IBM. 한국어 로그인 (또는 등록)
developerWorks. 기술 토픽 다운로드 커뮤니티 행사 및 세미나

Eclipse V3.4 완전 정복, Part 3: JDT 텍스트 편집기 팁과 비결

Prashant Deva, 창립자, Placid Systems

요약 : "Eclipse 완전 정복" 연재 기사는 Eclipse를 처음 접하는 사용자들에게 Eclipse IDE를 속속들이 가르쳐 줍니다. 이 연재가 끝날 때쯤엔 고급 사용자와 어깨를 나란히 하게 될 것입니다. 이 기사에서는 Eclipse의 Java™ 편집기를 둘러보고 일부 고급 기능에 대해 학습합니다.

기사 게재일 : 2011 년 11 월 28 일
난이도 : 초급
원문 : 보기
페이지뷰 : 1104 회
의견 : 1 (보기 | 의견 추가 - 로그인)
평균 평가 등급 (0 투표하기)
아티클 순위

"Eclipse 완전 정복" 연재의 이 파트에서는 Eclipse의 Java 편집기에 대해 자세히 살펴본다. 편집기는 개발자가 대부분의 시간을 들이는 곳이므로 고급 기능을 이해하면 생산성을 크게 개선할 수 있다.

`Organize Imports` 명령

`Organize Imports` 명령은 누락된 가져오기를 추가하고 Java 파일에서 기존 가져오기 선언을 구성한다. **Ctrl+Shift+O**을 사용하여 현재 편집기에서 이 명령을 실행할 수 있다. `Organize Imports` 명령을 전체 프로젝트에 적용하려면 Project Explorer에서 프로젝트를 마우스 오른쪽 단추로 클릭하고 **Source > Organize Imports**를 선택한다.

Java 파일 어딘가에서 클래스를 사용했지만 이를 가져오는 것을 잊어버렸다고 가정하자. `Organize Imports`가 독자를 대신해 자동으로 가져올 수 있다. 해당 명령이 클래스의 위치를 잘 모르면, 선택하는 옵션 목록이 있는 창이 열린다. 예를 들어, 코드에서 `List` 클래스를 사용하는 경우, `Organize Imports` 명령을 실행할 때, 명령이 자체적으로 결정할 수 없기 때문에 `java.util.List`와 `javax.swing.List` 사이에 선택하도록 질문하는 창이 팝업될 수 있다.

`Organize Imports`도 일반적으로 사용되는 `.*` 스타일 가져오기 선언을 개별 가져오기로 나눈다. 예를 들어, 파일에서 `import java.util.*` 와 같은 가져오기가 있고 패키지에서 `List` 클래스만 사용한다고 가정한다. `Organize Imports` 명령은 원래 `import` 명령문을 `import java.util.List`로 바꾼다.

문자열 부분 골라내기

종종 두 개의 문자열 사이에 변수를 연결해야 한다. 대부분의 경우에 이러한 문자열은 단일 명령문의 부분이며, 문자열의 둘 중 하나의 측면에 공백을 생략하는 실수를 하기가 쉽다. 예를 들어, `You have 5 seconds left`가 아니라 `You have5seconds left`와 같은 출력이 될 수 있다. Eclipse는 다음과 같이 이 함정에 빠지지 않도록 돕는다.

1. 코드 내에서 일반적으로 하는 것처럼 다음과 같이 문자열을 입력한다.
```
static String getMsg(int time)
{
    return "You have 5 seconds left";
}
```
2. 변수로 바꾸려는 문자열 부분을 선택한다 — 이 예제에서는 숫자 5이다.
3. **Ctrl+1**을 누르고 **Pick out selected part of String**을 선택한다.

그림 1. Pick out selected part of String 선택
```
static String getMsg(int time)
{
    return "You have 5 seconds left";
}
```
Pick out selected part of String

그림 2에서 결과를 보여준다.

그림 2. 고른 문자열 부분이 선택됨
```
static String getMsg(int time)
{
    return "You have " + "5" + " seconds left";
}
```

4. 중간에 문자열을 변수와 바꾼다.

