

NRI

jcoverage(Cobertura)利用ガイド



掲載した製品名はそれぞれの会社の商標、あるいは登録商標です。

この製品の仕様、およびマニュアルに記載されている事項は、将来予告なしに変更することがあります。

乱丁、落丁があった場合はお取替えいたします。

第 1 版 第 1 刷

発行日 2005 年 4 月

© Nomura Research Institute, Ltd. 2005. All Rights Reserved.

変更履歴

このマニュアルの変更内容について、説明します。

バージョン	変更内容	箇所(ページ)
2005.04 版	新規作成	

はじめに

このマニュアルは、jcoverage(および継続プロジェクトの Cobertura)のインストール方法と、これを利用したカバレッジデータの取得方法について説明します。

【対象読者】

jcoverage を使用してカバレッジデータを取得する方を対象としています。

【構成】

次に示す 5 つの章から構成されています。

第 1 章 概要

jcoverage(Cobertura)の概要について説明します。

第 2 章 前提条件

jcoverage(Cobertura)を使用する際に必要な前提条件について説明します。

第 3 章 インストール

jcoverage の環境構築方法について説明します。

第 4 章 jcoverage(Cobertura)の利用方法

jcoverage を使用してレポートを作成する方法について説明します。

第 5 章 レポート機能

レポートに関する機能について説明します。

付録 A 付録

jcoverage(Cobertura)を使用する際の注意事項と、タスクについて説明しています。

【マニュアル中の表記】

次に示す表現を使っています。

表記方法	意味
[× × × × ×]	画面名、ダイアログ名、ボタン名、メニュー名、アイコン名を示します。

関連するマニュアルとして次のマニュアルがあります。

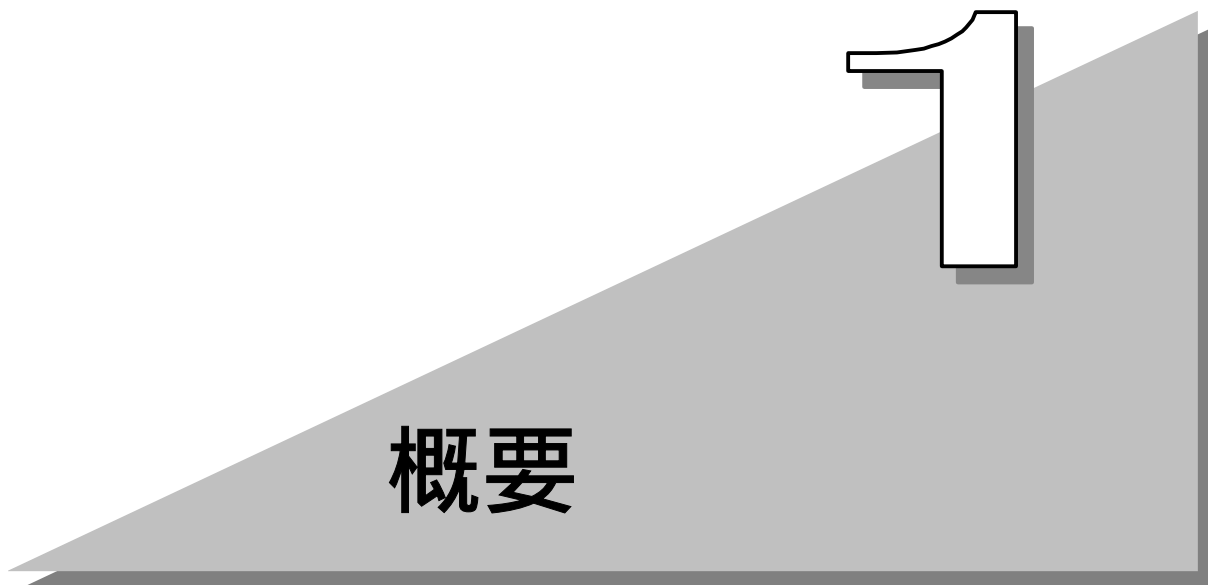
eclipse 3.0 利用ガイド(Tomcat5.0 版)

Ant on eclipse 3.0 利用ガイド

目次

1.	概要	1-1
1.1	マニュアルの位置付けと範囲	1-2
1.2	jcoverage(Cobertura)の概要	1-3
1.2.1	jcoverage(Cobertura)とは	1-3
2.	前提条件	2-1
2.1	前提条件	2-2
2.1.1	ハードウェア	2-2
2.1.2	ソフトウェア	2-2
2.2	インストール環境	2-3
3.	インストール	3-1
3.1	jcoverage(Cobertura)のインストール	3-2
3.1.1	インストール手順	3-2
4.	jcoverage(Cobertura)の利用方法	4-1
4.1	はじめに	4-2
4.1.1	概要	4-2
4.1.2	プロジェクトの構成	4-2
4.2	プロジェクト作成と各種設定	4-4
4.2.1	プロジェクトの作成	4-4
4.2.2	フォルダの作成と配置	4-4
4.2.3	プロジェクトへのクラスパスの設定	4-4
4.3	サンプルクラスの作成	4-5
4.4	テストケースクラスの作成	4-6
4.5	build.xml の記述	4-7
4.5.1	build.xml (jcoverage 使用前)	4-7
4.5.2	build.xml (jcoverage 使用)	4-8
4.5.3	jcoverage(Cobertura)定義の追加内容	4-10
4.6	Ant の実行	4-13
4.6.1	Ant のクラスパス設定	4-13
4.6.2	Ant の実行	4-14
4.6.3	実行後の出力ファイル	4-15
4.6.4	テストコードの追加	4-17
5.	レポート機能	5-1
5.1	レポートの概要	5-2
5.1.1	レポート出力までの流れ	5-2
5.2	HTML レポート	5-3
5.2.1	HTML レポートの出力	5-3
5.2.2	HTML レポートの内容	5-3
5.3	XML レポート	5-6
5.3.1	XML レポートの出力	5-6
5.3.2	XML レポートの内容	5-6
5.4	merge 機能	5-7
5.4.1	merge 機能の使用	5-7

A.	付録.....	A-1
A.1	注意事項.....	A-2
A.1.1	jcoverage を使用する際の注意事項.....	A-2
A.2	jcoverage タスク.....	A-3
1.	<check>タスク.....	A-3
A.2.1	<check>タスクの使用について.....	A-3



1. 概要

この章では、jcoverage(Cobertura)の概要および本マニュアルの位置付けについて説明します。

1.1	マニュアルの位置付けと範囲	1-2
1.2	jcoverage(Cobertura)の概要	1-3

1.1 マニュアルの位置付けと範囲

本マニュアルは、jcoverage(Cobertura)の利用方法や eclipse との連携方法を理解し、eclipse 上で jcoverage(Cobertura)の実行ができるようになることを目的としています。

jcoverage(Cobertura)と eclipse を連携させることによりテスト実施後のカバレッジ取得作業を IDE(eclipse)上で行うことができます。

第 1 章(本章)では、本マニュアルの位置付けと範囲及び jcoverage(Cobertura)の概要を記述しています。第 2 章では、本マニュアルを利用する際の前提条件について記述しています。

第 3 章では、インストール方法について記述しています。第 4 章では eclipse の jcoverage(Cobertura)連携機能を用いて jcoverage(Cobertura)を実行する方法について具体例を元に解説しています。

第 5 章では jcoverage(Cobertura)のレポート機能について記述しています。

付録 A では jcoverage(Cobertura)を使用する際の注意事項と、タスクについて説明しています。

1.2 jcoverage(Cobertura)の概要

1.2.1 jcoverage とは

jcoverage は、テストを行った際にテスト対象クラスのコードをどれだけ網羅したか(カバレッジ)をレポートとして出力できるツールです。jcoverage(Cobertura)の特徴として、出力レポート形式はHTML形式とXML形式の2種類があります。当初はGPLライセンスのもと、オープンソースとして開発されていましたが、現在は従来どおりのGPLライセンス版(jcoverage/gpl)と有償版(jcoverage+)の2種類のバージョンが存在します。

