 때까지의 시간을 나타낸다. 시스템의 성능을 평가하는 지표로 주로 사용한다.
- **Latency** : 요청을 보낸 후 데이터를 받기 시작할 때까지 시간이다.
- **Think Time** : 하나의 요청에 응답을 수신하고 다음 요청을 보낼 때까지 시간을 의미한다. 테스트에서 실제 사용자의 사용 패턴과 유사한 패턴을 구현하기 위해서는 이 Think Time을 적절히 적용해야 한다.
- **Request Interval Time** : 요청을 보낸 후 다음 요청을 보낼 때까지 시간을 의미한다.
- **Load Time vs Latency** : [그림 1-2]는 Load Time/Latency/Think Time/Request Interval Time의 관계를 이해하기 쉽도록 그림으로 나타낸 것이다.

[그림 1-2] 시간 관계도



[그림 1-2]를 보면 항상 Load Time ≥ Latency가 성립된다. 두 개를 왜 나눠놨을까? 이것은 Latency와 Load Time을 구분함으로써 성능을 분석할 때 요

긴하게 사용할 수 있다.

A와 B 사이트에 동일한 크기(10MB 정도)의 파일을 올려놓고 다운로드 테스트를 진행한다고 가정하자. A 사이트와 B 사이트의 결과를 비교해 보니 B 사이트의 Load Time이 2배 이상 컸다. 하지만 Latency는 거의 비슷했다. 이렇게 차이가 나는 이유는 무엇일까?

Load time에서 Latency를 빼면 데이터를 전송받는 데 걸리는 시간을 나타낸다. 즉, B 사이트가 A 사이트보다 데이터를 내려받는 속도가 느리다고 볼 수 있다. 따라서 B 사이트는 처리량을 늘리기 위해 웹 서버를 튜닝하기보다는 네트워크의 대역폭(Bandwidth)을 늘리는 것을 고려해야 한다.

성능 테스트 사전 작업

실제 대용량 웹 성능 테스트를 수행하는 데 필요한 사전 준비 작업을 알아본다. 웹 성능을 테스트하기 위한 Test Plan을 만들어 보고, 웹 성능 테스트를 하기 위해 꼭 필요한 JMeter의 Element 사용법과 특징을 간단한 예제를 통해서 자세히 알아본다.

2 | JMeter 설치와 환경 설정

JMeter는 아파치 프로젝트Apache Project에 속한 오픈소스로, 아파치 소프트웨어 재단 홈페이지[01]에서 내려받을 수 있다. 다운로드 페이지로 가면 Binaries와 Source 두 가지 버전을 확인할 수 있다. JMeter를 변경 없이 그대로 사용하려면 Binaries 버전을 내려받고, 소스 코드를 수정 또는 추가하거나 컴파일해서 사용하려면 Source 버전을 내려받는다. Source 버전을 이용해서 컴파일한 후 사용하는 방법은 나중에 알아보고 우선 Binaries 버전을 내려받자.

2.1 실행 환경

JMeter는 순수 자바 애플리케이션이므로 Java JDK(JRE)만 설치되어 있으면 구동하는 데 문제없다. 최신 버전의 JMeter는 Java JDK 6 이상[02]이 필요하다.

JMeter는 별도의 설치 과정 없이 압축 파일을 풀고 apache-jmeter-2.xx/bin 디렉터리에 있는 시작 명령어만 실행하면 바로 구동할 수 있다.

[표 2-1] 디렉터리 구조

디렉터리	설명
apache-jmeter-2.11\bin	JMeter를 실행하기 위한 실행 파일과 설정 파일이 있는 디렉터리다.
apache-jmeter-2.11\docs	API 관련 문서 디렉터리다.
apache-jmeter-2.11\extras	추가 유틸리티가 있는 디렉터리다.
apache-jmeter-2.11\lib	JMeter Components나 플러그인을 실행하는 데 필요한 유틸리티와 Dependency Jars가 있는 디렉터리다.
apache-jmeter-2.11\lib\ext	JMeter에서 사용하는 Components와 플러그인이 있는 디렉터리로, 기본으로 제공되는 Components 외에 추가로 설치된 Component나 플러그인도 이 디렉터리에 두면 JMeter가 구동될 때 자동으로 참조한다.

