Tadikarawo:手力雄

Akio MIWA
http://triring.net/

2009 年 6 月 30 日 火曜日

概 要

これは，Tadikarawo という開発中の個々の作業時刻を記録するためのツールのマニュアルです.

図 1: tadikarawo タイトル

第1章　はじめに

Tadikarawo は，開発中の個々の作業時刻を記録するためのツールです．

nttdocomo の i アプリの開発を支援するソフトウェアである i α ppli Development Kit にヒントを得て作成しました．

ご自身の作業記録を解析する LifeHack ツールとして活用してください．

1.1　開発動機

このアプリケーションを作成した目的は以下の 2 つです．

- 作業時間を記録し，アプリケーションが完成するまでにどれだけの時間が費やされたか?1 日当りどれくらい make を行っているのか?プログラミングの過程に好調，不調はあるのか？などを調べてみたかった．——＞プログラム作業の見える化
- cygwin の環境では，大量の出力があるとコンソールからコンパイル結果や実行結果を参照して，必要部分を切り出すことがやりにくかった．これを簡単に，カット＆ペーストできるようにしたかった．

1.2　特徴

基本的な機能は，以下の 3 つです．

1. 以下のイベント時刻をログファイルに記録します．
   - このアプリケーションの起動時刻，終了時刻
   - それぞれのコマンドの実行した時刻
   - ファイルを開いた時刻
   - コメントを書き込んだ時刻
2. コマンドリストに登録された内容をコマンドボタンに割付けます．これにより，それぞれのボタンのクリックだけで，実行が可能になります．
3. 同じ作業ディレクトリ内のファイルを日常使用しているエディタで開くことができます．

第2章　実行環境，インストール，設定，実行

2.1　実行環境

Java で作成したので、基本的には、Java 1.4 以上の JVM さえあれば、どのようなプラットホームでも動くはずです。

実行環境がない方は、以下のサイトより、JavaTM 2 Runtime Environment (JRE) を入手してインストールしてください。

http://java.sun.com/getjava/ja/[1]

2.2　インストール

以下のファイルをダウンロードしてください．必要に応じて，ドキュメントもダウンロードしてください．

- 2009-06-30 - Tadikarawo v0.80[2](236K)
- 2009-06-30 - manual v0.80[3](72K)

入手したパッケージファイルを展開し，以下の 3 つのファイルを取り出してください．

1. 実行ファイル Tadikarawo.jar
2. 設定ファイル tadikarawo.ini
3. コマンドファイル sample.cmd

この 3 つのファイルを作業するディレクトリにコピーしてください．

[1] http://java.sun.com/getjava/ja/
[2] Tadikarawo080.zip
[3] manual080.pdf

2.3 ini ファイルの設定

次に，tadikarawo.ini を開いて，以下の 2 項目を設定してください．

- 作業ログを保存するファイル名を設定する
- 編集に使用するエディタの実行ファイルを設定する

予め，主要なエディタの実行ファイルの保存場所が書かれています．お好みのエディタを選び，コメントをはずしてください．デフォルトはメモ帳に設定してあります．

ログファイルは，デフォルトで，同じディレクトリ内の worklog.csv に保存されるようになっています．

```
#Sun May 24 14:13:15 JST 2009
Logfile=worklog.csv
Editor=C:\\WINDOWS\\notepad.exe
#Editor=C:\\Program Files\\sakura\\sakura.exe
#Editor=C:\\Program Files\\TeraPad\\TeraPad.exe
#Editor=C:\\Program Files\\WZ Editor\\WZEDITOR.EXE
```

図 2.1: tadikarawo.ini の記述例

注意：ディレクトリの区切りとして'\' が 2 つずつ書かれています．これは，間違いではありません．これは仕様です．必ず、'\\' と記述してください。

2.4 コマンドファイルの設定

sample.cmd ファイルを開き，以下のように使用したいコマンドを登録してください．

- 先頭に'#' が書かれている行は，コメントとして扱われます．
- 1 行にコマンド，ラベル，コメントの順に，それぞれを':' で区切って記述してください．
- このデータを元に，コマンドボタンが生成されフォームに配置されます．
- コマンドは，それぞれのコマンドボタンに登録され，そのボタンがクリックされた時点実行されます．この時の時刻がログに記録されます．
- ラベルは，コマンドボタンのテキストとして登録されます．

- コメントは，ToolTip として登録されます．したコマンドは，コマンドボタンとしてフォームに配置されます．
- 日本語のコメントは，不具合が発生する場合があります．不具合が発生した場合は，その行を削除してください．

