

Total Eclipse

IDE의 미래와 오픈 소스의 흐름

JLab.net 허원진 · 전지원 공저

진정한 이클립스 프로젝트를 위한 책

- 혁신적 IDE, 이클립스의 사용방법
- 최고의 자바 그래픽 컴포넌트 SWT를 통한 응용프로그램 개발
- 이클립스 기능의 무한 확장 플러그인, PDE를 통한 개발

YoungJin.com Y.
영진닷컴

Beyond Press

BIPress

Total Eclipse

JLab.net 허원진 · 전지원 공저

YoungJin.com **Y.**
영진닷컴

Total Eclipse

Copyright © 2003 by Youngjin.com
1623-10 Seocho-dong Seocho-Gu
Seoul, 137-878
Korea

All rights reserved. First edition Printed 2003.
Printed in Korea

ISBN 89-314-2562-7

Microsoft는 Microsoft Corp.의 등록상표입니다.
Windows는 Microsoft Corp.의 등록상표입니다.
Java는 Sun Microsystems, Inc.의 등록상표입니다.

이외의 상표는 각 회사의 등록상표입니다.

만든 사람들

저자 / 허원진, 전지원 | 오픈 퍼블리쉬 참여자 / 이아스, 정원용, 박재성 | 기획 / 비아이프레스
기획 책임 / 엄진영 | 진행 / 윤은숙 | 편집 / 민기희 | 표지 / 플릭스

저자 : 허원진 | 한국 이클립스 유저 모임 제이랩 운영자 / 이클립스 프로젝트 관련 카운셀러 활동 / 자바 관련 전문 강사 /
인하대학교 컴퓨터 공학과 재학 중

전지원 | 한국 이클립스 유저 모임 제이랩 집필진 / MPC 음성정보연구소 근무

오픈 퍼블리쉬 참여자 : 이아스 | 티맥스 소프트 근무 / 이클립스용 제우스 서버 플러그인 개발
정원용 | JCO 주체 자바 S/W 공모전 은상 / 광운대학교 네트워크 연구회 (K-NET) 회원
박재성 | www.javajigi.net 자바지기 운영자

머리말

2001년 이클립스라는 생소한 이름의 공개 소프트웨어가 배포되기 시작했습니다. 국내에서는 이 때만해도 검색을 하면 나이트 클럽과 자동차 관련 페이지가 검색되었습니다. 2003년 이클립스는 공개 소프트웨어는 어떻다는 고정관념을 깨고 엄청난 속도로 확산되고 있습니다. 2002년 필자에게 들어온 기업의 상담 내용은 이클립스를 이용해서 상용 애플리케이션 수준의 개발이 가능하다는 질문이 주류를 이루었습니다. 하지만 지금은 어떻게 이클립스 플랫폼을 통해 사업을 할 것인가가 주류를 이루고 있습니다.

이미 IBM은 이클립스를 기반으로 한 IDE를 발표하였습니다. 1년 사이의 변화, 이제 사람들은 이클립스라는 흐름이 바다로 갈 것인가에 대한 의문 보다는 이 흐름을 이용해서 자신이 어떻게 이익을 볼 것인가에 대한 생각을 하고 있습니다. 언어를 공부하는 학생, 컴퓨터 언어 개발자들을 비롯해서 지구상에서 IDE를 생각하고 사용하는 분들은 이 흐름을 외면하지 않았으면 합니다.

이클립스에는 IBM과 볼랜드 등 주요 툴 개발회사들의 노하우를 바탕으로 제작되었습니다. 이클립스에는 IDE 개발의 정수와 새로운 요구사항들의 해결책이 들어 있습니다. IDE적 가치를 제외하고도 이클립스는 정말 대단한 '생각'입니다. 중복 개발을 없애고 이상적인 IDE로의 길, 이것이 이클립스 프로젝트입니다. 이클립스가 여러분에게 주는 이익은 두 가지입니다. 이 이상적인 툴을 이용해서 즐겁고 편하게 개발하는 것과, 그 이상적인 툴을 만드는 프로젝트에 직접 참여하는 것입니다. 이클립스라는 거대한 흐름 위에 수력 발전소를 건설하든 아니면 아이처럼 물장난을 하던 자신에게 맞게 즐기면 됩니다. 프로젝트 이니까요.

이 책은 제이랩의 초대 집필진인 심우곤님과 후에 들어오신 손동익님의 도움으로 만들 수 있었습니다. 큰 도움을 주신 이 분들께 작은 글귀를 빌어 감사 드립니다. 또한 이 책은 저자만의 책이 결코 아닙니다. 오픈 퍼플리쉬라는 독자와의 공개적인 집필 제도를 도입했습니다. 책은 저자를 위해서가 아니라 독자를 위해서 존재하는 것입니다. 이 공개 편집에 참여해 주신, 이아스님, 정원용님, 박재성님께 감사 드립니다. 이 책을 쓸 수 있는 발판을 마련해준 제이랩의 운영진, 집필진 모두와 제이랩 유저 분들께 감사 드립니다. 집필을 도와주시고 챙겨주신 (주)블루엠티내셔널 문석원님께 감사 드립니다. 마지막으로 군에 있는 동안 자식을 위해 30Kg 넘는 책을 보내주신 부모님, 허준님, 윤성욱님 당신을 사랑합니다.

일러두기

Part 1 Eclipse 프로젝트

이클립스 프로젝트의 성격과 내용을 설명하고 있습니다. 이클립스 프로젝트 탄생 목적 파악이 중요합니다. 또한 이클립스 IDE를 이용해서 자바 개발하는 방법에 대한 내용을 다루고 있습니다. 책만으로는 이해하기 어려운 부분도 많으므로 꼭 따라 하는 것이 필요합니다.

Part 2 SWT

현존하는 최고의 자바 그래픽 툴킷인 SWT에 대해서 다루고 있습니다. GUI 경험이 있건, 처음 공부를 하건 SWT 특성과 애플리케이션 구조를 이해하는 것이 중요합니다. SWT의 기반은 JNI(Java Native Interface)입니다. 책의 난이도를 고려해서 제외했지만, SWT를 제대로 다루려면 JNI에 대한 이해가 필요합니다.

Part 3 PDE

이클립스에 실제로 기능을 부여하는 것은 플러그인입니다. 기본적인 플러그인 구조에 대해 살펴보고 실제로 간단한 플러그인과 XML 에디터 플러그인을 만들어 봅니다. 플랫폼과 확장, 확장점에 대한 이해가 중요합니다.

Part 4 Eclipse Resource

이클립스 프로젝트를 더 잘 이해하기 위한 공식 FAQ 번역 문서와 뉴스 그룹 이용법, 커뮤니티 소개가 있습니다. 책 이상의 것을 배우려면 이 파트의 내용을 잘 이용해야 합니다.

차례

Part 1 | Eclipse 프로젝트

Chapter 1 Eclipse	17
01 문답으로 알아보는 이클립스 프로젝트와 이클립스 IDE	18
02 이클립스와 오픈 소스에 대한 필자의 생각	19
03 이클립스 프로젝트 이야기	21
Eclipse platform	22
JDT(Java Development Tools)	23
PDE(plug-in development environment)	23
Chapter 2 Eclipse Basic	25
01 이클립스 설치와 실행	26
설치 전 고려사항	26
윈도우용 설치	31
리눅스용 설치	35
그 외 플랫폼 설치	37
이클립스 Update / Uninstall	38
02 이클립스 IDE 친해지기	39
워크벤치 (Workbench)	39
퍼스펙티브 (Perspective)	40
숏컷바(Short cut bar)	42
툴바 (Tool bar)	43
뷰(View)	43
에디터(Editor)	45

Chapter 3 Eclipse IDE 시작하기

01 Hello, World!부터 만들어 볼까?	48
프로젝트 만들기(Ctrl + N)	48
자바 퍼스펙티브로 보기	49
클래스 만들기	50
코딩하기	51
컴파일하기(Ctrl + S)	52
실행하기(Ctrl + F11)	52
02 JDt 자세히 보기 (코딩 부분)	53
소스의 주석을 나에게 맞게 형식화하기	55
클래스 경로 추가하기	58
자바독 이용하고 보기	59
향상된 클래스 생성 마법사	62
QuickFix 이용(Ctrl + 1)	65
자동 코드 블록 닫기 기능	66
자동 예외처리 코드 삽입 기능	67
Setter와 Getter 만들기	68
소스 형식화 기능	70
03 JDt가 제공하는 뷰 기능 이용하기	70
아웃라인	71
선언 열기	72
계층 구조 보기	72
로컬 히스토리 기능	74
04 이클립스로 디버그를 해보자	75
즐거운 디버깅을 위한 준비	76
브레이크 위치를 정하자	77
디버깅 퍼스펙티브로 전환하자	78
관찰 대상을 정하자	79
실행포인트를 움직이자	81
스텝필터	82
에디터에서 확인하기	82



Chapter 4 이클립스에 탑재된 강력한 기능	85
<hr/>	
01 JUnit	86
JUnit 준비	87
TestCase를 만들자	89
TestCase 실행하기	92
TestSuite	93
02 CVS(Concurrent Versions System)	95
CVS란?	95
CVS 저장소 등록하기	96
CVS 저장소와 프로젝트 연결하기	97
CVS 통한 프로젝트 관리	99
버전 관리(태그)	102
버전 관리(브랜치)	103
03 Ant	104
Ant란 무엇인가?	105
Ant를 왜 써야 하나요?	106
Ant의 컨셉	106
IDE와 Make와의 비교	108
이클립스에서의 Ant	110
Chapter 5 이클립스의 기능 무한 확장 플러그인	117
<hr/>	
01 이클립스 플러그인에 대해서	118
이클립스 플러그인 설치 방법	119
플러그인 업데이트	119
플러그인 언인스톨	120
02 V4ALL – Visual Builder for Eclipse.	120
다운로드/설치하기	121
플러그인 시작하기	121
03 Omundo UML	128
다운로드/설치하기	128
플러그인 시작하기	129

04 CDT C/C++	132
다운로드 설치하기	133
C/C++ 프로젝트 생성하기	135
소스 파일 작성하기	135
컴파일을 위해서 컴파일러 설치하기	137
컴파일/실행하기	141
디버깅하기	143

Part 2 | SWT

Chapter 6 SWT(Standard Widget Toolkit) 151

01 문답으로 알아보는 SWT	152
02 SWT에 대해서	153
AWT(Abstract Windowing Toolkit package)	153
Swing	154
SWT	156
03 SWT에 대한 필자의 생각	157
SWT는 자바의 표준이 아니다	157
새로운 기술 습득의 부담	158
배포의 문제	158
Low Level API 부족	158
예외 처리	159

Chapter 7 SWT Basic 161

01 나의 첫 번째 SWT 애플리케이션	162
SWT 라이브러리 설치	162
SWT 클래스 등록	164



SWT 애플리케이션 프로젝트 만들기	165
02 SWT 애플리케이션 구조	167
03 SWT 애플리케이션 배포	172
Chapter 8 SWT Graphics	173
<hr/>	
01 기본적 이해	174
02 그래픽 API와 그래픽 이벤트	175
03 Text	180
04 Image	183
Chapter 9 SWT Layout	187
<hr/>	
01 FillLayout	189
02 RowLayout	191
03 RowData	194
04 GridLayout	196
05 GridData	199
06 FormLayout	203
Chapter 10 Advanced SWT	209
<hr/>	
01 Print	210
02 Custom widget	217
Widget의 이식성	217
Custom widget은 어디부터 시작하나	218
어떤 식으로 만들까	218
Simple Custom widget	218

이벤트 처리를 통한 더 향상된 CustomWidget	226
SubClassing으로 CustomWidget 구현하기	227
03 Custom Layout	228
Widget 생성에서 Layout 배치까지	228
CustomLayout 구조	229
구현	230
Chapter 11 SWT Widget BIBLE	243
<hr/>	
01 Canvas	246
02 Clipboard	253
03 ComboBox	258
04 Control	261
Keyboard Control	261
Mouse Control	266
05 Display	271
그래픽 처리	271
이벤트 관리	273
Shell 관리	274
06 Drag & Drop	276
07 File & Directory Dialog	283
08 List	288
09 Menu	292
드롭 다운 메뉴	293
팝업 메뉴	297
10 Program	301
11 Shell	306
Shell의 모양	307
Modal과 Modeless 윈도우	309