01 https://jmeter.apache.org/download_jmeter.cgi

02 http://www.oracle.com/technetwork/java/javase/downloads

디렉터리	설명
apache-jmeter-2.11\ licenses	non-ASF 소프트웨어의 라이선스 정보가 담겨 있는 디렉터리다.
apache-jmeter-2.11\ printable_docs	도움말 문서가 있는 디렉터리다.

[표 2-2] Linux/Unix 실행 명령어

명령어	설명
jmeter	GUI 모드로 실행하기 위한 명령어다(Default).
jmeter-server	Server 모드로 실행된다(Server 모드는 분산 테스트 설정에서 자세히 설명한다).
shutdown.sh	non-GUI 모드로 실행할 때 정상 종료(Gracefully)하게 하는 명령어다.
stoptest.sh	non-GUI 모드로 실행할 때 즉시 종료(Abruptly)하게 하는 명령어다.

[표 2-3] Windows 실행 명령어

명령어	설명
jmeter.bat jmeter.cmd	• GUI 모드로 실행하기 위한 명령어다(Default). • jmeter.bat으로 실행하면 cmd 창이 뜬 상태로 실행되고, jmeterw.cmd로 실행하면 cmd 창 없이 실행된다. • 중간에 출력되는 JMeter 메시지를 보려면 jmeter.bat으로 실행한다.
jmeter-n.cmd	• non-GUI 모드로 실행하기 위한 명령어다. • non-GUI 모드로 실행해야 할 때 추가 인자로 실행될 JMX 파일명을 입력한다. 예) c:\> jmeter-n.cmd test.jmx
jmeter-n-r.cmd	jmeter-n.cmd와 기능은 같지만, local에서 실행되는 것이 아니라 remote_hosts에 등록된 jmeter-server를 이용해서 실행된다.
jmeter-t.cmd	jmeterw.cmd와 같이 GUI 모드로 실행된다. 단, 입력 인자로 JMX 파일을 입력해야 한다.
jmeter-server.bat	Server 모드로 실행된다.
shutdown.cmd	non-GUI 모드로 실행했을 때 정상 종료하게 하는 명령어다.
stoptest.cmd	non-GUI 모드로 실행했을 때 즉시 종료하게 하는 명령어다.

2.2 JVM 옵션 설정

JVM 옵션을 변경해서 실행하려면 JVM_ARGS 변수값을 설정한다.

Windows

jmeter.bat 파일에 다음을 추가하고 실행한다.

```
set JVM_ARGS="-Xms1024m -Xmx1024m -Dpropname=propvalue"
```

Linux/Unix

항상 같은 설정값을 이용한다면 jmeter 파일에 변수값을 설정해도 되지만, 실행할 때마다 설정값을 조금씩 수정해야 하는 상황이라면 다음과 같이 실행 시 변수값을 설정하고 실행할 수도 있다.

```
JVM_ARGS="-Xms1024m -Xmx1024m" jmeter -t test.jmx
```

2.3 Overriding Property

Java System Property와 JMeter의 Property, Logging Property는 jmeter.properties 파일에 설정되어 있다. JMeter를 실행하면 해당 파일을 읽어 들여서 Property를 설정한다. 하지만 변경이 잦을 때는 파일을 수정하기보다는 커맨드라인에서 Property 값을 재정의하는 것이 편하다.

[표 2-4] Property 옵션

옵션	설명
-D〈Prop_name〉=〈value〉	Java System Property 값을 정의한다.
-J〈Prop_name〉=〈value〉	Local JMeter Property를 정의한다.
-G〈Prop_name〉=〈value〉	모든 remote_hosts에 전달될 Property를 정의한다.

옵션	설명
-G〈property_file〉	Property 내용이 저장된 파일을 remote_hosts에 전송한다.
-L〈category〉=〈priority〉	카테고리(Category)별로 로깅 수준(Level)을 결정한다.
-L〈priority〉	최상위 로깅 수준으로 설정할 수 있다.

[예제]

```
jmeter -Duser.dir=/home/mstover/jmeter_stuff -Jremote_hosts=127.0.0.1
-Ljmeter.engine=DEBUG
```

2.4 non-GUI Mode

non-GUI 모드는 커맨드라인 모드라고도 한다. Linux/Unix에서는 GUI를 실행할 수 없는 환경일 때가 종종 있다. 이럴 때 non-GUI 모드를 사용한다. 그러나 결과를 그래프나 수치로 바로 확인하면서 작업하면 테스트 도중에 발생할 수 있는 문제점이나 비정상 결과값을 즉시 확인할 수 있으므로 가능하면 GUI 모드를 사용하는 것이 좋다.

