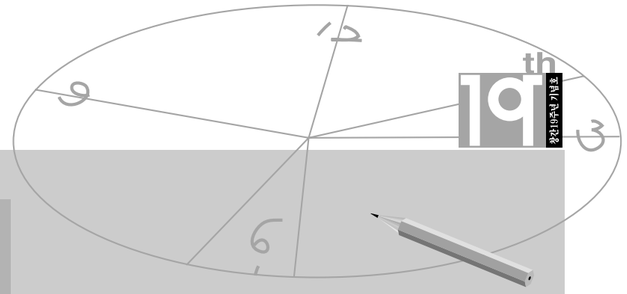
특집 2-3부



난 이렇게 개발한다 3

"자신의 몸에 착 달라붙는 운명적인 툴을 손에 쥐어라"

인터넷과 공유정신의 만남은 개발자들이 자유롭게 사용할 수 있는 무료 개발 툴의 황금기를 낳았다. 이전에는 돈을 주고 구입해야 할만한 개발 툴을 이젠 수많은 오픈 소스 프로젝트 속에서 어렵지 않게 발견할 수 있다. 심지어 그들 속을 들여다 볼 수 있고 라이선스의 범위 안에서 합법적으로 소스를 수정해 자신의 개발 환경에 맞출 수도 있다. 자신의 몸에 착 달라붙는 운명적인 툴을 손에 쥐는 것은 개발 속도와 작업 산출물의 품질을 향상시킬 수 있는 강력한 수단을 확보하는 것이다. 이 글에서는 자바 개발자들이 부딪히는 여러 문제들을 각종 무료 툴들을 통해 도전하는 방법들을 살펴볼 것이다. 이 중에서 독자의 손에 들릴 무기가 있기를 희망한다.

소프트웨어 툴의 정의는 다양하지만 본 기사에서는 '소프트웨어 개발을 도와주는 각종 프로그램'으로 한정할 것이다. 여기서 '소프트웨어 개발'이란 소프트웨어의 생명주기(lifecycle)에서 발생하는 각종 작업(task)들을 의미한다. 소프트웨어 분석 · 설계 · 구현 · 설치 · 운영 · 유지보수의 과정 속에서 각종 세부 작업들을 수행할 때 이전보다 더 빨리, 더 값싸게, 더 품질이 좋게 도와주는 프로그램들이 소프트웨어 툴이다.

왜 툴의 참맛을 알아야 하는가?

명성이 높은 툴이라고 모두가 받아들이는 것은 아니다. 필자가 외부에서 강의할 때 앞으로 소개할 이클립스(eclipse)와 CVS(Concurrent Versions System)를 수강생들에게 가르치고 사용을 권유한 적이 있다. 그런데 수강생들 대부분이 일주일 만에 원래 사용하던 텍스트 에디터를 사용하는 것을 보았다. 새로 도입한 툴에 적응하지 못한 것이다. 이유는 간단했다. "이클립스 단축키가 에디트 플러스하고 달라요. CVS보다 공유 디렉토리에 쓰는 게 더 편해요."

새로운 툴의 수많은 장점들과 효과를 배우고 적용할 때 공짜로 되는 것은 아니다. 시간과 노력을 투자해야 한다. 만약 그 가치를 확신하고 인정하지 못하면 툴의 참맛을 알기 전에 기존 시스템으로 회귀한다. 툴의 효과를 제대로 맛보려면 얼마 간의 학습과 적용 시간이 반드시 필요하다. 이것을 낭비로 생각하면 결국 툴을 포기하게 된다. 결국 이들 수강생들은 프로젝트를 진행할수록 시간이 지연되거나 경직된 코드들을 만들게 됐다. 차라리 처음에 시간이 좀 걸렸어도 이클립스나 CVS를 제대로 배우고 프로젝트에 적용했더라면 전체 개발 시간은 단축됐을지 모른다. 필자의 경험상, 개인이나 팀이 유용한 툴들을 성공적으로 도입하려면 다음과 같은 과정을 밟는 것이 좋다.

박지훈 outinsider@orgio.net

소프트웨어 개발자와 조직들이 인터넷을 적극적으로 이용해 효율적으로 능력을 향상할 수 있는 방법론과 수단에 대해 관심이 많다. 그리고 국제적인 소프트웨어 프로세스 심사 및 개선 표준들(SPICE, CMM)과 자연스럽게 연동하며 네트워크와 커뮤니티, 협동에 근거한 효과적인 소프트웨어 개발 모델과 방법론, 유연한 툴의 개발을 연구 목표로 삼고 있다.