[参照] <http://www.jcoverage.com/>

1.2.2 Cobertura とは

Coberturaは、jcoverageがベースとなったカバレッジツールです。現在、SOURCE FORGE上(<http://sourceforge.net/>)で開発が管理されています。ライセンスは、CoberturaのAntタスク機能のみASL(Apache Software Licence)、その他の機能はGPLライセンスとなっています。

[参照] <http://cobertura.sourceforge.net/index.html>

現時点で jcoverage-gpl 版の開発および保守は完了しているため、本ドキュメントでは bugfix などの継続メンテナンスを行っている Cobertura を前提としています。

2

前提条件

2. 前提条件

この章では、本マニュアルの前提条件について説明します。

2.1	前提条件	2-2
2.2	インストール環境	2-3

2.1 前提条件

2.1.1 ハードウェア

eclipse3.0 の稼動条件に従います。

2.1.2 ソフトウェア

以下に示すソフトウェアがインストール済みであることを前提とします。

(1) オペレーティングシステム

- ・ Windows2000

(2) eclipse

- ・ eclipse3.0

(3) Ant

- ・ 本マニュアルでは eclipse3.0 に付属の Ant(1.6.2)を使用しています。

(4) jcoverage(Cobertura)

- ・ Cobertura1.2

(5) JUnit

本マニュアルでは Cobertura1.2 に付属の JUnit 3.8.1 を使用しています。

(6) その他

eclipse3.0 を稼動させるために、Sun Microsystems 社が開発、配布している Java の実行環境である Java2SE SDK がインストールされている必要があります。

2.2 インストール環境

本マニュアルでは以下のインストール環境を前提にしています。

- ・ 環境変数

```
JAVA_HOME= C:¥j2sdk1.4.2_08 1  
PATH=%JAVA_HOME%¥bin  
(%JAVA_HOME%¥bin は PATH の先頭に設定することを推奨しま  
す)
```

- ・ バージョン

```
Java2SE SDK : j2sdk-1_4_2_08-windows-i586-p.exe  
eclipse : eclipse-SDK-3.0.1-win32.zip  
eclipse 国際化モジュール : NLpack-eclipse-SDK-3.0.x-win32.zip  
Ant : 1.6.2 ( eclipse3.0 に付属 )  
Cobertura : cobertura-1.2-bin.zip  
JUnit : 3.8.1 ( eclipse3.0 に付属 )
```

- ・ インストール先

```
Java2SE SDK : C:¥j2sdk1.4.2_07 1  
eclipse : C:¥usr¥local¥eclipse  
Cobertura : C:¥usr¥local¥Cobertura
```

注 1 2005年4月現在の最新版です。「eclipse 3.0 利用ガイド」を元にインストールを実行した
場合を記述しています。

3

インストール

3. インストール

この章では、jcoverage(Cobertura)のインストール方法について説明します。

3.1	jcoverage(Cobertura)のインストール.....	3-2
-----	----------------------------------	-----

3.1 jcoverage(Cobertura)のインストール

3.1.1 インストール手順

<http://prdownloads.sourceforge.net/cobertura/cobertura-1.2-bin.zip?download> をダウンロードしてください。

本マニュアルではバージョン 1.2(cobertura-1.2-bin.zip,2005 年 4 月現在の最新版)を扱っています。

jcoverage(Cobertura)にはインストーラーはありません。ダウンロードしたファイルを適当な場所で解凍すると cobertura-1.2 ディレクトリができます。本章では例として C:\usr\local\Cobertura に配置します。jcoverage(Cobertura)の実行には同ディレクトリ直下の lib ディレクトリと cobertura.jar を使用します。

4

jcoverage(Cobertura) の利用方法

4. jcoverage(Cobertura)の利用方法

この章では、jcoverage(Cobertura)の使用方法について説明します。

4.1	はじめに	4-2
4.2	プロジェクト作成と各種設定	4-4
4.3	テスト対象ファイルの作成.....	4-5
4.4	テストケースクラスの作成.....	4-6
4.5	build.xmlの記述.....	4-7
4.6	Antの実行	4-13

4.1 はじめに

4.1.1 概要

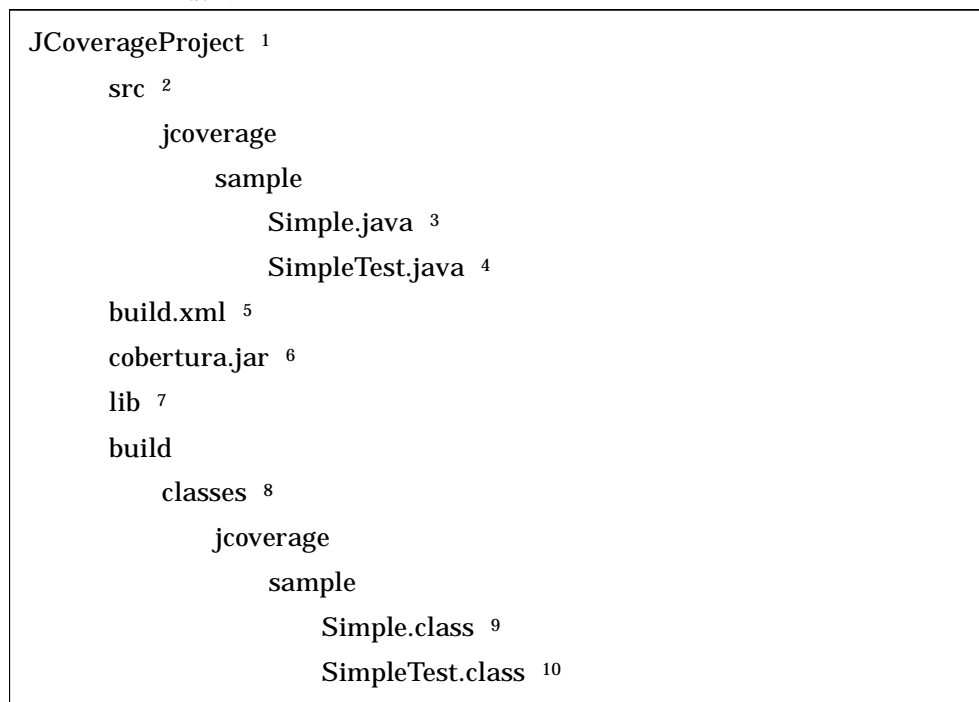
本章では、jcoverage(Cobertura)を使用してカバレッジデータを取得するためにプロジェクト、テスト対象クラス、テストケースクラスを作成し、Ant を用いてコンパイル、テスト実施、カバレッジデータの作成までを行います。その手順は以下の通りです。

- eclipse の Java プロジェクト作成と各種設定
- サンプルクラス Simple.java の作成
- サンプルクラスのテストケースクラス SimpleTest.java の作成
- build.xml の記述
- eclipse より Ant 実行

4.1.2 プロジェクトの構成

本章で使用する eclipse プロジェクトは以下の構成を前提とします。

Fig. 4-1 プロジェクト構成



- 注 1 プロジェクトディレクトリです。
- 注 2 Java ソースを格納するディレクトリです。
- 注 3 サンプルの Java ソースです。
- 注 4 サンプルのテストコードを記述した Java ソースです。
- 注 5 Ant 実行の定義ファイルです。
- 注 6 jcoverage(Cobertura)の JAR ファイルです。
- 注 7 jcoverage(Cobertura)を実行するのに必要な JAR ファイルを格納するディレクトリです。
- 注 8 コンパイルしたクラスファイルを格納するディレクトリです。
- 注 9 サンプルの Java ソースをコンパイルしたクラスファイルです。

注 10 サンプルのテストコードを記述した Java ソースをコンパイルしたクラスファイルです。

4.2 プロジェクト作成と各種設定

4.2.1 プロジェクトの作成

Fig4-1 に示した eclipse のプロジェクト(JCoverageProject)を作成します。ディレクトリの作成は [ファイル]-[新規]-[ディレクトリ]で行います。また、ソース・ディレクトリを「JCoverage/src」、デフォルト出力先ディレクトリを「JCoverageProject/build/classes」に設定します。設定方法は「eclipse 3.0 利用ガイド」を参考にしてください。