[표 2-5] non-GUI 옵션

옵션	설명
-n	non-GUI 모드를 실행하는 옵션이다.
-t 〈testplan name〉	테스트에 사용될 Test plan 파일명을 입력한다.
-l 〈logfile name〉	결과(Sample Result)가 저장될 로그 파일명을 입력한다.
-j 〈jmeter log name〉	JMeter 로그 정보가 저장될 파일명을 입력한다. (Default: jmeter.log)
-r	JMeter Property 중 remote_hosts에 설정된 jmeter-server를 실행한다.
-R 〈list of remote servers〉	remote_hosts에 설정된 jmeter-server가 아니라 직접 jmeter-server를 지정해서 테스트할 수 있다.
-H 〈proxy server host〉	프락시 서버(Proxy Server)를 이용해서 접속할 때 해당 프락시 서버의 호스트(host)를 설정한다.
-P 〈proxy server port〉	프락시 서버를 이용해서 접속할 때 해당 프락시 서버의 포트를 설정한다.

[예제]

```
jmeter -n -t test.jmx -l log.jtl -j my_jmeter.log -H my.proxy.server -P 8000
```

[실행 결과]

```
jacojang@jacojang-Desktop:~/apache-jmeter-2.11/bin$ ./jmeter -n -t test.jmx
Creating summariser <summary>
Created the tree successfully using test.jmx
Starting the test @ Sat Apr 05 17:50:42 KST 2014 (1396687842690)
Waiting for possible shutdown message on port 4445
summary +     72 in    17s =    4.3/s Avg:   653 Min:   489 Max:  1006 Err:     0
(0.00%) Active: 3 Started: 3 Finished: 0
summary +    102 in  24.3s =    4.2/s Avg:   660 Min:   603 Max:  1008 Err:     0
(0.00%) Active: 0 Started: 3 Finished: 3
summary =    174 in    41s =    4.3/s Avg:   657 Min:   489 Max:  1008 Err:     0
(0.00%)
Tidying up ...    @ Sat Apr 05 17:51:23 KST 2014 (1396687883952)
... end of run
```

3 | 간단한 Test Plan 작성하기

JMeter에서는 테스트 스크립트를 'Test Plan'이라고 표현한다. 이 장에서는 웹 서버를 테스트하기 위한 간단한 Test Plan을 만들어 본다. 설명한 대로 따라 하면 쉽게 만들 수 있다.

JMeter는 부하 발생을 목적으로 하는 프로그램이어서 외부 웹 서버에 접속해서 테스트하면 외부 서버에 부하가 발생하거나 내부 네트워크 트래픽에 과부하를 줄 수 있으므로 자신의 PC에 테스트 타깃 서버(Target Server)를 만들어 놓고 테스트를 진행하는 것이 좋다.

테스트에 필요한 URL을 테스트할 수 있는 WAR 파일은 다음 주소에서 내려받을 수 있다. 톰캣을 설치한 후 내려받은 파일을 배포하면deploy 이 책에 사용하는 모든 테스트 URL을 사용할 수 있다.

http://www.jacojang.com/jmeter/jmeter.war

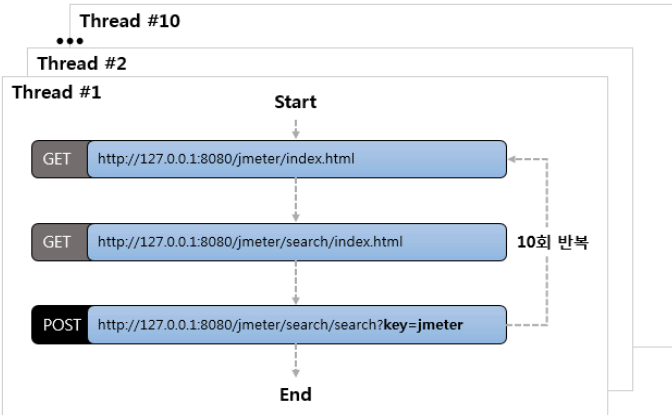
톰캣 설치 방법과 WAR 파일 배포 방법은 부록을 참고하기 바란다.

