SVN, Mantis, JFeature, JUnit 사용법 및 CTIP 개론

Software Modeling & Analysis

소프트웨어 모델링 및 분석

보고서 #2

Team. T1

201111388 조연호

201211374 이창오

201211379 장종훈

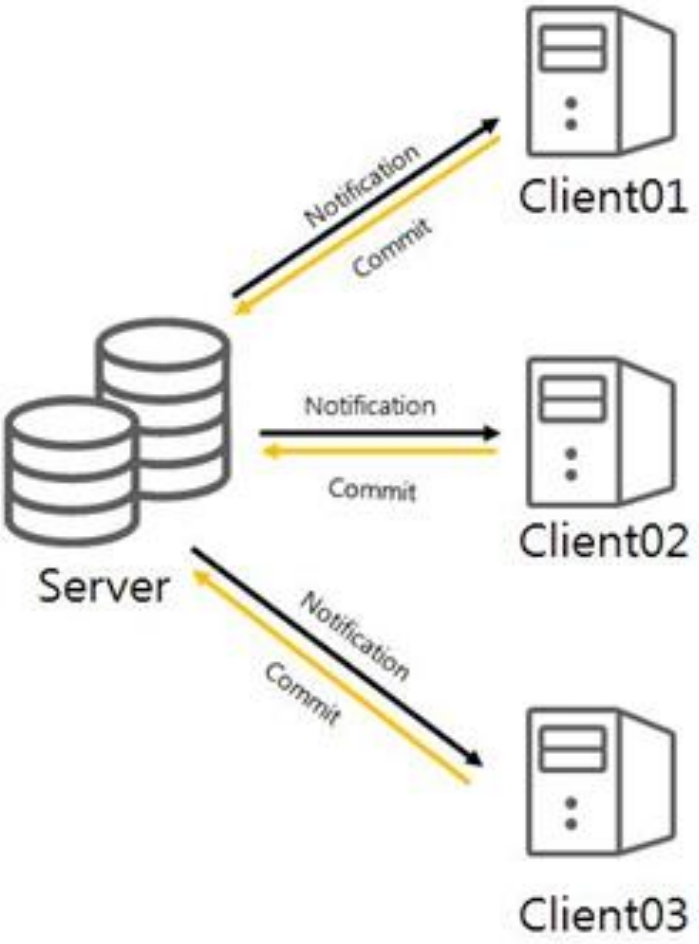
201314196 양동혁

목차

1. SVN

A. 소개

SVN은 자유 소프트웨어 버전 관리 시스템이다. 원래 CVS를 사용했으나 디렉토리의 이동이나 이름 변경 등과 같은 한계점 때문에 SVN이라는 개량 시스템이 나오게 되었다. 한 프로젝트의 소스 코드를 하나의 중앙 저장소(서버)에 위치하고, 해당 저장소는 포함하고 있는 파일과 디렉토리의 모든 변경 사항을 기억하고 있게 된다.



또한 서브버전은 여러 컴퓨터에서 네트워크를 통해 접근할 수 있으며, 다수의 사람들이 파일을 수정할 수 있는 협업을 가능하게 한다. 모든 작업에 대해서는 버전이 자동으로 매겨지므로 잘못된 수정이 이뤄졌어도 쉽게 되돌릴 수 있다.

B. 사용 모델

SVN은 Copy-Modify-Merge 모델을 사용한다.

각각의 사용자는 서버 저장소에 개인적인 작업 복사본을 만들어 작업하며 수정이 완료된 시점에서 저장소에 변경사항을 반영한다. 또한 변경 대상에 대해 다른 수정

자에 의한 변경사항이 존재한다면 자신의 복사/수정본과 병합할 수 있다. 병합 시 새로운 변경사항 중 자신이 수정한 부분과 겹치는 부분에 대해서는 사용자끼리 의견 교환 후 수동으로 선택한다.

이렇게 변경된 작업들은 위에서 말한 것처럼 버전이 자동으로 매겨져 저장이 되는데, 버전 관리를 하는 이유는 다음과 같다.

- 여러 사람이 한 프로젝트의 소스 코드를 관리할 수 있다.
- 대규모 수정 작업을 안전하게 진행할 수 있다.
- 소스 코드의 변경 사항을 추적 할 수 있다.
- 새롭게 개발한 부분을 검증 후 프로젝트에 병합 할 수 있다.
- 프로젝트의 영향을 최소화 하면서 새로운 부분 개발을 할 수 있다.

C. 기능

i. Directory Versioning

가상 버전 파일 시스템을 구현하여, 디렉토리 트리 전반의 모든 변화를 기록한다. 그리고 파일과 디렉토리에 함께 버전이 부여된다.

ii. 실제적인 버전 히스토리

파일과 디렉토리 모두 추가, 삭제, 복사, 이름 변경이 자유롭다. 또한 새로운 파일이 추가될 때마다 해당하는 새로운 히스토리가 시작된다.

iii. 원자적 Commit

개별적인 수정 사항들이 모두 저장소에 적용이 되든지, 아니면 하나도 적용이 되지 않는다. 이는 사용자들이 수정 사항들을 논리적 단위로 적용하여, 수정 사항의 일부만 적용되어 발생되는 문제를 예방해준다.

iv. 버전화된 메타 데이터

각각의 파일과 디렉토리는 키와 값의 속성 집합을 가지고 있어서 임의의 키와 값을 저장할 수 있다. 이 속성 값 또한 파일과 같이 계속 버전화 된다.

v. 네트워크 레이어의 선택

SVN은 새로운 네트워크 메커니즘을 구현하기 쉽도록 하는 추상적 저장소 접근법을 가지고 있으며 Apache HTTP 서버에 확장 모듈로 플러그인 될 수 있기 때문에 견고하고 상호 동작이 쉽다.

vi. 일관된 데이터 핸들링

SVN은 파일의 차이점을 바이너리 대조 알고리즘을 이용하기 때문에 텍스트와

바이너리 파일 모두에서 동일하게 동작한다. 두 종류의 파일 모두 저장소에 압축되어 저장되며 차이점은 네트워크를 통해 양방향으로 전송된다.

vii. 효율적인 브랜칭과 태깅

SVN은 브랜치와 태그를 단순히 프로젝트를 복사하는 방법으로 생성하므로 시간이 절약된다.

D. 용어

용어	설명
Repository	서버 저장소
Working Copy	Repository에서 받은 개인적인 작업 복사본
Import	최초로 Repository 에 자료를 올리는 행위
Export	Repository에서 버전 관리 정보를 제외한 자료를 내려 받는 행위
Check Out	Repository에서 버전 관리 정보를 포함한 자료를 내려 받는 행위
Commit	개인 작업 영역에서의 변경내역을 Repository에 반영
Revision	갱신 번호. 저장소에서의 수정이 발생할 때 마다 자동으로 증가
Trunk	개발 가지들 중에 가장 중심이 되는 줄기
Branch	특정 버전으로의 뻗어나가기(부분 개발)
Merge	두 저장소의 트리를 비교하고 그 차이점을 작업본에 적용
Tag	개발 버전의 스냅샷

E. SVN 툴

i. TortoiseSVN

Windows 기반의 SVN 툴로 SVN의 Client 역할을 해준다. 자체 동작 기능을 가지고 있지 않기 때문에 Windows의 Poplist 기능으로 동작한다. 자체 GUI를 가지고 있다.

ii. SVN X

Mac OS에서 사용되는 SVN 툴로써 GUI로 구현되어있다.