그림 3. 문자열을 변수와 바꾸기
```
static String getMsg(int time)
{
    return "You have " + time + " seconds left";
}
```

이제 공백에 대해 실수하지 않았다고 확신할 수 있다.

자동으로 로컬 작성

종종 메소드를 호출하고 값을 새 로컬 변수로 지정해야 한다. Eclipse에서는 이를 매우 간편하게 만들기 때문에 다음과 같이 로컬을 이전 방식으로 절대 지정하지 않을 것이다.

1. 다음과 같이 호출을 메소드로 입력한다.
```
public void foo()
{
    getMsg(3);
```

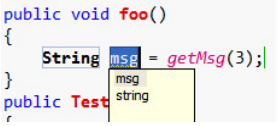
```
}
```

2. 삽입 마커를 이동하지 않은 상태로 **Ctrl+1**을 누르고 **Assign statement to a new local variable**을 선택한다(그림 4 참조). 로컬 변수가 아니라 필드를 사용하려는 경우 **Assign statement to a new field**를 선택할 수 있다.

그림 4. Assign statement to a new local variable 선택

메소드의 리턴 값과 동일한 유형의 새 로컬 변수가 작성되었다. 적절한 이름이 제공되지만 원하는 경우 이를 변경할 수 있다(그림 5 참조). **Enter**를 눌러 이름을 수락한다.

그림 5. 스스로 변수의 이름을 변경할 수 있음

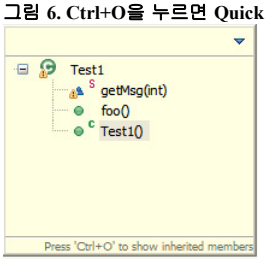


Quick Outline

Outline 보기는 Java 파일에서 메소드로 간편하게 이동할 수 있기 때문에 유용하다. 하지만 이 보기는 귀중한 화면 상의 공간을 많이 차지한다.

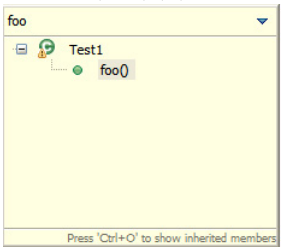
Quick Outline 보기는 화면을 차지하는 보기가 필요하지 않은 상태로 Outline 보기의 모든 기능을 제공한다. 편집기 내에서부터 Quick Outline 보기를 활성화하려면 **Ctrl+O**을 누른다. 팝업 창이 파일의 아웃라인을 표시하면서 나타난다.

그림 6. Ctrl+O을 누르면 Quick Outline 보기가 표시됨



화살표 키를 사용하여 어느 메소드로나 탐색할 수 있다. 메소드로 훨씬 더 빠르게 이동하려면 이름을 입력하기 시작한다. 목록이 필터하기 시작하여, 입력하는 문자로 시작하는 메소드만 표시된다.

그림 7. 입력을 시작하면 Outline 보기가 필터링을 시작함

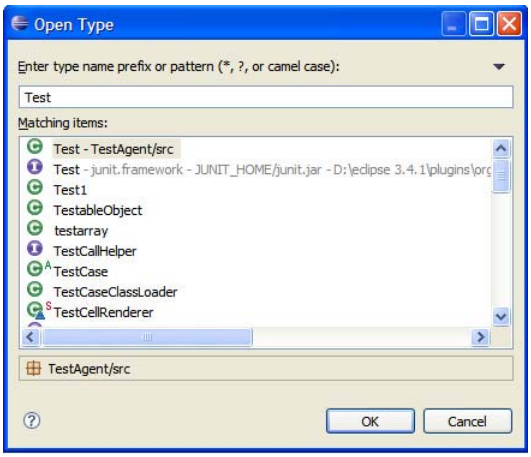


Quick Open Type

Open Type 창은 Quick Outline이 Outline 보기에 하는 작업을 Package Explorer에 수행한다.