일러스트레이터 조경보 siren71@hitel.net

가. 우리 팀이 자주 부딪히는 문제가 무엇인가?
예 : 소스 코드의 클래스, 메쏘드, 파라미터의 이름을 자주 바꾸는데 그때마다 클라이언트 코드를 제대로 변경할 수 없거나 시간이 많이 걸리는 문제가 발생한다.

나. 이 문제를 해결할 수 있는 툴이 있는가?
예 : 이클립스에 소스 코드 내의 각종 이름들을 자동으로 변경해주는 리팩토링(refactoring) 기능이 있다.

다. 간단한 문제를 임의로 만들거나 선정해 툴로 해결해 본다. 반드시 기존에 사용했던 툴이나 방법과 비교해 새로운 툴의 효과를 공식적으로 인정할 때까지 해야 한다.
예 : 이클립스로 간단한 소스 코드를 작성해보며 클래스 이름이나 메쏘드, 파리미터 등의 이름들을 변경해 본다. 기존의 텍스트 에디터 툴로 소스 코드의 변경을 처리했을 때와 이클립스를 사용했을 때를 비교해 어떤 것이 효과가 높은지 비교해 본다. 중요한 것은 고개를 끄덕이며 온 맘으로 인정할 때까지 실험하고 테스트하는 것이다. "어라, 이게 뭐야?" 하는 반응이면 그 툴이 아무리 명망이 높아도 도입에 실패할 가망성이 높다.

라. 만약 적당하다고 판단되면 실제 프로젝트에서도 사용해본다. 새로운 툴이 익숙하지 않기 때문에 생기는 초기의 각종 지연 현상은 '다'의 효과를 생각하면서 무시한다. 도입비용으로 생각한다.

또는 주위의 친한 동료나 유명한 개발자들이 인정하는 툴을 무턱대고 써보며 자신의 어떤 문제를 해결하는지 주의 깊게 관찰하는 것도 하나의 방법이 되겠다. 그렇다면 전 세계의 수많은 자바 개발자들은 어떤 툴들을 사용하는 것일까?

무료 개발 툴 파레이드

개인이나 기업이 무료로 사용할 수 있는 무료 툴 중 많은 개발자들이 인정하고 사용하는 것들을 주로 선정했다. 주로 오픈 소스의 결과물들이 많다. 일단 애플리케이션 프레임워크(framework)나 유틸리티 등은 제외한다.

이클립스, 통합 개발환경

IBM이 자 바 언어로 만든 통합 개발환경 툴로서 오픈 소스로 대중에게 공개됐다. JDT가 기본 설치된 이클립스는 자바 소스 코드의 편집, 컴파일, 디버깅 등의 IDE 기본 기능 외에 다양한 종류의 플러그인 프로그램이 제공된다.

이클립스에서 주목해야 할 기능은 무엇일까.

소스 코드 변경의 자동 처리

소스 코드의 변경을 자동으로 처리해주는 리팩토링 기능이 매우

〈화면 1〉 이클립스

편리하다. 이런 기능이 없는 IDE에서는 클래스나 메쏘드, 파라미터의 이름을 변경하려면 Find/Replace를 이용해야만 한다. 패키지들이 많은 프로젝트에서는 이것도 완전하게 동작하지 않는다. 결국 Find/Replace 이후 컴파일해 에러가 발생하면 해당 위치를 찾아 수정하는 번거로운 작업을 해야 한다. 이클립스는 이름 변경뿐만 아니라 다른 패키지로 클래스를 이동하고 메쏘드 파리미터의 위치를 변경하는 리팩토링 기능도 제공한다.

필자는 리팩토링의 기능 중 주로 Rename(이름 변경)을 자주 이용한다. 클래스나 메쏘드의 이름을 변경하면 자연스럽게 코드의 질이 높아지는 계기가 된다. 이름을 바꾸면 코드 구조의 변경을 촉발시키는 경우가 많다. 이클립스의 리팩토링 기능 덕분에 부담 없이 소스 코드를 변경할 수 있고, 이것은 코드 품질의 향상으로 이어진다.