12 Table	311
13 TabFolder	323
14 Toolbar	327
15 Tree	333
Chapter 12 Face – 플러그인 UI 프레임워크	339
01 JFace란?	340
02 org.eclipse.jface.viewers	341
Viewer 구조	341
ListViewer	342
TableView	347
TreeView	353
TableTreeView	356
03 org.eclipse.jface.action	357
Actions	358
Contributors	360
04 org.eclipse.jface.resource	361
05 org.eclipse.jface.dialogs	371
정의된 Dialog	371
사용자 다이얼로그 만들기	373
06 org.eclipse.jface.wizard	377
마법사의 구조	377
사용자 정의 마법사 만들기	378
07 org.eclipse.jface.operation	384
08 org.eclipse.jface.util	385

Part 3 | 플러그인 개발

Chapter 13 PDE 389

01 배경지식	391
이클립스 플랫폼	391
워크벤치	392
확장점과 확장	394
JDT	396
SWT & JFace	397
02 HelloWorld 제작	398
플러그인 제작	398
플러그인 실행 및 테스트	403
03 PDE 오버뷰	406
호스트 & 런타임 워크벤치 인스턴스	406
외부 & 작업 공간 플러그인	407
PDE 관련 툴	407

Chapter 14 간단한 XML 에디터 제작 417

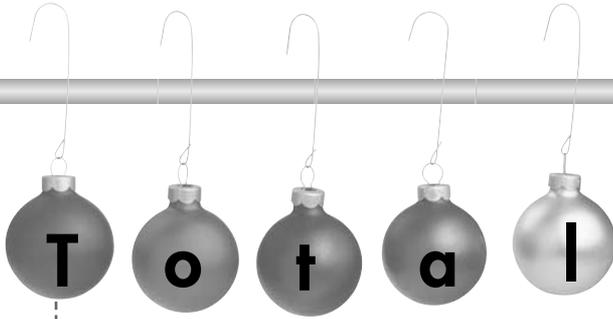
01 템플릿 활용하기(XML 에디터 템플릿 이용)	418
소스 분석	430
02 XML 문서에서 태그 이름을 굵은 글씨로 처리하기	435
03 아웃라인 보기에 아웃라인 표시	439
아웃라인 페이지 작성	439
IAdaptable 인터페이스	450
특성 보기에 XML 태그의 속성 표시하기	452
04 이클립스 워크벤치의 이벤트 처리에 대해	461



05 OLE 활용하기	463
OLE 객체 생성	463
객체 활성화 시키기	465
OLE Automation	465
인터넷 익스플로러로 XML 문서 디스플레이 하기	467
06 멀티 페이지 에디터로 확장	475
plugin.xml 수정	475
템플릿 MultiPageEditor 클래스 분석	478
MultiPageEditor 클래스 수정	481
07 환경 설정 페이지 제작하기	485
플러그인 클래스 제작	485
환경 설정 페이지 작성	489
Chapter 15 플러그인 국제화와 배포	519
<hr/>	
01 플러그인 국제화	520
프로퍼티 파일 만들기	521
Manifest 파일 국제화	525
단편 만들기	526
외부 도구 활용하기	532
02 플러그인 패키징 및 배포	536
기능 프로젝트 생성	536
기능 프로젝트에서 배포 패키지 만들기	539
온라인 업데이트 사이트	545
updatesite 프로젝트 생성	547
Chapter 16 2% 부족할 때...	551
<hr/>	
01 도움말 활용	552
02 이클립스의 소스 활용	553

Part 4 | Eclipse Project Resource

Chapter	17 이클립스 프로젝트 리소스	561
01 이클립스 프로젝트 공식 FAQ		562
살펴보기		562
시작하기		564
플러그인 개발		573
Miscellaneous		577
02 이클립스 뉴스그룹 사용하기		577
03 미리 보는 이클립스 2.2와 3.0		580
발표일정과 플랫폼		580
왜 2.2 대신에 3.0인가?		582
언어문제와 호환성		582
다음 이클립스의 새로운 기능		583
04 이클립스 관련 사이트		583

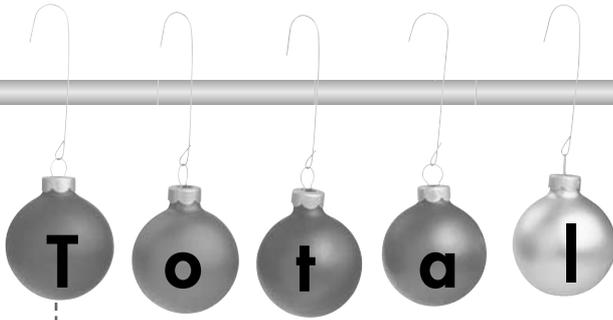


Eclipse

T o t a l E c l i p s e

Part **1**

Eclipse 프로젝트



Eclipse

T o t a l E c l i p s e

Chapter **1**

Eclipse

2001년도 eclipse.org의 시작과 함께 필자와 이클립스와의 관계도 시작되었습니다. 그 후로 가장 많이 질문 받은 것이 이클립스를 설명해 달라는 것 입니다. 저의 답은 이렇습니다.

“난 이클립스를 IDE의 미래라 부른다.”

여러분들이 이클립스를 보고 있다면 IDE의 미래와 오픈 소스 흐름의 미래를 보고 있는 것입니다.

이클립스의 시작은 OTI(Object Technology International Inc)입니다. 기술 개발이 사업 방향인 회사로 IBM에 속해 있습니다. IBM의 유명한 IDE인 Visual Age가 OTI와 IBM의 협력으로 만들어진 결과물입니다. 이클립스 또한 이들의 협력으로 만들어진 프로젝트입니다. IBM은 이클립스 플랫폼 기술을 자사의 상품(웹스피어)에 사용하고 있습니다. 우리는 이클립스 프로젝트의 성격과 의미에 대해서 알아볼 것 입니다.

01 문답으로 알아보는 이클립스 프로젝트와 이클립스 IDE

Q&A

Q→이클립스를 처음 접하는 사람입니다. 간단히 설명해 주세요.

A→이클립스 프로젝트의 결과물이 이클립스입니다. 이클립스 프로젝트는 동일한 IDE 플랫폼 위에서 IDE를 개발하여 소모적인 중복 개발을 막고, IDE 개발자들이 IDE 개발 자체에만 몰두 할 수 있도록 만든 프로젝트입니다. 기본적으로 자바 개발 환경을 지원하고 있고 성능 또한 상용 IDE 수준에 버금갑니다.

Q→이클립스 IDE는 셰어웨어(Shareware)인가요?

A→이클립스 프로젝트는 공개 프로젝트입니다. 프로젝트의 결과물인 IDE 또한 공개 프로그램입니다. 소스와 문서 모두 공개되어 있어서 직접 컴파일 하거나 다운로드 받거나 자신에게 맞게 개작해서 사용할 수 있습니다. 하지만 일부 이클립스 IDE 플러그인은 구입을 해서 사용해야 하는 경우도 있습니다.

계속 ▶

Q→이클립스 IDE로 좋은 개발을 할 수 있는지 궁금합니다.

A→이클립스에는 많은 주목을 받고 논의된 기술들이 사용되고 있습니다. JFace, JUnit, XML, ANT 등 지금의 기능성과 앞으로의 가능성을 가지고 있는 기술들입니다. 또한 팀 프로젝트에 기본이 되는 CVS도 지원합니다. IDE로서의 기능은 충분합니다. 따라서, 충분히 만들 수 있습니다. 필자 또한 여러 프로젝트를 이클립스로 개발했고, 현재도 진행하고 있습니다. 일부 회사에서는 정책적으로 이클립스 플랫폼 도입을 추진하는 것으로 알고 있습니다.

Q→IDE라는 말을 처음 듣는데 뭔가요?

A→통합 개발 환경인 'Integrated Development Environment'의 약자로 하나의 애플리케이션에서 에디팅, 컴파일, 실행, 디버깅 등 개발에 필요한 모든 것을 지원하는 환경입니다.

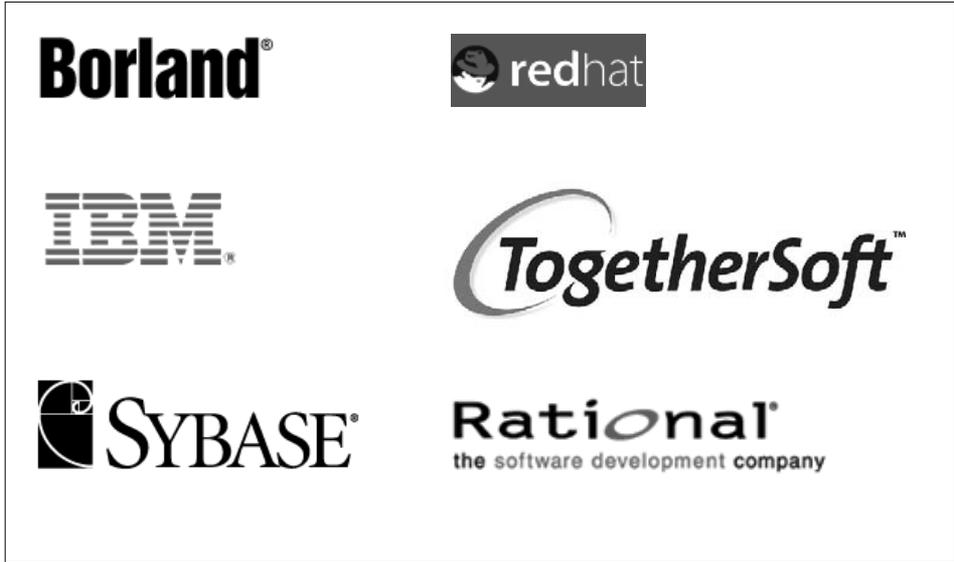
Q→에디터 플러스나 울트라 에디터를 쓰고 있는 개발자입니다. 저에게도 이클립스 IDE가 좋을 까요?

A→에디터 플러스, 울트라 에디터 모두 좋은 에디터입니다. 하지만 개발에 편리한 환경은 제공해 주지 않습니다. 예를 들어 팀 프로젝트를 하기 위해서는 별도의 CVS 클라이언트 프로그램을 설치해야 하고, 디버그 환경도 좋지 못합니다. 물론, IDE를 쓰더라도 더 편리할 뿐이지 개발자의 실력이 늘거나 하지는 않습니다. 단지 개발에 더 편리하고 실수를 줄일 수 있는 환경을 제공할 뿐입니다.

02 이클립스와 오픈 소스에 대한 필자의 생각

eclipse.org는 IDE 발전을 위한 단체입니다. 표준화 되고 검증된 플랫폼을 기반으로 그 위에서 IDE 발전을 생각합니다. MS의 비주얼 스튜디오, 볼랜드의 제이빌더, IBM의 비주얼 에이지 등 수 많은 훌륭한 IDE가 있지만 어떻게 보면 소모적인 개발을 하는 것 입니다.

예를 들어 선택스 하이라이팅 기능(의미 있는 문장을 다른 색 같은 방법으로 돋보이게 하는 기능)을 만든다고 할 때, 이 단순한 작업에도 키 입력과 입력에 따른 키워드 검색, 파서 등 수 많은 개발자의 노력이 필요합니다. 이러한 정보가 공유된다면 이들에 의해 검증되고 발전 될 수 있습니다. 이것이 바로 오픈 소스의 힘입니다.



▶ 그림 1-1 이클립스 컨소시엄 참여 대표 기업

위 회사들은 이클립스 컨소시엄에 참여하는 회사들 중 일부입니다. 그런데 왜 이클립스에 투자할까요? 볼랜드와 같이 IDE를 개발하는 회사도 있습니다. 오히려 자사 IDE 판매에 문제가 될 것이라 생각되는데 말입니다. 이들은 현명한 투자자이며 노련한 기업가입니다. 왜냐하면 이클립스 프로젝트 자체가 IDE의 미래이기 때문입니다. 이들은 이클립스 프로젝트에 투자, 참여함으로써 검증되고 강력한 기술을 마음대로 쓸 수 있습니다.