3.1 Test Plan 작성

[그림 3-1]과 같이 10명의 사용자가 다음 테스트 페이지를 10번 반복 요청한다고 가정한다.

- http://127.0.0.1:8080/jmeter/index.htm(Method: GET)
- http://127.0.0.1:8080/jmeter/search/index.html(Method: GET)
- http://127.0.0.1:8080/jmeter/search/search?key=jmeter(Method: POST)

[그림 3-1] 테스트 구성도



Test Plan은 다음 순서로 작성한다.

- **Thread Group 추가 및 설정** : 가상 사용자(Thread)의 숫자와 반복 횟수, 반복 시간을 설정한다.
- **Config Element**[01] **추가 및 설정** : 이번 장에서는 HTTP Request Defaults만을 사용해서 작업한다.
- **HTTP Request Sampler**[02] **추가 및 설정** : 테스트 페이지 목록에 해당하는 세 개의 Sampler를 추가한다.
- **Listener**[03] **추가 및 설정** : 테스트 결과를 보기 위해서는 Listener를 꼭 추가해야 한다. View Result Tree와 Summary Report를 추가한다.

모든 작업이 완료되면 [그림 3-2]와 같은 형태가 된다.

---

01 JMeter에서는 Test Plan 아래에 추가되는 노드를 Element라고 한다.
02 Sampler는 실제로 서버에 요청을 보내는 Element를 말한다.
03 Listener는 테스트 결과를 보기 위한 Element를 말한다.

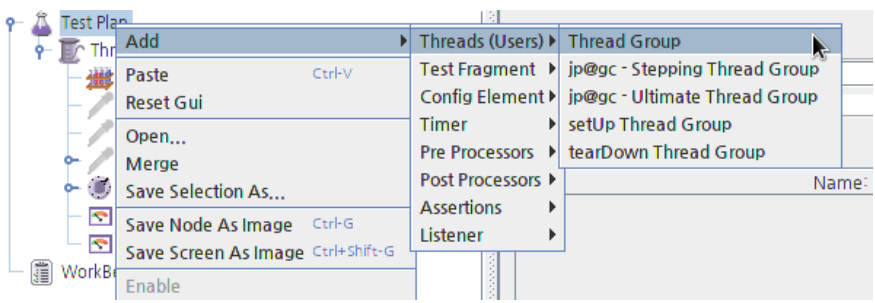
[그림 3-2] Test Plan 계층 구조

3.1.1 Thread Group 추가 및 설정

Thread Group 추가

Test Plan에서 'Add → Threads (Users) → Thread Group'을 선택하여 추가한다.

[그림 3-3] Thread Group 추가

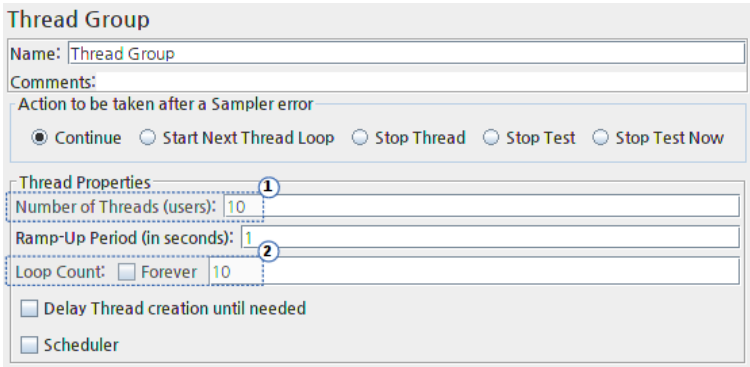


Thread Group 설정

① Number of Threads(users): 10은 10개의 Thread를 생성하라는 의미다.

② Loop Count: 10은 10명이 10번씩 Test Plan을 반복하라는 의미이므로 '10(명) × 10(반복) = 100회'를 수행한다.

