

NRI

JsUnit 利用ガイド

*Advanced
Information
Technology
Division*

技術力をベースに未来を創発する



Copyright © 株式会社野村総合研究所 2006 All rights reserved.

本書は、本書に記載した要件、技術、方式に関する内容が変更されないこと、および出典を明示いただくことを前提に、無償でその全部または一部を複製、翻案、翻訳、転載、引用、公衆送信、譲渡等して利用いただけます。なお、全体を複製、翻案、翻訳された場合は、本書にある著作権表示を明示してください。

本書の著作権者は、本書の記載内容に関して、その正確性、商品性、利用目的への適合性等に関して保証するものではなく、その利用により生じた損害について、法律上のいかなる責任も負いません。

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。

第1版 第1刷

発行日 2006年2月

はじめに

このマニュアルは、JsUnit「<http://www.edwardh.com/jsunit/> (2006/01 現在)」の利用方法を記述しています。

【対象読者】

JsUnit を用いて JavaScript プログラムのテストを行いたい方を対象にしています。

【構成】

次に示す 6 つの章と 1 つの付録から構成されています。

第 1 章 概要

JsUnit の概要について説明します。

第 2 章 前提

JsUnit の前提条件・本ドキュメントにおける前提条件について説明します。

第 3 章 インストール

WindowsXP における JsUnit とその他のアプリケーションのインストール方法について説明します。

第 4 章 テストの記述と単一環境におけるテスト

JsUnit の実行方法について説明します。

第 5 章 複数環境におけるテスト

JsUnit Server の実行方法について説明します。

第 6 章 アンインストール

WindowsXP における JsUnit とその他のアプリケーションのアンインストール方法について説明します。

付録 A WTP1.0 による JavaScript の編集

WTP1.0 で JavaScript を開発する方法について説明します。

【マニュアル中の表記】

次に示す表現を利用します。

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

目次

1. 概要	1-1
1.1 JsUnit とは	1-2
1.1.1 JsUnit.....	1-2
1.1.2 JsUnit Server	1-2
1.2 マニュアルの位置付けと範囲	1-3
2. 前提	2-1
2.1 前提条件	2-2
2.1.1 ソフトウェア	2-2
2.2 インストール環境	2-3
2.2.1 モジュール名	2-3
2.2.2 インストール先.....	2-3
2.2.3 環境変数	2-3
3. インストール	3-1
3.1 Apache のインストール	3-2
3.1.1 インストールの手順.....	3-2
3.1.2 インストールの確認.....	3-6
3.2 JsUnit のインストール.....	3-7
3.2.1 ローカルで利用する場合	3-7
3.2.2 Web サーバーを利用する場合	3-8
3.3 JDK のインストール.....	3-9
3.3.1 インストールの手順.....	3-9
3.3.2 インストールの確認.....	3-11
3.4 Ant のインストール	3-12
3.4.1 インストールの手順.....	3-12
3.4.2 インストールの確認.....	3-12
4. テストの記述と単一環境におけるテスト	4-1
4.1 JsUnit の概要.....	4-2
4.2 テストの準備	4-3
4.2.1 JavaScript プログラムの開発	4-3
4.2.2 テストスクリプトの記述	4-5
4.3 テストの実行.....	4-12
4.3.1 ローカルで利用する場合	4-12
4.3.2 Web サーバーを利用する場合	4-15
5. 複数環境におけるテスト	5-1
5.1 JsUnit Server の概要	5-2
5.2 Standalone Test	5-3
5.2.1 テストの概要	5-3
5.2.2 テストの配備	5-3
5.2.3 テストの実行	5-5
5.3 Distributed Test.....	5-9
5.3.1 テストの概要	5-9
5.3.2 テストの準備	5-10
5.3.3 テストの実行	5-13

6.	アンインストール	6-1
6.1	JsUnit のアンインストール	6-2
6.1.1	アンインストールの手順	6-2
6.2	Apache のアンインストール	6-3
6.2.1	アンインストールの手順	6-3
6.3	Ant のアンインストール	6-4
6.3.1	アンインストールの手順	6-4
6.4	JDK のアンインストール	6-5
6.4.1	アンインストールの手順	6-5
A.	WTP1.0 による JavaScript の編集	A-1
A.1	WTP	A-2
A.2	WTP1.0 による JavaScript の編集	A-3
A.2.1	JSP ファイルにおける JavaScript の編集	A-3
A.2.2	外部スクリプトファイルにおける JavaScript の編集	A-7



1. 概要

この章では、JsUnit の概要について説明します。

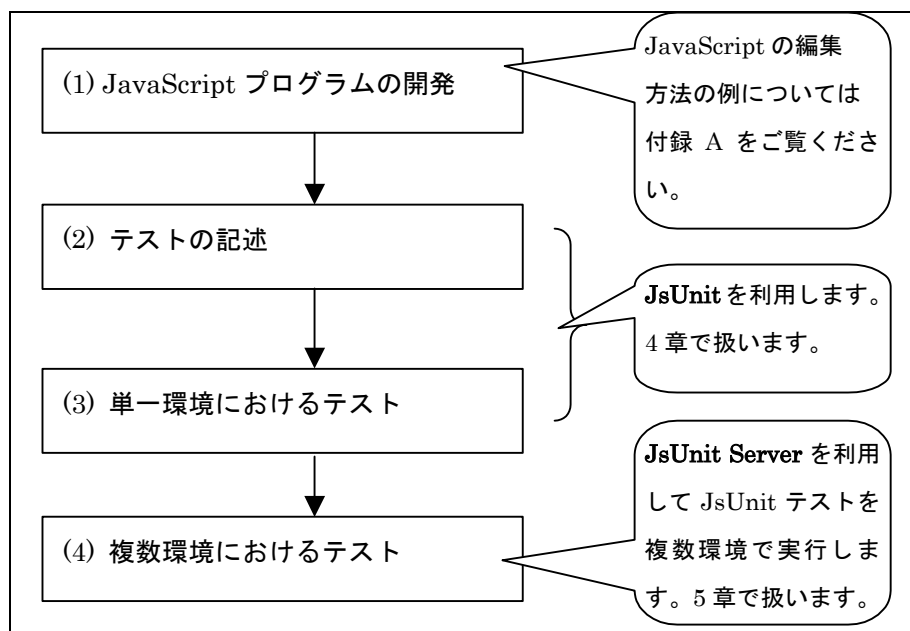
1.1	JsUnitとは	1-2
1.2	マニュアルの位置付けと範囲	1-3

1.1 JsUnit とは

JsUnit「<http://www.edwardh.com/jsunit/> (2006/01 現在)」は、オープンソースの JavaScript 用 テスティング・フレームワークです。テストは JavaScript で記述しますので、手軽に導入することができます。またテストの記述方法は JUnit を踏襲していますので、単体テスト・フレームワークの経験者は簡単にテストを記述することができます。

JsUnit を利用した開発手順を以下に示します。

Fig. 1-1 開発手順



1.1.1 JsUnit

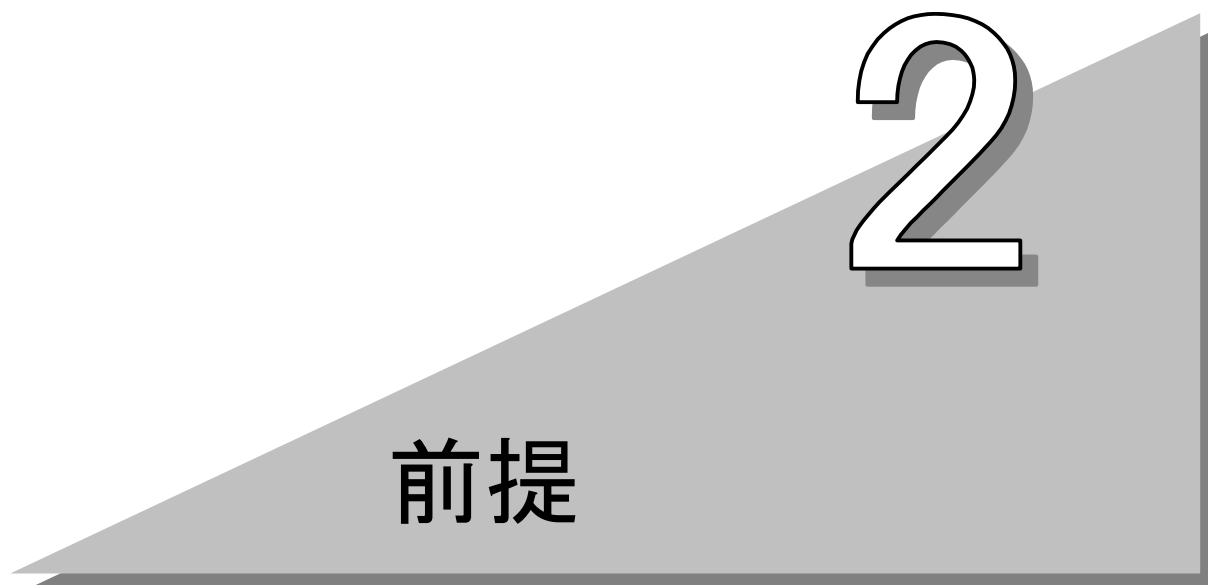
JavaScript プログラムの単体テストを記述・実行するためのテスト・フレームワークです。JavaScript で記述されています。

1.1.2 JsUnit Server

JsUnit テストを複数環境(複数のブラウザ・複数のマシン)で実行する機能です。Java で記述されています。JsUnit Server を利用する前に、JsUnit を用いてテストを記述する必要があります。

1.2 マニュアルの位置付けと範囲

本マニュアルでは JavaScript 開発者を対象として、JsUnit と JsUnit Server の利用方法を説明します。



2. 前提

この章では、JsUnit の前提条件・本ドキュメントにおける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 ソフトウェア

(1) 必要なソフトウェア

(a) JsUnitに必要なソフトウェア

- ・ ブラウザー ((2)ブラウザーの種類を参照)
- ・ Web サーバー(テストに Web サーバーを利用する場合)

(b) JsUnit Serverに必要なソフトウェア

- ・ ブラウザー ((2)ブラウザーの種類を参照)
- ・ JDK
- ・ Ant + JUnit
- ・ Web サーバー(テストに Web サーバーを利用する場合)

(2) ブラウザーの種類

- ・ Internet Explorer 5.5 以上
Windows NT、Windows 2000、Windows XP、Windows 95、Mac OS 9、Mac OS X
- ・ Mozilla 0.9.4 以上
全プラットフォーム
Firefox0.9 以上・Netscape6.2.3 以上の Gecko ベースブラウザーを含む
- ・ Konqueror 5 以上
KDE 3.0.1 (Linux)

(3) 本ドキュメントが前提とする環境

#	環境名	値
1	OS	Windows XP Professional
2	ブラウザー	Internet Explorer 6.0 SP2

2.2 インストール環境

本ドキュメントで利用するインストール環境は以下のとおりです。

2.2.1 モジュール名

JsUnit	jsunit2_1.zip
JDK	j2sdk-1_4_2_09-windows-i586-p.exe
Ant	apache-ant-1.6.5-bin.zip
Apache	apache_2.0.55-win32-x86-no_ssl.msi

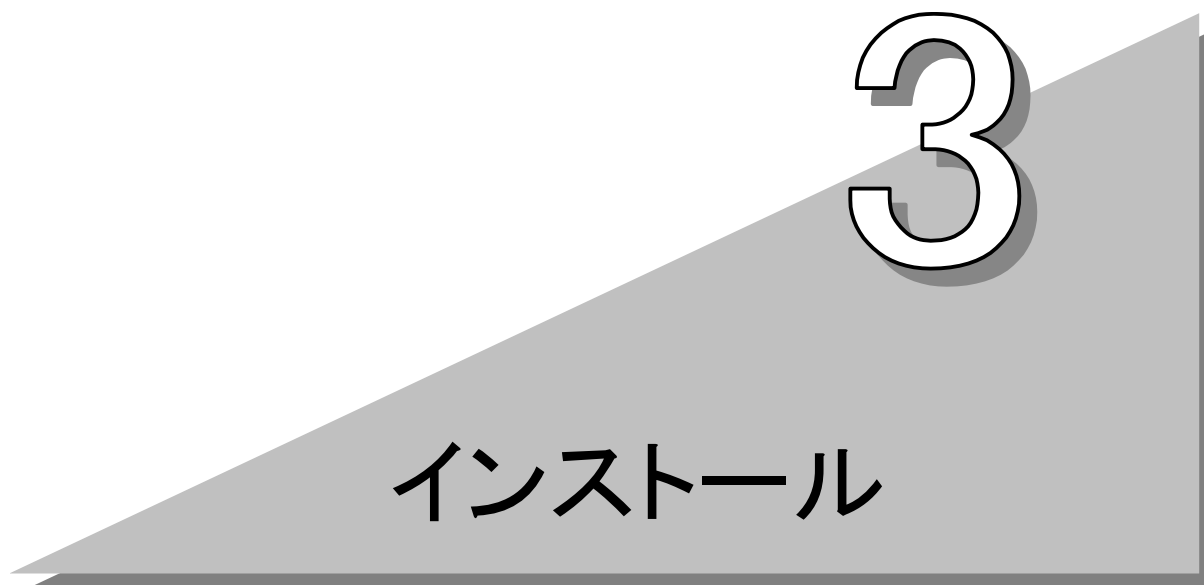
※ JUnit は JsUnit に付属するモジュールを利用します。

2.2.2 インストール先

JsUnit	C:\%usr%\local\%jsunit
JDK	C:\%j2sdk1.4.2_09
Ant	C:\%usr%\local\%apache-ant-1.6.5
Apache	C:\%Program Files%\Apache Group%\Apache2