F. 사용 방법

i. Check Out

Check Out은 저장소에 있는 파일을 최초로 사용자의 컴퓨터로 가져올 때 실행하는 명령어로써 최초 한 번만 사용한다.

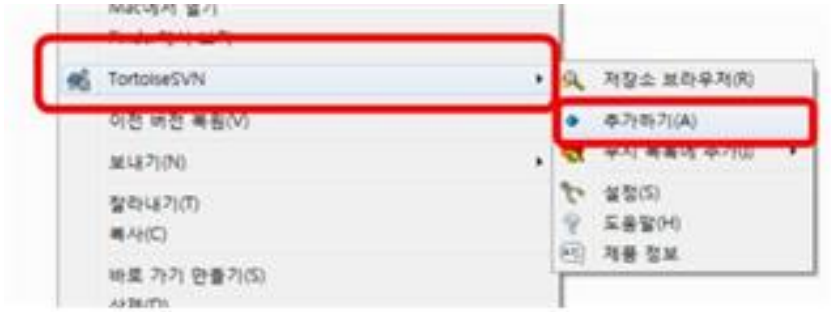
URL of repository에 서버 도메인 또는 IP/서버DP 생성한 저장소 폴더를 입력한다.

계정을 입력한다.

완료 시 위와 같이 폴더 이미지가 변화한다.

ii. Commit

Check Out 한 폴더에 원하는 파일을 만든 후 파일 추가하기를 선택한다.



SVN Commit을 선택한다.

Commit이 완료되면 Revision이 올라간다.

iii. Update

저장소에 저장된 최신의 Revision으로 자신의 소스를 변경하는 작업이다. 마우스 오른쪽 클릭 시 나타나는 SVN 업데이트를 선택한다.

계정 입력 시 자신의 파일이 저장소의 파일로 변경된다.

2. Mantis

A. 소개

Mantis는 웹 기반의 무료 버그 관리 시스템이다. 설치가 쉽고, 사용하기 편리하여 이슈를 관리해야 하는 공동 작업 프로젝트 개발 시 유용하게 사용되고 있다.

Kenzaburo Ito가 개발한 Mantis는 PHP 스크립트 언어로 작성되었고, 웹 서버(Apache, IIS)와 MySQL이 설치되어 있는 Windows, OS/2, MacOS, UNIX 등의 환경에서 사용할 수 있다. 웹 기반의 시스템이기 때문에 대부분의 웹 브라우저에서 문제없이 기능 수행이 가능하다. GNU General Public License(GPL) 라이선스에 의거한 오픈 소스가 특징이며, 현재 버전 1.2.19가 최신 버전이다.

B. 특징

i. 복수 프로젝트 대응

Mantis는 한 번의 설치로 여러 개의 프로젝트를 생성하여 관리할 수 있다. 또한 부모 프로젝트 하위에 한 단계 낮은 레벨의 자식 프로젝트를 생성하여 구조적인 관리가 가능하다. 카테고리 기능도 지원하여 프로젝트 내 이슈에 쉽게 접근할 수 있다.

ii. 국제화 대응

오픈 소스인 Mantis는 세계 각국의 개발자들로부터 유지 보수가 되고 있다. 포럼은 물론이고, 실제 Mantis를 활용한 버그 트래커도 직접 운영하고 있어 다양한 언어를 완벽하게 지원한다는 특징이 있다.

iii. 이메일 통지

프로젝트에서 발생하는 다양한 이슈를 이메일로 알려주는 기능도 포함하고 있다. 이 기능을 통해 개발자는 놓치는 이슈 없이 즉각 대응할 수 있다는 장점이 있다.

iv. 커스텀 필드 추가 기능

Mantis는 프로젝트 별로 개발자가 임의로 생성하는 커스텀 필드 추가 기능을 제공한다. 문자 또는 숫자, 부동소수, 체크 박스, 날짜 등 다양한 형식을 지원하여 개발자들의 편의를 도모한다.

v. 변경 이력 관리

버그 추적 시스템의 필수 기능인 프로젝트 변경 이력 관리 기능도 강력하다. 프로젝트에 참여한 개발자들의 작업 내용을 쉽게 관리할 수 있고, 프로젝트 내 변경 사항이 발생할 때마다 변경 사항을 항시 기록한다.

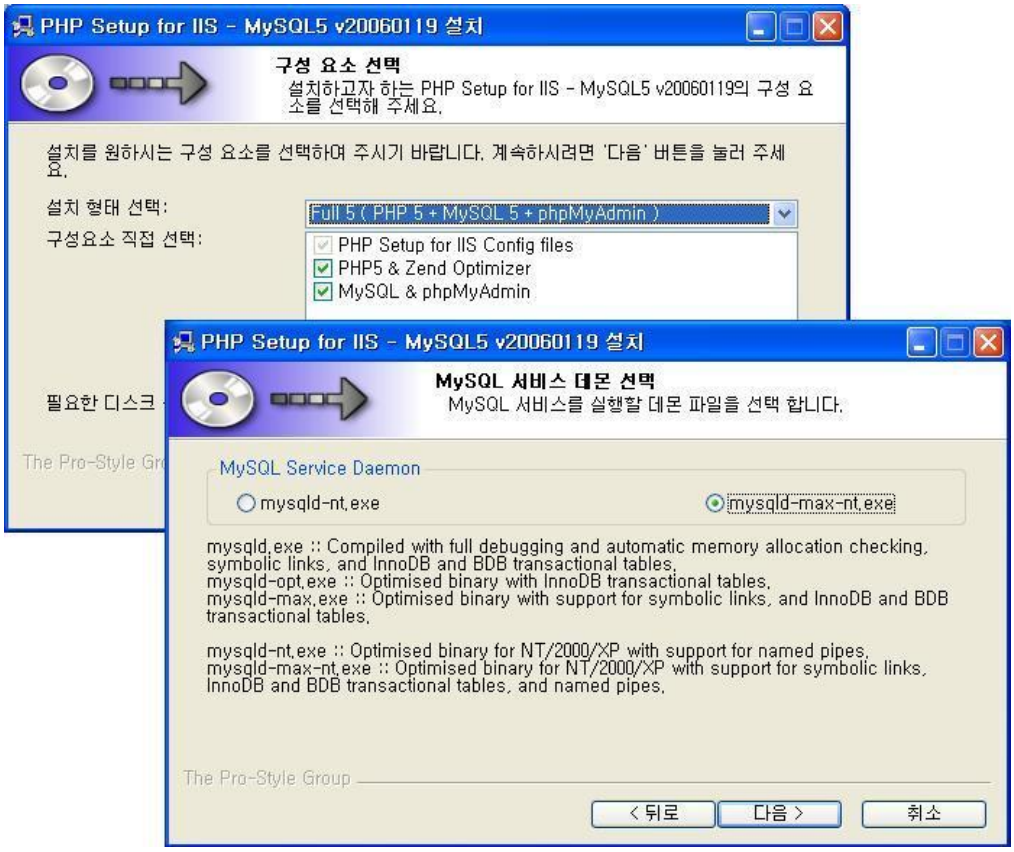
vi. 다양한 출력 형식

Mantis는 CSV 또는 Microsoft Excel, Word 형식으로 내보낼 수 있는 기능을 지원한다. 언제든지 작업 내용을 다양한 형식으로 출력할 수 있으며, 웹 서버가 오프라인일 경우에도 활용할 수 있다는 장점이 있다.