[그림 3-4] Thread Group 설정

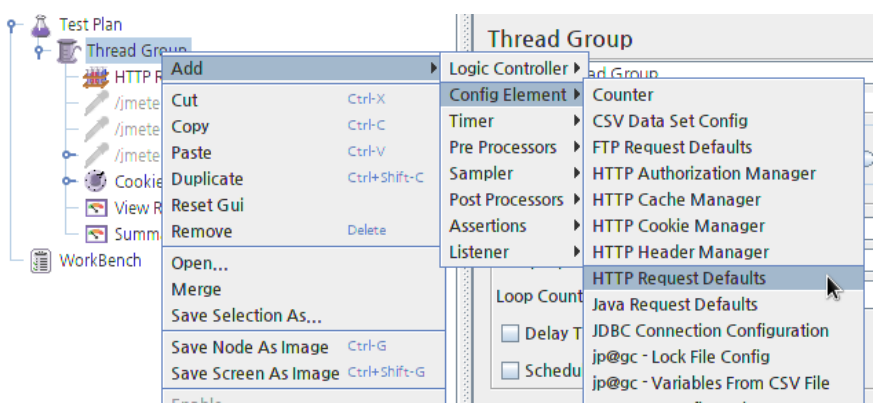


3.1.2 Config Element(HTTP Request Defaults) 추가 및 설정

Config Element 추가

Thread Group에서 'Add → Config Element → HTTP Request Defaults'를 선택하여 추가한다.

[그림 3-5] Config Element 추가



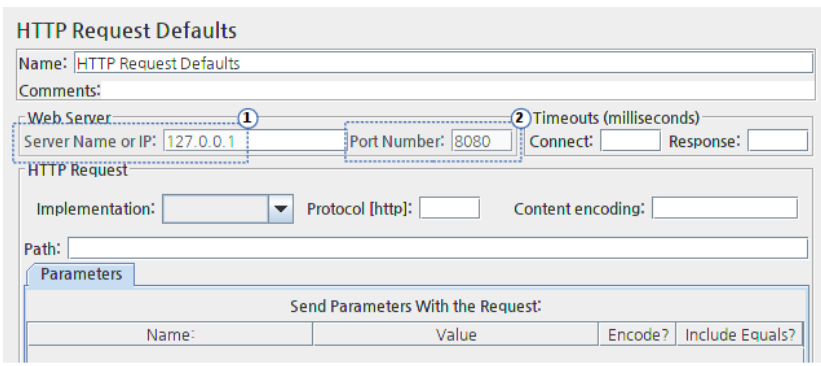
Config Element 설정

HTTP Request Sampler에 설정되는 정보 중에 중복되는 부분을 HTTP Request Defaults에 설정하면 다음에 나올 HTTP Request Sampler의 설정을 간소화할 수 있고, 변경 사항이 생겼을 때 작업량이나 오류 발생이 줄어드는 장점이 있다.

① Server Name of IP: 127.0.0.1

② Port Number: 8080

[그림 3-6] HTTP Request Defaults 설정



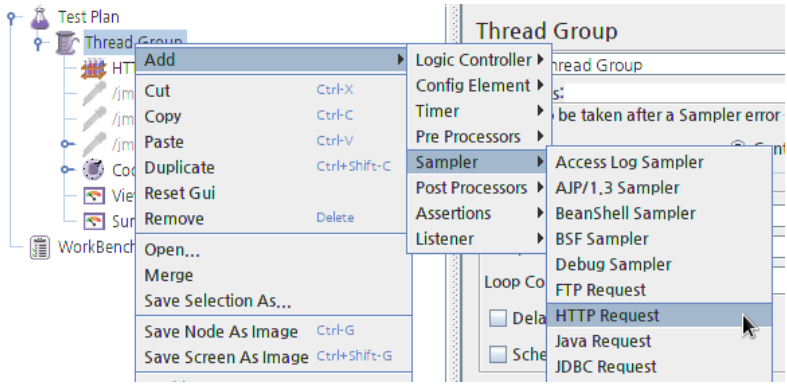
3.1.3 HTTP Request Sampler 추가 및 설정

웹 서버에 요청을 보낼 경우에는 HTTP Request라는 Sampler를 이용한다.

HTTP Request Sampler 추가

Thread Group에서 'Add → Sampler → HTTP Request'를 선택하여 추가한다. 테스트 페이지 목록에 3개의 URL이 있으므로 3개의 Sampler가 필요하다. 따라서 이 작업을 3번 반복한다.

[그림 3-7] HTTP Request Sampler 추가



HTTP Request Sampler 설정

① Name: Name은 알맞은 값으로 수정한다. 다음에 설명할 Listener에서 결과를 정리할 때 이 Name을 기준으로 보여주므로 구분되는 이름으로 설정해야 한다.