2.2.3 環境変数

JAVA_HOME=	C:\%j2sdk1.4.2_09
ANT_HOME=	C:\%usr%\local\%apache-ant-1.6.5
Path=	%JAVA_HOME%\%bin;%ANT_HOME%\%bin



3. インストール

この章では、WindowsXP における JsUnit とその他のアプリケーションのインストール方法について説明します。

3.1	Apacheのインストール	3-2
3.2	JsUnitのインストール	3-7
3.3	JDKのインストール.....	3-9
3.4	Antのインストール.....	3-12

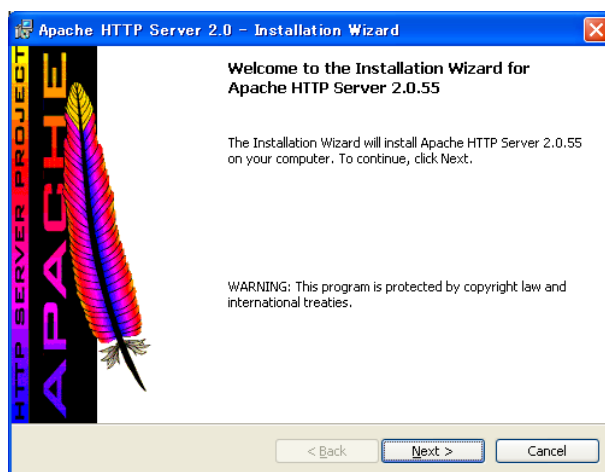
3.1 Apache のインストール

この節では、Apache のインストール方法を説明します。Web サーバーを利用して JsUnit を実行する場合、Web サーバーが必要です。本ドキュメントでは、Web サーバーとして Apache を利用します。

3.1.1 インストールの手順

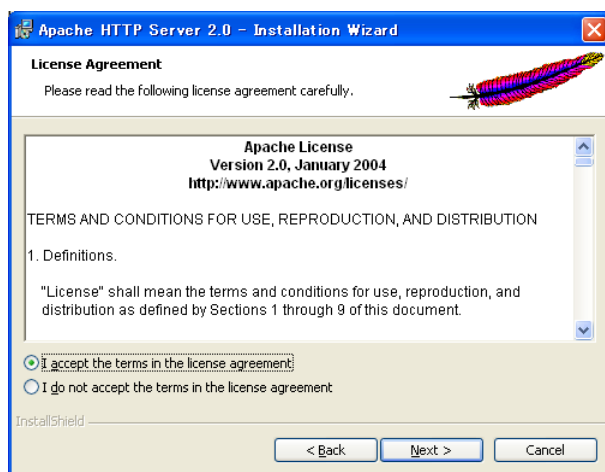
- ① Apacheグループのウェブサイト「<http://httpd.apache.org/> (2006/01 現在) より、Apache2 のインストーラー(`apache_2.0.55-win32-x86-no_ssl.msi`)を入手します。
- ② インストーラーを実行します。インストールウィザードが表示されます。

Fig. 3-1 Apache HTTP サーバー インストールウィザード : 開始



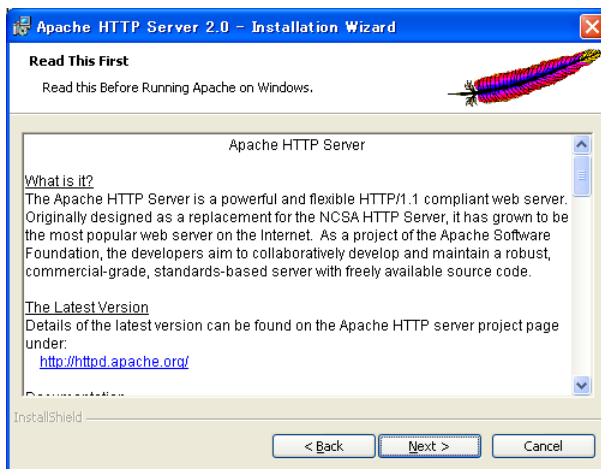
- ③ [Next]ボタンをクリックします。

Fig. 3-2 Apache HTTP サーバー インストールウィザード : ライセンスの同意



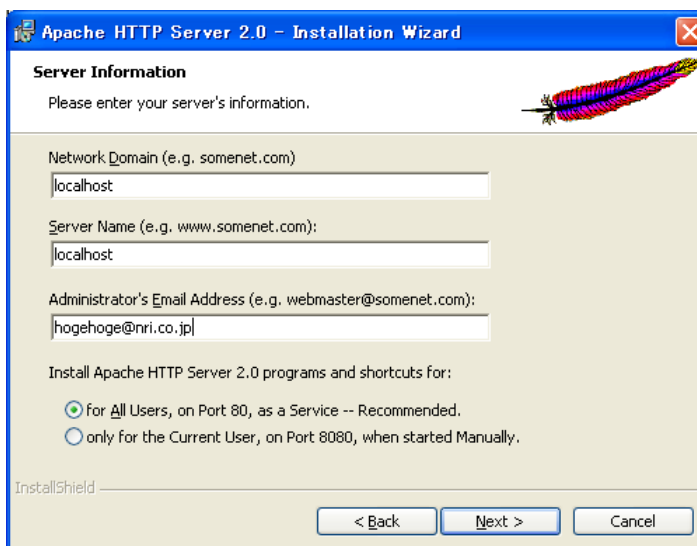
- ④ ライセンスの同意事項を読み、[I accept the terms in the license agreement]ラジオボタンを選択し、[Next]ボタンをクリックします。

Fig. 3-3 Apache HTTP サーバー インストールウィザード : 説明



- ⑤ [Next]ボタンをクリックします。

Fig. 3-4 Apache HTTP サーバー インストールウィザード : サーバー情報

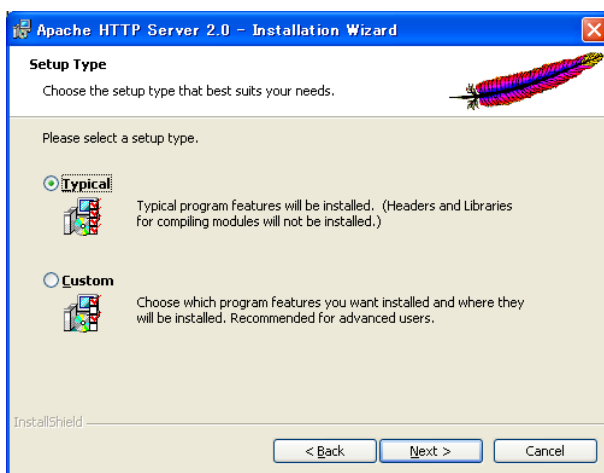


- ⑥ サーバー情報を入力し、[Next]ボタンをクリックします。

Tab. 3-1 サーバー情報設定内容

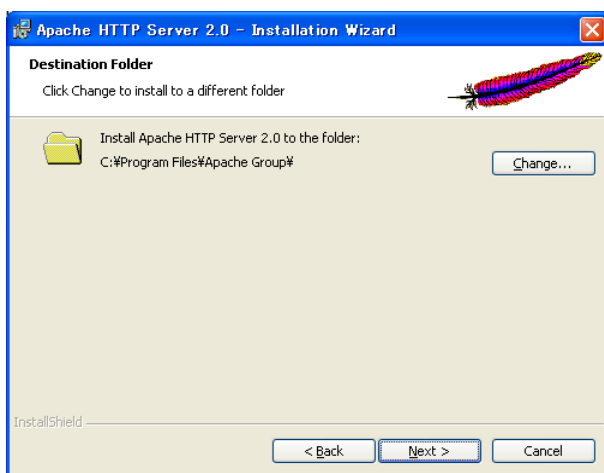
#	項目名	意味	例
1	Network Domain	ドメイン名	localhost
2	Server Name	サーバー名	localhost
3	Administrator's Email Address	管理者の E メールアドレス	hogehoge@nri.co.jp

Fig. 3-5 Apache HTTP サーバー インストールウィザード : セットアップのタイプ



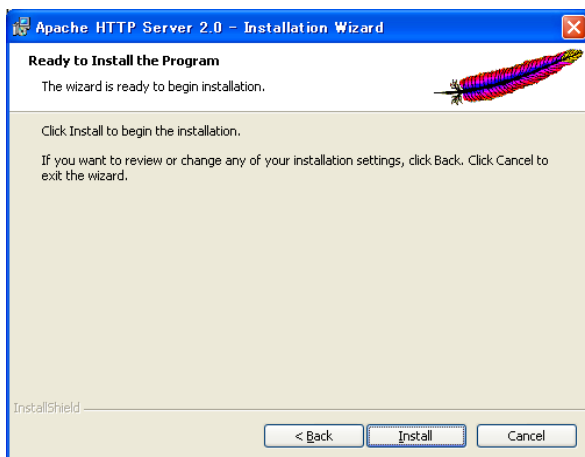
- ⑧ [Next]ボタンをクリックします。

Fig. 3-6 Apache HTTP サーバー インストールウィザード : ディレクトリーの選択



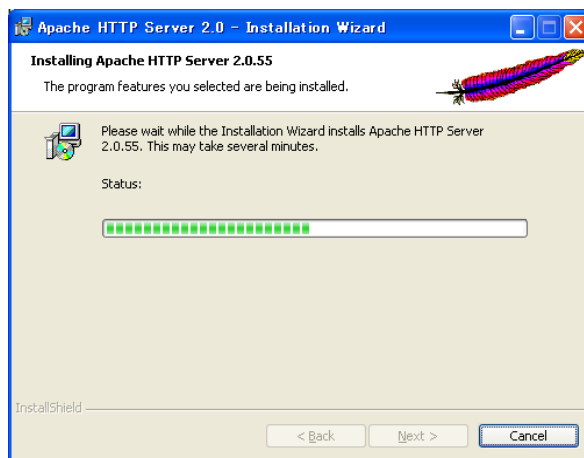
- ⑨ [Next]ボタンをクリックします。

Fig. 3-7 Apache HTTP サーバー インストールウィザード : インストールの開始



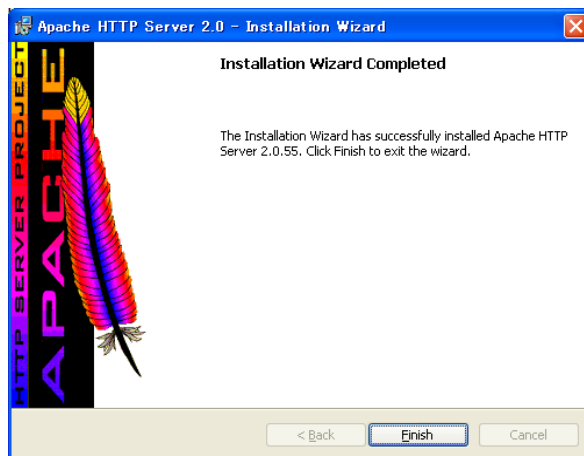
- ⑩ [Install]ボタンをクリックします。

Fig. 3-8 Apache HTTP サーバー インストールウィザード : インストール中



- ⑪ インストールが終了するまで待ちます。

Fig. 3-9 Apache HTTP サーバー インストールウィザード : インストール完了



- ⑫ [Finish]ボタンをクリックします。Apache が自動的に起動します。

Fig. 3-10 タスクバーに表示される Apache Monitor



3.1.2 インストールの確認

- ① Apache が起動していない場合、Windows のスタートメニューから [Apache HTTP Server v2.0.55]-[Control Apache Server]-[Start] を実行し、Apache を起動します。
- ② ブラウザーで以下のページを表示します。

http://localhost/

- ③ 以下のような画面が表示されることを確認します。

Fig. 3-11 Apache インストール時のテストページ



3.2 JsUnit のインストール

この節では、JsUnit のインストール方法を説明します。JsUnit のインストール方法は、ローカルで利用する場合と Web サーバーを利用する場合で異なります。両方法を、以下に示します。

3.2.1 ローカルで利用する場合

この項では、Web サーバーを利用せずに JsUnit を実行するための、JsUnit のインストール方法説明します。

(1) インストールの手順

- ① JsUnitホームページ「<http://www.edwardh.com/jsunit/> (2006/01 現在)」より、jsunit2_1.zip を入手します。
- ② 同 zip ファイルを解凍します。
- ③ 解凍したファイルを「C:\%usr%\local」に配置します。