이것이 바로 오히려 자사 상품 판매에 도움이 되는 것 입니다. MS의 제품들은 매우 훌륭합니다. 하지만 소비자들은 도대체 어떤 기술이 사용되고 앞으로 발생할 문제에 대해서 전혀 예측을 할 수 없습니다. 또한 치명적인 버그로 세상을 가끔 떠들썩하게 만듭니다. 검증되지 않았거나, 충분한 검증이 안되었기 때문입니다.

03 이클립스 프로젝트 이야기

앞에서 eclipse.org에 관해 알아 보았습니다. 이번에는 이클립스 프로젝트에서 구체적으로 무엇을 하는지 이야기할까 합니다.

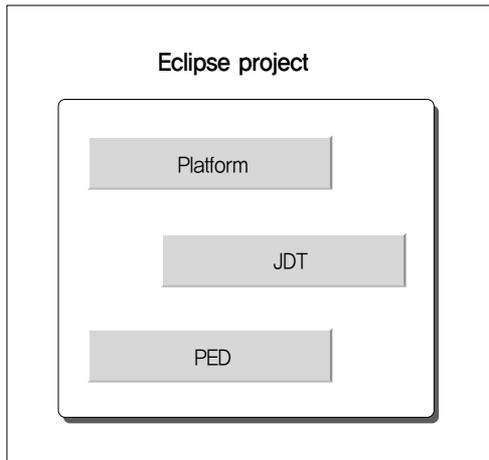


그림 1-2 이클립스 프로젝트 구성

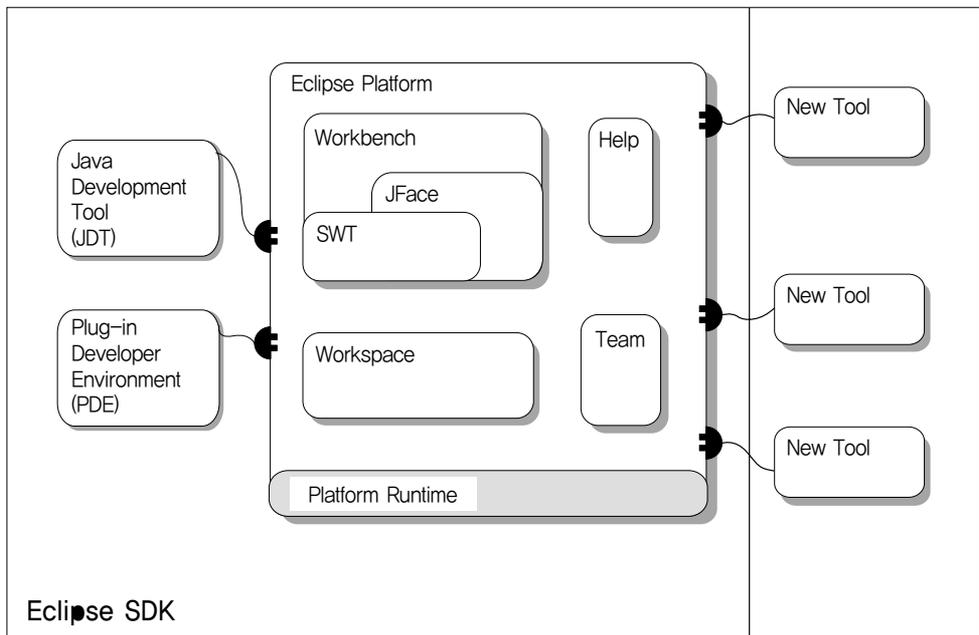


그림 1-3 Eclipse Platform 구성도

Eclipse platform

이클립스 프로젝트의 기반이 되는 프로젝트입니다. 개발자들이 IDE의 발전 자체에만 몰두할 수 있도록 IDE 개발의 표준화된 환경을 제공합니다. 차례차례 보도록 합시다. 꽤 의미 있고 중요한 부분입니다.

플랫폼 런타임(Platform Runtime)

이클립스 플랫폼 구성도(그림 1-3)의 오른쪽을 보면 New Tool들이 플러그로 연결되어 있습니다. 이 부분이 extension point(확장 포인트)로 플러그인과 플랫폼의 연결, 플러그인과 플러그인의 연결을 통해 이클립스에 기능성을 제공합니다. JDT(Java Development Tool)나 PDT(Plug-In Development Tool)도 확장 포인트를 통해 플랫폼에 연결된 플러그인에 불과합니다. 확장 포인트의 사용은 이클립스 SDK API에서 제공합니다. 이런 API 문서를만 봄으로써 쉽게 사용하고 플랫폼 소스 코드를 분석해서 IDE의 기반을 튼튼히 하고 더 유연하게 만들 수 있습니다.

워크벤치(Workbench)

이 부분을 통해 이클립스 IDE에 표준화된 인터페이스를 제공합니다. IDE에서 실제 사용자가 접하는 부분입니다. 메뉴, 구조 뷰 창, 편집 창, 상태 창, 메시지 창, 단축키 등을 지원 합니다. 이클립스에서 제공하는 창과 뷰 등 UI(User Interface)의 표준입니다.

JFace/SWT

구성도의 워크벤치 부분을 자세히 보면 워크벤치에 SWT와 JFace가 있고 SWT가 JFace 앞에 있지만 둘 사이가 포함관계가 아니라는 것을 알 수 있습니다. SWT(Standard Widget Toolkit)는 그래픽 라이브러리입니다. SWT는 우리가 실제 사용하고 보는 부분이지만 JFace는 UI에서 발생하는 상황(입력 발생, 뷰 창의 변화)들을 처리하기 위한 툴킷입니다.

워크스페이스(Workspace)

워크스페이스는 자원(프로젝트, 파일, 폴더) 관리를 담당하는 부분입니다.

도움말(Help)

로컬시스템의 파일이나 온라인 문서와 IDE 간의 연결을 제공합니다.

VCM(Version Configuration Management)

팀 프로젝트 작업을 위한 파일 동기화, 업데이트 등을 지원합니다. 보통 이 부분에 많이 사용되는 CVS를 지원합니다.

JDT(Java Development Tools)

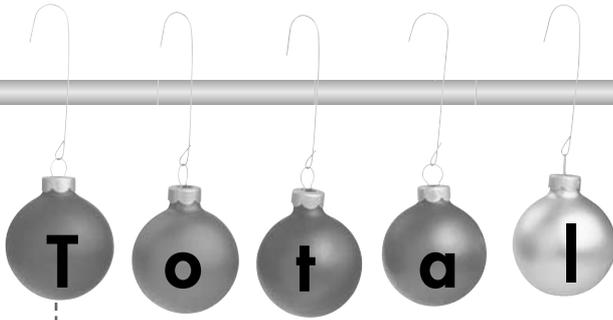
이클립스를 이용한 자바 개발 환경 프로젝트입니다. 물론 이클립스는 특정 컴퓨터 언어나 국가, 환경을 위한 프로젝트가 아닙니다. 이미 CDT를 통해 C/C++가 지원되고 있고 많은 다른 언어들도 논의되고 있습니다.

JDT 구성

- **JDT Core** : UI를 제외한 부분의 기반으로 컴파일, 자바 요소들의 트리 구성(검색과 코딩을 돕기 위한), 리팩토링 등을 지원합니다.
- **JDT UI** : 이클립스 IDE를 통해 실제 접하는 부분으로 코드 편집을 기본으로 코딩을 위한 모든 지원(신택스 하이라이팅, 클래스의 구조화된 뷰)를 지원합니다.
- **JDT Debug** : 자바 코드 디버그를 위한 부분으로 실행, 디버깅, JVM Stack 접근, 다이내믹 클래스 리로딩(디버그 중에 코드 편집 바로 적용을 위한) 등을 지원합니다.

PDE(plugin-in development environment)

이클립스의 성능과 가능성을 무한히 만들어줄 프로젝트입니다. 이클립스에 플러그인 개발 환경 프로젝트로서 이클립스를 위한 플러그인을 쉽게 개발할 수 있는 환경을 지원합니다. JDT나 PDT(플러그인 개발툴)도 단순히 플러그인에 불과합니다. 실제 이클립스에 기능을 부여하는 부분입니다.



Eclipse

T o t a l E c l i p s e

Part **2**

SWT



Eclipse

T o t a l E c l i p s e

Chapter **6**

SWT
(Standard Widget Toolkit)

SWT는 이클립스 IDE에 사용된 그래픽 툴킷입니다. 여러 장점을 가지고 있는 현존하는 최고의 자바 그래픽 툴킷입니다. 우리는 SWT를 사용하기 위해 SWT의 구조와 원리 그리고 사용법을 익힐겁니다.

SWT의 역사는 AWT나 Swing에 비해서 매우 짧은편입니다. 여러분들은 아직 SWT와 기존의 AWT, Swing의 차이를 모를 것입니다. 우리는 앞으로 이들의 장점과 단점은 알아보고 왜 SWT를 선택해야 하는지 알아볼겁니다.

01 문답으로 알아보는 SWT

Q&A

Q→SWT도 자바처럼 한번의 코딩으로 모든 플랫폼에서 사용할 수 있나요?

A→할 수 있습니다. 엄밀히 말하면 SWT 라이브러리가 지원되는 플랫폼에서 가능합니다. 현재 자바를 사용할 수 있는 플랫폼의 종류와 비슷합니다(임베디드 기기 포함).

Q→SWING이나 AWT도 모르는데 SWT를 배워야 하나요?

A→SWT는 현존하는 최고의 자바 그래픽 라이브러리입니다. 빠른 GUI를 원한다면 익히는 것이 좋습니다. 하지만 자바를 공부하는 분들보다는 자바 애플리케이션 개발자들에게 권합니다.

Q→Swing과 AWT를 공부해본 사람입니다. SWT 공부하는데 도움이 될까요?

A→SWT의 기본 개념은 Swing과 AWT와 통하는 곳이 많습니다. Swing과 AWT를 해보셨다면 아주 쉽게 접근할 수 있습니다.

Q→임베디드 기기에서의 SWT 전망을 어떻게 보시나요?

A→SWT는 단일 레이어를 통해 자원을 적게 사용합니다. 자원이 제한적인 임베디드 기기에서도 좋은 결과를 보여 줍니다.

02 SWT에 대해서

자바를 배우기 시작해서 콘솔 환경을 넘어설 무렵 AWT와 Swing이라는 단어가 다가옵니다. 둘 다 자바의 GUI 방식입니다. AWT의 경우 자바 탄생 무렵에 나왔고 Swing은 후반에 나왔습니다. 여기에서 살펴볼 내용은 AWT, Swing, SWT의 비교와 설명입니다.

AWT(Abstract Windowing Toolkit package)

AWT는 해석하면 추상적 윈도우 툴킷 패키지가 됩니다. 프로그래밍 언어에서 쓰는 추상적이라는 의미는 메소드는 정의되어 있지만 실체가 없는 객체 즉, 실제로 구현해 주어야만 쓸 수 있는 객체를 뜻합니다. 그렇다고 AWT를 사용하기 위해 수많은 API를 실제로 구현해 주어야 하는 것은 아니고 운영체제마다 추상화 부분이 다르게 구현되어 있습니다.

여기에서 사용하는 구현이라는 단어의 뜻은 형과 파라미터만 선언되어 있는 객체를 실제 코딩한다는 의미입니다. AWT는 자바 코드를 해석해서 운영체제의 API를 이용해 화면에 GUI를 보여 줍니다. 따라서 모습이 운영체제에 따라 달라집니다.



그림 6-1 윈도우에서의 자바 애플릿



그림 6-2 리눅스에서의 자바 애플릿



그림 6-3 매킨토시에서의 자바 애플릿

이것은 운영체제를 반영하는 것으로 사용자에게는 친근한 인터페이스를 제공한다는 이점이 있습니다. 하지만 AWT는 근본적인 문제가 있습니다. 자바 코드를 운영체제에 전달하는 과정에서 피어(Peer)를 사용합니다.