편집기에서 **Ctrl+Shift+T**를 누른다. 작업공간에서 어느 클래스 이름이나 입력할 수 있는 창이 열린다(그림 8 참조). **OK**를 클릭하면, 해당 클래스가 편집기에서 즉시 열린다. 원하는 클래스를 열기 위해 Package Explorer에서 트리를 샅샅이 뒤질 필요가 없다.

그림 8. Open Type 창을 통해 작업공간의 어느 클래스로나 즉시 이동할 수 있음

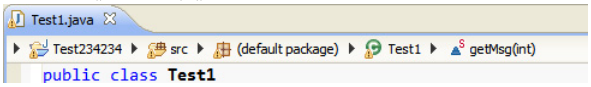


텍스트 상자 아래 목록을 필터링하기 위해 클래스 이름을 입력할 때 와일드카드를 사용할 수 있다. 예를 들어, `*Exception`을 입력하여 모든 `Exception` 클래스를 표시한다.

브레드크럼 막대(Breadcrumb bar)

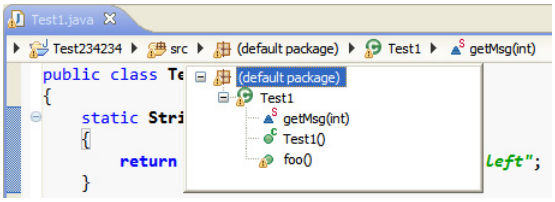
브레드크럼 막대는 Eclipse V3.4에 도입되었다. 이는 탭 바로 아래 편집기 창의 맨 위에 놓여 있고, Package Explorer 및 Outline 보기 둘 다의 기능을 제공한다(그림 9 참조). 활성화되지 않으면 **Alt+Shift+B**를 눌러 활성화할 수 있다.

그림 9. 브레드크럼 막대



브레드크럼 막대는 현재 작업공간과 관련된 현재 편집기의 경로를 가리킨다. 요소의 컨텐츠를 표시하는 검은색 화살표 중 하나를 클릭하면 막대에서 전체 작업공간을 찾아볼 수 있다.

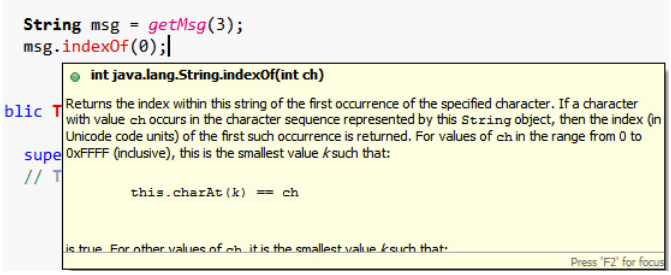
그림 10. 브레드크럼 막대를 통해 해당 막대에서 작업공간을 찾을 수 있음



Quick Javadoc

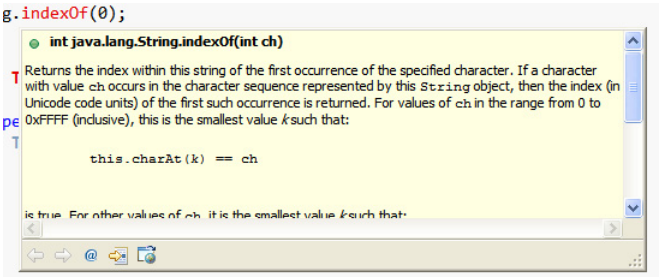
메소드가 무엇인지 알아보기 위해 시도하고 있는가? Eclipse는 어느 메소드에 대한 Javadoc이나 간편하게 표시할 수 있다. 메소드에 커서를 갖다 대면 Javadoc 창이 나타난다.

그림 11. 커서를 Java 요소에 갖다 대면 관련된 Javadoc이 표시됨 -- 이 경우에 `indexOf` 메소드



하지만 이 기능은 이것이 끝이 아니다. 커서를 팝업 창 내로 이동하면 창이 스크롤 가능하게 된다(그림 12 참조). 맨 아래 단추를 사용하여 내부 창에서 Javadoc을 열 수 있으며, 보고 있는 Javadoc의 요소의 선언으로 이동할 수 있다.