소스 코드 타이핑 중 에러 경고 기능

소스 코드 타이핑 중에도 문법 에러가 발생되면 즉시 해당 위치에 빨간 줄이 쳐지면서 경고해준다. 초보 개발자들은 컴파일 중에 발생하는 엄청난 문법 에러들을 감당 못하고 개발을 포기하는 경우가 많다. 그렇지만 타이핑 중에 문법 에러가 즉시 발견되면 끝까지 프로그램을 완성할 수 있다. 필자는 이 기능을 보고 '이클립스는 나를 위해서 태어난 거야!' 라고 외쳤던 수강생을 본적이 있다.

플러그인 프로그램을 통한 시스템 기능 확장

기본 기능 이외에 데이터베이스 관리, XML 편집, 온라인 짝 프로그래밍 등 다양한 종류의 플러그인 프로그램이 존재하기 때문에 개발자의 상황에 따라 이클립스의 기능을 쉽게 확장할 수 있다. 다음에 소개된 플러그인 전문 사이트에서 자신이 원하는 플러그인을 쉽게 찾을 수 있다.

CVS 클라이언트의 기본 내장

여러 명이 공동으로 소프트웨어를 개발할 때 소스 코드를 통합적으로 관리해주는 시스템이 CVS이다. 이것은 CVS 서버와 클라

이언트 프로그램으로 구성되는데, 이클립스는 CVS 클라이언트 프로그램이 내장되어 있다. 이클립스와 완전히 통합된 형태이기 때문에 사용하기가 매우 편리하다.

만약 EditPlus 또는 UltraEdit가 최고라고 생각하는 개발자는 이 툴을 써보길 바란다. 단, 소스 코드의 각종 이름을 변경하고 코드를 이동시키면서 소스 코드의 품질을 개선하는 실험을 해보자. 만약 Ant, JUnit 등을 통합적으로 사용하려면 이 툴은 거의 필수적이라고 해도 과언이 아니다. 이클리스 홈페이지 및 관련 사이트는 다음과 같다.

◆ **이클립스 홈페이지** : http://www.eclipse.org
◆ **이클립스 홈페이지 내의 플러그인 소개** : http://www.eclipse.org/community/plugins.html
◆ **이클립스 플러그인 전문 사이트** : http://eclipse-plugins.2y.net/eclipse/index.jsp

단위 테스팅 툴 및 객체지향 설계 유도기 , JUnit

JUnit은 켄트 벡(Kent Beck)과 동료들이 만든 단위 테스팅(Unit Testing) 프레임워크이다. 이것을 사용하면, 개발자가 의도한 대로 클래스가 제대로 동작하는지 반복적으로 확인할 수 있는 테스트 프로그램을 쉽게 만들 수 있다. 특히 XP(eXtreme Programming)의 TDD(Test Driven Development)에서 핵심적으로 사용되는 툴이다(자세한 내용은 현재 마소에서 연재되고 있는 TDD 기사를 참고할 것). JUnit의 주목해야 할 기능을 살펴보자.

이미 구현한 클래스의 기능 테스트

먼저 원하는 클래스를 구현한 다음, 그것에 대한 테스트 프로그램을 나중에 만들어 클래스의 기능을 검증하는 용도로 사용될 수 있다. 즉, A 클래스를 만들고 컴파일한 이후에, 제대로 만들었는지 확인하기 위해 JUnit을 이용해 ATest 클래스를 만드는 경우다. 기존에는 클래스가 제대로 만들었는지 확인하려면 변수의 내용과 메쏘드의 반환 값을 화면에 출력하는 테스트용 프로그램을 개발자가 직접 만들어야 했다. 그리고 화면에 출력되는 각종 값들을 읽고 분석해 테스트를 통과했는지를 확인해야 했다. 반면 JUnit를 이용해 테스트 프로그램을 만들면 '합격/불합격'의 명쾌한 테스트 결과를 즉시 확인할 수 있다. 더군다나 클래스의 기능이 확장될 경우에도 테스트 프로그램을 더불어 확장할 수 있다.