4.2.2 ディレクトリの作成と配置

JCoverageProject ディレクトリ直下に cobertura-1.2 ディレクトリ内の以下のディレクトリとファイルを配置してください。

- ・ lib ディレクトリ
- ・ cobertura.jar ファイル

4.2.3 プロジェクトへのクラス・パスの設定

JCoverageProject に

「C:\usr\local\eclipse\workspace\JCoverageProject\lib\junit\3.8.1\junit.jar」のパスを追加する必要があります。「ナビゲーター」もしくは「パッケージ・エクスプローラー」ビューで [JCoverageProject] を選択し、右クリックし [プロパティ] を選択します。[Java のビルド・パス] の [ライブラリー] タグから [外部 JAR の追加] ボタンをクリックして上記 JAR ファイルを選択し、[OK] ボタンをクリックします。

4.3 サンプルクラスの作成

テスト対象となるクラスファイルを作成します。

このクラスファイルでは文字列と数値をそれぞれ入力し、その内容を表示するプログラムです。

以下のソースを参考に Fig.4-1 に示した場所にそれぞれファイルを作成してください。

(1) Simple.java

```
package jcoverage.sample;

public class Simple {

    private String sName;
    private int iNum;
    public Simple() {
        sName = "";
        iNum = 0;
    }
    public void setName(String arg0) {
        if (arg0 == null) {
            sName = "";
        } else {
            sName = arg0;
        }
    }
    public String getName() {
        return sName;
    }
    public void setNum(int arg0) {
        if (arg0 < 0) {
            iNum = 0;
        } else {
            iNum = arg0;
        }
    }
    public int getNum() {
        return iNum;
    }
    public String toString() {
        StringBuffer sb = new StringBuffer();
        sb.append("jcoverageSample[");
        sb.append("Name : "+sName+" ");
        sb.append("Num : "+iNum+"]");
        return sb.toString();
    }
}
```

4.4 テストケースクラスの作成

テスト対象クラスのテストケースクラスを作成します。

記述するテストケースの内容を示します。

testSimple

Simple コンストラクタでは文字列に「""」、数値に「0」が設定されているかを確認する。

testSetName1

setName メソッドに文字列を渡した場合、値が正しく設定されているかを確認する。

testSetNum1

setNum メソッドにマイナスの数値を渡した場合、値「0」を設定するか確認する。

testSetNum2

setNum メソッドに0以上の数値を渡した場合、その数値を正しく設定するか確認する。

getName、getNum メソッド自体のテストケースは実行内容が簡易なため、省略しています。

～ で使用することで正しく処理していることが確認できます。

以下のソースを参考に Fig.4-1 に示した場所にそれぞれファイルを作成してください。

(1) SimpleTest.java

```
package jcoverage.sample;

import junit.framework.TestCase;

public class SimpleTest extends TestCase {

    public void testSimple() {
        Simple sample = new Simple();
        String name = sample.getName();
        int num = sample.getNum();
        assertEquals("", name);
        assertEquals(0, num);
    }

    public void testSetName1() {
        Simple sample = new Simple();
        sample.setName("TestString");
        String result = sample.getName();
        assertEquals("TestString", result);
    }

    public void testSetNum1() {
        Simple sample = new Simple();
        sample.setNum(-1);
        int result = sample.getNum();
        assertEquals(0, result);
    }

    public void testSetNum2() {
        Simple sample = new Simple();
        sample.setNum(1);
        int result = sample.getNum();
        assertEquals(1, result);
    }

}
```

4.5 build.xml の記述

Ant の実行に必要な build.xml を作成します。

build.xml ファイルの作成方法は、JCoverageProject を選択、右クリックし、[新規] - [ファイル] でファイル作成ウィザードが表示されます。[ファイル名]欄に"build.xml"と入力し、[終了]ボタンをクリックすると build.xml が JCoverageProject 直下に作成されます。

4.5.1 build.xml(jcoverage 使用前)

ビルドファイルを使って Ant で実行する主な処理は以下になります

- ・ ソースのコンパイル
- ・ テストケースの実行

作成したビルドファイルに次の内容を記述してください。

(1) build.xml

```
<project name="jcoverage.examples" default="main" basedir=".">
  <property name="build.dir" value="${basedir}/build"/>
  <property name="build.classes.dir" value="${build.dir}/classes"/>
  <property name="build.reports.dir" value="${build.dir}/reports"/>
  <property name="dist.dir" value="${basedir}"/>
  <property name="lib.dir" value="${dist.dir}/lib"/>
  <property name="src.dir" value="${basedir}/src"/>
  <target name="main" depends="clean,init,compile,test" description="clean build and unit test"/>

  <path id="junit">
    <fileset dir="${lib.dir}">
      <include name="junit/3.8.1/*.jar"/>
    </fileset>
  </path>
  <path id="log4j">
    <fileset dir="${lib.dir}">
      <include name="log4j/1.2.8/*.jar"/>
    </fileset>
  </path>

  <target name="clean" description="clean up build artifacts">
    <delete quiet="true">
      <fileset dir="${build.dir}"/>
    </delete>
  </target>

  <target name="init" description="create build directories">
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${build.classes.dir}"/>
    <mkdir dir="${build.reports.dir}"/>
  </target>

  <target name="compile" description="compile all classes">
    <javac srcdir="${src.dir}" destdir="${build.classes.dir}" failonerror="yes" debug="yes">
      <classpath refid="junit"/>
      <classpath refid="log4j"/>
    </javac>
  </target>

  <target name="test" description="Unit test the application">
    <junit fork="yes" dir="${basedir}" errorProperty="test.failed" failureProperty="test.failed">
      <classpath location="${build.classes.dir}"/>
      <formatter type="xml"/>
      <test name="${testcase}" todir="${build.reports.dir}" if="testcase"/>
      <batchtest todir="${build.reports.dir}" unless="testcase">
        <fileset dir="${src.dir}">
          <include name="**/*Test.java"/>
        </fileset>
      </batchtest>
    </junit>
  </target>
</project>
```

4.5.2 build.xml(jcoverage 使用)

jcoverage(Cobertura)を使用するためにビルドファイルに変更を加えます。変更後は以下のようになります。各変更内容については次の項で説明します。

(1) build.xml(修正後)

```
<project name="jcoverage.examples" default="main" basedir=".">

  <property name="build.dir" value="${basedir}/build"/>
  <property name="build.classes.dir" value="${build.dir}/classes"/>
  <property name="build.instrumented.dir" value="${build.dir}/instrumented-classes"/>
  <property name="build.coverage.dir" value="${build.dir}/coverage"/>
  <property name="build.reports.dir" value="${build.dir}/reports"/>
  <property name="dist.dir" value="${basedir}"/>
  <property name="lib.dir" value="${dist.dir}/lib"/>
  <property name="src.dir" value="${basedir}/src"/>

  <target name="main" depends="clean,init,compile,instrument,test,coverage" description="clean build, instrument and unit test"/>..... 注

  <taskdef classpath="cobertura.jar" resource="tasks.properties"/>

  <target name="clean" description="clean up build artifacts">
    <delete quiet="true">
      <fileset dir="${build.dir}"/>
      <fileset dir="${basedir}">
        <include name="cobertura.ser"/>
        <include name="cobertura.log"/>
      </fileset>
    </delete>
  </target>

  <target name="init" description="create build directories">
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${build.classes.dir}"/>
    <mkdir dir="${build.coverage.dir}"/>
    <mkdir dir="${build.instrumented.dir}"/>
    <mkdir dir="${build.reports.dir}"/>
  </target>

  <target name="compile" description="compile all classes">
    <javac srcdir="${src.dir}" destdir="${build.classes.dir}" failonerror="yes" debug="yes">
      <classpath>
        <fileset dir="${lib.dir}">
          <include name="junit/3.8.1/*.jar"/>
        </fileset>
        <fileset dir="${lib.dir}">
          <include name="log4j/1.2.8/*.jar"/>
        </fileset>
      </classpath>
    </javac>
  </target>

  <target name="instrument" description="Add jcoverage instrumentation">
    <cobertura-instrument todir="${build.instrumented.dir}">
      <fileset dir="${build.classes.dir}">
        <include name="**/*.class"/>
      </fileset>
    </cobertura-instrument>
  </target>

  <target name="test" description="Unit test the application">
    <junit fork="yes" dir="${basedir}" errorProperty="test.failed" failureProperty="test.failed">
      <classpath location="${build.instrumented.dir}"/>
      <classpath location="${build.classes.dir}"/>

      <classpath path="${basedir}/cobertura.jar"/>
      <formatter type="xml"/>

      <test name="{testcase}" todir="${build.reports.dir}" if="testcase"/>
    </junit>
  </target>
</project>
```

```

<batchtest todir="${build.reports.dir}" unless="testcase">
  <fileset dir="${src.dir}">
    <include name="**/*Test.java"/>
  </fileset>
</batchtest>
</junit>
</target>

<target name="coverage" description="HTML and XML coverage reports can be found in
build/coverage">
  <cobertura-report srcdir="${src.dir}" destdir="${build.coverage.dir}"/>
  <cobertura-report srcdir="${src.dir}" destdir="${build.coverage.dir}" format="xml"/>
</target>
</project>