C. 설치 및 실행

다음은 Windows 기반에서 PHP와 IIS, MySQL 환경으로 Mantis를 설치하는 과정이다.

i. PHP 및 MySQL 설치

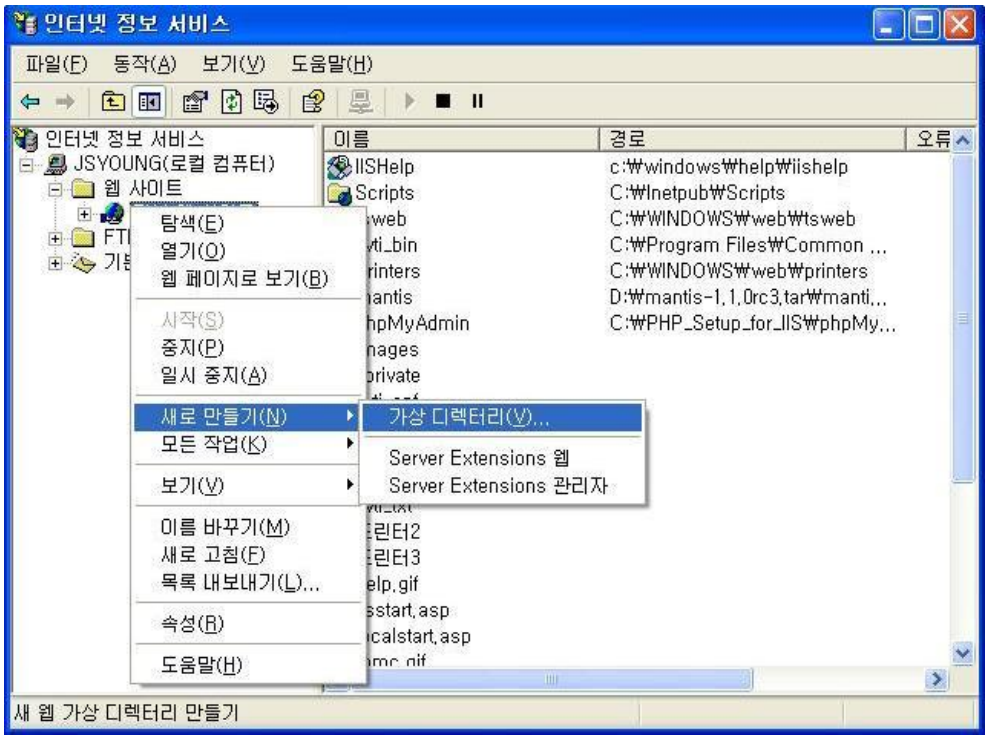


IIS 설치 후 PHP 및 MySQL을 설치한다. 개별로 다운 받아 설치하는 방법도 있지만, 'PHP Setup for IIS'라는 프로그램을 활용하면 두 가지를 한 번에 설치할 수 있어 편리하다.

ii. Mantis 설치

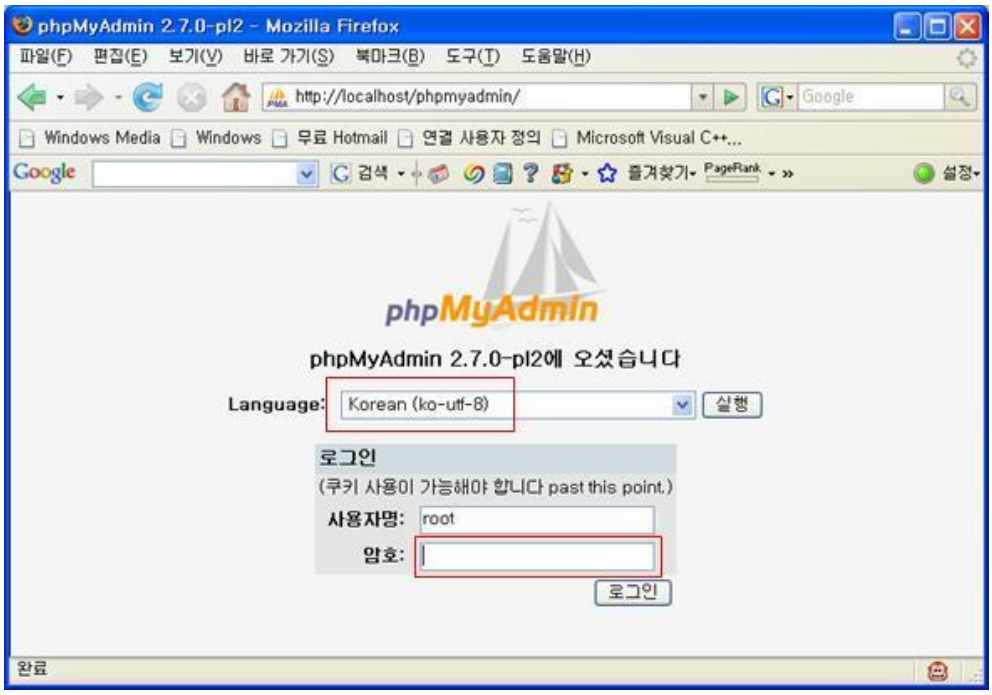
Mantis 공식 홈페이지로 들어가 최신 버전의 Mantis를 내려 받는다. 따로 설치 과정 없이 압축만 풀어서 사용하면 된다.