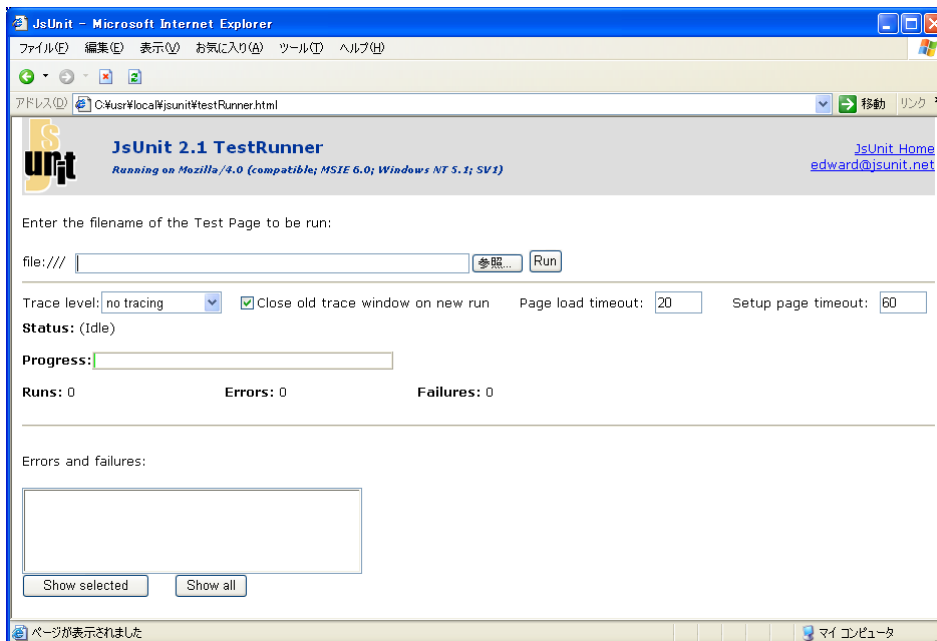
(2) インストールの確認

- ① ブラウザーで以下のページを表示します。

C:\%usr%\local\%jsunit%\testRunner.html

- ② Fig. 3-12 のような画面が表示されることを確認します。

Fig. 3-12 ローカルの testRunner.html



3.2.2 Web サーバーを利用する場合

この項では、Web サーバーを利用して JsUnit を実行するための、JsUnit のインストール方法説明します。

(1) インストールの手順

- ① 3.2.1 ローカルで利用する場合のインストール手順を実行します。
- ② Apache のドキュメントルート「C:\Program Files\Apache Group\Apache2\htdocs」に、「C:\usr\local\jsunit」配下にある以下のファイルをコピーします。
 - ・ testRunner.html
 - ・ app
 - ・ css
 - ・ images

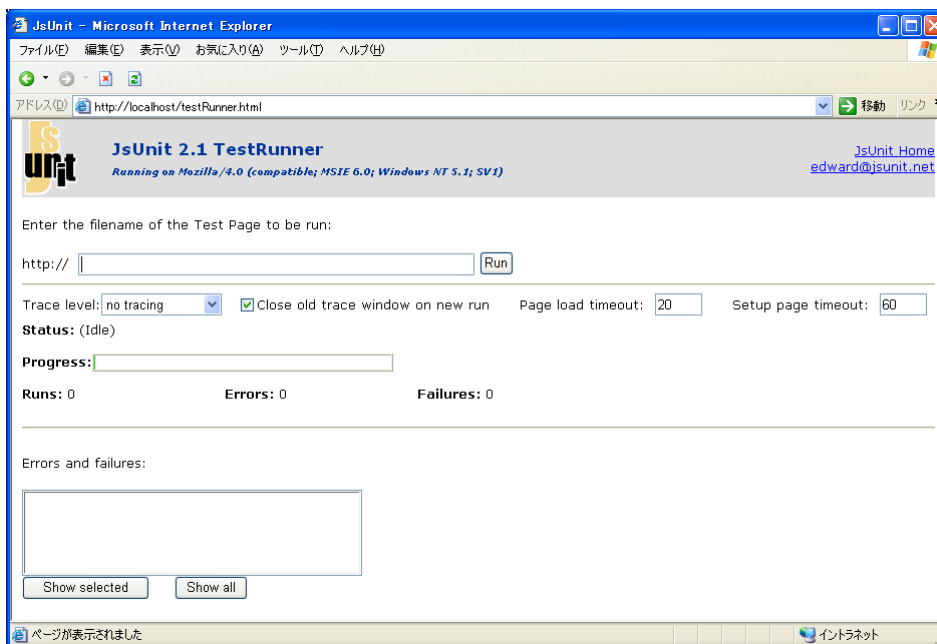
(2) インストールの確認

- ① Apache が起動していない場合、Windows のスタートメニューから [Apache HTTP Server v2.0.55]-[Control Apache Server]-[Start] を実行し、Apache を起動します。
- ② ブラウザーで以下のページを表示します。

http://localhost/testRunner.html

- ③ 以下のような画面が表示されることを確認します。

Fig. 3-13 Web サーバーの testRunner.html



3.3 JDK のインストール

この節では、JDK のインストール方法を説明します。JsUnit Server を利用する場合、Ant を実行するために JDK が必要です。

3.3.1 インストールの手順

- ① サン・マイクロシステムズのウェブサイト「<http://java.sun.com/products/archive/> (2006/01 現在)」より、j2sdk-1_4_2_09-windows-i586-p.exeを入手します。
- ② 同 exe ファイルを実行します。

Fig. 3-14 JDK インストールウィザード : 開始

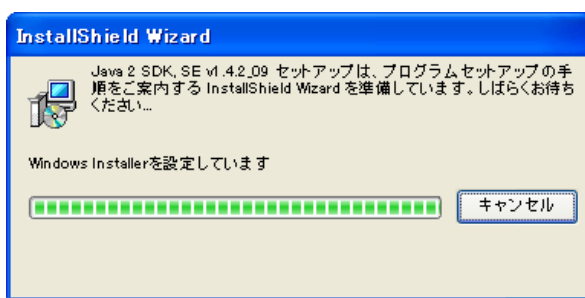
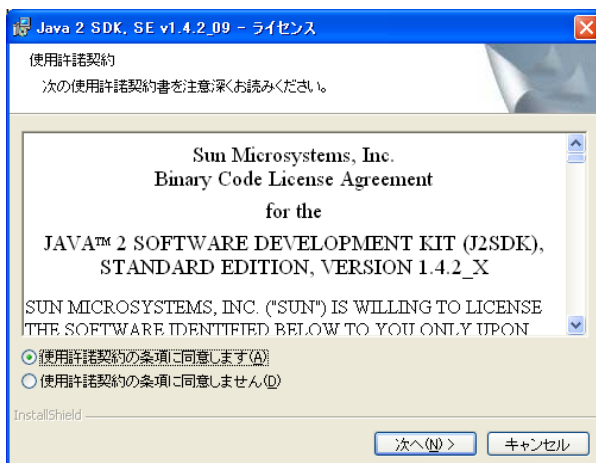
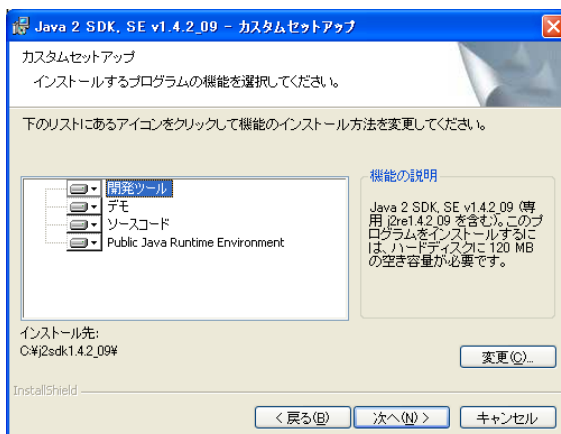


Fig. 3-15 JDK インストールウィザード : ライセンスの同意



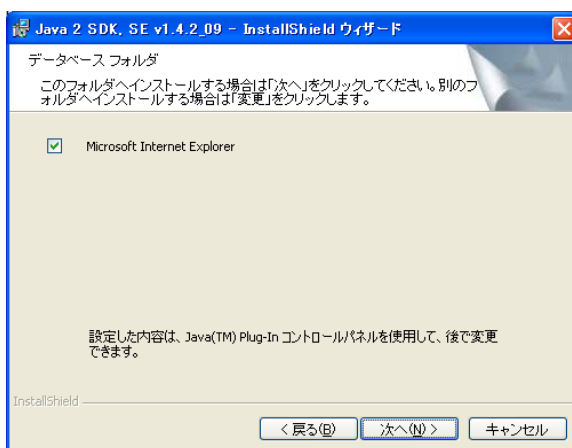
- ③ [使用許諾契約の条項に同意します]ラジオボタンを選択し、[次へ]ボタンをクリックします。

Fig. 3-16 JDK インストールウィザード : インストール先



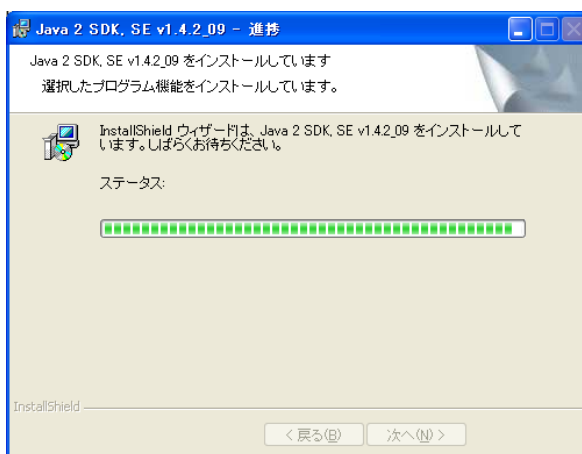
- ④ [次へ]ボタンをクリックします。

Fig. 3-17 JDK インストールウィザード : データベースフォルダ



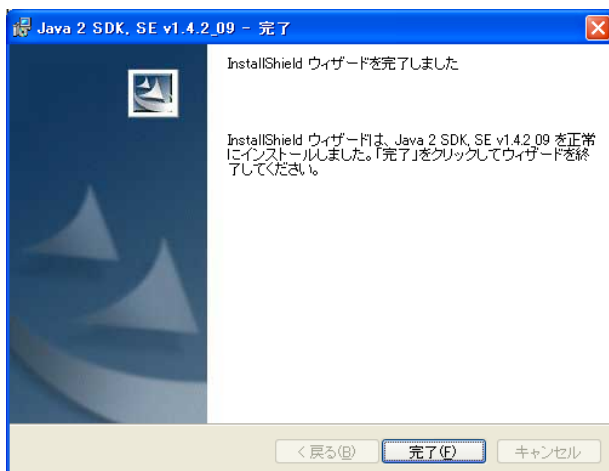
- ⑤ [次へ]ボタンをクリックします。

Fig. 3-18 JDK インストールウィザード : インストール中



- ⑥ インストールが終了するまで待ちます。

Fig. 3-19 JDK インストールウィザード : インストール完了



- ⑦ [完了]ボタンをクリックします。
- ⑧ Windows の[システムのプロパティ]画面を開き、[詳細設定]タブで以下の環境変数を追加します。

Tab. 3-2 追加する環境変数

#	環境変数名	値	備考
1	JAVA_HOME	C:\¥j2sdk1.4.2_09	環境変数を新規作成する。
2	Path	%JAVA_HOME%\¥bin	既存の環境変数の値を編集し、「;」で区切って先頭に追加する。

3.3.2 インストールの確認

コマンドプロンプトを起動し、以下のコマンドを実行します。

```
java -version
```

Java のバージョンが表示されることを確認します。

```
C:\¥>java -version
java version "1.4.2_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_09-b05)
Java HotSpot(TM) Client VM (build 1.4.2_09-b05, mixed mode)
```

3.4 Ant のインストール

この節では、Ant のインストール方法を説明します。JsUnit Server を利用する場合、Ant が必要です。Ant とは、Java ベースのビルドツールです。実行には JDK が必要です。

3.4.1 インストールの手順

- ① Apache Antのホームページ「<http://ant.apache.org/> (2006/01 現在)」より `apache-ant-1.6.5-bin.zip` を入手します。
- ② 同 zip ファイルを解凍します。
- ③ 解凍されたファイルを「`C:\usr\local`」に配置します。
- ④ Windows の[システムのプロパティ]画面を開き、[詳細設定]タブで以下の環境変数を追加します。

Tab. 3-3 追加する環境変数

#	環境変数名	値	備考
1	ANT_HOME	<code>C:\usr\local\apache-ant-1.6.5</code>	環境変数を新規作成する。
2	Path	<code>%ANT_HOME%\bin</code>	既存の環境変数の値を編集し、「;」で区切って追加する。

- ⑤ JsUnit Server で利用するため、「`C:\usr\local\jsunit\java\lib\junit.jar`」を、「`C:\usr\local\apache-ant-1.6.5\lib`」にコピーします。

3.4.2 インストールの確認

コマンドプロンプトを起動し、以下のコマンドを実行します。

```
ant -version
```

Ant のバージョンが表示されることを確認します。

```
C:\>ant -version
Unable to locate tools.jar. Expected to find it in C:\Program Files\Java\jre1.5.0_05\lib\tools.jar
Apache Ant version 1.6.5 compiled on June 2 2005
```

4

テストの記述と 単一環境におけるテスト

4. テストの記述と単一環境におけるテスト

この章では、JsUnit の実行方法について説明します。

4.1	JsUnitの概要	4-2
4.2	テストの準備	4-3
4.3	テストの実行	4-12

4.1 JsUnit の概要

JsUnit は単体テストを記述・実行するためのテスト・フレームワークです。JavaScript でテストを記述します。JsUnit テスト・フレームワーク自身も JavaScript で記述されています。

4.2 テストの準備

この節では、JsUnit テストの記述方法を説明します。JsUnit で JavaScript プログラムのテストを行うには、JavaScript で JsUnit の規則に則ったテストを作成する必要があります。以下に手順を示します。

4.2.1 JavaScript プログラムの開発

JavaScript プログラムを開発します。JsUnit でテストを行う場合、JavaScript プログラムは外部スクリプトファイルに記述されている必要があります。