```

この main ターゲットを実行することで以下の処理が行われます。

1. 既存のクラスファイルを削除
2. 必要なディレクトリの作成
3. ソースコンパイル実行
4. class のリコンパイル
5. JUnit のテスト実行
6. カバレッジレポートの出力

4.5.3 jcoverage(Cobertura)定義の追加内容

上記の build.xml の追加内容について説明します。追加した定義は以下になります。

```

jcoverage(Cobertura)で使用するディレクトリの変数定義
main ターゲットへの jcoverage(Cobertura)タスク追加
jcoverage(Cobertura)のタスクを使用する宣言
jcoverage(Cobertura)ファイルのクリア
jcoverage(Cobertura)で必要となるディレクトリの作成
クラスのリコンパイル
  で作成されたクラスファイルをテスト対象に追加
実行結果をレポートとして出力

```

次に各項目の記述内容と詳細を説明します

jcoverage で使用するディレクトリの定数定義

```

<property name="build.instrumented.dir" value="${build.dir}/instrumented-classes"/>
<property name="build.coverage.dir" value="${build.dir}/coverage"/>

```

main ターゲットへの coverage タスク追加
 ファイルのクリアからカバレッジレポート出力まで一括して行う main ターゲットに coverage タスクを追記します。

```

<target name = "main" depends="clean,init,compile,instrument,test,coverage" description = "clean
build,compile,Instrument,unit test and coverage"/>

```

cobertura.jar の使用宣言

Ant に予め用意されていないタスクを使用する場合にはタスクを宣言します。使用するクラスパスとタスク定義の含まれたリソース名を指定します。<taskdef>タグを記述します。

jcobertura を使用する場合、classpath には[jcobertura.jar]を記述します。

```
<taskdef classpath="cobertura.jar" resource="tasks.properties"/>
```

jcobertura(Cobertura)ファイルのクリア

jcobertura(Cobertura)を行う場合に前回出力したファイルをクリアします。clean ターゲットに対象となるファイルを指定します。

```
<fileset dir="${basedir}">
  <include name="cobertura.ser"/>
  <include name="cobertura.log"/>
</fileset>
```

jcobertura(Cobertura)で必要となるディレクトリの作成

カバレッジ取得のためにリコンパイルしたクラスファイルとカバレッジ結果を別ディレクトリに格納します。格納用ディレクトリを新規作成します。

```
<mkdir dir="${build.coverage.dir}"/>
<mkdir dir="${build.instrumented.dir}"/>
```

クラスファイルのリコンパイル

ソースファイルをコンパイルしたクラスファイルに新たに instrumentation 命令を挿入し、リコンパイルします。instrumentation 命令が JVM により実行されたかをカウントします。

リコンパイルしたクラスファイルを instrumented-class ディレクトリに作成されます。コンパイラターゲットとテストターゲットの間に次のターゲットを記述してください。

```
<property name="build.instrumented.dir" value="${build.dir}/instrumented-classes"/>

<target name="instrument" description="Add jcoverage instrumentation">
  <cobertura-instrument todir="${build.instrumented.dir}">
    <fileset dir="${build.classes.dir}">
      <include name="**/*.class"/>
    </fileset>
  </cobertura-instrument>
</target>
```

注意 1 jcoverage を使用すると、指定した classes ディレクトリに作成される通常の class ファイル以外に instrumented-classes ディレクトリにテスト用の class ファイルを作成します。jcobertura はテスト用 class ファイルをコンパイルする際、ソースに独自のコードを埋め込んでいます。そのまま使用すると性能の低下につながる恐れがあります。テスト以外の目的に使用する場合は classes ディレクトリの class ファイルを使用してください。

作成されたクラスファイルをテスト対象に追加
作成された instrumented-class ディレクトリ内のテスト用クラスファイルがテスト実施タスクで行われるようにテスト対象として追加します。

```
<classpath location="${build.instrumented.dir}"/>
<classpath path="${basedir}/cobertura.jar"/>
```

実行結果をレポートとして出力

テストケース実行タスクの後に<coverage>タスクが実行されるように定義を追加します。
<coverage>タスクにはテスト結果のレポートを出力するため<report>タグが使用されます。
<report>タグの属性は以下のとおりです。

- srcdir : カバレッジを取得したいサンプルソースの存在するディレクトリ名を指定します。
- destdir : カバレッジを取得したレポートの出力先ディレクトリ名を指定します。
ディレクトリが存在しない場合には新規作成します。
- format : 出力するレポートの形式を指定します。HTML 形式と XML 形式を指定することができます。指定しなかった場合には HTML 形式となります。

次の設定では、format の異なる 2 つの<report>タグを併記することで、HTML 形式と XML 形式の 2 種類の出力レポートを作成しています。

```
<property name="build.coverage.dir" value="${build.dir}/coverage"/>
<target name="coverage" description="HTML and XML coverage reports can be found in build/coverage">
  <cobertura-report srcdir="${src.dir}" destdir="${build.coverage.dir}"/>
  <cobertura-report srcdir="${src.dir}" destdir="${build.coverage.dir}" format="xml"/>
</target>
```

4.6 Ant の実行

4.6.1 Ant のクラスパス設定

本マニュアルでは、eclipse 3.0 に付属している Ant1.6.2 を用います。Ant のクラスパス設定の確認を以下の手順で行います。

[ウィンドウ] - [設定]から設定ウィンドウを開きます。

左ペインの[Ant - ランタイム]を選択します。

[クラスパス]タブを選択し、[グローバル項目]をクリックします。

[外部 JAR の追加]をクリックします。

C:\usr\local\eclipse\workspace\JCoverageProject\lib\junit\3.8.1\junit.jar を選択し、[OK]をクリックします。