iii. IIS 설정

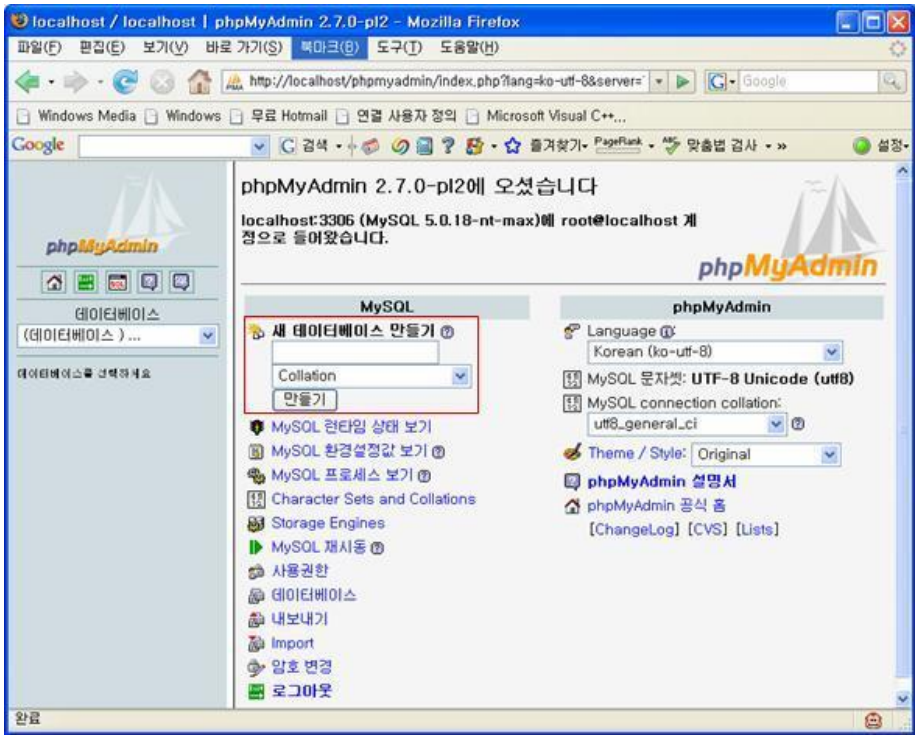


가상 디렉토리 이름을 입력하고, 압축을 푼 Mantis의 경로를 지정한다.

iv. DB 생성



Language를 'Korean (ko-uft-8)'로 수정하고, 초기 암호 '123456'을 입력하여 로그인 한다.



'새 데이터베이스 만들기'에서 'Mantis' 혹은 원하는 이름을 입력하여 새로운 데이터베이스를 생성한다. 아래의 '암호 변경'을 선택하여 초기 암호 '123456' 대신 자신만의 암호를 사용하는 것이 보안 상 안전하다.

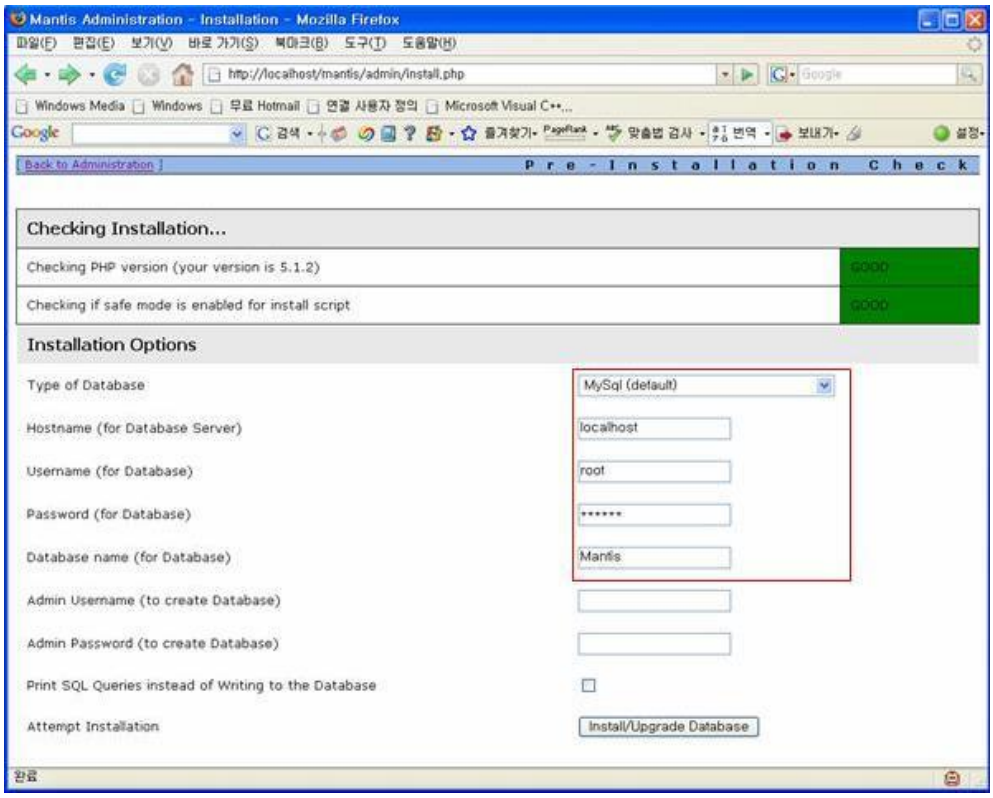
v. Mantis 설정

Mantis 설치 경로로 이동하여 'config_default_inc.php' 파일을 'config_inc.php' 파일로 복사한다. 그리고 다음과 같이 수정한다.

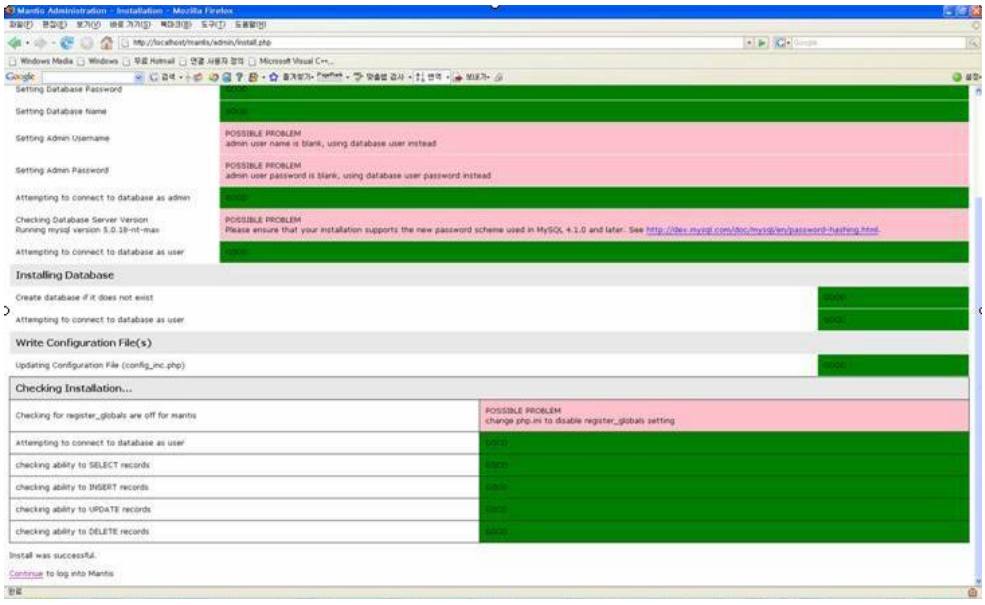
```
[DB 설정]
$g_hostname = 'localhost';
$g_db_username = 'root';
$g_db_password = '';
//DB 생성 과정에서 암호를 변경하였다면 변경한 패스워드를 입력한다. (초기 암호 '123456')
$g_database_name = '';
//DB 생성 과정에서 생성한 새로운 데이터베이스 이름을 입력한다.
$g_db_type = 'mysql';

[Mail 설정]
$g_phpMailer_method = 2;
$g_smtp_host = 'localhost';
$g_smtp_username = 'syjung'; //사용자에 맞도록 메일 설정
$g_smtp_password = 'zero04'; //사용자에 맞도록 패스워드 설정
```