本ドキュメントでは JavaScript プログラムの開発方法を扱いません。以下に、本ドキュメントで利用する JavaScript プログラム「pppicheck.js」を示します。

Fig. 4-1 pppicheck.js

```

var __gReYYYYMMDD = new RegExp("^[Yy]{8}$");
var __gReYYMMDD = new RegExp("^[Yy]{6}$");
var __gReCapitalAlpha = new RegExp("^[A-Z]+$");
var __g20Century = "19";
var __g21Century = "20";

function p3i_isYYYYMMDD(strDate_) {
    if (!__isObjectEnabled(strDate_)) {
        return false;
    }
    if (!__isMatch(strDate_, __gReYYYYMMDD)) {
        return false;
    }
    return __isDate(strDate_);
}

function p3i_isYYMMDD(strDate_) {
    if (!__isObjectEnabled(strDate_)) {
        return false;
    }
    if (!__isMatch(strDate_, __gReYYMMDD)) {
        return false;
    }
    if (parseInt(strDate_.substr(0, 2), 10) >= 70) {
        return __isDate(__g20Century.concat(strDate_));
    } else {
        return __isDate(__g21Century.concat(strDate_));
    }
}

function __isMatch(strToBeMatched_, rePattern_) {
    var objMatched = strToBeMatched_.match(rePattern_);
    if (objMatched == null) {
        return false;
    }
    return true;
}

function __isObjectEnabled(obj_) {
    if ((obj_ != null) && (obj_ != undefined) && (obj_.length != 0)) {
        return true;
    }
    return false;
}

function __isDate(strDate_) {
    if (strDate_.length != 8) {
        return false;
    }
    if (isNaN(__getIntegerYear(strDate_))) {
        return false;
    }
    if (__getIntegerYear(strDate_) < 1 || __getIntegerYear(strDate_) > 9999) {
        return false;
    }
    if (!__isMonth(__getIntegerMonth(strDate_))) {
        return false;
    }
    if (!__isDay(__getIntegerYear(strDate_), __getIntegerMonth(strDate_), parseInt(strDate_.substr(6, 2), 10), true)) {
        return false;
    }
    return true;
}

function __getIntegerYear(strDate_) {
    return parseInt(strDate_.substr(0, 4), 10);
}

function __getIntegerMonth(strDate_) {
    return parseInt(strDate_.substr(4, 2), 10);
}

function __isMonth(nMonth_) {

```

```

    if (isNaN(nMonth_)) {
        return false;
    }
    if (nMonth_ < 1 || nMonth_ > 12) {
        return false;
    }
    return true;
}
function __isDay(nYear_, nMonth_, nDay_, blnUseYear_) {
    if (isNaN(nDay_)) {
        return false;
    }
    var arrayYear = new Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    if (((nYear_ % 100 != 0) && (nYear_ % 4 == 0))
        || ((nYear_ % 400 == 0) && (nYear_ % 4 == 0))
        || blnUseYear_ == false) {
        arrayYear[1] = 29;
    }
    if (nDay_ < 1 || nDay_ > arrayYear[nMonth_ - 1]) {
        return false;
    }
    return true;
}
}

```

本ドキュメントでは、「pppcheck.js」に含まれる以下のメソッドをテストします。

Tab. 4-1 テスト対象メソッド

#	メソッド名	メソッドの仕様
1	p3i_isYYYYMMDD(strDate_)	strDate_の書式がYYYYMMDDである場合 true。それ以外の場合 false。
2	p3i_isYYMMDD(strDate_)	strDate_の書式がYYMMDDである場合 true。それ以外の場合 false。

4.2.2 テストスクリプトの記述

(1) JsUnit テストファイルを作成する

JsUnit テストを記述するための JsUnit テストファイルを作成します。html ファイルとして作成し、以下のファイルを JsUnit テストファイルからの相対パスでインクルードします。

Tab. 4-2 JsUnit テストファイルにインクルードするファイル

#	インクルードするファイル	例
1	JsUnit の/app/jsUnitCore.js	./app/jsUnitCore.js
2	テスト対象外部スクリプトファイル	pppcheck.js

Fig. 4-2 JsUnit テストファイルの例

- ※ JsUnit テストファイルと pppcheck.js を「C:\¥usr¥local¥jsunit」に配置する場合の例です。
- ※ インクルードに失敗するとJsUnitが動作しませんので、パスの記述に注意してください。

```

<html>
  <head>
    <title>Test Page for pppicheck.js</title>
    <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <script type="text/javascript" src="/app/jsUnitCore.js"></script>
    <script type="text/javascript" src="pppicheck.js"></script>
    <script type="text/javascript">
      <!--
      //ここに JsUnit テストメソッドを記述する。
      // -->
    </script>
  </head>
  <body>
  </body>
</html>

```

(2) JsUnit テストメソッドを用意する。

テストスクリプトを記述するための、JsUnit テストメソッドを用意します。JsUnit テストメソッドは「test」で始まる必要があります。

Tab. 4-3 本ドキュメントで記述する JsUnit テストメソッド

#	JsUnit テストメソッド	テスト対象メソッド
1	testP3i_isYYYYMMDD	p3i_isYYYYMMDD(strDate_)
2	testP3i_isYYMMDD	p3i_isYYMMDD(strDate_)

Fig. 4-3 JsUnit テストメソッドの例

```

<html>
  <head>
    <title>Test Page for pppicheck.js</title>
    <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <script type="text/javascript" src="/app/jsUnitCore.js"></script>
    <script type="text/javascript" src="pppicheck.js"></script>
    <script type="text/javascript">
      <!--
      //p3i_isYYYYMMDD メソッドのテスト。
      function testP3i_isYYYYMMDD() {
        //ここにテストスクリプトを記述する。
      }
      //p3i_isYYMMDD メソッドのテスト。
      function testP3i_isYYMMDD() {
        //ここにテストスクリプトを記述する。
      }
      // -->
    </script>
  </head>
  <body>
  </body>
</html>

```

(3) テストスクリプトを記述する

以下の JUnit メソッドを利用して、テストスクリプトを記述します。(d)にテストスクリプトの例を示します。

(a) assert メソッド

テスト結果と期待値を比較するメソッドです。

Tab. 4-4 assert メソッド一覧

#	メソッド	説明	例
1	assert([comment(※1), booleanValue)	booleanValue が true であることをチェックします。	assert(p3i_isYYMMDD("060101"));
2	assertTrue([comment], booleanValue)	booleanValue が true であることをチェックします。	assertTrue(p3i_isYYMMDD("060101"));
3	assertFalse([comment], booleanValue)	booleanValue が false であることをチェックします。	assertFalse(p3i_isYYMMDD("aaBBcc"));
4	assertEquals([comment], value1, value2)	val1 と val2 が等しいことをチェックします。	assertEquals("平成 160101", p3i_convertADToJapaneseEra("20060101", true));
5	assertNotEquals([comment], value1, value2)	val1 と val2 が等しくないことをチェックします。	assertEquals("平成 160101", p3i_convertADToJapaneseEra("20060101", false));
6	assertNull([comment], value)	value が null であることをチェックします。	assertNull(p3i_convertADToJapaneseEra(null, true));
7	assertNotNull([comment], value)	value が null でないことをチェックします。	assertNotNull("", p3i_convertADToJapaneseEra("", true));
8	assertUndefined([comment], value)	value が undefined であることをチェックします。	assertUndefined(p3i_focusFirstObject(""));
9	assertNotUndefined([comment], value)	value が undefined でないことをチェックします。	assertNotUndefined(p3i_focusFirstObject(""));
10	assertNaN([comment], value)	value が NaN(※2)であることをチェックします。	assertNaN(p3i_convertADToJapaneseEra("20060101", true));

#	メソッド	説明	例
11	assertNotNaN([comment], value)	value が NaN でないことをチェックします。	assertNotNaN(p3i_convertADToJapaneseEra("20060101", false));
12	fail(comment)	テストを強制的に失敗させます。	fail("テストは失敗しました。");

※1 #1～#11 の comment 引数は省略可能です。comment 引数にテストコメントを記述すると、テストが失敗した場合に詳細情報の一部として参照できます。

※2 NaN : Not a Number

例えば、p3i_YYYYMMDD(strDate_)メソッドの引数に"20060101"を指定した場合に true が戻ることをテストする場合、以下のように記述します。

```
assertTrue("引数に 20060101 を指定した場合は true が戻ります。", p3i_isYYYYMMDD("20060101"));
```

メソッドの戻り値が true であることをテストするため、assert メソッドもしくは assertTrue メソッドを利用します。

第 1 引数はコメントです。省略可能ですが、記述しておくと同メソッドでテストが失敗した場合に詳細情報として参照できます。

第 2 引数は p3i_isYYYYMMDD メソッドの実行です。

(b) 初期化・終了メソッド

初期化処理や終了処理を記述するメソッドです。これらのメソッドをオーバーライドして利用します。assert メソッドとは記述方法が異なります。

Tab. 4-5 初期化処理・終了処理メソッド一覧

#	メソッド	説明
1	setUpPage()	JUnitテストファイルが読み込まれた際に、一度だけ実行されます。ページの初期化処理等を記述します。同メソッドの最後で、setUpPageStatus 変数に"complete"をセットする必要があります。JUnit等にはないJUnit独自のメソッドです。
2	setUp()	(2)で用意する JUnit テストメソッドが実行される前に毎回呼び出されます。メソッド単位の初期化処理等を記述します。
3	tearDown()	(2)で用意する JUnit テストメソッドが実行された後に毎回呼び出されます。メソッド単位の終了処理等を記述します。

例えば、JUnit テストメソッド毎に共通の初期化処理を行う場合、以下のように記述します。

```
function setUp() {
  //初期化処理。テキストフィールドをクリア。
  document.form1.input1.value = "";
}
```

setUp メソッドの中に初期化処理スクリプトを記述します。setUpPage メソッドおよび tearDown メソッドも同様です。

(c) トレースメソッド

後述する Tracing 画面(4.3.1(2)(a))に情報を出力するメソッドです。

Tab. 4-6 トレースメソッド一覧

#	メソッド	出力される トレースレベル(※1)			例
		warning	inform	debug	
1	warn(message, [value(※2)])	○			warn("変数 x は 5 より大きくなれ ばならない", x);
2	inform(message, [value]) info(message, [value])	○	○		inform("変数 x の値", x);
3	debug(message, [value])	○	○	○	debug("変数 x の値", x);

※1 詳しくは 4.3.1(1) 実行方法を参照してください。

※2 #1～#3 の value 引数は省略可能です。

(d) テストスクリプトの例

JUnit テストファイルの例(mytest1.html、mytest2.html)を示します。

Fig. 4-4 mytest1.html

```

<html>
  <head>
    <title>Test Page for pppcheck.js</title>
    <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <script type="text/javascript" src="./app/jsUnitCore.js"></script>
    <script type="text/javascript" src="pppcheck.js"></script>
    <script type="text/javascript">
    <!--
    //ページの初期化処理等を記述。
    function setUpPage() {
      //Tracing 画面に情報を出力。
      inform("mytest1.html");
      //setUpPage の終わり。
      setUpPageStatus = "complete";
    }
    //p3i_isYYYYMMDD メソッドのテスト。
    function testP3i_isYYYYMMDD() {
      assertTrue("引数に 20060101 を指定した場合は true が戻ります。",
        p3i_isYYYYMMDD("20060101"));
      assertFalse("引数に aaaaBBcc を指定した場合は false が戻ります。",
        p3i_isYYYYMMDD("aaaabbcc"));
    }
    // p3i_isYYMMDD メソッドのテスト。
    function testP3i_isYYMMDD() {
      assertTrue("引数に 060101 を指定した場合は true が戻ります。",
        p3i_isYYMMDD("060101"));
      assertFalse("引数に aaBBcc を指定した場合は false が戻ります。",
        p3i_isYYMMDD("aaBBcc"));
    }
    // -->
    </script>
  </head>
  <body>
  </body>
</html>

```

Fig. 4-5 mytest2.html

テストの値にフォームを与えることもできます。

```

<html>
  <head>
    <title>Test Page for pppicheck.js</title>
    <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <script type="text/javascript" src="./app/jsUnitCore.js"></script>
    <script type="text/javascript" src="pppicheck.js"></script>
    <script type="text/javascript">
    <!--
    //ページの初期化処理等を記述。
    function setUpPage() {
      //Tracing 画面に情報を出力。
      inform("mytest2.html");
      //setUpPage の終わり。
      setUpPageStatus = "complete";
    }
    //JsUnit テストメソッド毎の初期化処理を記述。
    function setUp() {
      //初期化処理。テキストフィールドをクリア。
      document.form1.input1.value = "";
    }
    //p3i_isYYYYMMDD メソッドのテスト。
    function testP3i_isYYYYMMDD() {
      document.form1.input1.value = "20060101";
      assertTrue("引数に 20060101 を指定した場合は true が戻ります。",
        p3i_isYYYYMMDD(document.form1.input1.value));
      document.form1.input1.value = "aaaaBBcc";
      assertFalse("引数に aaaaBBcc を指定した場合は false が戻ります。",
        p3i_isYYYYMMDD(document.form1.input1.value));
    }
    // p3i_isYYMMDD メソッドのテスト。
    function testP3i_isYYMMDD() {
      document.form1.input1.value = "060101";
      assertTrue("引数に 060101 を指定した場合は true が戻ります。",
        p3i_isYYMMDD(document.form1.input1.value));
      document.form1.input1.value = "aaBBcc";
      assertFalse("引数に aaBBcc を指定した場合は false が戻ります。",
        p3i_isYYMMDD(document.form1.input1.value));
    }
    // -->
  </script>
</head>
<body>
  <!-- テストで利用するテキストフィールド。 -->
  <form name="form1">
    <input name="input1" type="text">
  </form>
</body>
</html>