リスト 1: sample.cmd

```
# Prime Makefile test
make clean:クリア:中間ファイル等を削除します.
make all:コンパイル:
make run:実行:出来上がったオブジェクトを実行します.
make backup:圧縮:パッケージを作成のため，圧縮ファイルを作成します.
```

2.5 起動方法

パッケージ内のサンプルとして入れてある Makefile[リスト 3] と sample.cmd[リスト 1] と Primes.java[リスト 2] を同じディレクトリ内に置いて，実行してみます．

リスト 2: Primes.java 素数を求めるプログラム

```
/**
 *  100 までの素数を求める
 */

public class Primes {
    static int[] generatePrimes(int n) {  // n 個の素数を生成
        int k = 0,  prime[] = new int[n];
        prime[k++] = 2;
        for (int x = 3; k < n; x += 2) {
            int i = 0;
            while (i < k && x % prime[i] != 0) i++;
            if (i == k) prime[k++] = x;
        }
        return prime;
    }

    public static void main(String[] args) {
        final int N = 25;
        int[] prime = generatePrimes(N);
        System.out.print("100 までの素数");
        for (int i = 0; i < N; i++) {
            System.out.print(prime[i] + " ");
        }
        System.out.println();
    }
}
```

リスト 3: Makefile

```
# Prime test
# テスト
JC = javac

clean:
        rm *.class
all:
        $(JC) Primes.java
run:
        java Primes
backup:
        jar cvf primes.zip Makefile Primes.*
        jar  -i primes.zip
```

起動は，cygwin のコンソールから，以下のコマンドで起動してください.

```
java -jar Tadikarawo.jar -c sample.cmd
```

図 2.2: tadikarawo.jar の起動

図 2.3 のような起動画面が表示されます.

2.6 実行

Alt+1,Alt+2,Alt+3,Alt+4 の順に、キー操作をするか、コマンドボタンを左から順に，クリックしてください.

図 2.5 のように、実行内容とその結果やエラーがコンソールに出力されます.

表 2.1: サンプルの sample.cmd に登録されたコマンド

	コマンド	ラベル	コメント
1	make clean	クリア	中間ファイル等を削除します.
2	make all	コンパイル	
3	make run	実行	出来上がったオブジェクトを実行します.
4	make backup	圧縮	パッケージを作成のため，圧縮ファイルを作成します.