vi. Mantis – DB 연동 테스트

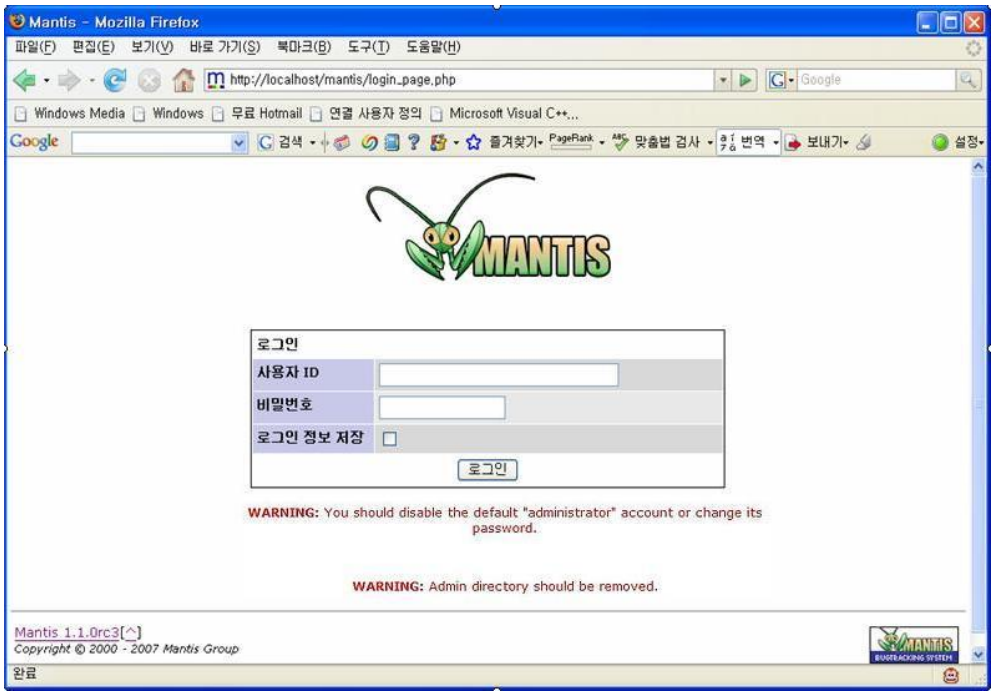


Type of Database와 Hostname, Username, Password, Database name을 입력하고 'Install/Upgrade Database' 버튼을 누른다.



위와 같은 화면이 출력된다면 정상적으로 연동이 되었음을 의미한다. 아래의 'Continue'를 클릭하여 Mantis를 사용할 수 있다.

vii. Mantis 시작



최초 시작 시 두 가지의 오류 메시지가 출력된다. Administrator의 암호를 변경한 후 Mantis 설치 경로에서 Admin 폴더를 삭제하면 두 가지의 오류 메시지 모두 처리되어 정상적으로 사용할 수 있다.

3. JFeature

A. 개요

JFeature는 Technobuff사에서 제작된 오픈 소스로, 개발한 코드의 요구사항에 집중할 수 있는 Feature/Requirement Coverage Tool이다. 요구사항들을 기록하고, 이들을 JUnit Test Case와 매치함으로써 프로세스를 단순화할 수 있는 도구이다.

B. 소개

요구사항들을 기록하고 이들을 JUnit Test Case와 매치함으로써 프로세스를 단순화할 수 있다.

i. 주요 기능

요구사항 기록 및 Report 생성

ii. 카테고리

Requirement Analysis & Management

iii. 세부 카테고리

요구사항 모델링

iv. 라이선스 형태 및 비용

Common Public License 1.0 – 무료

Commercial License – 유료

v. 사전 설치 도구

Eclipse 3.4 ~ 4.3

vi. 운영체제

Windows, Linux, UNIX, Mac OS X

vii. 장점

1. JUnit의 Test Case 메소드와 통합
2. 요구사항의 수정사항 발생 시 즉시 뷰를 제공하므로 빠른 편집 가능
3. 요구사항의 반영 여부를 Coverage 형태로 쉽게 확인 가능
4. 요구사항들을 편집하는 확장 인터페이스 추가 리포팅 포맷(Commercial Version의 추가 기능)

5. 요구사항 Spec 반출(Commercial Version의 추가 기능)

viii. 관련 도구

IBM Rational DOORS, CaliberRM, RequisitePro

ix. 제작사

Technobuff (http://www.technobuff.net)

C. 특징

i. 개발 중간에 요구사항에 맞는 기능을 확인할 수 있다.

ii. 이클립스 플러그인을 통하여 요구사항을 편리하게 관리하는 편집 창과 요구사항에 대한 코드의 커버리지를 볼 수 있는 뷰를 제공한다.

iii. JUnit과 연동하여 요구사항에 대한 Break, No Coverage 등과 같이 요구사항 커버리지에 대한 자세한 보고서를 제공한다.

iv. 커스텀 Ant와 연동하여 자동화된 빌드에 맞춰서 전체 프로젝트에 대한 요구사항 커버리지를 만들어낼 수 있다.

v. Round Trip Engineering을 통해 프로젝트 진행 중간에 외부 요구사항 파일과 JFeature의 요구사항 파일을 동기화 시킨다.

vi. 리팩토링으로 단위 테스트의 변경에 따라 요구사항 파일을 최신상태로 유지한다.

vii. CSV, XML 파일로부터 요구사항들을 추출한다.

viii. 테스트 메소드 에디터를 통하여 요구사항에 맞는 테스트 메소드를 정의한다.

ix. 요구사항 셀렉터를 통하여 요구사항 간의 의존성을 정의한다.

D. 실행 환경

i. OS

1. Windows XP, 7 : 32, 64-bit 지원
2. Linux : 32, 64-bit 지원
3. UNIX : 32, 64-bit 지원
4. Mac OS X : 32, 64-bit 지원

ii. JDK

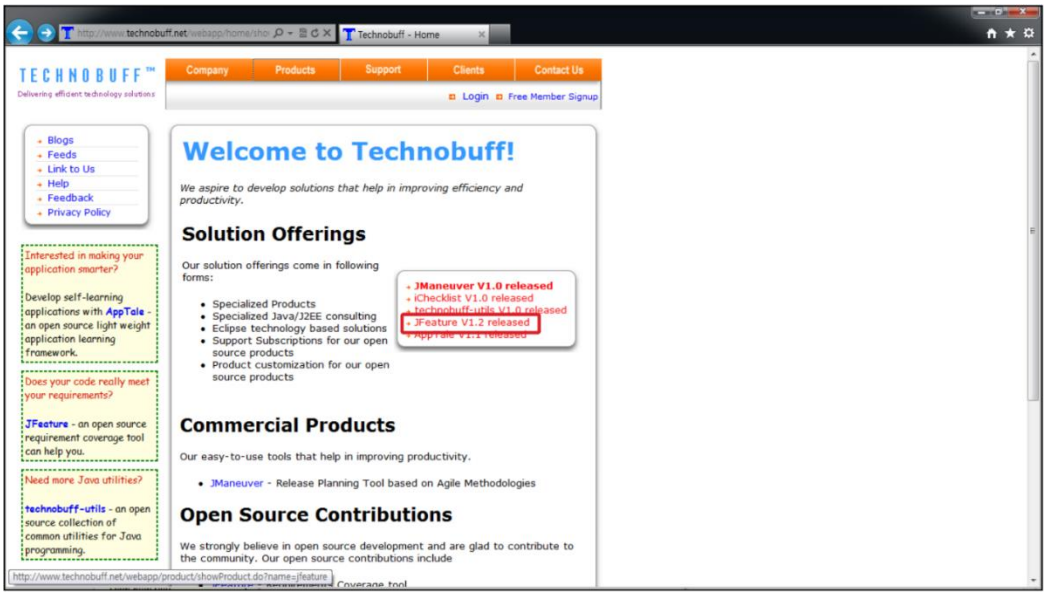
iii. Eclipse

코드 및 Eclipse 플러그인, 도구에 포함되어 있는 형태로 제공된다.