Fig. 4-2 Ant クラスパス(Ant ホーム項目)の設定確認画面

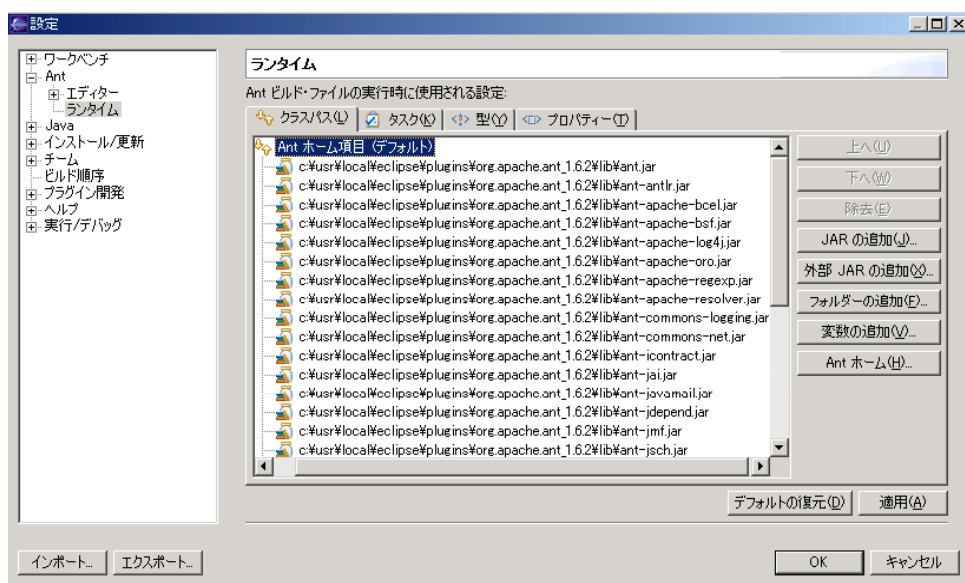
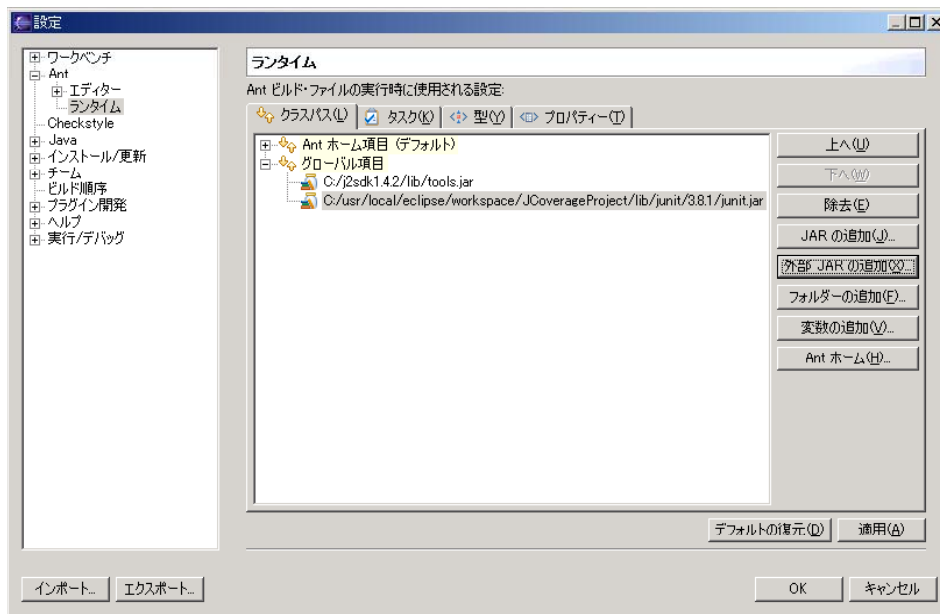


Fig. 4-3 Ant クラスパス(Ant グローバル項目)の設定確認画面



4.6.2 Ant の実行

4.5.3 で作成したビルドファイルを使用して Ant の実行を行います。
実行手順を以下に示します。

build.xml ファイルを右クリックし、[実行] - [2 Ant ビルド]を選択します。

Fig. 4-4 Ant の実行

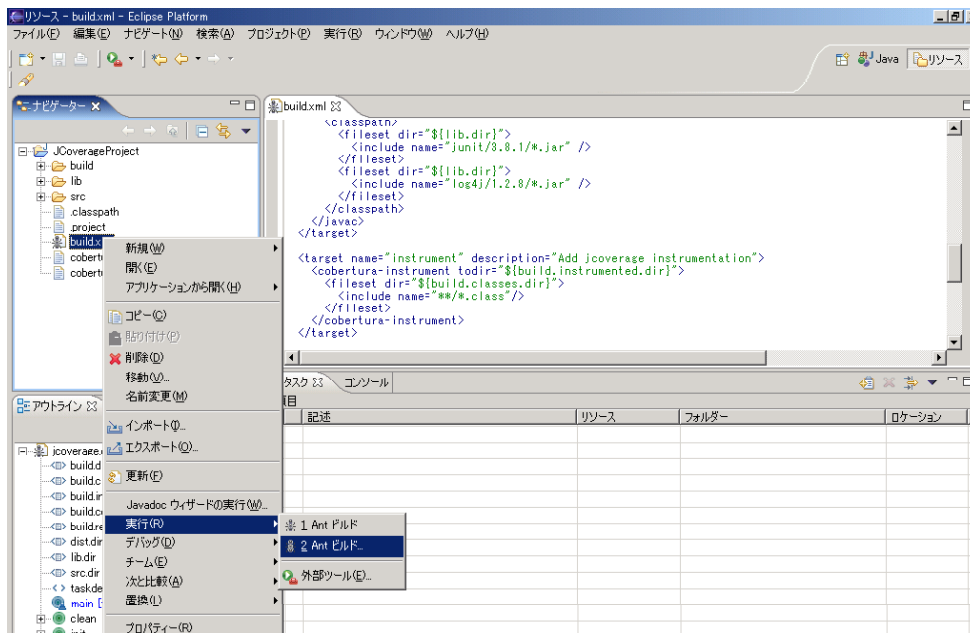


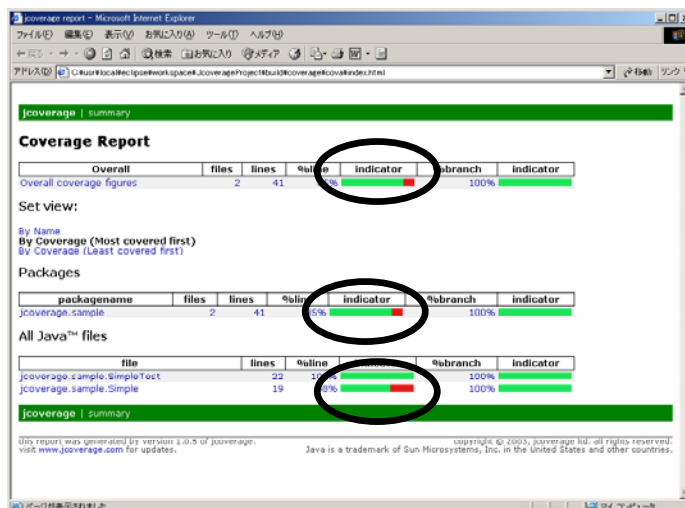
Fig. 4-7 プロジェクト構成

```
JcoverageProject
  build
    coverage 1
      cova
      covd
      coverage.xml 2
      images
      index.html 3
      jcoverage.sample.html
      jcoverage.sample.Simple.html
      jcoverage.sample.SimpleTest.html
    instrumented-classes
  reports
  cobertura.ser 4
```

- 注 1 カバレッジ用ディレクトリです。
- 注 2 XML 形式のカバレッジレポートです。
- 注 3 HTML 形式のカバレッジレポートです。
- 注 4 カバレッジデータです。

coverage ディレクトリにある「index.html」を表示します。項目の詳細については 5 章を参照してください。

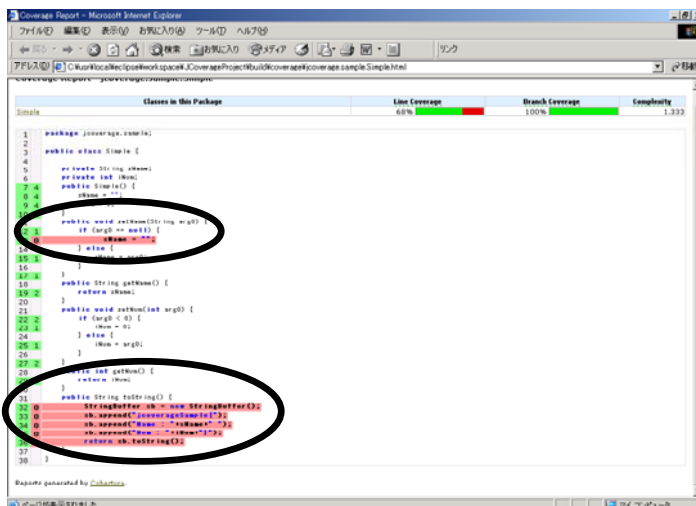
Fig. 4-8 カバレッジ結果



上記画面で「jcoverage.sample.Simple.html」のリンクをクリックすると Simple.java ファイルのソースコードを表示します。表示内容の詳細は 5 章を参照してください。

テストケースがテスト対象クラスのステップを網羅していない場合、実行されなかったステップはステップごとに赤色に色づけされて表示されます。メソッドが実行されなかった場合にはメソッド内部の全てが赤色に色づけされて表示されます。