```

(4) TestSuite を作成する

JsUnit テストファイルを複数作成した場合、新しい JsUnit テストファイルを用意して TestSuite を作成し全ての JsUnit テストファイルを呼び出すように記述すると、複数の JsUnit テストファイルを一度に実行できます。JsUnit を実行する場合は、個々の JsUnit テストファイルではなく、TestSuite を記述した JsUnit テストファイルを指定します。

Fig. 4-6 mytestsuite.html

```

<html>
  <head>
    <title>TestSuite Page</title>
    <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <script type="text/javascript" src="./app/jsUnitCore.js"></script>
    <script type="text/javascript">
    <!--
      function suite() {
        //TestSuite を作成します。
        var newsuite = new top.jsUnitTestSuite();
        //jsUnitTestSuite クラスの addTestPage メソッドを利用して
        //個々の JsUnit テストファイルを TestSuite に登録します。
        newsuite.addTestPage("mytest1.html");
        newsuite.addTestPage("mytest2.html");
        return newsuite;
      }
    // -->
    </script>
  </head>
  <body>
  </body>
</html>

```

4.3 テストの実行

この節では、JsUnit テストの実行方法を説明します。

4.3.1 ローカルで利用する場合

この項では、Web サーバーを利用せずに JsUnit を実行する方法を説明します。

(1) 実行方法

テストの実行方法を以下に示します。

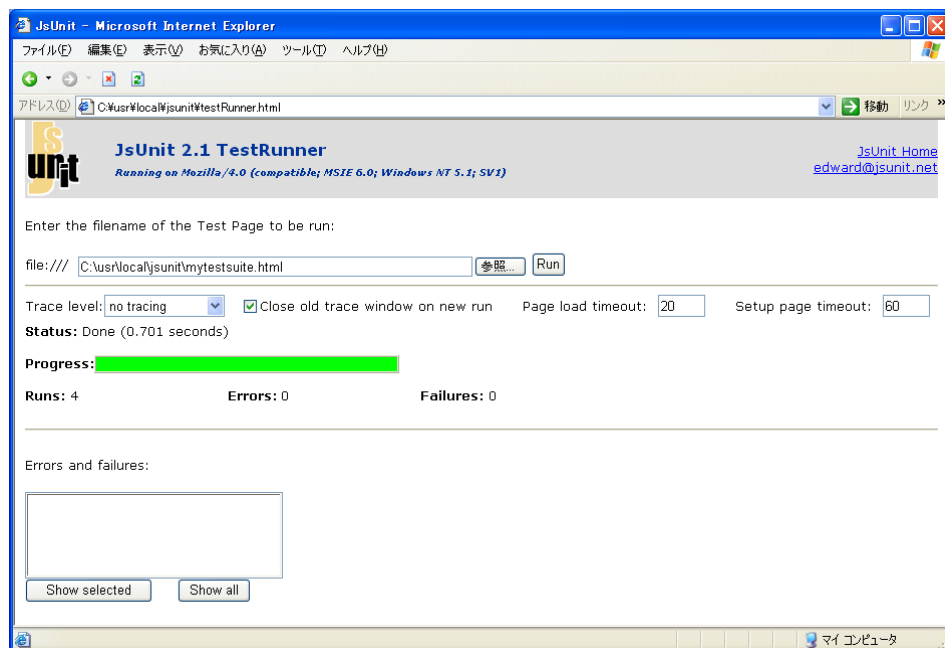
- ① ブラウザーで「C:\usr\local\jsunit\testRunner.html」を表示します。
- ② [参照]ボタンをクリックして、作成した JsUnit テストファイルを選択します。
- ③ [Trace level:]を選択します。トレースレベルにより、Tracing 画面に出力されるトレースメソッドが異なります。

Tab. 4-7 トレースレベル

#	トレースレベル	出力されるトレースメソッド		
		warn	inform	debug
1	no tracing			
2	warning	○		
3	inform	○	○	
4	debug	○	○	○

- ④ [Run]ボタンをクリックします。

Fig. 4-7 実行結果(テストが成功した場合)



注 ブラウザーによってはキャッシュに保存されたJsUnitテストファイルがテスト対象になり、

正しくテストが実行できない事があります。正しくテストが実行できない場合は、キャッシュの設定を変更してください。

(2) 実行結果の参照方法

テスト実行結果の参照方法を以下に示します。

(a) Tracing 画面

トレースメソッドの実行結果を出力します。Tab. 4-7 トレースレベルで選択したトレースレベルのトレースメソッドの実行結果を出力します。

Fig. 4-8 Tracing 画面



(b) Progress

テストの進捗状況を表示します。テスト結果により色が変わります。

Tab. 4-8 Progress

#	テスト結果	Progress の色
1	テストが全て成功した場合	緑
2	テストに失敗が含まれる場合	赤

(c) Runs / Errors / Failures

テストの数を表示します。

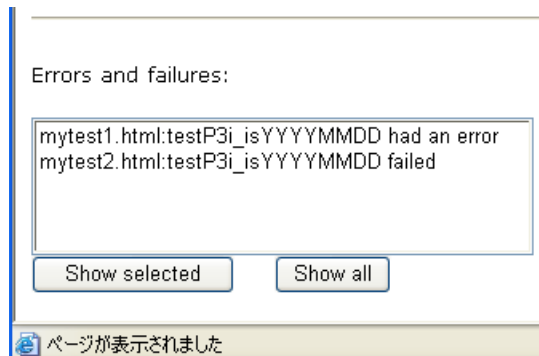
Tab. 4-9 Runs / Errors / Failures

#	項目名	Progress の色
1	Runs	実行したテストの数。assert メソッドの数ではなく、JsUnit テストメソッドの数。
2	Errors	発生したエラーの数。JavaScript の構文エラーなどが含まれる。
3	Failures	失敗したテストの数。

(d) Errors and failures

Tab. 4-9 Runs / Errors / Failures の Errors / Failures について、詳細情報を表示します。

Fig. 4-9 Errors and failures



- ① 個々の Error/Failure 情報を参照するには、参照する行を選択して[Show selected]ボタンをクリックします。Failure 情報には、assert メソッドで記述したコメントが含まれます。

Fig. 4-10 [Show selected]ボタン 実行結果



- ② 全ての Error/Failure 情報を参照するには、[Show all]ボタンをクリックします。

Fig. 4-11 [Show all]ボタン 実行結果



4.3.2 Web サーバーを利用する場合

この項では、Web サーバーを利用して JsUnit を実行する方法を説明します。

(1) 実行方法

テストの実行方法を以下に示します。

- ① テスト対象外部スクリプトファイルと JsUnit テストファイルを Web サーバーのドキュメントルート(Apache の場合は「C:\Program Files\Apache Group\Apache2\htdocs」)に配置します。
- ② ブラウザーで「<http://localhost/testRunner.html>」を表示します。
- ③ [[http://](#)]に JsUnit テストファイルのパスを記述します。(パスに [http://](#)は要りません。)
- ④ [Trace level:]を選択します。(4.3.1(1)③を参照)
- ⑤ [Run]ボタンをクリックします。

(2) 実行結果の参照方法

テスト実行結果の参照方法は、4.3.1(2)実行結果の参照方法と同じです。

5

複数環境における テスト

5. 複数環境におけるテスト

この章では、JsUnit Server の実行方法について説明します。

5.1	JsUnit Serverの概要	5-2
5.2	Standalone Test	5-3
5.3	Distributed Test.....	5-9

5.1 JsUnit Server の概要

JsUnit テストを複数環境(複数のブラウザー・複数のマシン)で実行する機能です。以下の2つの機能に分けられます。

(1) Standalone Test

1台のマシンにおいて複数のブラウザーに対し、JsUnit テストを実行することができます。

(2) Distributed Test

複数のマシンにおいて複数のブラウザーに対し、JsUnit テストを実行することができます。

なお、以下の理由により本ドキュメントでは Web サーバーを利用して JsUnit テストを実行しません。

- ・ テスト対象マシンに JsUnit をインストールすることで、Web サーバーを用いずにテストを行うこともできますが、JsUnit テストファイルの配布が煩雑になってしまいます。
- ・ JsUnit テストを呼び出す URL は、全てのブラウザーで同一である必要があります。

5.2 Standalone Test

この節では、Standalone Test の概要と実行方法を説明します。

5.2.1 テストの概要

Standalone Test は、1 台のマシンにおいて複数のブラウザに対し、JUnit テストを自動的に実行します。

Standalone Test には、以下のソフトウェアが必要です。3 章を参照してインストールしてください。

- ・ JUnit
- ・ JDK
- ・ Ant + JUnit
- ・ Web サーバー (Web サーバーを利用して JUnit を実行する場合に必要)

5.2.2 テストの配備

この項では、Standalone Test の準備方法を説明します。

(1) テストの準備

- ① 4.1 テストの準備を参照して、JUnit テストファイルを用意します。
- ② 4.2 テストの実行を参照して、JUnit テストファイルを配備します。
- ③ ブラウザーの一つで、テストが実行できることを確認します。
- ④ JUnit に付属する build.xml ファイルを編集します。(2) build.xml の編集を参照してください。

※ Web サーバーを利用せずにローカルにテストを配備してテストを実行する場合、テスト対象ブラウザで ActiveX を利用不可に設定していると、テストが途中で止まってしまいます。ActiveX を利用可能に設定するか、Web サーバーを利用してテストしてください。

(2) build.xml の編集

JUnit テストファイルやその他の環境情報を設定するため、JUnit に付属する Ant 用 build.xml ファイル「C:\%usr%\local%\jsunit%\build.xml」を編集します。同 xml ファイルを開き、以下の <property>タグを修正します。

(a) browserFileNames

テスト対象ブラウザのパスを設定します。複数のブラウザを指定する場合は、「,」で区切ります。

(例)

```
<property name="browserFileNames" value="c:\program files\internet explorer\iexplore.exe,c:\program files\Mozilla Firefox\Firefox.exe"/>
```

※ (例)は、表示の都合上複数行に分けて記述していますが、build.xml には 1 行で記述します。

(b) port

JsUnit Server が利用するポート番号を指定します。指定がなければ、JsUnit Server はデフォルトのポート番号 8080 を利用します。

(例)
`<property name="port" value="8000"/>`

(c) url

ブラウザから JsUnit テストの URL を指定します。以下のクエリー文字列も指定します。クエリー文字列同士は「&」で連結します。

- ・ **testPage** :
JsUnit テストファイルを指定します。Web サーバーを利用する場合、「http://」は付けません(例②)。
- ・ **autoRun** :
テストを自動的に実行するため、「true」を指定します。
- ・ **submitresults** :
実行結果を受け取る JUnit Server のサーブレットを指定します。以下のように指定します。
「<IP アドレス>:<(b) port で指定したポート番号>/jsunit/acceptor」

(例① ローカルで利用する場合)
`<property name="url" value="file:///c:/usr/local/jsunit/testRunner.html?testPage=c:%usr%local%jsunit%mytestsuite.html&autoRun=true&submitResults=localhost:8080/jsunit/acceptor"/>`

(例② Web サーバーを利用する場合)
`<property name="url" value="http://localhost/testRunner.html?testPage=localhost/mytestsuite.html&autoRun=true&submitResults=localhost:8080/jsunit/acceptor"/>`

※ (例)は、表示の都合上複数行に分けて記述していますが、**build.xml**には1行で記述します。

(d) logsDirectory

テスト結果を出力するディレクトリーを指定します。

(例)
`c:%usr%local%jsunit%logs`

(e) build.xml ファイルの例

```
<project name="JsUnit" default="create_distribution" basedir=".">
    <property name="browserFileNames" value="c:%program files%Mozilla Firefox%firefox.exe,c:%program files%internet explorer%iexplore.exe"/>
    <property name="url"
value="http://localhost/testRunner.html?testPage=localhost/mytestsuite.html&autoRun=true&submitResults=localhost:8080/jsunit/acceptor"/>
    <property name="remoteMachineURLs" value="" />
    <property name="port" value="" />
    <property name="resourceBase" value="" />
    <property name="logsDirectory" value="c:%usr%local%jsunit%logs"/>
-- 以下省略 --
```

5.2.3 テストの実行

この項では、Standalone Test の実行方法を説明します。

(1) standalone_test ターゲットの実行

- ① コマンドプロンプトを起動します。
- ② 「C:¥usr¥local¥jsunit」をカレントディレクトリにします。

```
cd C:¥usr¥local¥jsunit
```