그림 12. 커서를 팝업 내로 이동하면 스크롤 가능한 창으로 변환함



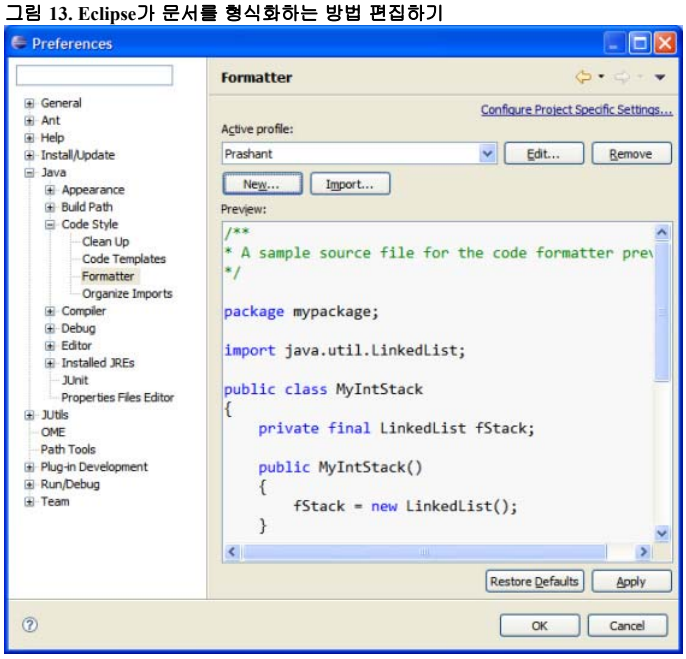
자동 형식화

문서를 자동으로 형식화하기 위해 Eclipse의 자동 형식화 기능을 사용할 수 있다. Package Explorer에서 문서를 마우스 오른쪽 단추로 클릭하고 **Source > Format**을 선택한다. Package Explorer에서 개별 파일이 아니라 프로젝트를 마우스 오른쪽 단추로 클릭하여 전체 프로젝트를 형식화할 수도 있다.

Eclipse가 코드를 형식화하는 방법을 구성하려면 다음을 수행한다.

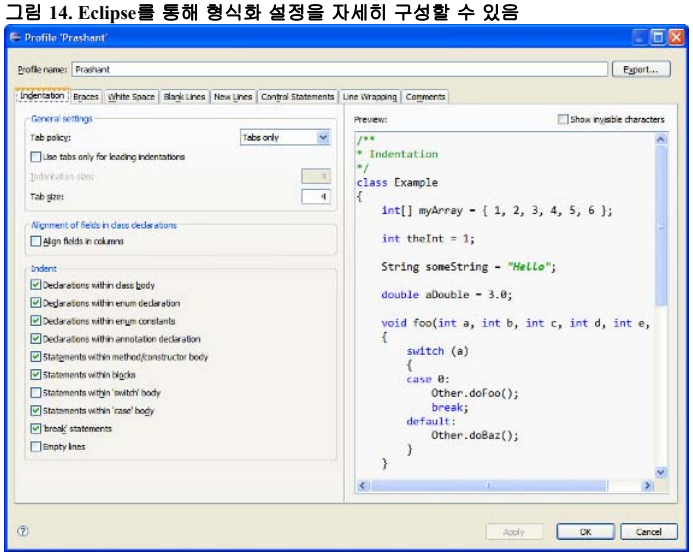
1. **Window > Preferences**를 선택한다. 그 다음에 **Java > Code Style > Formatter**를 찾는다.