Fig. 4-9 ソースコードの表示



4.6.4 テストコードの追加

ソースコードの網羅率が 100%未満だった場合、実行されなかったステップを実行するテストコードを記述します。

カバレッジデータを参照すると実施したテストでは次のケースを追加する必要があります。

- setName メソッドに「null」を渡した場合、「」を設定する
- toString メソッドの指定した値を正しく出力する

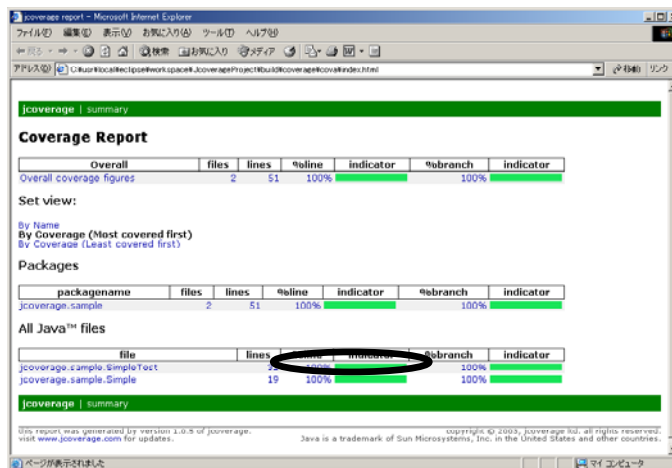
テストケースクラスに次のテストコードを記述してください。

テストコードを追加し、Ctrl-s で保存してください。

```
public void testSetName2() {
    Simple sample = new Simple();
    sample.setName(null);
    String result = sample.getName();
    assertEquals("",result);
}
public void testToString() {
    Simple sample = new Simple();
    sample.setName("TestString");
    sample.setNum(1);
    assertEquals("jcoverageSample[Name : TestString Num : 1]",sample.toString());
}
```

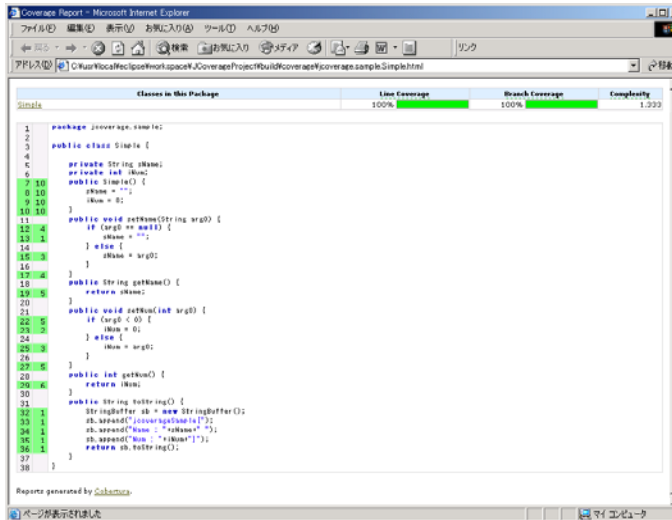
4.6.2 と同様に Ant を実行し、coverage ディレクトリにある「index.html」を表示します。ソースコードの網羅率が 100%になっていることが確認できます。

Fig. 4-10 修正後のカバレレッジ結果



「jcoverage.sample.Simple.html」をクリックし、ソースコードを表示します。
赤色になっている部分がなくなり、全てのステップが実行されたことが確認できます。

Fig. 4-11 修正後のソースコード



5

レポート機能

5. レポート機能

この章では、jcoverage(Cobertura)のレポート機能について説明します。

5.1	レポートの概要	5-2
5.2	HTMLレポート	5-3
5.3	XMLレポート	5-6
5.4	merge機能	5-7

5.1 レポートの概要

5.1.1 レポート出力までの流れ

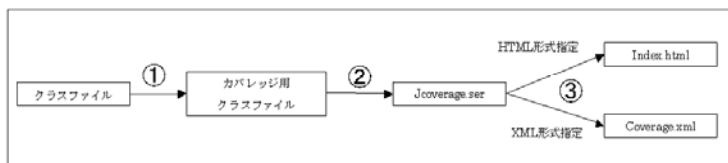
jcovrage(Cobertura)を使用して、作成したテスト対象クラスとテストケースクラスからカバレッジ結果のレポートを出力するまでの流れを説明します。

通常のクラスファイルから Ant を使用してカバレッジ用クラスファイルを作成します。

jcovrage(Cobertura)を使用し、カバレッジデータ(cobertura.ser)ファイルを取得します。

取得したカバレッジデータファイルから Ant を使用し、指定した形式でカバレッジ結果のレポートを出力します。jcovrage は取得したカバレッジデータから 2 種類の形式でレポートを出力することができます。

Fig. 5-1 レポート出力までの流れ



5.2 HTML レポート

5.2.1 HTML レポートの出力

HTML レポートについて説明します。

cobertura.ser から HTML レポートを出力するためには、ビルドファイルの<coverage>ターゲットに次の<report>タスクを追加します。

```
<target name="coverage" description="HTML coverage reports can be found in build/coverage">  
  <cobertura-report srcdir="${src.dir}" destdir="${build.coverage.dir}"/>  
</target>
```

5.2.2 HTML レポートの内容

5.2.1 の<report>タスクを含む build.xml を実行すると、cobertura.ser から以下のような HTML 形式のレポートが出力されます。レポートの出力先は<cobertura-report>タグの destdir 属性で指定したディレクトリ(C:\usr\local\eclipse\workspace\JCoverageProject\build\coverage)となります。同ディレクトリにある index.html をクリックすると下図(Fig.5-2)のような全パッケージ分の HTML レポートが表示されます。また、同図の左上の丸で囲んだ箇所(リンク)をクリックすると、下図(Fig.5-3)のようなパッケージ単位の HTML レポートが表示されます。

Fig. 5-2 HTML レポート画面(全パッケージ分)

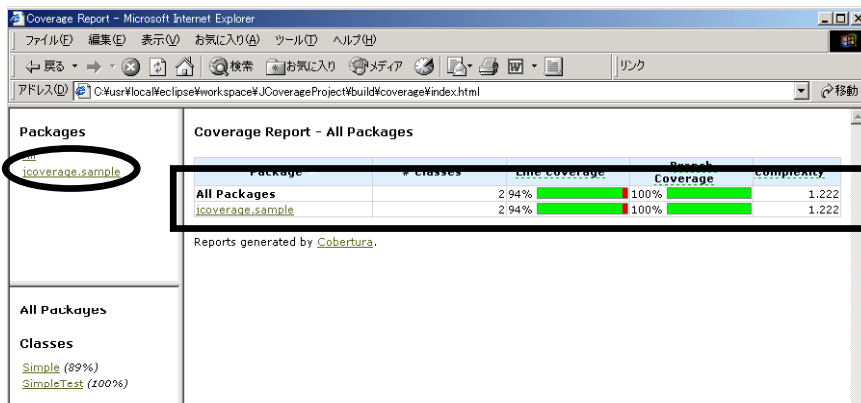


Fig. 5-3 HTML レポート画面(個別パッケージ分)

Package	# Classes	Line Coverage	Branch Coverage	Complexity
jcoverage.sample	2	65%	100%	1.2
Classes in this Package				
Simple		68%	100%	1.333
SimpleTest		100%	100%	1

Reports generated by Cobertura.