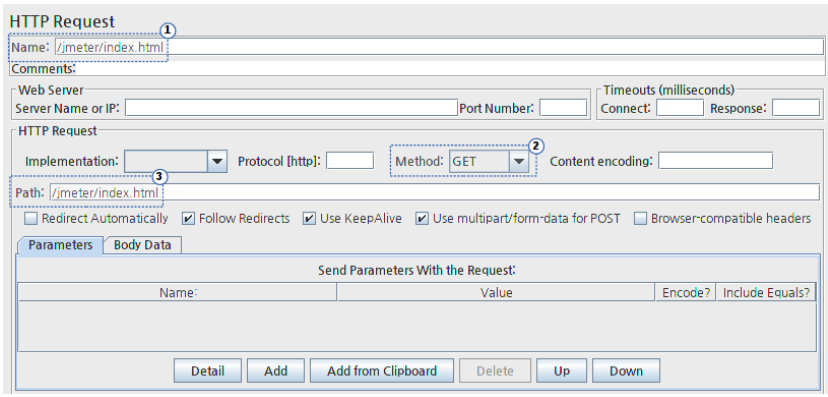
- /jmeter/index.html
- /jmeter/search/index.html
- /jmeter/search/search

② Method: HTTP Request가 생성될 때 기본적으로 GET으로 설정된다. /jmeter/search/search만 POST 방식을 사용하므로 이 Sampler만 POST로 변경한다.

③ Path: 세 개의 HTTP Request Sampler의 Path에 각각 다음 값을 입력한다.

- /jmeter/index.html
- /jmeter/search/index.html
- /jmeter/search/search

[그림 3-8] HTTP Request Sampler 추가



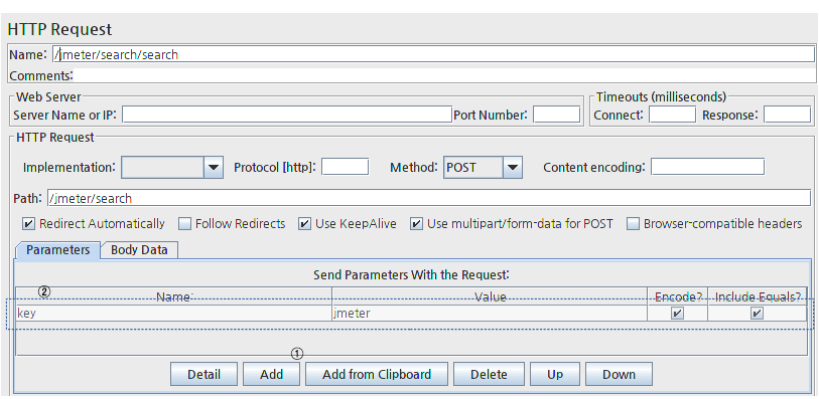
Parameter 입력

/jmeter/search/search에는 검색 키워드를 함께 보내기 위해서 Parameters를 추가한다.

① Add 버튼을 누른다.