- ③ Ant で standalone_test ターゲットを実行します。

```
ant standalone_test
```

(2) 実行結果の参照方法

(a) Ant に表示される結果

全てのテスト対象ブラウザでテストが成功した場合、コマンドプロンプト上の Ant 実行結果に「BUILD SUCCESSFUL」が表示されます。

Fig. 5-1 Ant 実行結果 (テストが成功した場合)

```
C:\usr\local\jsunit>ant standalone_test
Buildfile: build.xml

standalone_test:
[junit] 10:34:04.107 EVENT Checking Resource aliases
[junit] 10:34:04.177 EVENT Starting Jetty/4.2.x
[junit] 10:34:04.197 EVENT Started HttpContext[/jsunit]
[junit] 10:34:04.217 EVENT Started SocketListener on 0.0.0.0:8080
[junit] 10:34:04.217 EVENT Started JsUnit Server
[junit] Wed Jan 18 10:34:04 JST 2006: port: 8080
[junit] resourceBase: C:\usr\local\jsunit\
[junit] logsDirectory: c:\usr\local\jsunit\logs
[junit] browserFileNames: [c:\program files\Mozilla Firefox\firefox.exe, c:\program files\internet explorer\iexplore.exe]
[junit] url: http://localhost/testRunner.html?testPage=localhost/mytestsuite.html&autoRun=true&submitresults=true
[junit] Wed Jan 18 10:34:04 JST 2006: StandaloneTest: launching c:\program files\Mozilla Firefox\firefox.exe
[junit] Wed Jan 18 10:34:04 JST 2006: StandaloneTest: waiting for c:\program files\Mozilla Firefox\firefox.exe to submit result
[junit] Wed Jan 18 10:34:08 JST 2006: ResultAcceptorServlet: Received request
[junit] Wed Jan 18 10:34:08 JST 2006: ResultAcceptorServlet: Done
[junit] 10:34:09.224 EVENT Started HttpContext[/]
[junit] Wed Jan 18 10:34:09 JST 2006: StandaloneTest: launching c:\program files\internet explorer\iexplore.exe
[junit] Wed Jan 18 10:34:09 JST 2006: StandaloneTest: waiting for c:\program files\internet explorer\iexplore.exe to submit result
[junit] Wed Jan 18 10:34:13 JST 2006: ResultAcceptorServlet: Received request
[junit] Wed Jan 18 10:34:13 JST 2006: ResultAcceptorServlet: Done
[junit] 10:34:13.590 EVENT Stopping Acceptor ServerSocket[addr=0.0.0.0/0.0.0.0, port=0, localport=8080]
[junit] 10:34:13.621 EVENT Stopped SocketListener on 0.0.0.0:8080
[junit] 10:34:13.631 EVENT Stopped HttpContext[/jsunit]
[junit] 10:34:13.631 EVENT Stopped HttpContext[/]
[junit] 10:34:13.631 EVENT Stopped JsUnit Server

BUILD SUCCESSFUL
Total time: 12 seconds
```

テストが失敗した場合、コマンドプロンプト上の Ant 実行結果に「BUILD FAILED」が表示されます。テストは失敗が発生したブラウザで終了します。残りのブラウザに対するテストは行われません。

Fig. 5-2 Ant 実行結果 (テストが失敗した場合)

```
C:\usr\local\jsunit>ant standalone_test
Buildfile: build.xml

standalone_test:
[junit] 10:32:28.029 EVENT Checking Resource aliases
[junit] 10:32:28.099 EVENT Starting Jetty/4.2.x
[junit] 10:32:28.119 EVENT Started HttpContext[/jsunit]
[junit] 10:32:28.139 EVENT Started SocketListener on 0.0.0.0:8080
[junit] 10:32:28.139 EVENT Started JsUnit Server
[junit] Wed Jan 18 10:32:28 JST 2006: port: 8080
[junit] resourceBase: C:\usr\local\jsunit\
[junit] logsDirectory: c:\usr\local\jsunit\logs
[junit] browserFileNames: [c:\program files\Mozilla Firefox\firefox.exe, c:\program files\internet explorer\explore.exe]
[junit] url: http://localhost/testRunner.html?testPage=localhost/mytestsuite.html&autoRun=true&submitresults=true
[junit] Wed Jan 18 10:32:28 JST 2006: StandaloneTest: launching c:\program files\Mozilla Firefox\firefox.exe
[junit] Wed Jan 18 10:32:28 JST 2006: StandaloneTest: waiting for c:\program files\Mozilla Firefox\firefox.exe to submit result
[junit] Wed Jan 18 10:32:32 JST 2006: ResultAcceptorServlet: Received request
[junit] Wed Jan 18 10:32:32 JST 2006: Problems:
[junit] -----
[junit] mytest2.html:testP3i_isYYYYMMDD failed:
[junit] "20060101?? ??>?? $???" Call to assertTrue(boolean) with false Stack trace follows: > JsUnitException > _assert > assertTrue > testP3i_isYYYYMMDD
[junit] -----
[junit] Wed Jan 18 10:32:32 JST 2006: ResultAcceptorServlet: Done
[junit] 10:32:33.186 EVENT Stopping Acceptor ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=8080]
[junit] 10:32:33.196 EVENT Stopped SocketListener on 0.0.0.0:8080
[junit] 10:32:33.196 EVENT Stopped HttpContext[/jsunit]
[junit] 10:32:33.196 EVENT Stopped JsUnit Server

BUILD FAILED
C:\usr\local\jsunit\build.xml:69: Test net.jsunit.StandaloneTest failed

Total time: 7 seconds
```

(b) ログファイルに出力される結果

build.xml ファイルの logsDirectory プロパティで指定したディレクトリー(本ドキュメントでは「C:\usr\local\jsunit\logs」)にテストの実行結果を格納したログファイルが出力されます。実行結果はブラウザ毎に出力されます。ログファイルの名前は JUnit Server が自動的に採番します(例 : 1137555838765.xml)。

Fig. 5-3 ログ xml ファイル (テストが成功した場合)

```
<?xml version="1.0" encoding="UTF-8"?>
<testsuite id="1137555838765" errors="0" failures="0" name="JsUnitTestCase" tests="4" time="0.871">
  <properties>
    <property name="JsUnitVersion" value="2.1" />
    <property name="userAgent" value="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)" />
    <property name="remoteAddress" value="127.0.0.1" />
    <property name="baseUrl" value="http://localhost/mytestsuite.html" />
  </properties>
  <testcase name="mytest1.html:testP3i_isYYYYMMDD" time="0.0" />
  <testcase name="mytest1.html:testP3i_isYYMMDD" time="0.01" />
  <testcase name="mytest2.html:testP3i_isYYYYMMDD" time="0.0" />
  <testcase name="mytest2.html:testP3i_isYYMMDD" time="0.0" />
</testsuite>
```

テストが失敗した場合、<failure>タグに詳細情報が出力されます。

Fig. 5-4 ログ xml ファイル (テストが失敗した場合)

```
<?xml version="1.0" encoding="UTF-8"?>
<testsuite id="1137555842861" errors="0" failures="1" name="JUnitTestCase" tests="4" time="1.683">
  <properties>
    <property name="JUnitVersion" value="2.1" />
    <property name="userAgent" value="Mozilla/5.0 (Windows; U; Windows NT 5.1; ja; rv:1.8)
      Gecko/20051111 Firefox/1.5" />
    <property name="remoteAddress" value="127.0.0.1" />
    <property name="baseUrl" value="http://localhost/mytestsuite.html" />
  </properties>
  <testcase name="mytest1.html:testP3i_isYYMMDD" time="0.02" />
  <testcase name="mytest1.html:testP3i_isYYYYMMDD" time="0.01">
    <failure message="&quot;20060101??P??¥?・????????&quot;
      Call to assertTrue(boolean) with false Stack trace follows: &gt;
      JsUnitException &gt; _assert &gt; assertTrue &gt; testP3i_isYYYYMMDD" />
  </testcase>
  <testcase name="mytest2.html:testP3i_isYYMMDD" time="0.01" />
  <testcase name="mytest2.html:testP3i_isYYYYMMDD" time="0.0" />
</testsuite>
```

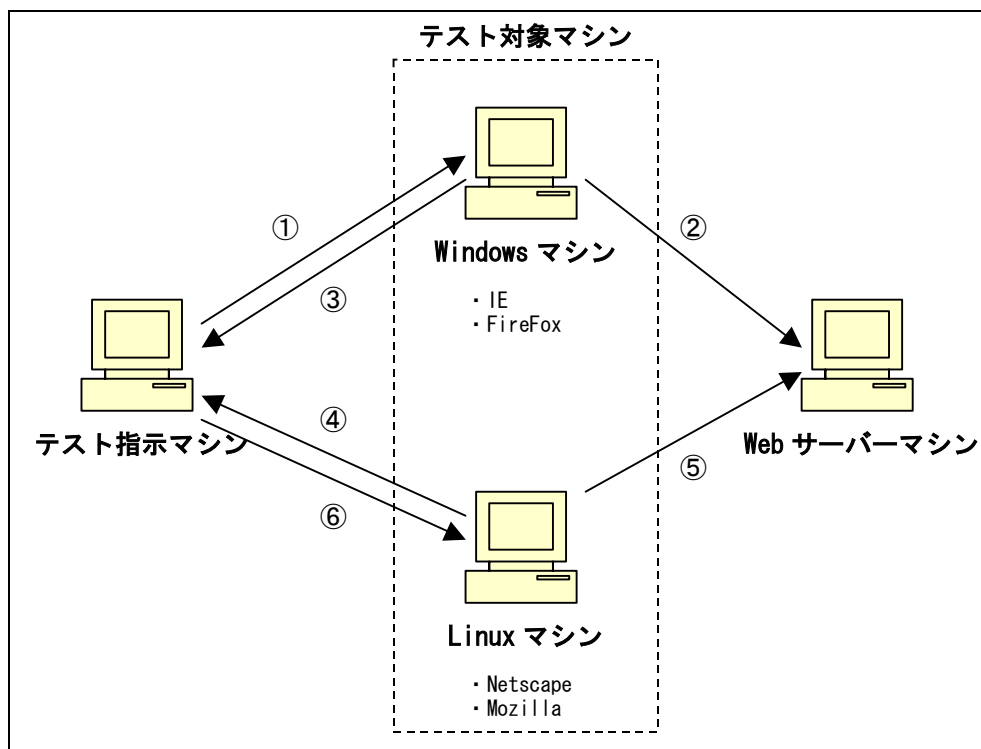
5.3 Distributed Test

この節では、Distributed Test の概要と実行方法を説明します。

5.3.1 テストの概要

複数のマシンにおいて複数のブラウザに対し、JsUnit テストを実行することができます。JsUnit Server は、各マシンに準備した Standalone Test を実行することで Distributed Test を実現します。

Fig. 5-5 Distributed Test の例



Tab. 5-1 Fig. 5-5 の流れ

#	流れの内容
①	テスト指示マシンが Windows マシンに JsUnit テストを指示する。
②	Windows マシンが Web サーバーマシンから JsUnit テストファイルを取得し、JsUnit テストを実行する。
③	Windows マシンがテスト指示マシンにテスト結果を返す。
④	テスト指示マシンが Linux マシンに JsUnit テストを指示する。
⑤	Linux マシンが Web サーバーマシンから JsUnit テストファイルを取得し、JsUnit テストを実行する。
⑥	Linux マシンがテスト指示マシンにテスト結果を返す。

※ テスト指示マシン・Web サーバーマシンは、Windows マシンもしくは Linux マシンと同一でも問題ありません。

5.3.2 テストの準備

この項では、Distributed Test の準備方法を説明します。

(1) Web サーバーマシンの準備

(a) JUnit の配備

3 章を参照して、Web サーバーマシンに Web サーバーと JUnit(3.2.2 Web サーバーを利用する場合 参照)をインストールします。テスト対象マシンやテスト指示マシンと異なり、Web サーバーマシンに JDK や Ant は必要ありません。

(b) テストの準備

- ① 4.1 テストの準備を参照して、JUnit テストファイルを用意します。
- ② 4.2.2 Web サーバーを利用する場合を参照して、JUnit テストファイルを Web サーバーに配備します。
- ③ 配備したテストが実行できることを確認します。

(2) テスト対象マシンの準備

テスト対象マシンでは、テスト指示マシンの Distributed Test から実行される Standalone Test の準備を行います。

(a) JUnit Server の配備

3 章を参照して、テスト対象マシン全てに以下をインストールします。

- ・ JUnit
- ・ JDK
- ・ Ant + JUnit

(b) Standalone Test の準備

- ① ブラウザーのプロキシの設定を確認します。ブラウザーでプロキシを利用している場合は、「プロキシを利用せずに直接接続するサイト」にテスト指示マシンの IP アドレスを加えます。
- ② JUnit テストファイルやその他の環境情報を設定するため、JUnit に付属する Ant 用 build.xml ファイル「C:\usr\local\jsunit\build.xml」を編集します。同 xml ファイルを開き、以下の<property>タグを修正します。

● browserFileNames

テスト対象ブラウザのパスを設定します。複数のブラウザを指定する場合は、「,」で区切ります。

(例)

```
<property name="browserFileNames" value="c:\program files\internet explorer\iexplore.exe,c:\program files\Mozilla Firefox\firefox.exe"/>
```

※ (例)は、表示の都合上複数行に分けて記述していますが、build.xml には 1 行で記述します。

● port

テスト対象マシン上の JUnit Server が利用するポート番号を指定します。指定がなければ、

JsUnit Server はデフォルトのポート番号 8080 を利用します。

```
(例)
<property name="port" value="8080"/>
```

● url

ブラウザから JsUnit テストを呼び出す URL を指定します。以下のクエリー文字列も指定します。クエリー文字列同士は「&」で連結します。

- ・ testPage :
JsUnit テストファイルを指定します。Web サーバーを利用する場合、「http://」は付けません(例)。
- ・ autoRun :
テストを自動的に実行するため、「true」を指定します。
- ・ submitresults :
実行結果を受け取る JUnit Server のサブレットです。以下のように指定します。
「localhost:<port> で指定したポート番号>/jsunit/acceptor」

```
(例)
<property name="url" value="http://webserver/testRunner.html
?testPage=localhost/webserver.html&autoRun=true
&submitResults=localhost:8080/jsunit/acceptor"/>
```