그림 13. Eclipse가 문서를 형식화하는 방법 편집하기



2. **New**를 클릭하고 열린 창에서 이름을 입력하여 형식화하기 위해 새 프로파일을 작성한다.
3. 프로파일을 작성한 후에 **Edit**를 클릭하여 해당 프로파일에 대한 형식화 환경 설정을 편집하기 시작한다.
4. 열린 **Profile** 창은 소스 코드의 형식화를 구성하기 위한 자세한 설정이 들어 있다(그림 14 참조). 소스 코드에서 각 요청에 어느 정도의 들여쓰기가 필요한지에서부터 가져오기 그룹 사이에 빈 줄이 얼마나 많이 표시되는지까지 모든 것을 구성할 수 있다.

그림 14. Eclipse를 통해 형식화 설정을 자세히 구성할 수 있음



5. **OK**를 클릭하여 **Profile** 창을 닫는다. **OK**를 다시 클릭하여 **Preferences** 창을 닫는다.

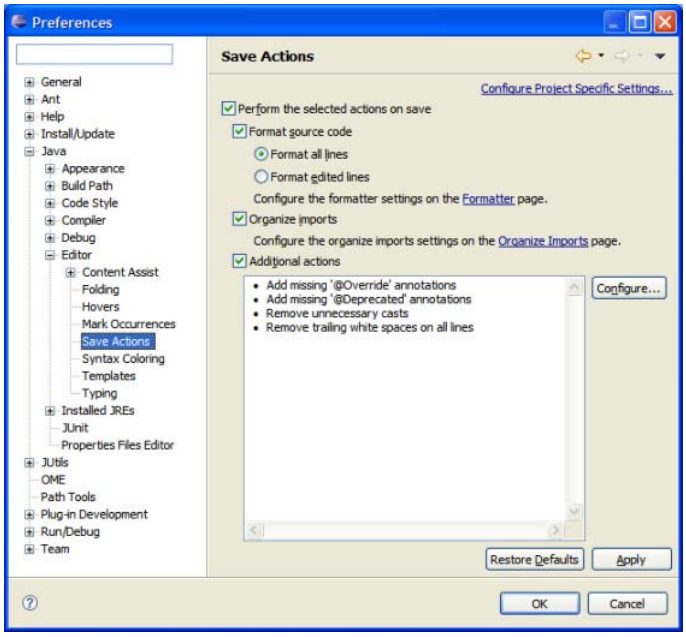
새 형식화 설정은 *다음에* 소스 코드를 형식화할 때 적용될 것임을 참고하자.

Save Actions

Save actions는 문서를 저장할 때 트리거된다. 이를 통해 코드를 형식화하는 것과 같이 작업을 수행할 수 있고 문서를 저장할 때 자동으로 가져오기를 구성할 수 있다. 저장 조치를 구성하려면 다음을 수행한다.

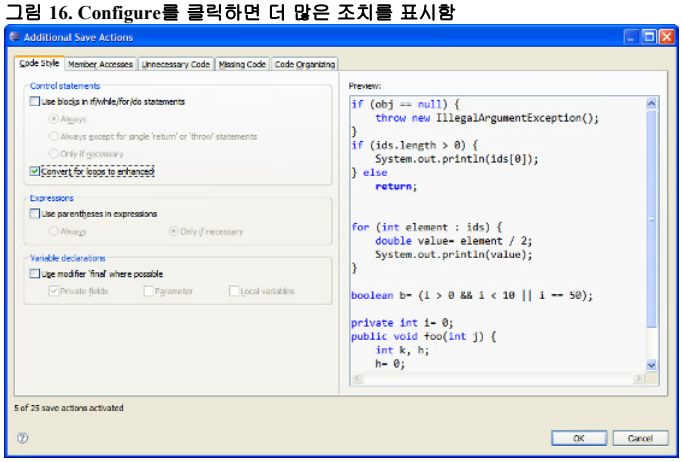
1. **Window > Preferences > Java > Editor > Save Actions**를 선택하여 Preferences 창을 연다.

그림 15. Preferences 창에서 Save Actions 구성하기



2. **Perform the selected actions on save** 선택란을 선택한다.
3. 저장에서 수행하려는 모든 조치를 선택한다. **Configure**를 클릭하여 조치를 더 많이 확인한다.