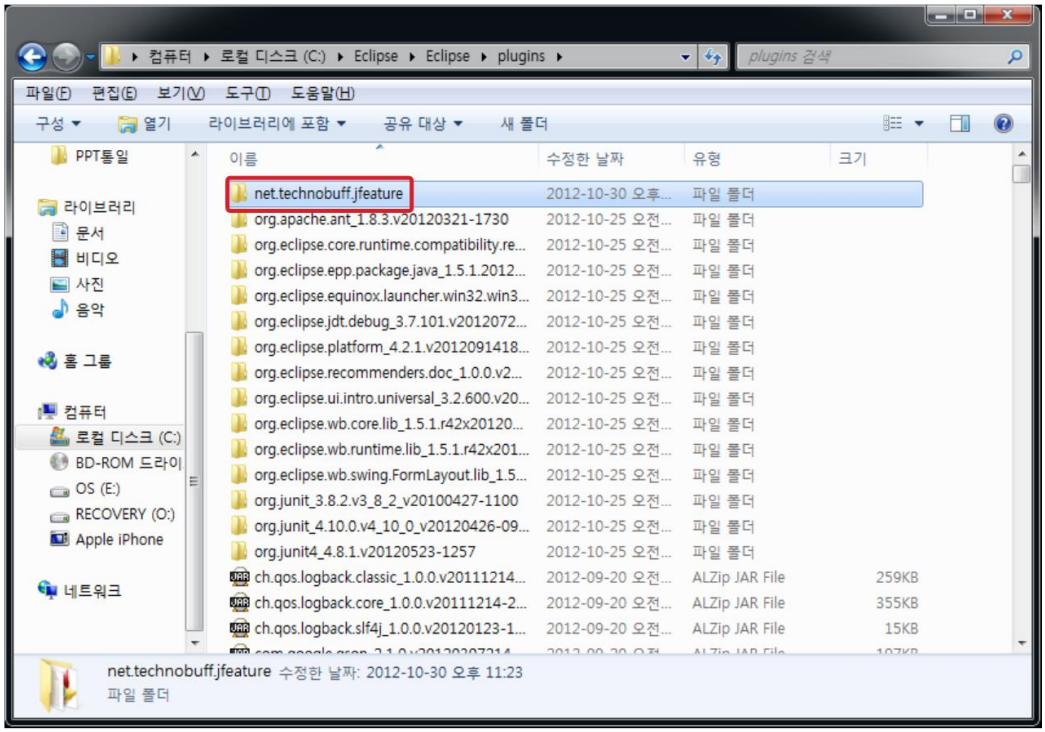
E. 기능

기능	지원 여부
요구사항 반영 여부 확인	지원(Coverage 형태)
타 플러그인 환경 지원	지원(JUnit)
코드 추적	지원
IDE 내 별도의 편집기 제공	지원
요구사항 Spec 반출	일부 지원(Commercial Version)
리포팅	기본 포맷 지원(Commercial Version)
Round Trip Engineering	지원(파일 보관 형태)

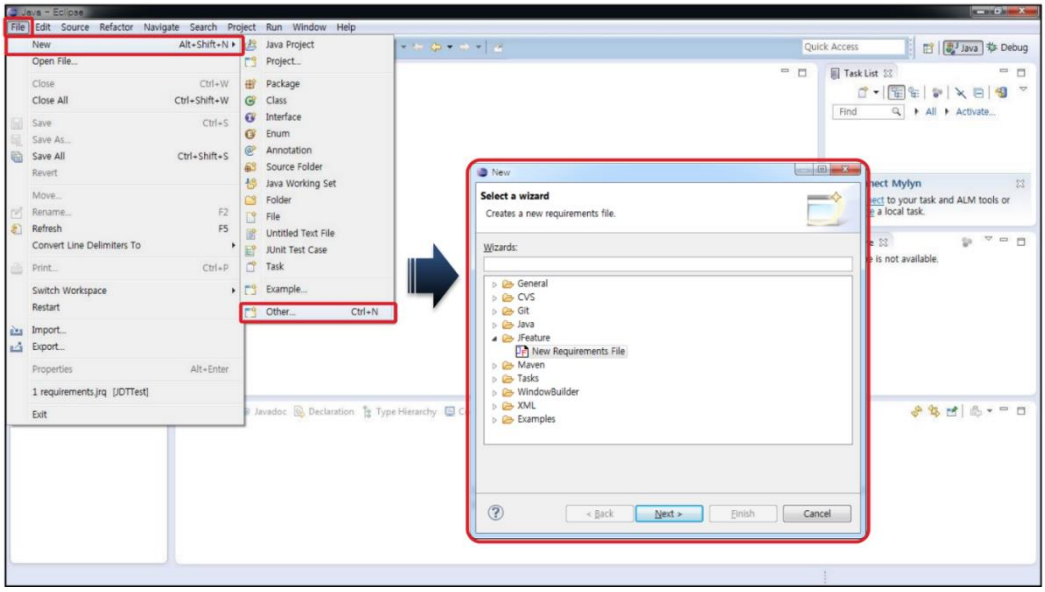
F. 설치 및 실행



Technobuff 홈페이지에서 JFeature를 다운로드한다.



다운로드 받은 압축 파일의 압축을 풀고, JFeature1.2를 Eclipse가 설치된 폴더 내의 'plugins' 폴더에 복사한다. (Eclipse 버전 3.4 ~ 4.3만 지원한다.)



Eclipse를 실행하여 추가된 플러그인으로 '새로 만들기' 한다.

4. JUnit

A. 소개

JUnit은 우리가 만든 프로그램이 원하는 방향으로 동작하는지 기대값과 결과값을 비교하여 알아볼 수 있는 테스트 툴이다.

대형 프로젝트의 경우 시나리오와 로직이 나온 후 로직의 모든 문제를 테스트하는 것은 시간적으로 문제가 많다. 또한 클래스 파일을 테스트할 때, 반드시 main 메소드를 만들어 겉으로만 테스트 하는 것은 엄연한 의미에서 테스트라고 보기 힘들다.

JUnit은 작은 범위만을 테스트한 후 그 범위들을 모아 전체적으로 이상이 없다고 판정을 내려준다. 비행기로 예를 들자면 비행기를 만들어 직접 하늘에 날게 하는 것이 아니라 부품 또는 기능마다 하나씩 살펴보고, 모든 것들이 각자 이상이 없다면 비행기 자체도 결합이 없다고 결론을 내리는 방식이다.