또한 팀으로 개발할 때는 각 개발자마다 자신에게 할당된 클래스들의 기능을 테스트할 때 사용할 수 있다. 만약 철수가 A 클래스를 만들고, 영이가 B 클래스를 만든다고 가정하자. 그런데 A 클래스는 B 객체를 호출한다. 영이 입장에서는 B 클래스를 에러 없이 구현해야 한다. 이때 B 클래스를 사전에 테스트하기 위한 용도로 JUnit을 사용하면 편리하다.

앞으로 구현할 클래스의 기능 테스트

이것은 XP의 TDD 방식에서 JUnit의 역할이다. 클래스를 먼저 만드는 것이 아니라, 구현할 클래스의 테스트 프로그램을 만들고, 이것을 통과하는 가장 단순한 클래스를 만드는 것이다. 전형적인 소프트웨어 개발 순서와 다르다. ATest라는 테스트 프로그램을 만들고, A라는 클래스를 개발한다. 이것은 클래스의 기능을 테스트하는 효과 외에 심플하고 느슨한 구조의 객체지향 설계를 유도한다. 또한 JUnit은 리팩토링의 안전장치 역할을 하게 된다. 반복 가능한 JUnit 테스트 프로그램이 있으면, 리팩토링을 했을 때 프로그램이 부서지는 현상을 방어할 수 있다.

가장 단순하며 지적인 툴

JUnit은 이제까지 나온 툴 중에서 가장 단순하며 지적인 툴에 속한다. JUnit을 사용해 TDD를 하면 프로그램은 느슨해지며 단순한 클래스 구조로 진화한다. 만약 A 클래스를 테스트하기 위해 B 클래스가 필요하다면 Mock Object 툴을 사용해 B를 구현하지 않고도 마치 구현한 것처럼 테스트할 수 있다. 또한 JUnit을 확장한 HttpUnit, Cactus를 사용하면 서버 쪽의 서블릿, EJB 등의 프로그램을 테스팅할 수 있다. JUnit을 잘 사용하면 개발자는 프로그램의 품질에 집중하게 되며, 결국 버그가 없는 프로그램의 지름길에 들어서게 된다. 이클립스에 JUnit 플러그인을 설치할 수 있으니 이클립스와 더불어 테스트해보길 추천한다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **JUnit 홈페이지** : http://www.junit.org/
◆ **Mock Object 관련 페이지** : http://www.mockobjects.com, http://www.easymock.org, http://www.mockmaker.org
◆ **HttpUnit 홈페이지** : http://httpunit.sourceforge.net/
◆ **Cactus 홈페이지** : http://jakarta.apache.org/cactus/
◆ **이클립스 JUnit 플러그인** : http://dev.eclipse.org/viewcvs/index.cgi/~checkout~/jdt-ui-home/plugins/org.eclipse.jdt.junit/index.html
◆ **켄트 벡의 TDD 서적 드래프트** : http://groups.yahoo.com/group/testdrivendevelopment/files/TDD17Jul2002.pdf

로깅 및 추적 툴, Log4J

Log4J는 아파치의 자카르타(Jakarta) 프로젝트의 서브 프로젝트로서, 애플리케이션의 각종 로그(log) 및 추적(trace) 정보들을 통합적으로 출력하고 유지해주는 API의 집합이다. Log4J의 주목할만한 기능은 무엇일까.

개발자들이 테스트 및 로그용으로 만드는 소스 코드 내의 수많은 System.out.println() 문장들을 이것으로 대체할 수 있다. 프로그램의 디버깅, 실행정보, 경고, 에러, 로그 등의 정보들을 통일되고 일관된 인터페이스를 사용해 화면에 출력하거나 저장할 수 있다.

프로그램을 만들어서 상업용으로 배포하려면 생각 없이 만든 소스 코드 내의 로깅 및 테스트 코드를 어떻게 취급해야 할지 고민된다. 기껏해야 특정 부분을 주석처리하거나 삭제하는 경우가 대부분이다. 나중에 그들을 다시 필요로 할 때는 일일이 수작업으로 그들을 복원해줘야 한다. 하지만 Log4J는 이런 로그 또는 각종 추적용 정보들을 통합적으로 다룰 수 있어 중급 이상의 복잡한 프로그램에 적용하면 개발 시간을 단축시켜 준다. 참고로 JDK 1.4는 Log4J보다 간단한 로깅용 API를 제공한다. Log4J는 JDK 1.1 버전 이상에서 동작한다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **Log4J 홈페이지** : http://jakarta.apache.org/log4j/
◆ **자바 1.4의 로깅 API** : http://java.sun.com/j2se/1.4/docs/guide/util/logging/