그림 16. Configure를 클릭하면 더 많은 조치를 표시함



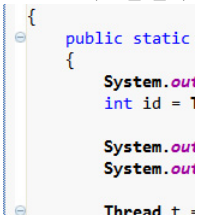
예를 들어, 문서를 저장할 때마다 모든 `for` 루프를 Java 5 스타일 향상된 루프로 변환하기 위해 **Code Style** 탭에서 **Convert for loops to enhanced** 선택란을 선택할 수 있다.

Save Actions는 시간을 많이 절약할 수 있다. 예를 들어, 형식화에 대해 걱정하지 않고 원하는 어느 방식으로나 코드를 입력할 수 있고, 이는 문서를 저장할 때마다 여전히 완벽하게 자동으로 형식화될 것이다.

코드 폴딩

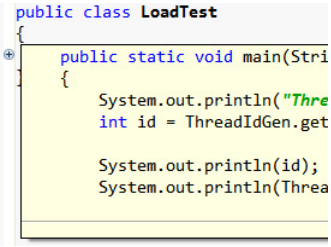
코드 폴딩을 통해 말 그래도 코드 조각을 접을 수 있으므로 편집기가 너무 지저분해지지 않는다. 메소드를 접으려면 왼쪽 눈금자에서 - 아이콘을 클릭한다.

그림 17. - 부호를 클릭하면 코드를 접음



코드를 접을 때, - 아이콘이 +로 변하며, 이는 메소드를 펼치기 위해 다시 클릭할 수 있다. 메소드가 접힐 때 커서를 + 부호에 갖다 대면, 팝업 창이 메소드 내의 텍스트를 표시한다.

그림 18. + 아이콘에 커서를 갖다 대면 접힌 코드의 미리보기를 표시함

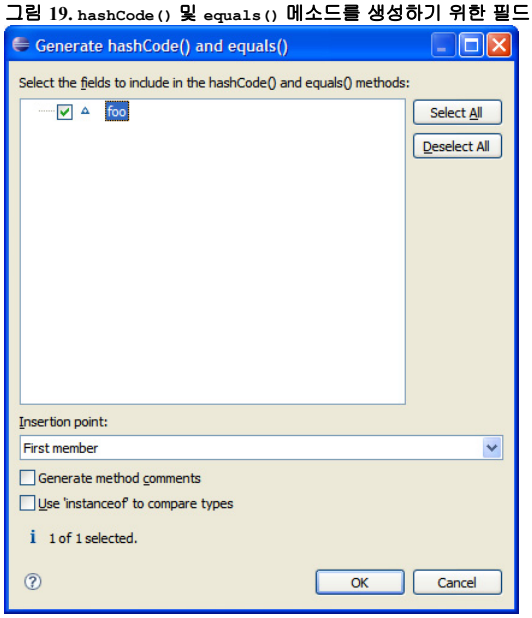


`hashCode()` 및 `equals()` 생성

Eclipse를 통해 클래스에 대한 `hashCode()` 및 `equals()` 메소드를 자동으로 생성할 수 있으므로 스스로 할 필요는 없다. 이 방법으로 이러한 메소드를 쓰는 동안 오류가 발생하는 것을 방지할 수 있다. 메소드를 생성하려면 다음을 수행한다.

1. **Source > Generate hashCode() and equals()**를 선택한다.
2. 창이 열리며 `hashCode()` 및 `equals()` 메소드가 클래스에서 호출될 때 어느 필드를 포함시키려는지 질문한다.

그림 19. `hashCode()` 및 `equals()` 메소드를 생성하기 위한 필드 선택



3. 원하는 필드를 선택하고 **OK**를 클릭한다. `equals()` 및 `hashcode()` 메소드가 생성된다.

개요 눈금자

개요 눈금자는 편집기 창의 오른쪽에 임베드된다. 이를 통해 어느 오류, 경고 또는 편집기에서 주목해야 하는 다른 마커를 즉시 확인할 수 있다. 편집기에서 오류가 있다면, 막대의 상단 오른쪽 모서리에 빨간색 상자가 표시된다.