表示内容の詳細を説明します。

～ は次の網羅率を表しています。 ¹

プロジェクト全体の網羅率

プロジェクトに含まれる各パッケージの網羅率

パッケージに含まれる各ファイルの網羅率

注 1 網羅率とはカバレッジ取得対象クラスの総ステップ数に対して、テスト実施時に実行したステップ数の割合を計算したものです。

のファイル名をクリックすると、指定したファイルのソースを表示することができます。

テストで実行されなかったステップは赤色で表示します。またメソッドが実行されなかった場合にはメソッド全体を赤色で表示します。

Fig. 5-4 HTML レポート画面(個別クラス分)

Coverage Report - Microsoft Internet Explorer

Classes in this Package: Line Coverage: 68% Branch Coverage: 100% Complexity: 1,333

```
1 package jcoverage.sample;
2
3 public class Simple {
4
5     private String sName;
6     private int iNum;
7     public Simple() {
8         sName = "";
9         iNum = 0;
10    }
11
12    public void setName(String arg0) {
13        if (arg0 == null) {
14            sName = "";
15        }
16        sName = arg0;
17    }
18    public String getName() {
19        return sName;
20    }
21    public void setNum(int arg0) {
22        if (arg0 < 0) {
23            iNum = 0;
24        } else {
25            iNum = arg0;
26        }
27    }
28    public int getNum() {
29        return iNum;
30    }
31
32    public String toString() {
33        StringBuffer sb = new StringBuffer();
34        sb.append("jcoverageSample");
35        sb.append("Name : " + sName + " ");
36        sb.append("Num : " + iNum + " ");
37        return sb.toString();
38    }
39 }
```

Reports generated by Cobertura.

ページが表示されました

5.3 XML レポート

5.3.1 XML レポートの出力

cobertura.ser から XML レポートを出力するためには、ビルドファイルに<report>タスクを追加します。XML 形式のレポートを得るために format 属性を"xml"に指定します。

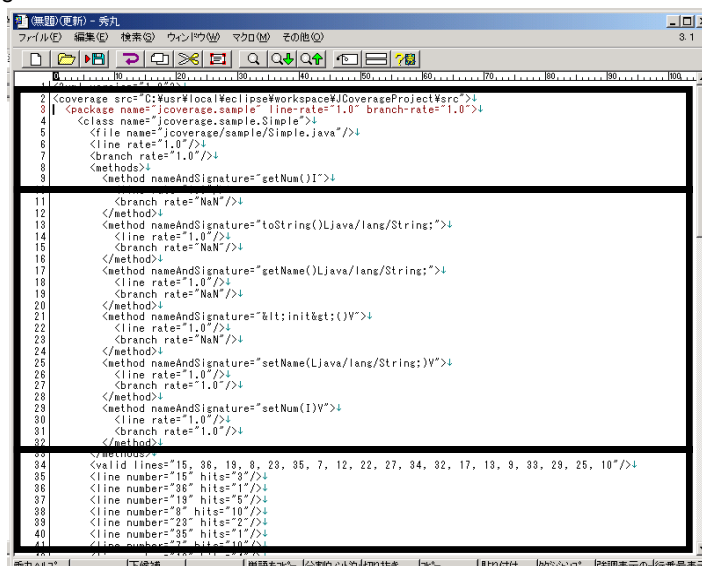
```
<target name="coverage" description="HTML coverage reports can be found in build/coverage">  
  <cobertura-report srcdir="${src.dir}" destdir="${build.coverage.dir}" format="xml">  
</target>
```

5.3.2 XML レポートの内容

5.2.1 の<report>タスクを含む build.xml を実行すると、cobertura.ser から以下のような XML 形式のレポートが出力されます。

出力先は C:\usr\local\eclipse\workspace\JCoverageProject\build\coverage となります。

Fig. 5-5 XML レポート



表示内容の詳細を説明します。

~ は次の内容を表しています。

各クラスのカバレッジデータ

ファイル名(file name)、ステップ網羅率(line rate)、分岐網羅率(branch rate)が記述されています。

各メソッドのカバレッジデータ

メソッド名(method nameAndSignature)、ステップ網羅率、分岐網羅率が記述されています。

各ステップのカバレッジデータ

各ステップのファイルにおける行番号(line number)とその実行回数(hits)が記述されています。

5.4 merge 機能

5.4.1 merge 機能の使用

merge 機能を使うことで、複数のカバレッジデータをマージ（併合）して1つのカバレッジデータにすることができます。

(1) カバレッジデータの用意

カバレッジデータのマージを行う場合、build.xml を実行することでプロジェクトのルートディレクトリ直下に生成される cobertura.ser を cobertura00.ser, cobertura01.ser のように適当なファイル名(coberturaXX.ser)に変えておきます。

(2) ビルドファイルの作成

4章で作成したレポートと同様にHTML形式とXML形式でレポートを出力することができます。マージを行うためには、build.xml 以下を加えます。

- ・ main ターゲットに merge タスク
 - ・ coverage タスクの前に merge タスクの定義
- 追加内容を以下に示します。

```
<target name="main" depends="clean,init,instrument,test,merge,coverage" description="clean,
init,,instrument,test, merge and coverage"/>

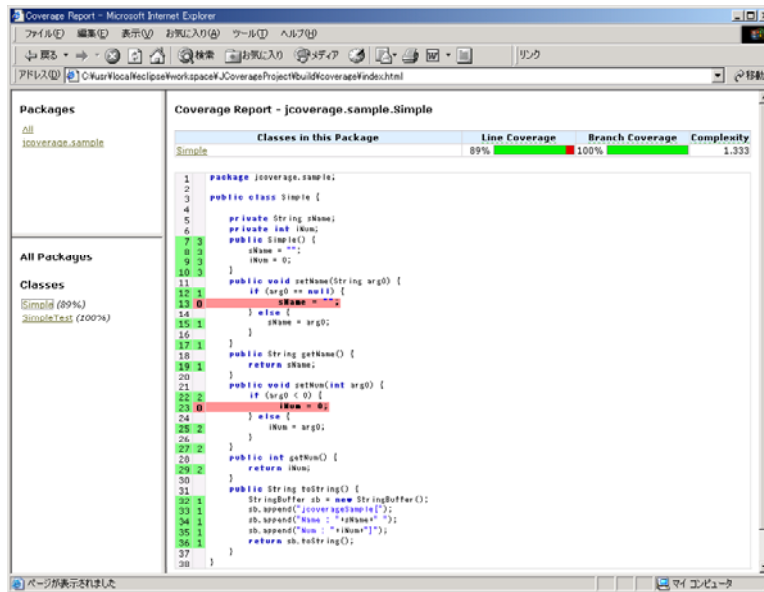
<target name="merge">
  <cobertura-merge>
    <fileset dir="{basedir}">
      <include name="**/cobertura*.ser"/>
    </fileset>
  </cobertura-merge>
</target>
```

上記のように、<coverage>タスクでレポートが出力される前に<merge>タスクが実行されるようにします。<include name="**/cobertura*.ser"/>の記述により、cobertura の後に任意の数字を付与したファイル名のカバレッジデータ(「cobertura*.ser」)が全てマージされ、新たに「cobertura.ser」というファイル名のカバレッジデータが得られます。

出力レポートの形式は<coverage>タスク内の<report>タグの属性で指定してください。

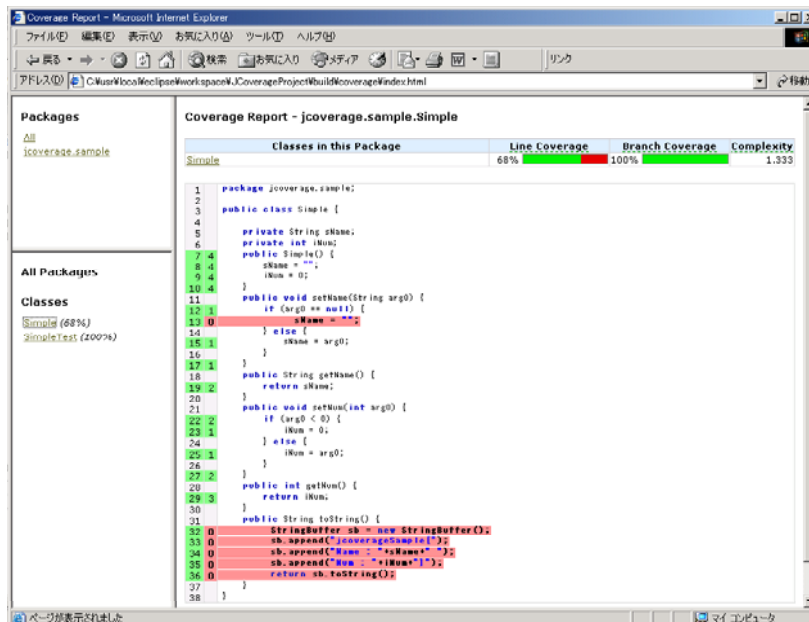
マージ元になっているカバレッジデータ(「jcoverage00.ser」)の Simple クラス出力レポートを示します。テストで実行されていない部分が赤色で表示されています。