※ (例)は、表示の都合上複数行に分けて記述していますが、build.xml には 1 行で記述します。

● logsDirectory

テスト結果を出力するディレクトリーを指定します。

```
(例)
c:\usr\local\jsunit\logs
```

● build.xml ファイルの例

```
<project name="JsUnit" default="create_distribution" basedir=". ">
    <property name="browserFileNames" value="c:\program files\Mozilla Firefox\firefox.exe,c:\program
files\internet explorer\iexplore.exe"/>
    <property name="url"
value="http://webserver/testRunner.html?testPage=webserver/mytestsuite.html&autoRun=true&su
bmitResults=localhost:8080/jsunit/acceptor"/>
    <property name="remoteMachineURLs" value=""/>
    <property name="remoteMachineURLs" value=""/>
    <property name="port" value="8080"/>
    <property name="resourceBase" value=""/>
    <property name="logsDirectory" value="c:\usr\local\jsunit\logs"/>
-- 以下省略 --
```

(c) start_server ターゲットの実行

以下のコマンドで「start_server」ターゲットを実行します。同ターゲットを実行することで、テスト対象マシンの Standalone Test が開始待ち状態になります。

```
ant start_server
```

なお、「start_server」ターゲットで起動した JUnit Server を停止するには、「stop_server」ターゲットを実行します。

```
ant stop_server
```

(3) テスト指示マシンの準備

(a) JsUnit Server の準備

3章を参照して、以下をインストールします。

- ・ JsUnit
- ・ JDK
- ・ Ant + JUnit

(b) build.xml の編集

「C:\usr\local\jsunit\build.xml」を編集し、以下のキーの値を修正します。

● remoteMachineURLs

クライアントと、クライアントで動いている JsUnit Server のポート番号を指定します。複数のリモートホストを指定する場合は、「,」で区切ります。

(例)
http://remotehost1:8080, http://remotehost2:8080

● build.xml ファイルの例

```
<project name="JsUnit" default="create_distribution" basedir=".">
  <property name="browserFileNames" value=""/>
  <property name="url" value=""/>
  <property name="remoteMachineURLs"
value="http://remotehost1:8080,http://remotehost2:8080"/>
  <property name="port" value=""/>
  <property name="resourceBase" value=""/>
  <property name="logsDirectory" value=""/>
  -- 以下省略 --
```

5.3.3 テストの実行

この項では、Distributed Test の実行方法を説明します。

(1) サーバーの実行方法

(a) distributed_test ターゲットの実行

- ① コマンドプロンプトを起動します。
- ② 「C:\usr\local\jsunit」をカレントディレクトリにします。

```
cd C:\usr\local\jsunit
```

- ③ Ant で standalone_test ターゲットを実行します。
同ターゲットを実行することで、テスト対象マシンで開始待ち状態になっていた Standalone Test が開始されます。

```
ant distributed_test
```

(2) テスト結果の参照方法

(a) Ant に表示される結果

全てのテスト対象マシンでテストが成功した場合、コマンドプロンプト上の Ant 実行結果に「BUILD SUCCESSFUL」が表示されます。

Fig. 5-6 Ant 実行結果 (テストが成功した場合)

```
C:\usr\local\jsunit>ant distributed_test
Buildfile: build.xml

distributed_test:

BUILD SUCCESSFUL
Total time: 10 seconds
```

テストが失敗した場合、コマンドプロンプト上の Ant 実行結果に「BUILD FAILED」が表示されます。テストは失敗が発生したマシンで終了します。残りのマシンに対するテストは行われません。

Fig. 5-7 Ant 実行結果 (テストが失敗した場合)

```
C:\usr\local\jsunit>ant distributed_test
Buildfile: build.xml

distributed_test:

BUILD FAILED
C:\usr\local\jsunit\build.xml:79: Test net.jsunit.DistributedTest failed

Total time: 7 seconds
```

(b) ログファイルに出力される結果

テスト対象マシンの `build.xml` ファイルの `logsDirectory` プロパティで指定したディレクトリにテストの実行結果を格納したログファイルが出力されます。参照方法は 5.2.3(2)(b) ログファイルに出力される結果を参照してください。

6

アンインストール

6. アンインストール

この章では、WindowsXP SP2におけるJsUnit とその他のアプリケーションのアンインストール方法について説明します。

6.1	JsUnitのアンインストール	6-2
6.2	Apacheのアンインストール	6-3
6.3	Antのアンインストール	6-4
6.4	JDKのアンインストール	6-5

6.1 JsUnit のアンインストール

この節では、JsUnit のアンインストール方法を説明します。

6.1.1 アンインストールの手順

- ① 「C:\usr\local\jsunit」を削除します。
- ② Web サーバー(Apache の場合は「C:\Program Files\Apache Group\Apache2\htdocs」)に配置した以下のファイルを削除します。
 - ・ testRunner.html
 - ・ app
 - ・ css
 - ・ images

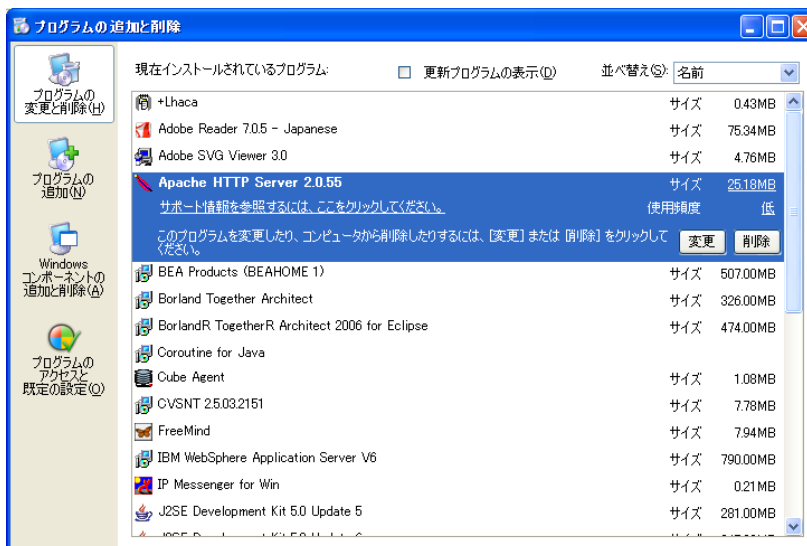
6.2 Apache のアンインストール

この節では、Apache のアンインストール方法を説明します。

6.2.1 アンインストールの手順

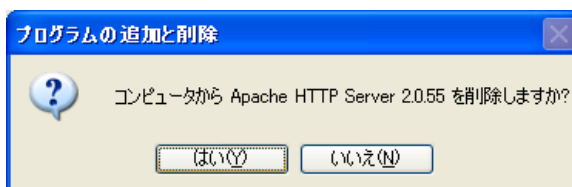
- ① Windows の[プログラムの追加と削除]画面を開きます。

Fig. 6-1 プログラムの追加と削除 一覧



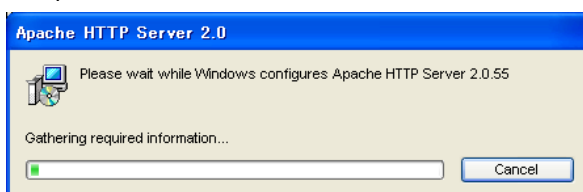
- ② [Apache HTTP Server 2.0.55]を選択し、[削除]ボタンをクリックします。

Fig. 6-2 プログラムの追加と削除 確認



- ③ [はい]ボタンをクリックします。

Fig. 6-3 Apache HTTP Server 削除中



6.3 Ant のアンインストール

この節では、Ant のアンインストール方法を説明します。

6.3.1 アンインストールの手順

- ① 以下の環境変数を修正します。

Tab. 6-1 修正する環境変数

#	環境変数名	値	備考
1	ANT_HOME	C:\usr\local\apache-ant-1.6.5	環境変数を削除する。
2	Path	%ANT_HOME%\bin	既存の環境変数の値を編集し、「%ANT_HOME%\bin」のみ削除する。

- ② 「C:\usr\local\apache-ant-1.6.5」を削除します。

6.4 JDK のアンインストール

この節では、JDK のアンインストール方法を説明します。

6.4.1 アンインストールの手順

- ① 以下の環境変数を修正します。

Tab. 6-2 修正する環境変数

#	環境変数名	値	備考
1	JAVA_HOME	C:\¥j2sdk1.4.2_09	環境変数を削除する。
2	Path	%JAVA_HOME%\¥bin	既存の環境変数の値を編集し、「%JAVA_HOME%\¥bin」のみ削除する。

- ② Windows の[プログラムの追加と削除]画面を開きます。

Fig. 6-4 プログラムの追加と削除 一覧



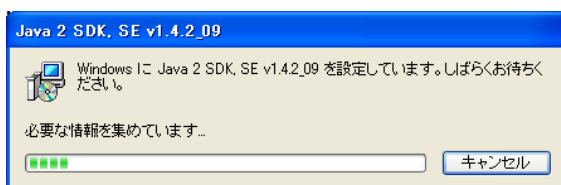
- ③ [Java 2 SDK, SE v1.4.2_09]を選択し、[削除]ボタンをクリックします。

Fig. 6-5 プログラムの追加と削除 確認



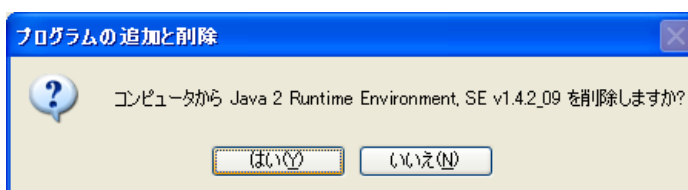
- ④ [はい]ボタンをクリックします。

Fig. 6-6 Java 2 SDK 削除中



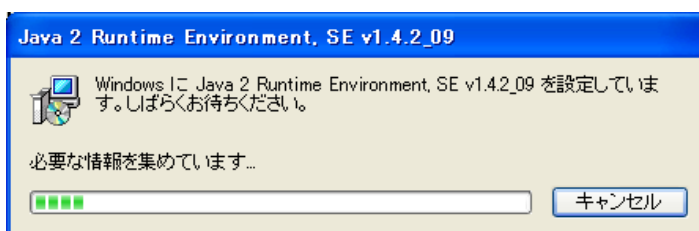
- ⑤ 再度[プログラムの追加と削除]画面で[Java 2 S Runtime Environment, SE v1.4.2_09]を選択し、[削除]ボタンをクリックします。

Fig. 6-7 プログラムの追加と削除 確認



- ⑥ [はい]ボタンをクリックします。

Fig. 6-8 Java 2 Runtime Environment 削除中



付録A

WTP1.0 による JavaScript の編集

A. WTP1.0 による JavaScript の編集

この章では、WTP1.0 で JavaScript を開発する方法について説明します。

A.1 WTP	A-2
A.2 WTP1.0 によるJavaScriptの編集	A-3

A.1 WTP

WTP(Eclipse Web Tools Platform)とは、J2EE Web アプリケーション作成を支援する Eclipse プラグインです。Eclipse とは、多機能かつ拡張可能なオープンソースの統合開発環境です。Eclipse に WTP をプラグインすると、アプリケーションの開発、デプロイ、テスト、デバッグといった一連の開発サイクルを Eclipse 上で行えます。また WTP は JavaScript の色分け表示やコードアシスト機能を備えたエディターも提供しています。2006 年 2 月の最新版は Eclipse3.1、WTP1.0 です。

この章では、WTP を利用した JavaScript の編集方法について説明します。Eclipse、WTP の詳細やインストール方法については「Eclipse3.1 利用ガイド」を参照してください。

A.2 WTP1.0 による JavaScript の編集

この節では、WTP で JavaScript を編集する方法について説明します。
準備として、WTP をプラグインした Eclipse で動的 Web プロジェクト(Dynamic Web Project)「JavaScriptSample」を作成する必要があります。動的 Web プロジェクトの作成方法については、「Eclipse3.1 利用ガイド」を参照してください。