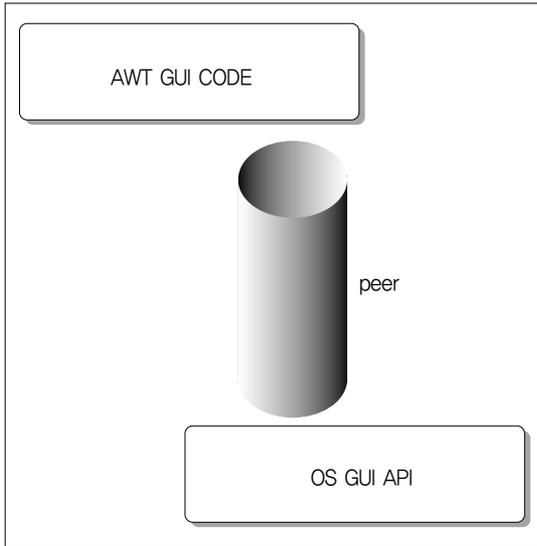


그림 6-4 AWT가 GUI를 보여주는 방식

모든 GUI적 이벤트들(키눌림, 마우스, 화면 변경 등)이 모두 이 피어를 거치게 되므로 비효율적이 됩니다. 피어는 크게 두 가지의 역할을 합니다. 첫 번째는 AWT와 지역 운영체제(어떤 운영체제도 될 수 있음)의 GUI 시스템과의 연결. 두 번째는 비슷한 폼의 GUI 형성을 위해서입니다. 여러 운영체제의 AWT에서 보다시피 보여지는 것이 다릅니다. 하지만 유저들은 동일하거나 비슷한 형태의 GUI를 원하기 때문에 피어를 통해 그 차이를 조정합니다. 위의 AWT의 특성을 종합해 보면 AWT는 운영체제에 의존적이고 유연성이 없다는 결론을 내릴 수 있습니다.

Swing

Swing은 자바2 플랫폼부터 시작된 GUI 컴포넌트입니다. AWT와 크게 대조되는 부분은 Swing의 경우는 운영체제에 의존적이지 않습니다. 운영체제는 단지 Swing 코드를 해석해서 그려 줍니다(AWT의 경우 운영체제의 GUI 자체를 사용합니다). 따라서 모든 플랫폼에서 동일한 GUI를 볼 수 있습니다.

또한 장점으로는 룩앤필(Look & Feel)의 변화 지원입니다. Swing 자체적으로 운영체제의 GUI를 모방해서 보여줌으로써 GUI의 모습(Look)과 GUI의 조작(Feel)을 바꿀 수 있습니다.

AWT의 경우 운영체제에 의존적이기 때문에 어쩔 수 없이 모습이 운영체제에 따라 다르지만, Swing에서 GUI의 모습은 단지 선택 사항입니다. 문제는 바로 모두다 Swing에서 그린다는데 있습니다. 결국 운영체제의 GUI도 그래서 보여주는 것이 아니냐고 반문할 수 있지만 방식의 차이가 있습니다.

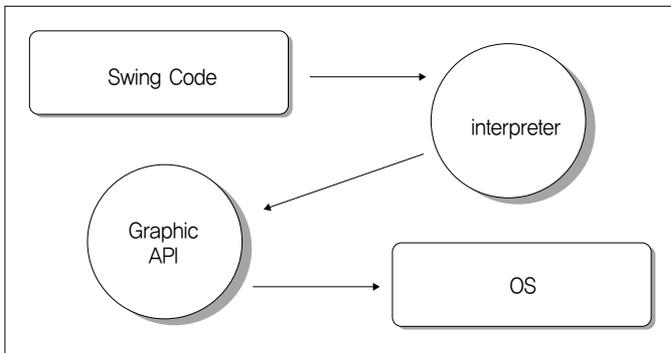


 그림 6-5 Swing이 GUI를 보여주는 방식

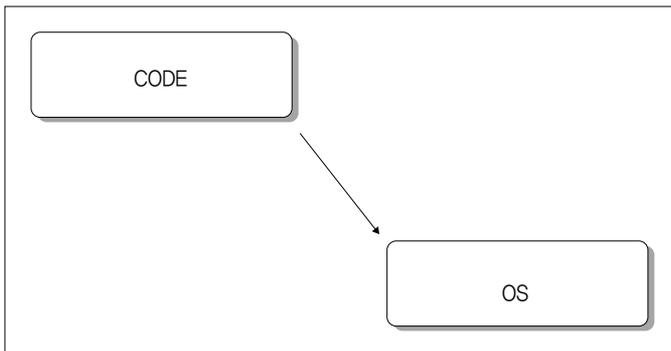
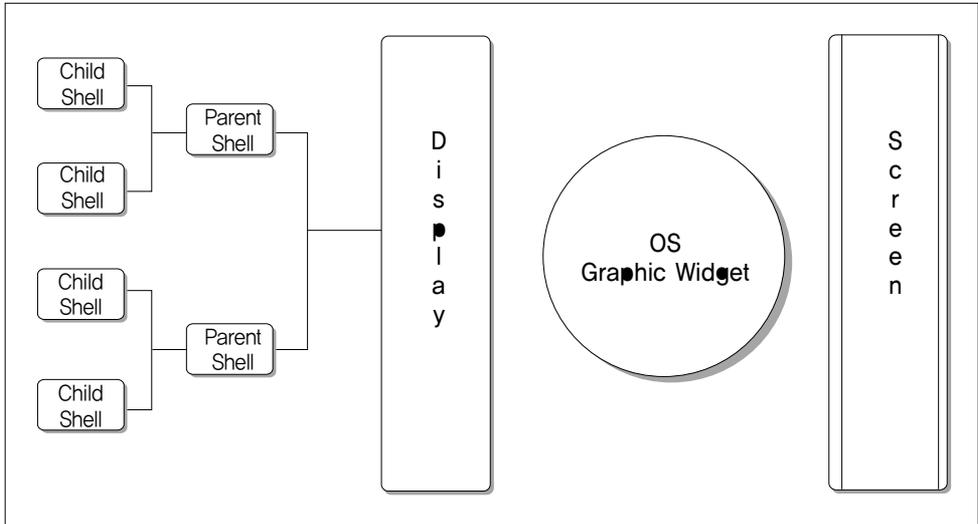


 그림 6-6 플랫폼 전용 코드가 GUI를 보여주는 방식

Swing은 그림 6-5처럼 해석기를 통해 어떻게 그릴지 결정하고, 운영체제의 그래픽 API를 통해 화면에 그려 줍니다. 이렇게 여러 단계를 거치는 동안 Swing 애플리케이션의 성능은 떨어집니다. 플랫폼 전용 코드도 당연히 화면에 그려주는 작업이 있겠지만 운영체제에 최적화된 GUI API를 통해 화면에 그려줌으로써 매우 빠르게 작동됩니다.

SWT



▶ 그림 6-7 SWT 구조

이제 SWT의 장점을 살펴 보겠습니다. 성능을 비교하자면 AWT에 비해 빠릅니다. AWT가 컴포넌트의 제어를 피어를 통해 하는 반면에 SWT는 하나의 레이어로 모든 작업을 처리합니다.

따라서 속도와 메모리 절약의 효과가 있습니다. 자바 코드처럼 한번 SWT로 코드를 만들면 SWT 라이브러리가 설치되어 있는 어느 플랫폼 위해서 실행이 가능합니다.

	이식성/표현능력	속도	메모리
AWT	중간	중간	중간
Swing	좋음	느림	많이 소모
SWT	좋음	빠름	적게 소모

▶ 표 6-1 AWT, Swing, SWT 비교표



위 비교표는 상대적으로 보십시오



또한 ActiveX 같은 운영체제에 의존적인 컴포넌트도 제어할 수 있습니다. SWT 속도의 비결은 하나의 레이어 구성과 직접 운영체제를 제어해서 화면에 그린다는 점입니다. 스윙이 코드를 해석해서 운영체제의 그래픽 API를 이용해 그리는 반면에 SWT는 운영체제가 지원하는 라이브러리(native interface)가 있다면 그 라이브러리를 사용하고 없다면 SWT 라이브러리를 통해 직접 운영체제 그래픽 장치를 이용해 그립니다.

즉, SWT AWT의 장점과 Swing의 장점을 결합한 훌륭한 라이브러리입니다. 또한 자신만의 룩 앤 필 제작도 가능합니다. 자바 코드의 범용성, 메모리 절약, 퍼포먼스때문에 임베디드(PDA, 휴대폰 등) 장치에도 적합합니다.

03 SWT에 대한 필자의 생각

앞서 AWT, Swing, SWT에 대해 알아보고 비교했습니다. 이번에는 필자의 생각을 다뤄봅니다. SWT가 지금의 GUI 방식 중 최고의 방식인 것은 사실이지만 알아야 할 것이 있습니다.

SWT는 자바 표준이 아니다

자바의 표준 GUI는 AWT와 Swing입니다. SWT는 서드파티 그래픽 라이브러리(3rd party Graphic Library: 표준이 아닌 외부에서 제작된)입니다. 따라서 앞으로의 자바 정책에 SWT는 고려 대상이 아닙니다.

이 문제로 자바가 바뀌면 SWT도 거기에 맞추어 바뀌어야 합니다. 따라서, 버전이 바뀌는 시점 1.3 → 1.4 → 1.5 사이에서 SWT 개발자는 JRE 모듈과 SWT 라이브러리가 만들어질 때까지 기다려야 합니다. 또한 이 과정에서 불가피하게 고유의 SWT 형태가 아닌 다른 형태로 바뀔 수 있습니다. 이전 버전에서의 소스 사용 문제와 이미 제작된 애플리케이션 모두 문제를 일으킬 수 있습니다.

새로운 기술 습득의 부담

지금도 계속 SWT에 대해 공부하고 있지만 Swing이나 AWT만큼 자료가 풍부하지도 않고 사용자도 적어서 어려움이 많습니다. 또한 API 사용도 기존의 다른 GUI API와 비슷하지만 다른 점도 많습니다. 빈약한 자료와 다른 방식 모두 부담이 됩니다. 물론 이러한 현상은 기술 개발 초기에는 당연한 현상입니다.

마이크로소프트의 게임/멀티미디어 그래픽 툴킷인 DirectX가 처음 소개되었을 때 많은 국내 개발자들이 어려움을 겪었습니다. 하지만 1년이 지나고 외국서적을 능가할 정도의 도서도 출판 되었습니다. SWT도 이러한 문제가 있습니다.

배포의 문제

SWT로 만든 애플리케이션을 사용하기 위해서는 SWT 라이브러리 설치가 필요합니다. 기존의 AWT나 Swing의 경우는 JRE가 모두 알아서 해주어 이러한 문제가 없지만, SWT의 경우 반드시 설치 프로그램으로 이러한 과정을 자동으로 해주는 것이 필요합니다. 물론 다른 자바 프로그램과 마찬가지로 JRE도 없을 수 있다는 가정 아래 JRE 설치도 포함해 주어야 합니다.

Low Level API 부족

SWT를 이용해서 일반적인 애플리케이션 제작에는 문제가 없지만 섬세한 애플리케이션이나 특별한 GUI 제작 시에 필요한 로우레벨 API(Low Level API: 저수준 API, 하드웨어 제어에 가까운)가 부족해서 다시 이 부분을 JNI를 통해서 구현해주어야 하는 경우도 있습니다. 다행히 SWT 업데이트가 상당히 빨리 이루어지고 있고 이 문제는 조만 간에 해결되리라 생각합니다.

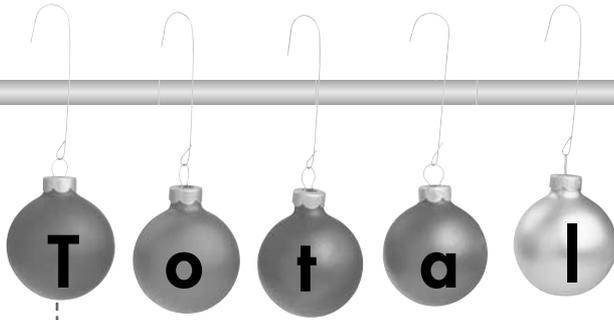


예외 처리

이 부분은 단점일 수도 있고 장점일 수도 있습니다. Swing, AWT의 경우 발생 가능한 예외 처리를 꼭 하도록 되어 있습니다. 하지만 SWT의 경우 그렇지 않습니다. 컴파일만 되었어도 실행도중 치명적 예외사항 발생으로 프로그램이 종료될 수 있습니다.