Fig. 5-8 カバレッジデータ(「jcoverage00.ser」)の出力レポート(jcoverage.sample.Simple)



マージ元になっているカバレッジデータ(「jcoverage01.ser」)の Simple クラス出力レポートを示します。テストで実行されていない部分が赤色で表示されています。

Fig. 5-9 カバレッジデータ(「jcoverage01.ser」)の出力レポート(jcoverage.sample.Simple)



マージされたカバレッジデータ（「cobertura.ser」）の出力レポートを示します。

マージにより得られるカバレッジデータにおいては、マージされたデータのうち少なくともひとつで実行されたステップ・分岐は網羅率に計上されます。

全てのマージ元のファイルで実行されなかったステップのみ赤色で表示されます。

Fig. 5-10 マージにより得られた出力レポート（jcoverage.sample.Simple）

Classes in this Package	Line Coverage	Branch Coverage	Complexity
Simple	95%	100%	1,333

```
1 package jcoverage.sample;
2
3 public class Simple {
4
5     private String sName;
6     private int iNum;
7     public Simple() {
8         sName = "";
9         iNum = 0;
10    }
11    public void setName(String arg0) {
12        if (arg0 == null) {
13            sName = "";
14        } else {
15            sName = arg0;
16        }
17    }
18    public String getName() {
19        return sName;
20    }
21    public void setNum(int arg0) {
22        if (arg0 < 0) {
23            iNum = 0;
24        } else {
25            iNum = arg0;
26        }
27    }
28    public int getNum() {
29        return iNum;
30    }
31    public String toString() {
32        StringBuffer sb = new StringBuffer();
33        sb.append("jcoverage.sample.");
34        sb.append("Name : " + sName + " ");
35        sb.append("Num : " + iNum + " ");
36        return sb.toString();
37    }
38 }
```

付録A

付録

A. 付録

この章では、注意事項、jcoverage のタスクなどについて説明します。

A.1	注意事項	A-2
A.2	jcoverageタスク	A-3

A.1 注意事項

A.1.1 jcoverage(Cobertura)を使用する際の注意事項

(1) 網羅率について

下記のように複数の条件が設定された判断式は、条件のうちどれか 1 つを満たしていればすべての条件を満たしていなくても網羅率の計上がされます。

Fig. A-1 複数の条件が設定された判断式

25		
26		public void setNum(int arg0) {
27	2	if (arg0 < 0 arg0 >= 256) {
28	1	iNum = 0;
29		} else {
30	1	iNum = arg0;
31		}
32		}

網羅率は「総ステップにおけるテストで実行したステップのパーセンテージ」を表しています。1つの条件式に複数の条件が「||」で記述されている場合など、条件の組み合わせを考慮していません。そのため出力レポートの網羅率のみから「全てのケースを満たしている」と判断できません。網羅率とステップの実行回数を参照する必要があります。

(2) クラスの手動実行

jcoverage(Cobertura)を使用してコンパイルを行ったクラスを Java コマンドにより手動実行を行った場合でもテストの実行、レポート作成は正常に行われます。また、本マニュアルではJUnitを使用していますが、JUnitを使用しない場合もテストの実行、レポート作成は正常に行われます。

A.2 jcoverage(Cobertura)のタスク

A.2.1 <check>タスクの使用について

jcoverage 上の<check>タスクに関して、以下に説明します。

テストを実施した際、網羅率がある一定の基準を上回っているかどうかを確認する際、<check>タスクを使用します。

このタスクは基準の網羅率を設定し、実際の網羅率はその基準を下回っていた場合、Ant 実行者に注意を促すメッセージを表示するものです。<check>タスクを使用する場合には次を参考にして build.xml に追記してください。

```
<project name="jcoverage.examples" default="main" basedir=". ">

  <property name="build.dir" value="${basedir}/build"/>
  <property name="build.classes.dir" value="${build.dir}/classes"/>
  <property name="build.instrumented.dir" value="${build.dir}/instrumented-classes"/>
  <property name="build.coverage.dir" value="${build.dir}/coverage"/>
  <property name="build.reports.dir" value="${build.dir}/reports"/>
  <property name="dist.dir" value="${basedir}"/>
  <property name="lib.dir" value="${dist.dir}/lib"/>
  <property name="src.dir" value="${basedir}/src"/>

  <target name = "main" depends="clean,init,compile,instrument,test,check,coverage" description =
"clean build, instrument and unit test"/>

  <taskdef classpath="cobertura.jar" resource="tasks.properties"/>

  <target name="check">
    <cobertura-check branch="80" line="80">
      <regex pattern="jcoverage.sample.*" branch="85" line="95"/>
    </cobertura-check>
  </target>

  <target name="clean" description="clean up build artifacts">
    <delete quiet="true">
      <fileset dir="${build.dir}"/>
      <fileset dir="${basedir}">
        <include name="cobertura.ser"/>
        <include name="cobertura.log"/>
      </fileset>
    </delete>
  </target>

  <target name="merge">
    <cobertura-merge>
      <fileset dir="${basedir}">
        <include name="**/cobertura*.ser"/>
      </fileset>
    </cobertura-merge>
  </target>

  ...以下、中略...
</project>
```

<check>タスクの属性について説明します。

- branch
分岐の網羅率の基準値です。単位はパーセントとなります。この値を下回っていた場合、コマンドプロンプトにてメッセージが出力されます。
- line
ステップの網羅率の基準値です。単位はパーセントとなります。この値を下回っていた場合、コマンドプロンプトにてメッセージが出力されます。

<check>タスクは内部に<regex>タグを定義することができます。このタグを使用することでパッケージごとの網羅率の基準値を設定することができます。<regex>タグの属性について説明します。

- pattern
パッケージ名を指定することができます。
- branch
pattern 属性で指定したクラス、パッケージの分岐の網羅率についての基準値。
- line
pattern 属性で指定したクラス、パッケージのステップの網羅率について基準値。

注 <regex>タグは必須タグではありません。ただし定義しない場合、プロジェクト/パッケージともに<check>タスクの branch/line 属性で定義した網羅率が基準の網羅率になります。

<check>タスクを実行するとコンソールタブに次のような結果が表示されます。jcoverage.sample.Simple のステップについての網羅率は 89.4%で、95.0%の line(基準値)を満たさなかったことを表示しています。また、基準値を満たしたパッケージに対してはこの内容が出力されません。

Fig. A-2 <check>タスクの実行

```
問題 Javadoc 宣言 コンソール
<終了> JcoverageProject build.xml [Ant ビルド] C:\jdk1.4.2_02\bin\javaw.exe (2005/04/04 11:27:07)
[cobertura-check] Cobertura 1.2
[cobertura-check] Copyright (C) 2003 jcoverage ltd.
[cobertura-check] Copyright (C) 2005 Mark Doliner <thekingant@users.sourceforge.net>
[cobertura-check] Cobertura is licensed under the GNU General Public License
[cobertura-check] Cobertura comes with ABSOLUTELY NO WARRANTY
[cobertura-check] Jcoverage.sample.Simple line coverage rate of: 89.4% (required: 95.0%)
coverage:
[cobertura-report] Cobertura 1.2
[cobertura-report] Copyright (C) 2003 jcoverage ltd.
[cobertura-report] Copyright (C) 2005 Mark Doliner <thekingant@users.sourceforge.net>
[cobertura-report] Cobertura is licensed under the GNU General Public License
[cobertura-report] Cobertura comes with ABSOLUTELY NO WARRANTY
[cobertura-report] Reporting time: 768ms
[cobertura-report] Cobertura 1.2
[cobertura-report] Copyright (C) 2003 jcoverage ltd.
[cobertura-report] Copyright (C) 2005 Mark Doliner <thekingant@users.sourceforge.net>
[cobertura-report] Cobertura is licensed under the GNU General Public License
[cobertura-report] Cobertura comes with ABSOLUTELY NO WARRANTY
[cobertura-report] Reporting time: 109ms
main:
BUILD SUCCESSFUL
Total time: 4 seconds
```