자동 빌드 툴, Ant

Ant는 아파치/자카르타 프로젝트의 서브 프로젝트로서, 자바 프로그램을 자동으로 빌드(build)해 주는 툴이다. 각종 환경설정 및 옵션(클래스 패스, 바이트 코드의 생성 위치, jar 파일 생성 여부)을 컴파일 할 때마다 일일이 지정하지 않고 특정 XML 파일에 저장해 자동으로 소프트웨어를 생성할 수 있게 해준다. Ant의 주목할만한 기능으로는 무엇이 있을까.

프로젝트를 여러 개 생성하면 그들 각각의 환경설정 정보들이 다른 경우가 많다. 필요한 라이브러리와 그들의 위치들도 각기 다르다. 따라서 각 프로젝트마다 javac 명령어를 사용할 때 일일이 그들만의 환경 정보들을 옵션으로 지정하는 번거로움이 있었다. 하지만 Ant를 사용하면 각 프로젝트마다 빌드에 필요한 모든 정보가 담긴 XML 파일에 따라 자동으로 프로그램이 생성된다. 따라서 컴파일과 배포 형식의 jar 파일을 자주 만들어야 하는 복잡한 프로젝트는 Ant를 사용하면 빌드 시간을 대폭 단축할 수 있다.

이것만큼 확실하게 시간 단축 효과를 보는 툴이 없다. 환경설정이 복잡한 프로젝트일수록 확실하게 보답을 해준다. javac를 할 때 클래스 패스를 신경 쓰지 않는 것이 어디인가? 더군다나 소프트웨어 개발 플랫폼이 변경되어도 프로젝트용 Ant 파일만 있으면 빌드 과정이 수월해진다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **Ant 홈페이지** : http://jakarta.apache.org/ant/

소스 코드 형상관리, CVS

소스 코드 형상관리의 대표 툴로서, 각 개발자가 수행한 소스 코드의 변경을 중앙에서 관리해준다. 모든 개발자들이 항상 최신 소스 코드를 갖고 작업하도록 도와준다.

누군가가 프로젝트의 모든 소스 코드를 CVS 서버에 올리면 다른 개발자들은 그것을 통째로 다운받을 수 있다. 이후 변경되거나 새로운 파일들을 각자 CVS 서버에 올리면, 다른 개발자들은 변경된 것들만 다운받아 사용할 수 있다. 이런 방식을 통해 모든 개발자들은 항상 최신 파일들과 동일한 프로젝트 소스 코드 환경 하에서 작업할 수 있게 된다.

만약 이제까지 여러 명이 개발할 때 전자메일, 메신저, 공유 디렉토리 등을 통해서 서로의 최신 코드들을 교환해 왔다면 지금 당장 CVS 서버를 설치하라. 일주일만 지나면 과거를 후회하게 될 것이다. 이클립스를 사용하면 쉽게 CVS의 기능을 체험할 수 있다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **CVS 홈페이지** : http://www.cvshome.org
◆ **윈도우용 CVS 클라이언트 홈페이지** : http://www.wincvs.org

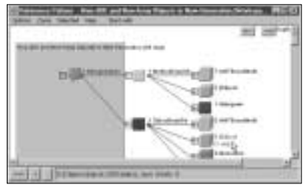
소스 코드 자동 리뷰, CheckStyle

오픈 소스로서 자신 또는 팀의 코딩 표준(coding standard)에 맞게 소스 코드가 작성됐는지 체크해주는 툴이다.

코딩 표준이란 자신/팀이 개발할 소스 코드가 지켜야 할 각종 규칙들을 의미한다. 변수의 이름 및 포맷, 들여쓰기(identation), 블럭의 형태 등을 구체적으로 정의한 것이다. 혼자가 아닌 팀 단위로 소프트웨어를 개발할 때 공동으로 합의된 표준이 있으면 소스 코드의 가독성이 높아진다. 팀원 누구라도 다른 팀원이 작성한 프로그램을 유지 보수할 때 쉽게 읽을 수 있고 일관된 포맷으로 수정할 수 있다.