Swing, AWT는 컴파일만 가능하면 매우 완벽한 상태의 애플리케이션을 만들 수 있습니다. 하지만 SWT의 경우는 예외의 경우를 생각해서 처리를 해주어야 합니다. 불필요한 예외처리를 생략함으로써 성능 향상을 얻을 수도 있습니다.

이같은 문제가 있지만 그래도 SWT는 현재 최고의 자바 GUI 입니다. 소비자들은 기다려주지 않습니다. 빠르고 강한 애플리케이션을 원합니다. 그렇다면 지금 선택은 SWT 밖에 없습니다.

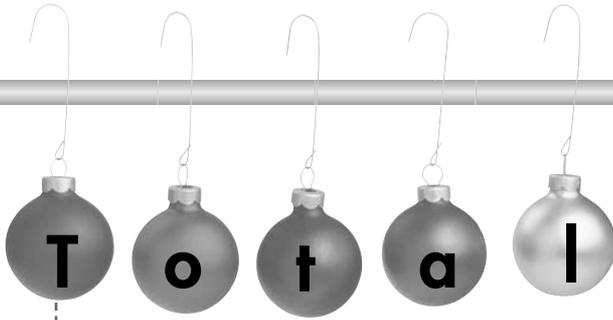


Eclipse

T o t a l E c l i p s e

Part **3**

플러그인 개발



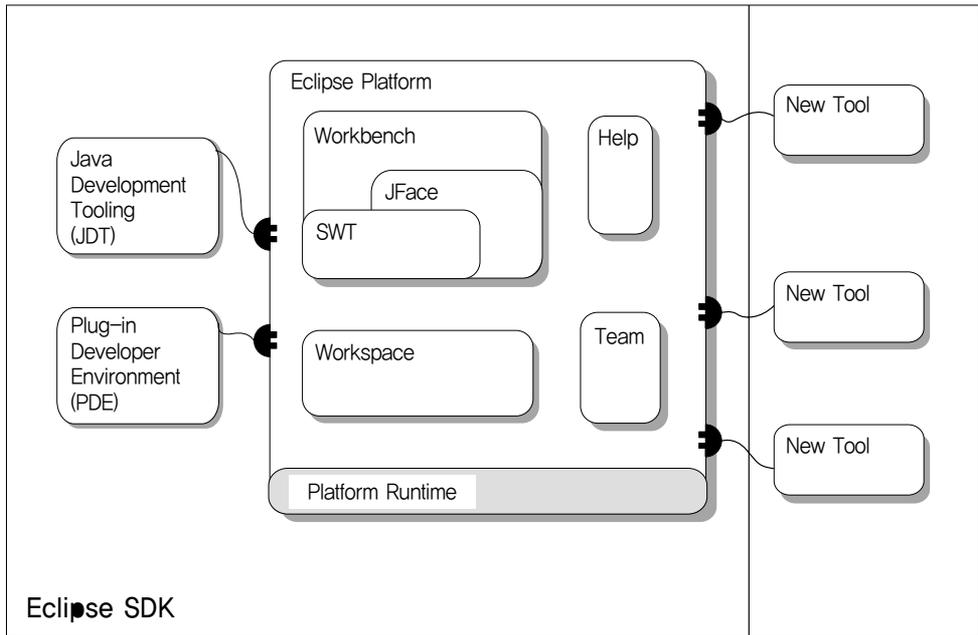
Eclipse

T o t a l E c l i p s e

Chapter 13

PDE

이클립스(Eclipse)는 독립적으로 개발된 툴들을 통합할 수 있도록 플랫폼의 API와 소스를 공개하고 있습니다. 현재 이클립스의 주요 툴 중 하나라고 할 수 있는 JDT도 이클립스 플랫폼 위에서 개발된 플러그인이라고 할 수 있습니다.



▶ 그림 13-1 이클립스 아키텍처(출처 : eclipse.org)

그림 13-1에서 볼 수 있듯이 이클립스 플랫폼은 통합에 필요한 UI, 도움말, 자원 관리, 팀 작업 등의 모델과 API를 제공합니다. 그리고 실제 필요한 도구는 플러그인 시스템을 통해 제작, 제공되는 구조입니다.

결국 이클립스의 모든 기능은 플러그인을 통해 제공된다고 할 수 있습니다. 이클립스 설치를 위해 제공되는 SDK에는 이클립스 플랫폼을 포함해 기본적인 플러그인과 JDT 등이 함께 제공됩니다. 플러그인의 개발을 돕기 위해 이클립스는 PDE(Plug-in Development Environment)를 SDK에 포함하고 있습니다. PDE를 이용하면 이클립스 플러그인을 빠르고 쉽게 만들 수 있습니다

이클립스 플랫폼은 공개되어 있고 확장이 가능하기 때문에 현장에서 새로운 도구에 대한

필요가 생길 때마다 빠른 시간 안에 플러그인 제작이 가능합니다. 이렇게 필요에 따라 개발된 플러그인은 통합 개발 환경으로 개발자에게 제공됩니다.

또한 현재까지 수많은 공개 플러그인들이 있기 때문에 새로운 플러그인 개발에 참조할 소스가 많아 플러그인 개발이 무척 쉽습니다. 이 점은 이클립스를 사용할 때 얻을 수 있는 큰 장점이라고 할 수 있습니다.

참고로 이클립스 플랫폼은 통합을 5단계로 구분해 각 단계별 통합 방법을 제공하고 있습니다. 보다 자세한 사항은 'Eclipse Article'의 'Level of Integration' 문서를 참조 바랍니다.

01 배경 지식

이 파트에서는 플러그인 개발에 필요한 배경 지식에 대해 살펴 보고 간단한 예제를 통해 이클립스의 각 요소와 API에 대해 설명하겠습니다. 먼저 배경 지식에 관해 살펴 보겠습니다.

Note

이클립스는 상당한 수준의 도움말을 제공하고 있습니다. 하지만 처음 플러그인을 제작하고자 하는 사람에게는 다소 어려울 수 있습니다. 예제 제작을 마치고 도움말을 다시 살펴본다면 도움말의 내용에 대한 이해가 쉬워질 것입니다.

이클립스 플랫폼

이클립스 플랫폼은 워크벤치(Workbench) UI와 IDE(Integrated Development Environment)를 구성하는데 필요한 라이브러리 및 시스템의 모음입니다.

즉, 그림 13-2와 같이 런타임 라이브러리와 자원 관리 시스템(Workspace), 도움말 시스템(Help), 팀 작업 지원(Team), 그리고 IDE의 외형인 워크벤치를 포함합니다.

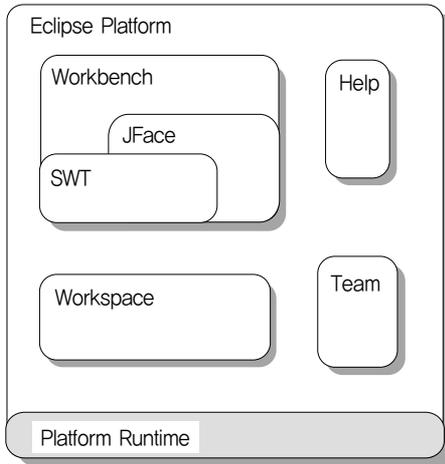


그림 13-2 이클립스 플랫폼(출처 : eclipse.org)

이클립스 플러그인은 플랫폼의 지원을 받아 플랫폼 및 다른 플러그인의 기능을 지원받습니다. 플랫폼은 플러그인과 플랫폼 혹은 플러그인 간의 통합을 이룹니다. 그리고 서로 다른 운영체제 지원 등을 보장합니다(서로 다른 운영체제 지원은 플랫폼이 담당합니다).

예를 들어 팀 작업의 경우 이클립스 플랫폼은 다른 워크스페이스와 팀 작업 모델 간의 통합 API를 제공합니다. 그리고 팀 작업 플러그인 제공에 필요한 API들을 제공하게 됩니다. 팀 작업 플러그인은 CVS, SourceSafe 등 팀 작업 공간의 종류에 따른 특성에 맞춰 필요한 내용을 구현을 해주게 됩니다.

워크벤치

워크벤치는 이클립스를 실행시켰을 때 나타나는 창입니다. 워크벤치는 화면과 키보드 마우스를 통해서 사용자의 입력을 받고 사용자의 명령에 대한 반응을 나타냅니다. 만약 우리가 제작하는 플러그인이 비주얼 인터페이스를 가지고 있다면 UI 측면에서 워크벤치와 일관성 있게 제작되는 것이 좋습니다. 보다 자세한 사항은 'Eclipse User Interface Guidelines' 문서를 참고 바랍니다.

플러그인 제작을 위해서는 워크벤치에 대해 확실하게 이해하는 것이 좋습니다. 그림 13-3은 워크벤치의 구성을 나타내는 그림입니다.

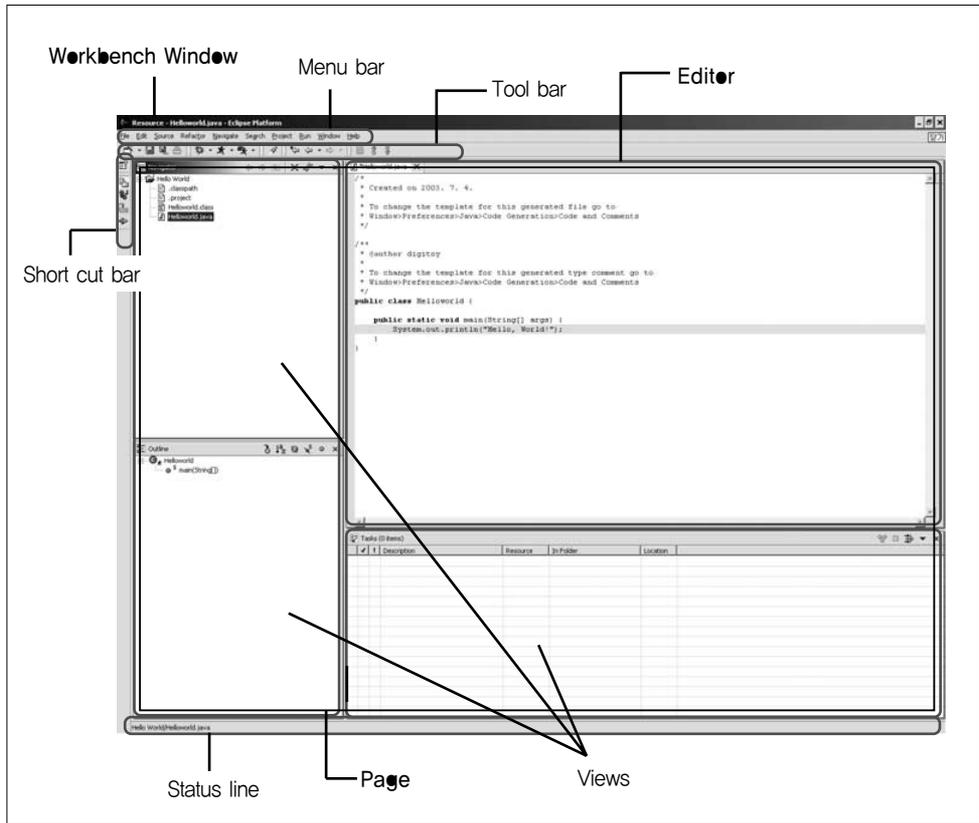


그림 13-3 워크벤치 구성(출처 : eclipse.org)

이러한 워크벤치의 구성 개념은 화면 분할 뿐만 아니라 API에도 반영되어 있습니다. 예를 들면 워크벤치 윈도우는 IWorkbenchWindow 인터페이스를 구현한 클래스의 인스턴스입니다.

워크벤치 윈도우

워크벤치 윈도우는 IWorkbenchWindow 인터페이스를 구현한 클래스의 인터페이스로 메뉴 표시줄, 도구 모음, 상태 표시줄, 바로 가기 막대 및 페이지 등이 있는 프레임입니다.