② Name : key / Value : jmeter

[그림 3-9] HTTP Request Parameter 추가



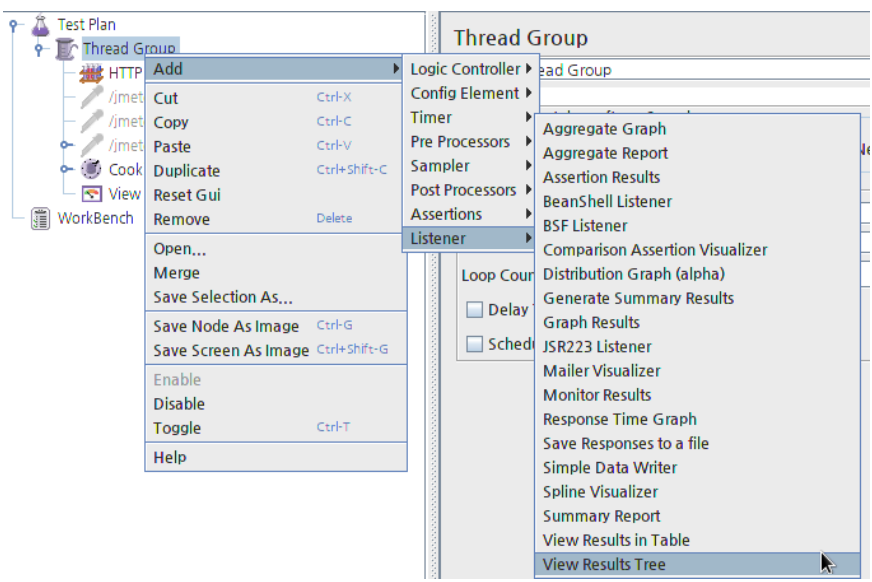
3.1.4 Listener 추가 및 설정

Listener는 Sampler의 요청에 대한 결과를 수집해서 그 결과값을 보여주는 Element를 의미한다. 요청을 보낸 후 성공/실패, 응답시간, 응답 메시지 등을 확인하려면 반드시 추가해야 한다. 여기서는 View Result Tree와 Summary Report를 추가한다.

View Results Tree 추가

Thread Group에서 'Add → Listener → View Results Tree'를 선택하여 추가한다.

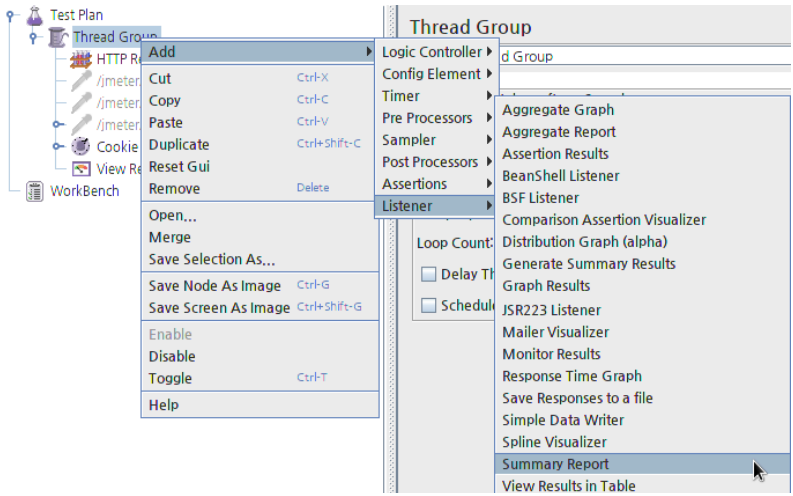
[그림 3-10] View Results Tree 추가



Summary Report 추가

Thread Group에서 'Add → Listener → Summary Report'를 선택하여 추가한다.

[그림 3-11] Summary Report 추가



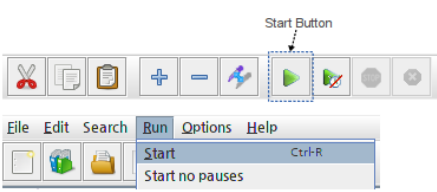
View Results Tree와 Summary Report는 별도의 설정이 필요 없다.

3.2 테스트 실행 및 결과 확인

GUI 모드에서 테스트를 실행하는 방법은 다음 세 가지가 있다.

- 메뉴 바에서 'Start' 버튼을 클릭한다.
- 메뉴에서 'Run → Start'를 선택한다.
- 단축키 'Ctrl-R'를 누른다.

[그림 3-12] 테스트 실행 화면

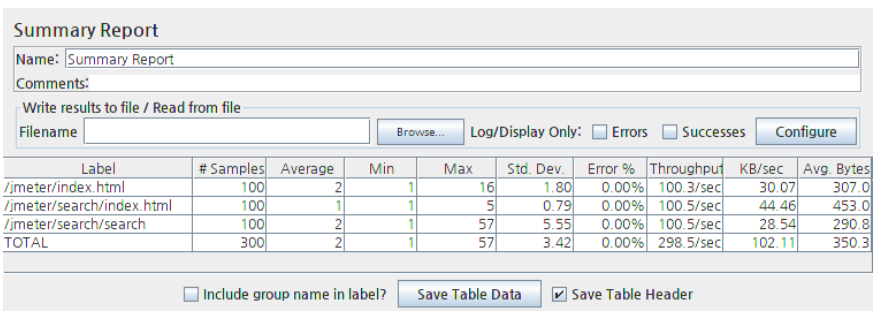


여기에서는 결과보는 방법을 간단히 설명하고 9.1 결과 분석에서 자세한 내용을 다루기로 한다.