그림 20. 개요 눈금자의 빨간색 상자는 편집기에서 하나 이상의 오류를 표시함

한편, 노란색 상자는 편집기에서 경고가 있음을 의미한다.

그림 21. 편집기에서 경고의 존재를 표시하는 개요 눈금자

개요 눈금자를 사용하면 편집기에서 어느 오류/경고나 즉시 이동할 수 있다. 눈금자는 편집기에서 오류/경고가 표시되는 곳마다 작은 노란색/빨간색 표시가 표시된다(그림 22 참조). 표시를 클릭하면 편집기는 코드에서 적절한 위치로 즉시 이동한다.

그림 22. 두 가지 경고와 한 가지 TODO 주석을 표시하는 개요 눈금자

또한 개요 눈금자는 편집기에서 관련된 다른 지점을 표시할 수 있다. 예를 들어, TODO 주석이 있거나 검색을 수행하면, 관련된 행은 개요 눈금자에서 표시된다. 마커를 클릭하면 해당 행으로 안내한다.

이름이 암시하는 대로 개요 눈금자는 편집기에서 관련된 행의 개요를 제공한다. Java 코드를 쓰면서 매우 유용한 동반자를 발견할 것이다.

결론

이 기사에서는 Eclipse에서 Java 편집기의 일부 고급 기능을 조사했다. 일반적인 코드 부분을 자동으로 생성하는 방법을 확인했다. Eclipse 편집기에서 제공된 단축키 덕분에 Eclipse 보기를 닫고 기능을 계속 사용하는 방법도 학습했다. 이 기사에서는 생산성을 크게 개선할 수 있는 Java 편집기의 다른 기능도 논의했다.

참고자료

교육

- "Eclipse V3.4 완전 정복, Part 1: Eclipse IDE 워크벤치"로 이 시리즈를 시작하자. 해당 시리즈는 "Eclipse V3.4 완전 정복, Part 2: JDT"로 계속된다.
- Java Development User Guide in Eclipse Help에서 자세한 정보를 찾아보자.
- Eclipse IDE Pocket Guide를 읽어보자.
- 공식 Eclipse FAQ를 방문하자.
- *Eclipse for Dummies*를 읽어보자.
- *Eclipse Distilled*를 읽어보자.
- "Recommended Eclipse reading list"를 확인하자.
- developerWorks에 있는 Eclipse 관련 컨텐츠를 모두 볼 수 있다.
- Eclipse 입문자라면 developerWorks 기사인 "Eclipse Platform 시작하기"에서 Eclipse Platform의 기원과 아키텍처 및 플러그인을 사용하여 Eclipse를 확장하는 방법을 읽어보자.
- IBM developerWorks의 Eclipse 프로젝트 자원을 활용하여 Eclipse 기술을 향상시키자.
- developerWorks 팟캐스트에서 소프트웨어 개발자의 흥미로운 인터뷰와 토론을 확인할 수 있다.
- developerWorks의 기술 행사 및 웹 캐스트를 통해 최신 정보를 얻을 수 있다.
- 무료로 제공되는 developerWorks On demand demos를 통해 IBM 및 오픈 소스 기술에 대해 배우고 제품 기능을 익히자.
- IBM 오픈 소스 개발자에게 유익한 컨퍼런스, 기술 박람회, 웹 캐스트 및 기타 행사를 확인하고 참여하자.
- developerWorks 오픈 소스 영역에서 오픈 소스 기술을 활용하여 개발 작업을 수행하고 이러한 기술을 IBM 제품과 함께 사용하는 데 도움이 되는 사용법 정보, 도구 및 프로젝트 업데이트를 확인할 수 있다.