페이지

IWorkbenchPage를 구현한 클래스로 워크벤치 구성 요소(IWorkbenchPart)들의 논리적인 묶음을 나타냅니다. 페이지는 에디터들과 뷰에 대한 정보들을 포함하고 있으며 페이지를 통해 에디터와 뷰에 접근이 가능합니다.

에디터

IEditorPart 인터페이스의 추상 구현인 EditorPart 클래스의 구체적인 구현입니다. 에디터는 문서나 입력 오브젝트를 편집하거나 찾아보기 위해 사용됩니다. 에디터에서의 수정은 외부 파일 시스템 에디터처럼 '열기-저장-닫기' 모델을 따릅니다. 플랫폼 문서 에디터와 자바 에디터는 워크벤치 에디터의 예입니다.

뷰

IViewPart 인터페이스의 추상 구현인 ViewPart 클래스의 구체적인 구현입니다. 뷰는 정보의 계층 구조를 탐색하거나, 에디터를 열거나, 활성 에디터의 특성을 표시하는데 사용됩니다. 예를 들어 네비게이터 보기를 사용하면 작업 공간 계층 구조를 탐색할 수 있습니다. 특성 및 아웃라인 보기에서는 활성 에디터의 객체에 대한 정보를 표시합니다.

확장점과 확장

확장점(Extension Point)은 이클립스에서 가장 중요한 개념입니다. 이클립스는 이 확장점 개념에 바탕을 둔 플러그인들의 집합이라 할 수 있습니다. 앞서 이클립스 플랫폼에 대해서 설명을 했습니다. 이클립스 플랫폼은 하나의 완벽한 구현일 수도 있지만 대개는 모델을 정의하고 기본적인 몇 가지 기능을 구현한 후 확장점을 제공하게 됩니다. 확장점을 확장해 추가적인 기능을 제공하는 것을 확장(Extension)이라고 합니다.

예를 들면 워크벤치의 경우 기본적인 에디터인 텍스트 에디터를 구현해 제공하고 있지만 에디터에 대한 확장점(org.eclipse.ui.editors)을 제공합니다. 플러그인에서는 .java 파일을 편집할 수 있도록 자바 에디터의 기능을 구현해 이 확장점에 통합하는 것을 확장을 한다고 하는 것입니다.

이러한 내용은 plugin.xml이라는 플러그인 Manifest 파일에 기재를 하고 이클립스 플랫폼이 이 내용을 해석해 자연스럽게 이클립스 워크벤치와 통합하게 됩니다.

다음 코드는 plugin.xml Manifest 파일의 일부입니다. 이 내용을 보면 org.eclipse.ui.editors 확장점에 기능을 추가해 확장하고 있으며 파일 확장자가 .xml인 파일에 대해서

“net.jlab.example.editors.XMLEditor” 클래스에서 에디터 기능을 구현하고 있음을 이클립스 플랫폼에 지시하고 있음을 알 수 있습니다.

```
<extension
    point="org.eclipse.ui.editors">
    <editor
        name="JLab XML 편집기"
        icon="icons/sample.gif"
        extensions="xml"

        contributorClass="org.eclipse.ui.texteditor.BasicTextEditorActionContributor"
        class="net.jlab.example.xmleditor.editors.XMLEditor"
        id="net.jlab.example.xmleditor.editors.XMLEditor">
    </editor>
</extension>
```

확장점은 이클립스 플랫폼 혹은 플러그인에서 기능을 추가할 수 있는 부분에 대한 정의입니다. 그리고 이렇게 정의된 확장점에 기능을 추가하는 것을 확장이라고 합니다. 위의 예에서 보면 워크벤치가 파일에 따른 에디터 기능을 추가할 수 있도록 “org.eclipse.ui.editors” 확장점을 정의하며 플러그인은 이를 확장해 따라 자바 파일, XML 파일에 대한 편집 기능을 제공하는 것입니다.

그림 13-4에서 보면 JDT는 플랫폼의 “org.eclipse.ui.newWizards” 확장점을 확장해 자바 프로젝트 작성 마법사, 클래스 작성 마법사, 인터페이스 작성 마법사 기능을 추가하게 됨을 알 수 있습니다. 또한 JDT는 ‘코드 컨벤션 룰(code convention rule)’에 따라 코드를 형식화 할 수 있도록 “org.eclipse.jdt.core.codeFormatter” 확장점을 제공합니다.

이에 따라 JDT 플러그인은 자바 에디터에서 Format 메뉴나, 명령 버튼이 눌리졌을 때 플러그인에서 정의한 코드 컨벤션 룰, 예를 들어 인스턴스 변수(Instance Variable)는 ‘i’로 시작한다는 등의 코드 작성 형식을 변경할 수 있습니다.

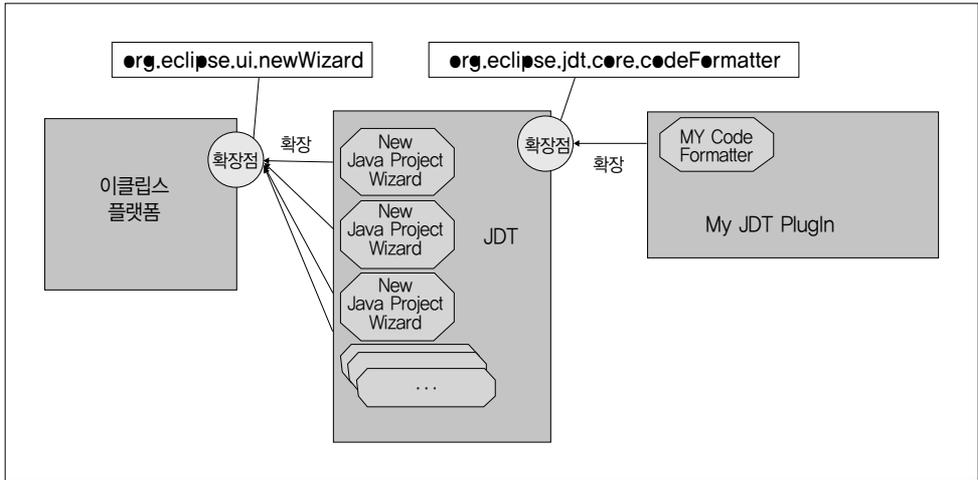


그림 13-4 확장점과 확장

현재 우리가 사용하는 이클립스는 이렇게 확장점을 확장해 기능을 구현한 플러그인들의 모음이라고 할 수 있습니다.

Note

개인적으로 이클립스가 마치 레고 블록 같다는 생각을 합니다. 레고 블록은 큰 판 위에 블록을 쌓아서 집도 만들고 성도 만들며 자동차 같은 것도 만듭니다. 이클립스도 플랫폼이라는 커다란 판 위에 SWT와 JFace와 같은 API 블록을 이용해 플러그인(집과 성)을 만드는 것과 같습니다. 레고 블록으로 만든 집이나 성은 다른 사람이 자기 기호에 따라 더 꾸밀 수가 있습니다. 이를테면 성 위에 깃발을 달거나 정원을 수정하거나 할 수 있습니다. 이처럼 이클립스의 플러그인도 플러그인의 형태를 변경하거나 더 꾸밀 수 있는 확장점에 확장을 해 플러그인의 형태를 변화시킬 수가 있는 것입니다. 확장점은 집이나 성에 사용된 레고 블록의 튀어나온 부분 중에 특별히 다른 블록을 추가, 변경할 수 있는 부분을 알려주는 설명서와 같은 것입니다.

JDT

JDT(Java Development Tooling)는 PDE와 함께 이클립스 SDK에 포함된 주요 플러그인 중 하나로 자바 개발을 지원하는 툴입니다. JDT는 자바 코드를 편집, 보기, 컴파일, 디버깅 및 실행하기 위한 기능을 제공하도록 워크벤치를 확장하고 있습니다.

플러그인의 코드는 자바 언어로 작성되도록 되어 있기 때문에 JDT는 플러그인 개발에 꼭 필요한 툴입니다.

JDT의 사용법은 이클립스 사용법과 함께 플러그인을 개발할 때 잘 익혀두어야 합니다. 제대로 익혀두면 코드 생산성 향상에 도움이 될 것입니다.

또한 JDT는 소스가 공개되어 있어 플러그인의 예제로도 활용할 수 있습니다. 'Eclipse Platform Technical Overview' 문서에서는 JDT를 케이스 스터디로 소개하면서 전체적인 구조와 플랫폼과의 연계성에 대해서 설명하고 있으니 참조 바랍니다.

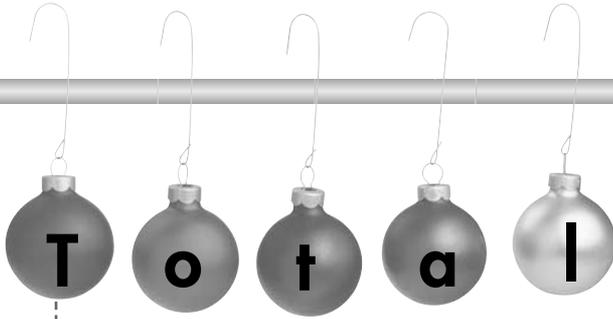
SWT & JFace

SWT(Standard Widget Toolkit)는 이클립스에서 제안된 새로운 그래픽 라이브러리입니다. 이클립스 플랫폼은 SWT와는 관계가 없습니다. 하지만 워크벤치 구현에 있어서 SWT에 의존하고 있습니다. 워크벤치의 부분을 SWT Widget을 이용해 화면에 그림을 그리거나 표시기를 작동시키고 있습니다. 워크벤치 코드의 상당 부분은 SWT의 Widget을 사용하고 있습니다.

JFace는 'User Interface Framework' 라고 이야기합니다. Swing이 MVC 모델을 이용해 GUI 프로그래밍시에 자주 사용되는 컴포넌트를 제공하듯이 이클립스에서는 JFace 프레임워크를 통해 이러한 컴포넌트를 제공합니다. JFace의 목표는 플러그인 제작에 필요한 부분을 제공해 제작 시간 및 노력을 줄이는데 있습니다.

Swing이 UI 프로그래밍에서 일반적이고 공통적인 문제 해결에 초점을 맞추었다면 JFace는 플러그인 개발시 발생할 수 있는 문제 해결에 초점을 맞추고 있습니다. 따라서 JFace는 우리가 Swing에서 처럼 기대했던 컴포넌트가 없을 수가 있습니다. 또 역으로 플러그인 개발을 위해서 한 부분에서만 사용할 수 있는 굉장히 용도가 협소한 컴포넌트를 찾을 수가 있습니다.

이클립스 개발자들은 워크벤치 API를 JFace에 중립적이 되도록 설계하려고 노력했다고 합니다. 하지만 실질적인 구현에 있어서는 JFace를 적극 활용하고 또 한편으로는 JFace에 의존적인 부분도 있습니다. 플러그인 개발자들은 JFace를 이용해 플러그인을 효율적으로 개발할 수 있을 것입니다.



Eclipse

T o t a l E c l i p s e

Part **4**

Eclipse Project Resource



Eclipse

T o t a l E c l i p s e

Chapter 17

이클립스 프로젝트 리소스

이 책을 읽으면서 중간 중간 궁금했던 부분들이 많았을 것입니다. 이 장에서는 이클립스를 처음 접하는 독자들이 궁금증을 가질 만한 내용들을 eclipse.org에서 정리해 놓은 내용을 소개하겠습니다. 그리고 장의 후반부에는 이클립스를 마스터 하기 위한 필수 항목인 뉴스레터 이용에 관한 부분을 소개했습니다. 많은 도움이 되길 바랍니다.

01 이클립스 프로젝트 공식 FAQ

살펴보기

Q&A

Q→이클립스 프로젝트란?

A→이클립스는 오픈 소스 프로젝트로 고도로 통합된 툴 개발을 목적으로 하고 있습니다. 이 프로젝트는 플랫폼, JDT, PDE라는 서브 프로젝트로 나눌 수 있습니다. 이클립스 플랫폼의 성공 여부는 툴 제작자 또는 업체들이 얼마나 자발적으로 참여할 것인가에 달려있습니다. 이클립스 프로젝트의 과제는 제작자와 사용자의 요구를 모두 만족시킬 수 있도록 이클립스 플랫폼의 완성도를 높이면서 각종 개발 툴들을 통합하는 것입니다.