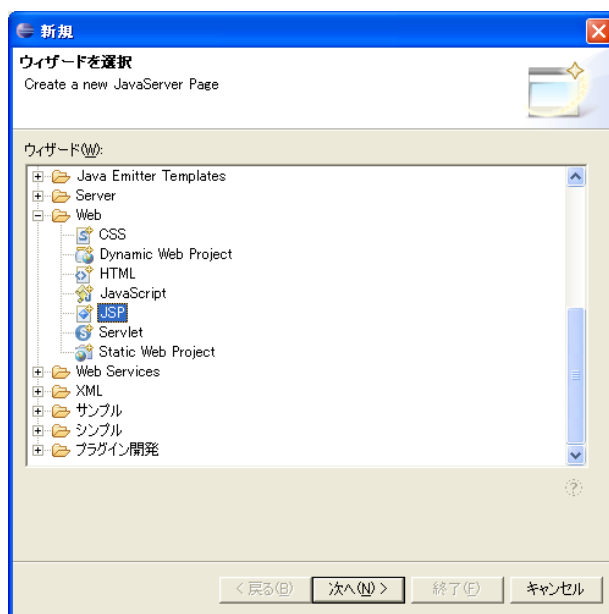
A.2.1 JSP ファイルにおける JavaScript の編集

JSP ファイルを作成し、JavaScript を開発する方法を説明します。

(1) JSP ファイルの作成

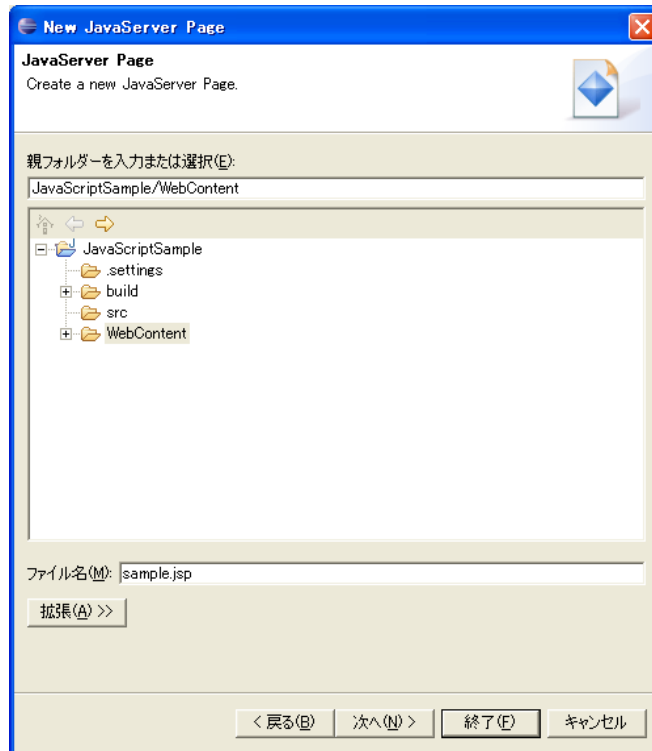
- ① Eclipse のメニューから[ファイル]-[新規]-[その他]を選択します。

Fig. A-1 新規 : ウィザードを選択



- ② [Web]-[JSP]を選択し、[次へ]ボタンをクリックします。

Fig. A-2 New JavaServer Page



- ③ 以下のように入力し、[終了]ボタンをクリックします。

Tab. A-1 New JavaServer Page 入力項目

#	項目	内容	例
1	親フォルダーを入力または選択	ファイルを作成するフォルダー	JavaScriptSample/WebContent
2	ファイル名	作成するファイルの名前	sample.jsp

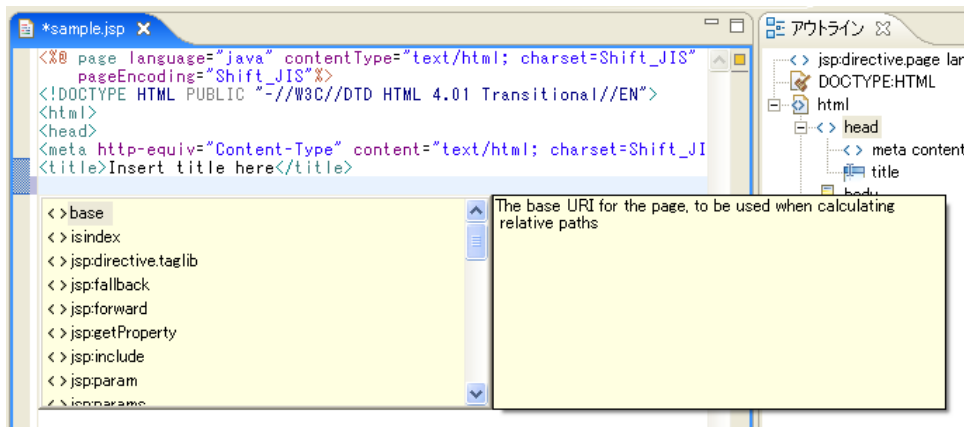
以上で JSP ファイルを作成することができます。

ヒント) HTML ファイルを作成する場合は、Fig. A-1 で[Web]-[HTML]を選択します。

(2) JavaScript の編集 – コードアシストの利用

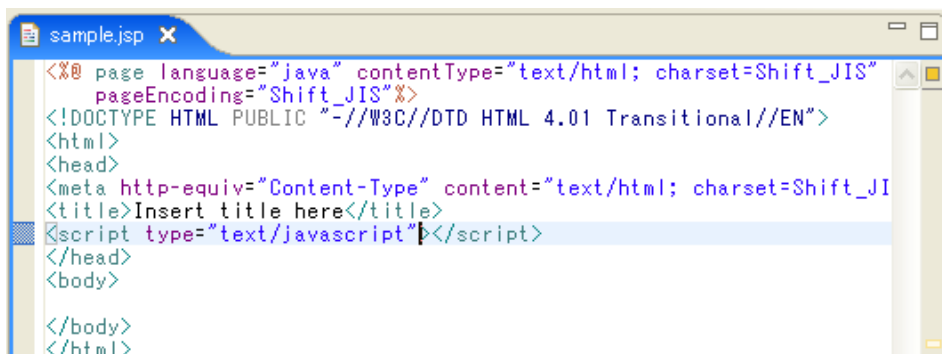
作成したファイル(sample.jsp)の</title>タグと</head>の間に一行挿入し、[Ctrl]+[Space]キーを押下します。以下のように、選択候補が表示されます。

Fig. A-3 タグの選択候補



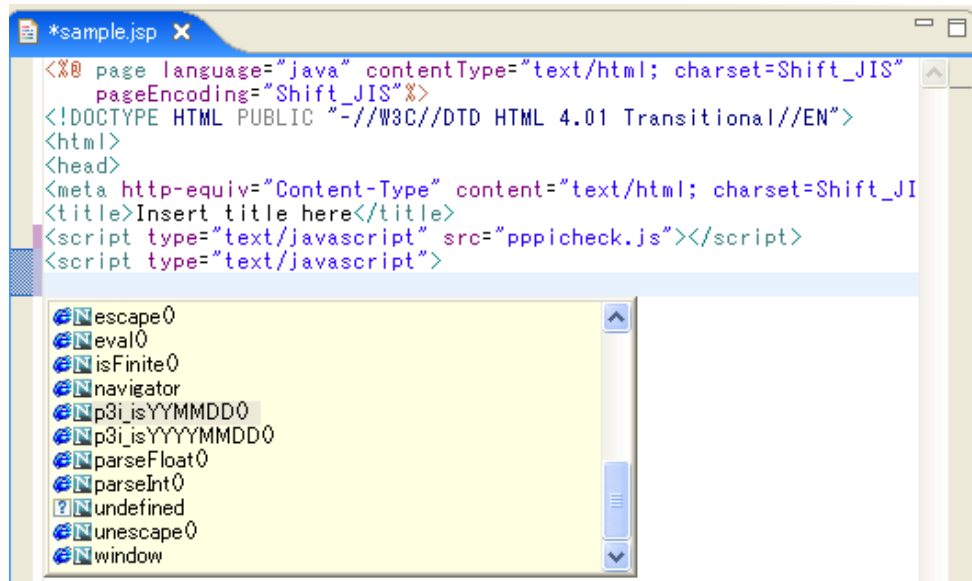
[s]キーを押下すると、先頭アルファベットが「s」である選択候補が絞り込まれます。「<script>」を選択し、[Enter]キーを押下します。

Fig. A-4 <script>タグを追加した状態



次に<script>タグの間に JavaScript を記述します。JavaScript の記述でも、[Ctrl]+[Shift]キーを押下することで選択候補を表示することができます。また<script>タグを利用して外部スクリプトファイルをインクルードした場合、外部スクリプトファイルに含まれる関数も選択候補に表示されます。

Fig. A-5 4章で使用した pppcheck.js をインクルードした場合の選択候補



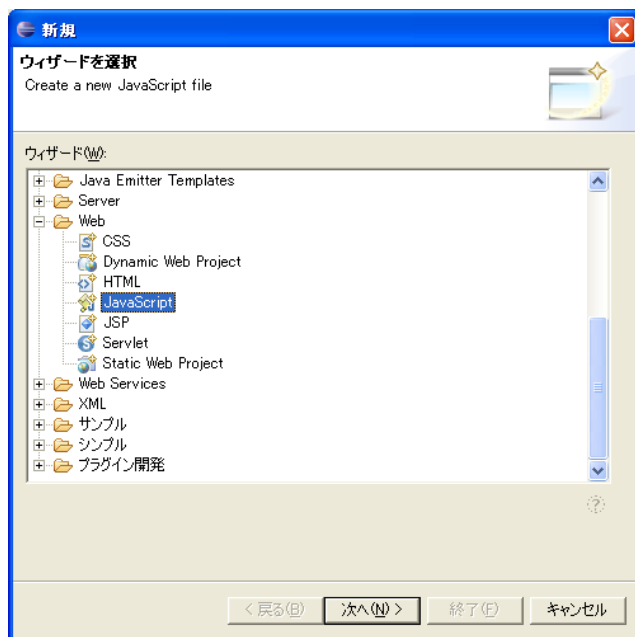
A.2.2 外部スクリプトファイルにおける JavaScript の編集

外部スクリプトファイルを作成し、JavaScript を開発する方法を説明します。

(1) 外部スクリプトファイル(js ファイル)の作成

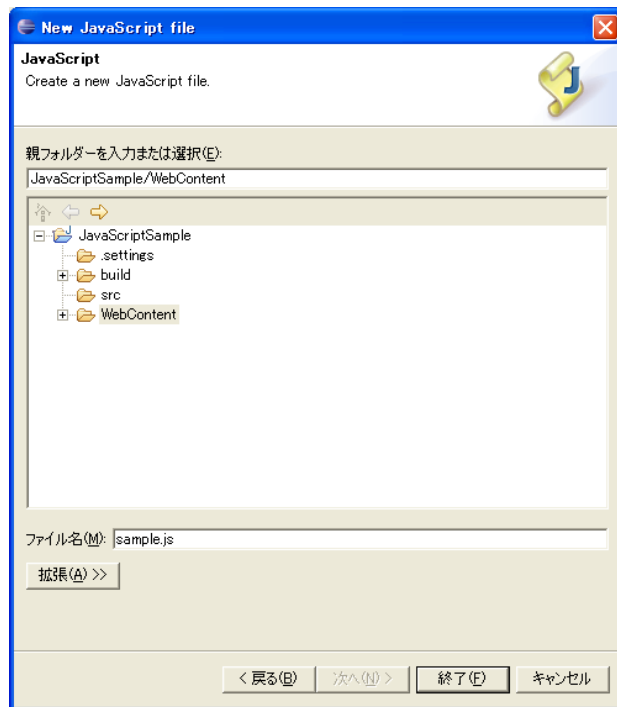
- ① Eclipse のメニューから[ファイル]-[新規]-[その他]を選択します。

Fig. A-6 新規 : ウィザードを選択



- ② [Web]-[JavaScript]を選択し、[次へ]ボタンをクリックします。

Fig. A-7 New JavaScript File



③ 以下のように入力し、[終了]ボタンをクリックします。

Tab. A-2 New JavaScript File 入力項目

#	項目	内容	例
1	親フォルダーを入力または選択	ファイルを作成するフォルダー	JavaScriptSample/WebContent
2	ファイル名	作成するファイルの名前	sample.js

以上で外部スクリプトファイルを作成することができます。

(2) JavaScript の編集 – コードアシストの利用

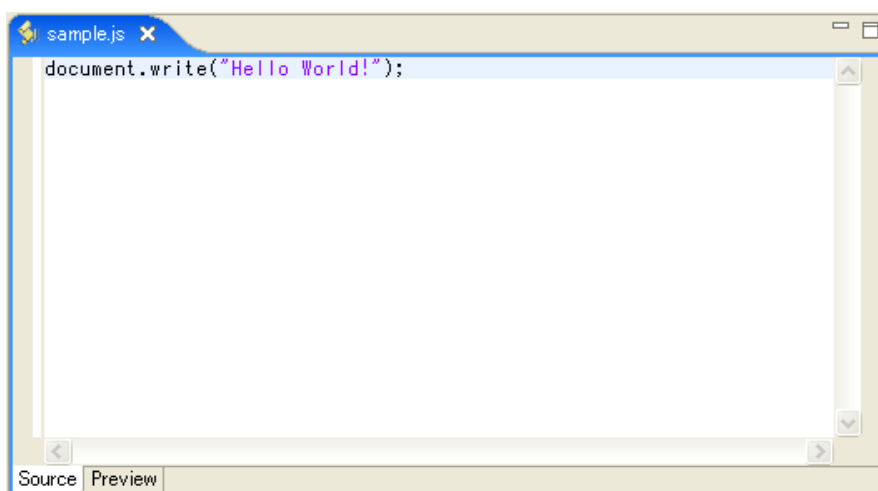
JavaScript でも、JavaScript エディターの [Source] タブ上で [Ctrl]+[Shift] キーを押下することで、JSP ファイルと同様のコードアシストを利用することができます。

(3) JavaScript の編集 – プレビューの利用

JavaScript エディターでは、[Source] タブで記述した JavaScript を [Preview] タブで実行することができます。JavaScript をプレビューするためにブラウザの更新ボタンを使用する必要がなく、効率の高い開発を行うことができます。以下に方法を示します。

- ① [Source] タブで JavaScript を記述します。

Fig. A-8 [Source] タブ



- ② [Preview] タブに切り替えます。[Source] タブで記述した JavaScript が実行されます。JavaScript に不具合があり実行時にエラーが発生した場合、エラーが発生する直前までの実行結果が表示されます。エラーについては警告されません。

Fig. A-9 [Preview] タブ