제품 및 기술

- IBM alphaWorks에서 최신 Eclipse 기술 다운로드를 확인하자.
- Eclipse Foundation에서 제공하는 Eclipse Platform 및 기타 프로젝트를 다운로드하자.
- IBM 제품 평가판을 다운로드하여 DB2®, Lotus®, Rational®, Tivoli®및 WebSphere®의 애플리케이션 개발 도구 및 미들웨어 제품을 사용해 볼 수 있다.
- DVD로 제공되거나 다운로드할 수 있는 IBM 시험판 소프트웨어를 사용하여 후속 오픈 소스 개발 프로젝트를 구현해 보자.

토론

- Eclipse에 관해 궁금한 점이 있으면 가장 먼저 Eclipse Platform 뉴스그룹을 살펴보자. (이 링크를 선택하면 기본 Usenet 뉴스 리더 애플리케이션이 실행되면서 eclipse.platform이 열린다.)
- Eclipse 뉴스그룹에는 Eclipse 사용 및 확장에 관심이 있는 사용자를 위한 리소스가 많이 있다.
- developerWorks 블로그를 통해 developerWorks 커뮤니티에 참여할 수 있다.

필자소개

Prashant Deva는 Placid Systems의 창립자이며, Eclipse용 ANTLR Studio 플러그인의 제작자다. ANTLR과 Eclipse 플러그인 개발에 관한 컨설팅도 한다. ANTLR과 Eclipse 플러그인 개발에 관한 여러 글을 썼으며, Eclipse 개발 팀에 많은 아이디어와 버그 리포트를 기여하고 있다. 지금은 멋진 차세대 개발 도구를 만드느라 여념이 없다.

닫기 [x]

디벨로퍼웍스 로그인

IBM ID:
IBM ID가 필요하세요?
IBM ID를 잊으셨습니까?

비밀번호 :
비밀번호를 잊으셨습니까?
비밀번호 변경

로그인한 상태로 머물기.

developerWorks 이용 약관에 동의하시는 경우 **제출**을 클릭하십시오. 이용 약관.

제출 취소

developerWorks에 처음 로그인하면 developerWorks프로파일이 생성됩니다.**귀하의 프로파일에서 동의하신 내용이 공개되지만 이 사항은 언제든지 변경 가능합니다.** 귀하의 성명(숨김으로 체크되어 있어도 표시됩니다)과 디스플레이 이름은 게시한 컨텐츠나 사이트 엑세스시 표시됩니다.

모든 정보가 안전하게 전송되었습니다.

닫기 [x]

화면상에 보여지는 닉네임을 정하세요.

처음 developerWorks에 로그인할 때 프로파일이 작성되므로, 이를 위해 디스플레이 이름을 선택해야 합니다. 선택하신 디스플레이 이름은 developerWorks에 게시한 컨텐츠에 표시됩니다.

3글자 이상 31글자 이하의 길이로 사용 가능합니다. dW커뮤니티 내에서는 보안상 이메일주소를 제외한 다른 이름을 지정하셔야 합니다.

디스플레이 이름 : 3개의 &이나 대쉬를 포함해주시고 31글자내로 제한해주세요.

developerWorks 이용 약관에 동의하시는 경우 **제출**을 클릭하십시오. 이용 약관.

제출 취소

모든 정보가 안전하게 전송되었습니다.

평균 평가 등급 (0 투표하기)

별 1 개

1 개
별 2 개

2 개
별 3 개

3 개
별 4 개

4 개
별 5 개

5 개

제출

의견 추가:

로그인 or 가입 댓글남기기.

댓글사이에 HTML은 지원되지 않습니다.

댓글이 달리면 알려줍니다.1000글자가 남았습니다.

의견 올리기

모든 댓글들 (1)

Thanks~ Eclipse ShortCut Key

04일 12월 2011년 캔파스님

부정사용 신고

이 페이지를 인쇄 이 페이지 공유 팔로우하기

developerWorks에 대하여
Help
컨텐츠 제출
피드와 앱
불건전 메세지 신고
이용 약관
개인정보 보호정책
Accessibility (영어)
Academic initiative
솔루션파트너 프로그램