Q→이클립스 플랫폼이란?

A→이클립스는 자바 기반의 개방적이고 확장성을 지닌 통합 개발 환경입니다. 이클립스 플랫폼은 통합 소프트웨어 개발툴을 만들고 실행할 수 있는 기반을 제공합니다. 이클립스 플랫폼은 툴 개발자들이 다른 개발자들이 사용하는 툴을 통합하는데 있어 독립적인 개발을 보장합니다.

Q→이클립스 SDK란 무엇인가요?

A→이클립스 SDK(Software Developer Kit)는 세 가지 프로젝트가 합쳐져 있습니다(Platform, JDT, PDE). 이 세 프로젝트의 통합으로 개발자들은 자유롭게 이클립스 플랫폼을 구성할 수 있는 개발 환경을 제공받습니다.

계속 ▶



Q→어떻게 이클립스 SDK에 관한 권리를 얻나요?

A→이클립스는 오픈 소스 프로젝트의 조합으로 이루어 졌습니다. 이클립스 프로젝트로 만들어지는 소프트웨어는 CPL(Common Public License - <http://www.eclipse.org/legal/cpl-v05.html>)을 따릅니다. 다른 컴포넌트들은 해당 컴포넌트가 허용하는 기능적, 라이선스적인 범위 안에서 활용이 가능합니다. 이클립스 사용자 동의서는 'www.eclipse.org/legal/notice.html'에서 확인할 수 있습니다. 다른 컴포넌트는 컴포넌트에 해당하는 디렉토리의 about.html 문서에서 라이선스 관련 내용을 확인할 수 있습니다

Q→이클립스 프로젝트는 어떻게 조직되어 있나요?

A→이클립스는 eclipse.org에서 운영하는 주요 프로젝트 중 하나입니다. 이클립스 프로젝트는 몇 개의 서브 프로젝트로 이루어져 있습니다. 각각의 프로젝트는 각각 하나 또는 하나 이상의 컴포넌트로 이루어졌습니다. 보다 자세한 사항은 다음 사이트를 참조 바랍니다.

- 이클립스 프로젝트에 대한 자세한 내용은 이클립스 프로젝트 차트(<http://www.eclipse.org/eclipse/eclipse-charter.html>)에서 확인할 수 있습니다.
- 이클립스 서브 프로젝트에 대한 내용은 이클립스 프로젝트 페이지(<http://www.eclipse.org/eclipse/index.html>)에서 확인할 수 있습니다.
- 컴포넌트에 대한 자세한 내용은 그 컴포넌트를 포함하는 프로젝트 페이지나 컴포넌트 페이지에서 확인할 수 있습니다.

Q→eclipse.org란 어떤 단체인가?

A→eclipse.org는 소프트웨어 개발 툴 제작자들의 개발을 위한 컨소시엄입니다. 대표적인 참여 업체는 'www.eclipse.org/org/index.html'에서 확인할 수 있습니다. 이클립스 프로젝트는 공개 표준형 통합 플랫폼이기 때문에 여러 개발자들이 함께 사용할 수 있다는 장점이 있습니다. 이클립스라는 핵심 통합 기술로 개발자들은 그들의 프로젝트와 신기술 개발에 힘을 모을 수 있게 되었습니다.

Q→새로운 이클립스 SDK에 대한 정보는 어떻게 얻을 수 있습니까?

A→'eclipse.org/eclipse/index.html'에서 최신 정보를 확인할 수 있습니다. 참고로 컴포넌트 페이지에서 자세한 진행 상황을 알 수 있습니다.

시작하기

Q&A

Q→어떤 버전의 이클립스를 다운 받아야 하나요?

A→최신 버전이나 가장 최근에 릴리즈 된 안정 버전을 받으면 됩니다. 최신 버전엔 여러 가지 버그가 있을 수 있습니다. 이런 면에서는 안정 버전이 안전합니다. 안전한 작업을 원하신다면 안정 버전을 활용하길 권합니다. 새로운 버전은 뉴스그룹을 통해 알 수 있습니다. 이클립스 다운로드 는 'eclipse.org/downloads/index.html' 에서 할 수 있습니다.

Q→페이지의 초록색 마크와 붉은색 X 마크는 뭔가요?

A→버전의 상태를 알 수 있습니다. 초록색은 테스트되었거나 이클립스 개발팀이 안전하다고 생각하는 버전입니다. 초록색도 버그가 있을 수 있으며 이전 버전으로 돌아갈 수도 있습니다. 다음 세 가지 항목중 하나라도 실패할 경우 붉은색 X 마크를 표시하게 됩니다.

- ① 플러그인 컴파일
- ② 자동 테스트를 모두 통과
- ③ 5분에서 10분의 테스트

Q→이클립스를 다운 받았습니다. 어떻게 실행하나요?

A→이클립스를 실행하기 위해서는 컴퓨터에 JRE(Java Runtime Environment)가 설치되어 있어야 합니다. 1.3이나 1.4의 J2SE JRE가 필요합니다. 이클립스는 JRE를 포함하고 있지 않습니다. JDK 에 JRE가 포함되어 있습니다. 이클립스 다운로드 페이지는 이클립스를 사용하기 위한 JRE와 JDK 링크를 안내합니다.

벤더	플랫폼	JDE, JDK
IBM	윈도우	IBM Developer Kit for Windows(R), Release 1.3.0
IBM	리눅스	IBM Developer Kit for Linux, Java 2 Technology Edition, Version 1.3
Sun	윈도우	JavaTM 2 SDK, Standard Edition Version 1.3.1_01 JavaTM 2 Runtime Environment, Standard Edition Version 1.3.1_01
Sun	리눅스	JavaTM 2 SDK, Standard Edition Version 1.3.1_01 JavaTM 2 Runtime Environment, Standard Edition Version 1.3.1_01
Sun	솔라리스	JavaTM 2 SDK, Standard Edition Version 1.3.1_01 JavaTM 2 Runtime Environment, Standard Edition Version 1.3.1_01

계속 ▶



JDK를 설치했다면 '-vm' 명령으로 이클립스 실행이 가능합니다. 만약 JDK가 설치된 곳이 'c:\jdk1.3' 이라면 다음 명령을 실행하면 됩니다.

```
eclipse.exe -vm c:\jdk1.3\bin\javaw.exe
```

참고로 어떤 JRE와 JDK는 자동으로 시스템 경로를 정하기도 합니다. 이럴 경우에는 '-vm' 명령만 내려도 됩니다.

Q→이클립스는 어떤 운영체제를 지원하나요?

A→1.0 릴리즈 버전은 윈도우 98/ME/2000/XP 그리고 레드햇 리눅스 7.1(x86/Motif)에서 실행되도록 만들어졌습니다. 최신 2.0 버전은 수세 리눅스 7.1(x86/Motif and x86/GTK)과 솔라리스 8 (SPARC/Motif)도 지원합니다. 향후 더 많은 플랫폼이 지원 할 예정입니다. 해당 플랫폼에 대한 정보는 사이트 다운로드 페이지에서 확인하세요. 버전으로 지원하는 플랫폼을 알 수 있습니다.

Q→"Hello World" 샘플 프로그램을 어떻게 만들지

A→먼저 이클립스를 실행합니다. 그리고 새로운 프로젝트를 다음과 같이 하나 만듭니다.

- ① 메뉴에서 'File' → 'New' → 'Project' 선택
- ② 종류는 자바를 선택
- ③ 프로젝트 리스트 중에 'Java Project' 를 선택하고 'Next' 을 누르면 됩니다.
- ④ 프로젝트 이름은 "Hello World Project"와 같은 식으로 하세요
- ⑤ 'Finish' 를 누르면 프로젝트를 볼 수 있는 창이 생깁니다.

이제 자바 파일을 만들어 보겠습니다.

- ① 툴바 버튼 중에 'Create a Java Class' 를 누르세요.
- ② 이름은 "HelloWorld"로 하세요.
- ③ 'public static void main(String[] args)' 메소드를 만들려면 체크박스에 체크하세요.
- ④ 'Finish' 클릭.

HelloWorld.java 에디터 창이 열릴 겁니다. 메인 함수에 다음과 같이 넣으세요

```
System.out.println("Hello World");
```

'**Ctrl** + **S**' 를 누르면 저장되고 자동으로 컴파일됩니다. 툴바에서 'run' 을 누르세요. 실행 환경을 묻는 창에서 'Java Application' 을 선택하고 'New' 를 클릭하세요. 'Run' 을 누르면 프로그램이 실행됩니다. 콘솔 창에 "Hello World"라는 글자가 보일 것입니다.

계속 ▶

Q→자바 소스를 어떻게 컴파일 하나요?

A→이클립스는 저장할 때 자동으로 컴파일하는 autobuild 기능을 가지고 있습니다. 'Window' → 'Preferences' → 'Workbench' → 'Perform build automatically' 에서 기능을 확인할 수 있습니다. 자동 빌드 기능을 끄고 수동으로 툴바에 있는 'build' 버튼으로 컴파일해도 됩니다.

Q→자바 프로그램 실행과 디버깅에 대해 알고 싶습니다.

A→패키지뷰 창에 있는 자바 프로그램의 메인 클래스를 선택하고 'Run' 이나 'Debug' 버튼을 누르면 됩니다. 한번 실행했으면 다음부터는 **F9**로 간단히 실행할 수 있습니다. 프로그램을 실행하면 코드에 의해 프로그램이 진행됩니다. 프로그램을 디버그해서 그 과정을 조절할 수 있습니다. 스텝이나 브레이크 포인트를 이용해 프로그램 내부를 조사할 수도 있습니다. 가상 머신에 디버거가 액세스할 수 있도록 속도를 늦추는 겁니다.

Q→CVS 저장소는 어떻게 액세스 하죠?

A→CVS 저장소를 액세스하려면 저장소 위치가 있어야 합니다. 새로운 저장소를 만들려면 다음과 같이 하면 됩니다.

① CVS 저장소 창을 엽니다. 메뉴에서 'Window' → 'Show View' → 'Other...' → 'CVS' → 'CVS Repositories' 를 선택합니다.

② CVS 창을 오른쪽 클릭하고 'New' → 'CVS Repository Location...' 을 선택

③ CVS 서버의 세부 사항을 입력하세요. 예를 들면 다음과 같습니다.

- **Host** : dev.eclipse.org
- **Repository path** : /home/eclipse
- **User** : anonymous
- **Password** : <leave empty>
- **Connection type** : pserver

④ 'Finish' 버튼을 누르면 됩니다.

저장소를 갖게 되면 저장소를 이용해서 프로젝트를 공유하고 수정할 수 있습니다. 프로젝트를 공유하려면 'Project' → 'right click' → 'Team' → 'Share Project...' 와 같이 저장소도 선택해야 합니다. 프로젝트가 수정되면 동기화 창이 나타날 겁니다. 코드를 수정하려면 수정에 대한 주석을 입력해야 합니다. 저장소를 통해 프로젝트를 공유한 후에 'Synchronize with Repository...' 기능을 통해서 최신 정보로 수정하고 다른 사람이 만든 코드를 얻을 수 있습니다. 참고로 저장소로 전송이 실패하면 다음과 같은 workspace/.metadata/.log파일을 확인해 보기 바랍니다. 모든 이를 위해 CVS 서버 디렉토리의 권한을 허용하세요.



```

4 org.eclipse.vcm.core.cvs 1 The CVS repository reported problems.
1=====
4org.eclipse.vcm.core.cvs 4 cvs server: failed to create lock directory
for `/home/cvs/CVSRROOT/Emptydir'
(/home/cvs/CVSRROOT/Emptydir/#cvs.lock):
Permission denied
4 org.eclipse.vcm.core.cvs 4 cvs server: lock failed - giving up
4 org.eclipse.vcm.core.cvs 4 cvs [server aborted]: lock failed
- giving up
1=====

```

Q→이클립스 매뉴얼은 어디에 있죠?

A→두 가지 방법이 있습니다. 첫 번째는 이클립스 SDK에 있는 매뉴얼을 이용하는 것으로 헬프 창에서 선택할 수 있습니다. 두 번째는 온라인 매뉴얼(<http://eclipse.org/documentation/main.html>)을 이용하는 것입니다. HTML과 PDF 형식이 있습니다.