Test Case는 일련의 테스트를 실행하기 위한 장치로서 Fixture와 기능, 원시코드 경로, 멤버 함수간의 상호작용을 정의하는 것이다. 전형적으로 작성한 모든 클래스는 Test Case를 가지고 있어야 한다. 테스트 Fixture는 테스트 수행에 필요한 자원인 Primitive 변수와 오브젝트를 제공하는 것이다. 동일하거나 유사한 오브젝트에 대한 테스트가 두 개 이상 있을 경우 테스트 환경을 Setup하기 위한 코드를 각 테스트에서 꺼내어 하나의 메소드에 넣어둔다. 동일하거나 유사한 환경에서 실행되는 테스트를 위한 설정을 테스트 Fixture라고 한다. Test Suite는 관련된 테스트 케이스를 모아 놓은 것을 말한다.

B. 사용 방법

JUnit 홈페이지(http://www.junit.org)에서 'junit.jar' 파일을 다운로드 하여 자바 Class Path에 jar 파일을 설정한다.

아래의 소스 코드는 JUnit을 이용한 Test Case를 extend 하여 메소드를 테스트하는 소스 코드이다.

```
package tddbook;
import junit.framework.*;
import junit.textui.*;

public class JUnitTutorialTest extends TestCase {
    public JUnitTutorialTest(String arg0) {
        super(arg0);
    }

    public void testNumber() {
```

```
        int expected = 10;
        assertEquals(expected, 2*5);
    }

    public static void main(String[] args) {
        TestRunner.run(JUnitTutorialTest.class);
    }
}
```

위와 같은 소스 코드가 전형적인 JUnit을 이용한 소스 코드의 틀이라고 할 수 있다. 위의 testNumber가 실제적인 테스트를 행하는 메소드이며, 이렇게 메소드명이 'test'로 시작하는 메소드들은 원하는 만큼 만들어서 사용할 수 있다.

testNumber 메소드를 보면 'assertEquals'라는 Test Case를 통해서 extend 받은 메소드를 이용하여 2*5의 결과값이 기대한 값(expected)와 일치하는지 비교한다.

소스 코드 실행 결과는 아래와 같다.

```
Time: 0.01
OK (1 test)
```

자세히 보면 가장 윗줄에 점(.)을 볼 수 있는데, 이것은 'test'로 시작하는 메소드들의 개수(테스트의 개수)를 의미한다. Time은 테스트 하는 데 소요된 시간을 의미하며 'OK'는 1개의 테스트가 성공했음을 알려준다. 이렇듯 텍스트로 그 결과를 보여주는 까닭은 우리가 main 메소드에서 junit.textui.TestRunner을 사용했기 때문이며 이외에도 awt나 swing을 이용한 비주얼적인 결과를 볼 수도 있다.

JUnitGui – awt, swing을 이용한 유닛 테스팅

```
package tddbook;
import junit.framework.TestCase;

public class JUnitTutorialTest extends TestCase {
    public JUnitTutorialTest(String arg0) {
        super(arg0);
    }

    public void testNumber() {
        int expected = 10;
```

```
        assertEquals(expected, 2*5);
    }

    public void testFailMessage() {
        int expected = 10;
        assertEquals(expected, 3*5);
    }
    public static void main(String args[]) {
        junit.textui.TestRunner.run(JUnitTutorialTest.class);
    }
}
```

위의 소스 코드는 testFailMessage 메소드를 추가한 소스 코드이다. 소스 코드를 보면 expected 값은 10이지만, 3*5의 값은 10이 아니다. 위의 소스 코드를 실행하면 아래와 같은 실행 결과가 나타난다.

```
..F
Time: 0.01
There was 1 failure:
1) testFailMessage(tddbook.JUnitTutorialTest)junit.framework.AssertionFailedError:
expected:<10> but was:<15>
        at tddbook.JUnitTutorialTest.testFailMessage(JUnitTutorialTest.java:17)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
        sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
        ava:39)
        at
        sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcces
        sorImpl.java:25)
        at tddbook.JUnitTutorialTest.main(JUnitTutorialTest.java:21)
FAILURES!!!
Tests run: 2,   Failures: 1,   Errors: 0
```

두 개의 점은 테스트의 개수를 뜻하며 오른쪽의 'F'는 테스트가 실패(Fail)되었음을 의미한다. 'There was 1 failure:' 아래에는 테스트의 실패 이유와 Trace가 나타난다. 예상대로 기대값과 결과값이 달라 'AssertionFailedError'가 발생했음을 알려준다. 마지막 줄은 총 2개의 테스트 중 1개의 Fail이 있고, Error는 0개임을 말한다. 테스트 코드에서 Fail과 Error는 다르다. Fail은 우리가 테스트한 기대값과 결과값이 다를 때 발생하

지만, Error는 코드상의 오류나 'NullPointerException'와 같이 예측 못한 Exception이 발생할 때 나타난다.

setUp

JUnit 테스트 코드의 setUp 메소드는 특별한 의미를 지니고 있다. 주로 소스 코드 내에서 사용할 리소스를 초기화 시킬 때 setup 메소드를 이용한다. 즉, 각각의 테스트 코드가 항상 'new Person()'이라는 statement를 실행한다면 이것을 setup 메소드에 선언하여 테스트 메소드가 실행될 때마다 수행하게끔 할 수 있는 것이다. 결국 setup 메소드는 각각의 테스트 메소드들이 수행되기 바로 직전에 매번 실행되는 것이다.

tearDown

setup 메소드와는 반대로 테스트 메소드가 종료될 때마다 수행되는 메소드이다. 사용한 리소스를 Clear 할 때 주로 사용된다.

C. Useful Methods

메소드명	기능
assertTrue(x);	x가 참인지 테스트한다.
assertFalse(x);	x가 거짓인지 테스트한다.
assertNull(x);	x가 Null 값인지 테스트한다.
assertNotNull(x);	x가 Null 값이 아닌지 테스트한다.
fail(msg);	무조건 실패시키고 msg를 출력한다. 주로 Exception 테스트 시 사용한다.

5. CTIP

A. 소개

CTIP은 지속적 통합(Continuous Integration, 이하 CI) 개념을 Java 기반의 개발 프로젝트에 쉽게 적용하기 위한 플랫폼이다.

CI는 수년 전부터 많은 이슈가 되어왔고, 상당한 성숙단계에 이른 개발 활동(Development Practice)이다. CI는 대표적인 Agile 개발 프로세스인 Extreme Programming의 개발 Practice 중 하나로 포함되어 있다. 손쉽게 CI를 적용하기 위한 다양한 CI 서버들이 개발되었고, 현재 많은 프로젝트에 도입되어 성공적인 프로젝트 수행에 공헌을 하고 있다.

기존의 많은 개발 프로젝트들의 막바지 통합 과정은 참으로 고통스러운 과정이었다. 개발자의 환경에서는 정상적으로 실행되던 코드들을 통합 서버에 올리기만 하면 알 수 없는 에러들이 발생했고, 모듈 간 약속된 인터페이스는 갑자기 변경되어 서로 연동이 되지 않는 문제가 생겼다.