기존에는 완성된 소스 코드와 코딩 표준을 수작업으로 비교하면서 표준에 어긋난 소스 코드를 찾아서 수정했다. 하지만 이 툴을 사용하면 특정 파일에 저장된 각종 코딩 표준 규칙(이름, 크기, 블럭 등)에 어긋나는 파일들을 자동으로 손쉽게 찾아낼 수 있다.

〈화면 2〉 Jinsight



〈화면 3〉 JMeter

만약 팀 단위의 코딩 표준을 만들고 모든 팀원이 그것을 지키기로 마음을 먹었다면 소스 코드를 수작업으로 리뷰하기 전에 이 툴을 먼저 돌려 결과를 보라. 그 후 이 툴이 잡을 수 없는 체크 사항들을 공동으로 리뷰하면 시간이 훨씬 단축될 것이다. 또한 Ant와 연동해 사용할 수 있다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **CheckStyle 홈페이지** : http://checkstyle.sourceforge.net/

프로그램 실행 분석, Jinsight

자바 프로그램의 실행을 분석해 비주얼하게 표현해주는 툴로서 IBM이 만들었다. 프로그램의 성능 분석, 메모리 점유율, 디버깅 정보, 메쏘드 호출 정보 등을 비주얼하게 보여준다. 이 정보를 바탕으로 개발자는 문제의 원인을 유추해낼 수 있다.

프로그램이 지나치게 느리거나 자원을 많이 점유한다면 이 툴을 실행해서 어떤 곳이 병목지점인지 확인해볼 수 있다. Jinsight는 메모리 점유율, 메쏘드 호출 시간 등의 정보를 종합적으로 보여줌으로써, 프로그램 실행시 발생하는 여러 문제의 원인을 개발자가 보다 정확하고 빨리 발견할 수 있게 도와준다. 복잡한 소프트웨어를 개발할 때는 System.out.println()을 통해 문제 해결을 시도하는 것보다 이 툴을 사용해 병목지점에 해당하는 소스 코드를 체계적으로 추적해 원인을 파악하는 것이 더 빠른 길일 수 있다.

프로그램 실행과 관련된 많은 정보들을 보여주기 때문에 실행시 발견되는 문제들의 원인을 보다 쉽게 발견할 수 있다. 하지만 문제의 해결은 개발자의 몫이다. 이 툴은 다른 것과 달리 사용방법이 쉽지 않다. 따라서 간단한 프로그램의 경우, 의심나는 부분에 System.out.println() 문장을 붙이는 것이 더 빨리 문제를 찾는 길일 수도 있다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **Jinsight 홈페이지** : http://www.alphaworks.ibm.com/tech/Jinsight

자원 테스팅 툴, JMeter

JMeter는 아파치/자카르트 프로젝트의 서브 프로젝트로서, 각종 파일, 서블릿, 서버 등의 자원들의 성능을 테스트하는 툴이다. 주로 웹 서버의 성능을 테스트할 때 요긴하게 사용된다. 그 외에 FTP, DB 서버들의 성능을 간단하게 테스트할 수 있다.

이 툴을 사용하면 많은 사용자들이 동시에 서버로 접속을 시도하는 효과를 낼 수 있다. 기존에는 자신이 구축한 서버의 접속 가능한 최대 사용자와 성능을 확인하려면 다른 사람들에게 접속을 부탁하거나 아니면 직접 클라이언트용 테스트 프로그램을 만들어야만 했다. 하지만 이 툴을 사용하면 접속하는 간격, 회수, 사용자 수 등을 인위적으로 지정해 마치 사람이 접속해 사용하는 효과를 낼 수 있다. 따라서 사람이 직접 테스트하는 것보다 편하고 빠르게 서버의 안정성을 테스트할 수 있다.

상업용 웹 애플리케이션을 구현했을 때 이 툴을 사용해 그들의 기초 체력과 성능을 측정해보자. 시스템이 견딜 수 있는 무게와 응답 속도를 대략적으로 확인할 수 있다. 만약 예상에 못 미치는 결과가 나왔다면 그 원인을 찾고 해결해야 할 것이다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **JMeter 홈페이지** : http://jakarta.apache.org/jmeter/

팀 지식 공유, Wiki 시스템

위키는 누구나 참여해 모두가 함께 지식을 축적하고 진화하도록 유도하는 시스템이다. 게시판과 달리 누구나 서로의 글을 수정할 수 있고 함께 정보를 축적해 나갈 수 있다.