Q→질문은 어떻게 하나요?

A→FAQ나 매뉴얼을 봐도 해결이 안되면 해당 문제를 뉴스그룹(<http://eclipse.org/newsgroups/index.html>)에 보내면 됩니다. 이를 위해서는 패스워드(<http://dev.eclipse.org/newsManager/newsRequestForm.html>)가 필요합니다. 기술에 관한 것은 뉴스그룹을 이용하기 바랍니다.

하지만 뉴스그룹은 공개된다는 것을 생각해야 합니다. 그렇기 때문에 비밀 정보는 포함하지 말기 바랍니다. "How to ask questions the smart way"(<http://www.tuxedo.org/~esr/faqs/smart-questions.html>)에릭 레이몬드가 쓴 이 글을 참조하세요. 참고로 버그 리포트는 버질라(<http://dev.eclipse.org/bugs/>)에 하면 됩니다.

Q→버그 리포트는 어떻게 하죠?

A→이클립스는 버그질라라는 버그 추적 시스템을 사용합니다. 버그 리포트는 버그질라 페이지에서 문서 작성만 하면 됩니다. 처음에는 계정과 패스워드가 필요합니다. 즉 등록을 해야 합니다. 버그 리포트 전에 버그질라에서 같은 리포트가 있는지 확인하세요. 이미 보고된 버그에 주석만 달 수도 있습니다.

같은 버그 리포트가 없으면 새로운 버그를 리포트 하세요. 보다 자세한 사항은 이클립스 버그 리포트에 관한 문서(<http://dev.eclipse.org/bugzilla.html>)를 참조 바랍니다. 버그 리포트를 유용하게 하기 위해 다음과 같은 내용이 필요합니다:

- ① 이클립스 버전(예 : Eclipse SDK 20011206)
- ② 컴퓨터 사양(운영체제, 메모리, 그리고 다른 부가 정보)

계속

- ③ 로그 파일 첨부(매우 중요합니다)
- ④ 이클립스가 작동하는 실행과 개발 플랫폼의 버전(JRE, JSDK)
- ⑤ 버그가 일어난 때
- ⑥ 버그에 대한 소견
- ⑦ 반복해서 일어나는 버그에 대한 설명

Q→로그 파일은 어디에 있습니까?

A→로그 파일은 workspace/.metadata 디렉토리에 있습니다. 로그 파일은 이클립스 런타임 에러 내용도 포함됩니다. 버그 리포트와 관련해서 볼 때 매우 중요한 정보를 담고 있다고 볼 수 있습니다.

Q→프로그램 개선 건의는 어떻게 합니까?

A→건의는 버그질라(<http://dev.eclipse.org/bugs/>)를 통해 하면 됩니다. 글을 올리기 전 버그질라를 통해 같은 의견이 있는지 확인하고 자세하게 적어주세요.

Q→자바 파일을 어떻게 이클립스로 옮기나요?

A→예전에 만든 파일은 프로젝트를 만들고 나서 프로젝트로 복사하는 것이 가장 좋습니다. 다음과 같은 방법이 있습니다:

- import 메뉴를 사용하는 방법(File → Import...)
- 프로젝트 폴더로 파일을 복사한 다음에 프로젝트를 오른쪽 클릭하고 'Refresh From Local'를 선택하면 됩니다.
- 윈도우의 경우 드래그 앤 드롭으로 복사할 수 있습니다.

파일 복사가 싫다면 프로젝트를 파일이 있는 곳에 만들어서 할 수 있습니다. 만약에 5,000개의 소스가 'c:\work'에 있다면 프로젝트의 위치를 'c:\work'로 프로젝트 생성시 'Use default location' 항목을 언체크(un check)해서 지정할 수 있습니다. 참고로 이클립스에서는 프로젝트 간의 중복되는 패스를 설정할 수 없습니다.

Q→프로젝트를 지울 때요 내용을 지우겠냐고 묻는데 이게 무슨 소리죠?

A→이클립스는 프로젝트를 디렉토리에 저장하고 있습니다. 기본적으로 워크스페이스와 프로젝트의 이름과 같은 곳에 파일이 위치합니다. 하지만 프로젝트가 기본 위치가 아닌 지정한 위치에 있는 경우 파일은 유지하고 워크스페이스에서 프로젝트만 삭제하는 것이 좋습니다. 참고로 기본 위치가 아닌 지정한 위치에서 프로젝트를 만들 때 프로젝트 위치 간의 중복으로 파일들이 삭제가 될 수 있으니 주의 바랍니다.



Q→폴더 이름에 의해 패키지가 정해지는데(예를 들어 sources.mypackage.MyClass) 어떻게 수정 하죠?

A→자바 클래스 폴더 위치를 설정할 수 있습니다.

- ① 정보 창을 엽니다. 그리고 메뉴에서 'Java Project' → 'right click' → 'Properties' → 'Java Build Path' 를 선택합니다.
- ② 소스 항목을 선택합니다.
- ③ 'Use source folders contained in the project' 를 선택합니다.
- ④ 'Add Existing Folders...' 를 클릭하세요
- ⑤ 소스 폴더를 정한 다음에 'OK' 를 누르세요.
- ⑥ 이클립스가 출력 파일 위치를 '/<project name>/bin' 로 바꿀 것이냐고 물으면 동의하세요.
- ⑦ 정보 창에 'OK' 버튼을 누르면 끝입니다.

Q→왜 리소스들이 나의 Output 폴더로 복사되나요?

A→출력 위치를 정하면 JDt는 컴파일된 클래스 파일을 패키지 폴더에서 출력 폴더로 복사합니다. 복사가 싫으면 다음과 같이 하세요.

- 소스 폴더 말고 출력 폴더를 프로젝트 폴더와 일치시키세요
- 프로젝트 폴더에 소스 폴더를 추가되지 않게 리소스들을 프로젝트로 복사하지 마세요

Q→다른 프로젝트와 관련된 소스를 개인 프로젝트에서 사용할 수 있나요?

A→현재로서 지원은 하지 않습니다. 프로젝트들은 각기 다른 위치에 있습니다. 만약의 소스가 많은 다른 위치에 널려 있다면 파일 시스템을 이용해서 연관시킬 수 있습니다.

Q→다른 JRE를 사용할 수 있나요?

A→다음과 같은 순서를 참조 바랍니다.

- ① 메뉴에서 'Window' → 'Preferences' → 'Java' → 'Installed JREs' 를 선택합니다
- ② 'Add' 를 누르고 JRE의 이름을 입력하세요(예 : JDK 1.3)
- ③ 'Browse' 를 누른 후 JRE가 있는 위치를 추가하세요.
- ④ JRE 홈 디렉토리를 선택했으면 JRE_SRC 변수는 여러분이 선택한 JRE를 가리키지 않습니다. 소스를 포함하고 있는 JRE들은 src.jar가 아닌 다른 곳에 있습니다. 만약에 JRE를 위해 변수를 설정하려면 'Use default library' 를 선택하지 마세요. 참고로 어떤 JRE들은 소스를 포함하고 있지 않습니다. 소스를 포함하고 있지 않는 JRE로 설정할 경우 디버거가 오작동을 할 수 있으니 주의 바랍니다.

계속 ▶

- ⑤ 설정이 완료되면 'OK' 버튼을 누르세요.
- ⑥ 모든 프로젝트에서 기본이 될 JRE를 선택하세요.
- ⑦ 'OK' 버튼을 누르면 끝입니다.

참고로 기본 JRE랑 다른 JRE를 사용해 실행하는 것은 가능합니다. 'Run' → 'Run...' → 'select your Launch configuration' 메뉴의 JRE 항목에서 수정할 수 있습니다.

Q→HotSwap 기능은 어떻게 쓰니까?

A→ 핫스왑 기능을 사용하려면 핫스왑을 지원하는 자바 런타임 아래서 디버그를 해야 합니다(1.4 버전의 자바 런타임은 이 기능을 지원하지 않습니다)..

- ① launch configuration에서 핫스왑을 지원하는 JRE로 바꾸세요.
- ② 브레이크를 에디터 왼쪽과 에디터 코드사이를 더블클릭해서 지정하세요.
- ③ 디버그하세요.
- ④ 브레이크가 걸리면 코드를 바꾸세요.
- ⑤ 저장하세요.
- ⑥ 자동 빌드 기능이 꺼졌다면 툴바의 빌드 버튼으로 컴파일하세요.
- ⑦ 컴파일 후 VM은 자동으로 새로운 코드를 실행합니다.

Q→엔트가 실행할 때 에러가 나는데 어떻게 하죠?

A→ 엔트는 다음과 같은 예러 메시지를 보일 수 있습니다.

• Cannot use classic compiler, as it is not available.JAVA_HOME 패스 문제입니다.

엔트 클래스 패스에 tools.jar를 추가하거나 엔트가 자바 프로그램을 컴파일할 때 쓰는 컴파일러를 이클립스가 포함하고 있는 컴파일러로 교체해서 문제를 해결할 수 있습니다. tools.jar 파일을 추가하려면 'Window' → 'Preferences' → 'Ant' → 'Customize'를 추가하면 됩니다. 이클립스 2.0 이전 버전에서는 지원을 하지 않습니다. 엔트 컴파일러 변경 방법을 소개하겠습니다. 먼저 스크립트에 다음 구문을 추가하세요.

```
<property name="build.compiler"
value="org.eclipse.jdt.core.JDTCompilerAdapter" />
```

이클립스를 사용하지 않고는 스크립트를 실행할 수 없습니다. 다음은 엔트 스크립트를 설정합니다. 먼저 'script'를 선택하고 마우스 오른쪽 버튼을 눌러서 'Run Ant...'를 선택하십시오. 인수는 다음과 같이 넣으면 됩니다.



```
-Dbuild.compiler=org.eclipse.jdt.core.JDTCompilerAdapter
```

참고로 위 내용은 이클립스 2.0에서만 가능합니다. 만일 1.0이나 2002년 6월 21일 이전 2.0 버전을 쓰고 있다면 'org.eclipse.jdt.core.JDTCompilerAdapter'를 'org.eclipse.pde.internal.core.JDTCompilerAdapter'로 바꾸어야 합니다.

Q→이클립스 SDK는 GUI 디자인 키트도 포함하나요?

A→이클립스 SDK에는 GUI 디자인 키트가 없습니다.

Q→다른 컴퓨터에서 실행되는 프로그램은 어떻게 디버그 하나요?

A→다음과 같이 원격으로 디버깅을 할 수 있습니다.

- ① 이클립스를 실행합니다.
- ② 메뉴에서 'Perspective' → 'Open' → 'Java'를 선택합니다.
- ③ 'Create a new Java Project'를 눌러 새로운 프로젝트를 만드세요. 만약 프로그램 소스가 있으면 프로젝트에 임포트하세요. 그래야 디버거가 디버깅 중에 소스를 보여 줍니다.
- ④ 다른 컴퓨터에 있는 자바 프로그램을 실행하세요. 명령어 줄에 프로그램 디버거가 가능하도록 다음을 추가하세요(참고로 여러분이 사용할 포트 주소로 바꿀 수 있습니다).

```
-Xdebug -Xnoagent -Xrunjdp:transport=dt_socket,  
server=n,suspend=n,address=8000 -Djava.compiler=NONE
```

- ⑤ 여러분이 만든 자바 프로젝트를 선택하세요. 그리고 'Run' → 'Debug...'를 선택합니다.
- ⑥ 원격 자바 프로그램을 선택하고 'New'를 누르세요.
- ⑦ 원격 자바 프로그램의 컴퓨터 이름과 포트 주소를 넣고 'Finish'를 누르세요. 디버거가 원격 프로그램으로 접속됩니다.

이 설정은 자바 프로그램이 디버거가 연결될 때까지 실행을 기다리는 설정입니다. 시작 중에 디버거가 필요 없으면 다음과 같이 명령어 줄을 수정하세요.

```
"server=n" to "server=y"
```

웹서버나 비슷한 프로그램 디버거에 매우 유용합니다. 이 방법은 여러분의 컴퓨터에서 디버깅할 때도 사용할 수 있습니다.