이러한 통합 과정의 문제를 해결하기 위해 Martin Fowler의 글 'Continuous Integration'에서 기존 개발 단계의 후반에 수행되던 통합 활동을 일상적인 개발 활동에 자연스럽게 녹이는 CI를 해결책으로 제시한다. CI 활동을 간단히 설명하면 다음과 같다.

- 개발자는 소스 저장소로부터 최신 소스를 내려 받는다. (Check Out)
- 개발자는 코드를 작성한 후 자신의 개발기에서 정상적으로 동작하는 지 충분히 검증한다.
- 검증이 끝난 후 개발자는 작업 내용을 소스 저장소에 올린다. (Check In)
- 개발자(또는 통합 관리자 또는 자동화된 시스템)는 통합 서버에서 방금 작업한 코드가 반영된 전체 코드 내용에 대한 빌드를 수행한다.
- 만약 빌드가 실패할 경우 실패 원인을 분석하고, 문제를 해결하여 빌드가 성공될 수 있도록 한다.

CI가 이상적으로 수행된다면 개발 막바지의 통합 작업이 필요 없게 된다. 통합 서버의 최신 빌드에는 개발자가 작업한 내용들이 자연스럽게 통합이 되어 있기 때문이다. 2008년 JOLT 상을 수상한 책인 'Continuous Integration: Improving Software Quality and Reducing Risk'에서는 CI의 수행을 통한 주요 이점을 다음과 같이 제시하고 있다.

- 위험을 줄일 수 있다.
- 수동으로 수행해야 하는 반복 작업을 줄일 수 있다.

- 언제, 어느 장소에서도 배포 가능한 소프트웨어를 만들 수 있다.
- 프로젝트에 대한 더 나은 가시성을 제공한다.
- 개발팀에게 소프트웨어 제품에 대한 자신감을 불어넣는다.

이러한 목적을 달성하기 위해서 Martin Fowler가 제시하는 성공적인 CI 수행의 요건은 다음과 같다.

- 단일 소스 저장소(Source Repository)를 유지하라.
- 빌드를 자동화하라.
- 빌드가 자체적으로 테스트 가능하도록 하라.
- 모든 사람은 매일 작업 내용을 커밋(Commit) 하라.
- 모든 커밋들은 통합 서버의 메인 라인에 반영되어야 한다.
- 각 빌드가 빨리 수행되도록 하라.
- 운영환경과 되도록 비슷한 환경에서 테스트 하라.
- 최신 결과물에 쉽게 접근할 수 있도록 하라.
- 현 빌드 상황을 쉽게 알 수 있도록 하라.

위의 요건 중 일부는 개발팀의 업무 프로세스 정의와 같은 정책적인 부분이고, 일부 요건들은 시스템적으로 뒷받침되어야 하는 부분이다. 이 중 시스템적으로 뒷받침되어야 하는 부분을 구현한 것이 CI 서버, 소스 버전 관리 시스템과 같은 도구들이다. CTIP도 이러한 시스템적 요건을 효과적으로 구성하기 위해 고안되었다.

B. 제공 기능

i. CI 서버를 통한 지속적 통합 및 빌드

ii. 품질 도구들을 통한 코드 품질 검토(테스트 및 정적 분석)

iii. 빌드 결과의 배포 및 관련자에게 통보

C. 특징

i. 단일 소스 저장소(Source Repository) 관리

소스코드 버전 관리 시스템을 도입하여 소스코드를 일관성 있게 관리한다.

ii. 빌드 자동화

CI 서버와 Ant 빌드 스크립트를 통해 빌드를 자동화한다.

iii. 자체적으로 테스트 가능한 빌드

코드 품질 관리 도구들을 통한 단위 테스트 등의 테스트 성공 여부와 서버의 deploy 성공 여부를 통해 빌드 성공 여부를 확인한다.

iv. 빠른 빌드 수행

CI 서버와 Ant 빌드 스크립트를 통해 단계적 빌드(Staged Build)를 구성합니다.

v. 운영환경과 유사한 환경 구성

CI 서버의 환경을 운영 환경과 유사하도록 구성한다.

vi. 최신 결과물에 대한 쉬운 접근

CI 서버를 통해 최신 빌드 결과물 및 빌드 리포트를 쉽게 내려 받을 수 있다.

vii. 손쉬운 빌드 상태 모니터링

CI 서버가 제공하는 RSS feed 혹은 이메일 전송 기능을 통해 빌드 상태를 모니터링 할 수 있다.

D. 영역

i. 소스 코드 버전 관리 영역

CVS, SVN 등의 소스 코드 버전 관리 시스템을 사용하여 프로젝트 전체 소스 코드의 일관성을 유지한다. 개발자는 버전 관리 시스템의 Repository로부터 최신 소스를 Check Out 하고, 작업 내용을 Check In 한다.

ii. 빌드 및 배포 관리 영역

CI 서버를 통해 지속적으로 빌드를 수행하고, 대상 서버에 대한 배포 작업을 수행한다. 빌드 주기는 정해진 시점에 수행(Nightly build)하거나 버전 관리 시스템의 Repository에 변경이 있을 경우 즉시 수행될 수 있다.

- 소스 코드 버전 관리 시스템으로부터 최신 소스를 받아온다.
- 코드 품질 관리 도구를 실행한다.
- 빌드 결과를 개발자에게 통보한다.
- 서버에 빌드된 어플리케이션을 배포한다.

iii. 코드 품질 관리 영역

오픈 소스 코드 검토 도구를 활용하여 코드 품질을 확인하고, 결과를 개발자에게 통보한다. 다양한 오픈 소스 및 상용 품질 관리 도구가 개발되어 있으므로 필요에 따라 선택하여 적용할 수 있다.

iv. 대상 서버군

운영 서버 배포를 위한 스테이징 서버, 테스트 실행을 위한 테스트 서버 등을 운영한다.

E. 도구

CTIP의 각 영역에서는 적절한 도구를 선정하여 영역에서 필요한 기능을 제공해야 한다. 기본적으로 CI 서버와 소스 코드 버전 관리 시스템, 빌드 시스템, 품질 관리 도구가 필요하다. 다양한 오픈 소스 및 상용 도구들이 개발되어 있기 때문에 특별한 기능이 필요하지 않은 경우 굳이 이러한 도구를 개발할 필요는 없다. 도구는 도구 자체의 기능, 프로젝트 특성, 팀의 도구에 대한 익숙함 등을 고려하여 선정한다.

6. 레퍼런스

A. 도서

i. **JUnit in Action : 단위 테스트의 모든 것**
피터 타치브 외 3인 저 | 이복연 역 | 인사이트 | 2011. 7. 3

B. 웹사이트

i. **TortoiseSVN**
http://tortoisesvn.net/

ii. **Mantis**
http://www.mantisbt.org/

iii. **Mantis 포럼**
http://www.mantisbt.org/forums/

iv. **Technobuff**
http://www.technobuff.net/

v. **JUnit**
http://junit.org/junit4/