3.2.1 Summary Report

테스트 결과를 요약Summary해서 보여준다. 통합된 요청량Sample Count, 응답시간Response Time/Load Time, 오류율, 단위 시간당 처리량Throughput 등을 확인할 수 있다. Label 부분이 HTTP Request Sampler에서 설정한 Name이다. 동일한 Name을 사용하면 구별이 어려우므로 주의한다. 응답시간은 Average, Min, Max 부분으로 1/1000초 단위로 표시된다.

[그림 3-13] Summary Report 화면



3.2.2 View Results Tree

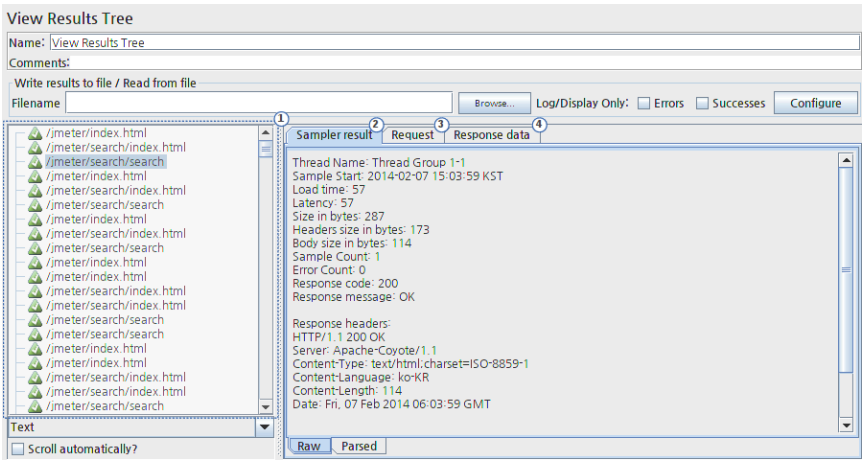
각 결과의 요청/응답Request/Response를 상세하게 살펴볼 수 있는 Listener다. 초기에 스크립트를 만들고 정상적으로 처리되는지 확인할 때 용이하다. Pre-Test할 때 사용하므로 기능에 대해서 잘 알아두는 것이 좋다.

[그림 3-14]는 각 Sampler의 결과를 Tree 형태로 보여준다.

① Sampler 목록 : 이중에서 하나를 선택하면 오른쪽 패널에 해당 Sampler의 상

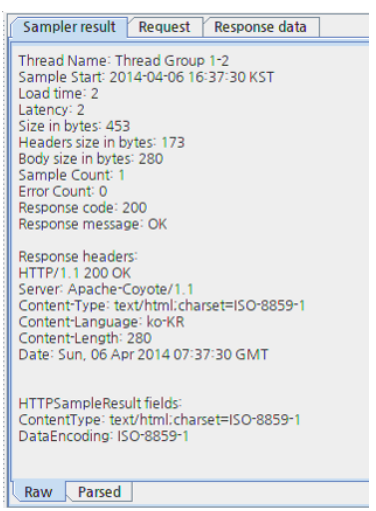
세 정보가 출력된다. 상세 정보 패널은 세 개의 탭으로 구성된다.

[그림 3-14] View Results Tree 화면



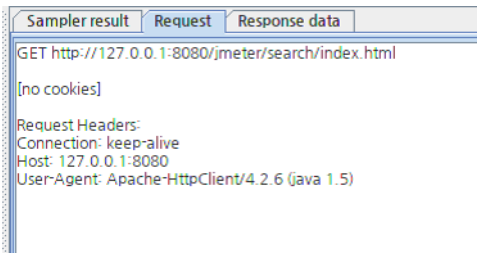
② Sampler result : 해당 Sampler의 요청 결과를 보여준다. 성공/실패 여부를 포함해서 응답시간과 크기Size 등을 보여준다.

[그림 3-15] Sampler result 화면



③ Request : 해당 Sampler가 웹 서버에 보낸 Request 정보를 볼 수 있다.

[그림 3-16] Request 화면



④ Response data : 해당 Sampler의 요청에 대한 응답 메시지를 보여준다. 웹 서버로 요청을 보냈으므로 Response data는 html 문서로 출력된다.

[그림 3-17] Response data 화면 뷰