이 툴을 사용하면 팀의 각종 개발 지식들을 한 곳에 축적해 나갈 수 있다. 공유 정신이 충만한 팀원들이라면 서로의 의견과 아이디어를 상호 발전시킴으로서 개발지식들이 유기적으로 연결된 팀의 자산을 만들 수 있다. 누구나 쉽게 접근하고 글을 수정할 수 있으며, 협동에 기반을 두기 때문에 소프트웨어의 에러나 수정 등의 추적 과정을 공동으로 관리하는 용도로 이용할 수도 있다.

위키 시스템의 위력과 가치는 누구나 인정한다. 이것을 팀의 재산을 만드는 도구로 활용해보자. 1, 2년이 지나면 서로의 개발

〈그림 1〉 All For One

지식과 경험을 상승시키는 놀라운 경험을 하게 될 것이다. 홈페이지 및 관련 사이트는 다음과 같다.

◆ **오리지널 위키 페이지** : http://www.c2.com
◆ **국내의 대표적인 위키 페이지** : http://www.no-smok.net

무엇이 좋은 툴인가

개발자마다 '좋은 툴'의 의미는 각기 다를 것이다. 필자는 개인적으로 다음 두 가지를 좋은 툴의 조건으로 생각한다. 그 하나는 바로 팀의 일체화를 유도하고 유지시켜주는 툴이다.

여러 사람들이 모이면 커뮤니케이션 문제로 잡음이 발생하는 경우가 많다. 그것은 팀의 일체화를 방해해서 팀 전체의 개발속도를 단계적으로 느리게 한다. 팀 성공의 기본적인 요소 중 하나는 팀원들이 일관성이 유지된 환경 속에서 협력하는 것이다. 툴은 이때 중요한 역할을 할 수 있다. 예를 들어 CVS는 팀 내의 소스 코드 변경 및 일관성의 유지, CheckStyle은 소스 코드 형식의 일체화를 보다 쉽게 실현하기 때문에 팀원들이 동일한 환경에서 일하도록 도와준다. 위키를 통한 공동 지식 축적 및 진화도 마찬가지이다. 개발 지식 및 노하우의 상향 평준화를 유도할 수 있다.

무료 툴 관련 사이트

1. http://www.sourceforge.net : 오픈 소스 사이트
2. http://alpahworks.ibm.com : IBM의 알파 테스트용 프로그램 사이트
3. http://www.onjava.com/pub/q/java_os_directory : 오렐리사의 오픈 소스 자바 디렉토리

〈그림 2〉 양복을 입으면 점잖아진다.

물론 모든 것은 팀의 정신에 달려 있다.

두 번째는 바람직한 개발 행위를 자연스럽게 유도하고 유지시켜주는 툴이다.

개발자에게 바람직한 행위를 강요하는 것이 아니라 자연스럽게 유도하고 촉발시키는 툴이야말로 누구에게나 필요한 것이다. JUnit의 표면상 기능은 클래스의 기능이 제대로 동작하는지 확인하는 것이다. 그런데 그것을 TDD의 툴로서 활용하면 테스팅뿐만 아니라 그와 동시에 바람직한 객체지향 설계를 유도하는 장치가 되어 버린다. 미리 만든 테스트에 통과하기 위해 개발자는 클래스들의 책임과 구조에 자연스럽게 집중하게 된다. 강요와 억지가 아니라 유도되는 것이다.

무료 툴을 잘 활용하세요

짧지만 앞에서 언급한 툴들이 지금 이 순간 모든 개발자에게 필요한 것은 물론 아니다. 필자는 단지 많은 사람들이 인정하는 강력한 무기들의 리스트를 나열함으로서 독자가 자신의 상황에 필요한 것을 직접 고를 수 있게 도와주려 노력했다. 앞에 소개한 각종 툴들은 무료로 사용할 수 있는 것으로 선정했다. 각 분야의 최고 베스트 툴이 아니다. 따라서 같은 기능의 상업용 툴들이 훨씬 좋을 수 있다. 애초에 기획했던 내용보다 많은 부분이 미흡하지만 아직 좋은 도구를 발견하지 못한 독자들에게 본 기사가 좋은 참고자료로 활용되길 바란다.

정리 : 조규형 jokyu@sbmedia.co.kr