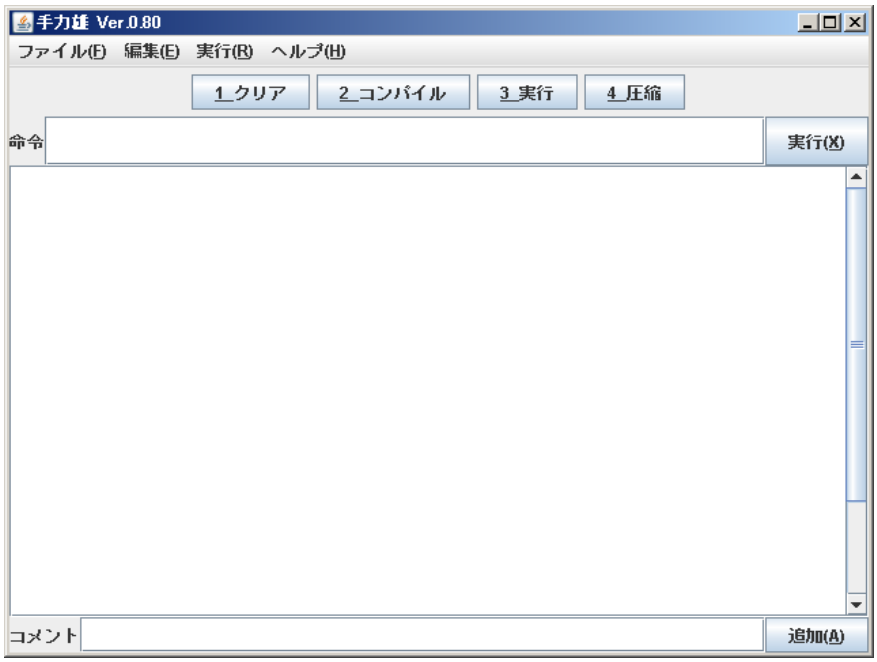

図 2.3: tadikarawo 起動画面

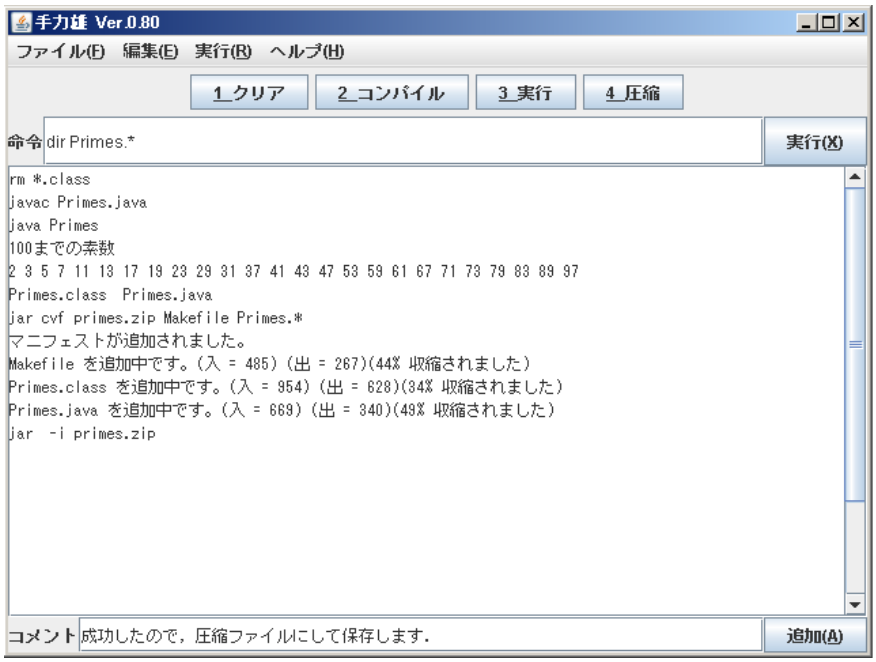

図 2.4: tadikarawo 実行画面

```
rm *.class
javac Primes.java
java Primes
100 までの素数
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 \
    89 97
jar cvf primes.zip Makefile Primes.*
マニフェストが追加されました。
Makefile を追加中です。(入 = 163) (出 = 123)(24% 収縮されました)
Primes.class を追加中です。(入 = 1076) (出 = 688)(36% 収縮されました)
Primes.java を追加中です。(入 = 756) (出 = 369)(51% 収縮されました)
jar  -i primes.zip
```

図 2.5: 実行の様子

この時の作業内容が，図 2.6 のようにタイムスタンプと共に記録されていきます。

コメント機能については，後述します.

2.7 メニューとメニューアイテムの解説

メニューとメニューアイテムの機能について説明します。

- ファイル (F)
  - 開く (O) ソースファイル等を登録されたエディタで開きます。
  - 終了 (X) このアプリケーションを終了します。
- 編集 (E)
  - コピー (C) コンソールの表示の選択された部分をクリップボードにコピーします。
  - クリアコンソールの表示を削除します。
  - プロパティー (P) 設定ファイルの編集に使用します。(現在、準備中)
- Make(M)

```
# logging start 2009-06-27T23:22:27+09:00,Sat
2009-06-27T23:22:27+09:00,Sat,comment, これより, 作業を開始します.

2009-06-27T23:22:48+09:00,Sat,comment, まず, 最初に古い実行ファイルを削除します.
2009-06-27T23:23:05+09:00,Sat,command,make \
    clean, クリア, 中間ファイル等を削除します.
2009-06-27T23:23:26+09:00,Sat,comment, 次に, ソースファイルを開きます.
2009-06-27T23:23:39+09:00,Sat,open \
    ,D:\nbdir\Tadikarawo\Primes.java
2009-06-27T23:24:05+09:00,Sat,comment, コンパイルをします.
2009-06-27T23:24:19+09:00,Sat,command,make all, コンパイル,
2009-06-27T23:24:46+09:00,Sat,comment, 実行します.
2009-06-27T23:24:58+09:00,Sat,command,make \
    run, 実行, 出来上がったオブジェクトを実行します.
2009-06-27T23:25:31+09:00,Sat,comment, コマンドラインから, dir を実行してみます.
2009-06-27T23:25:51+09:00,Sat,execute,dir Primes.*
2009-06-27T23:26:15+09:00,Sat,comment, 成功したので, 圧縮ファイルにして保存します.
2009-06-27T23:26:30+09:00,Sat,command,make \
    backup, 圧縮, パッケージを作成のため, 圧縮ファイルを作成します.
2009-06-27T23:27:37+09:00,Sat,comment, これで, 終了します.
2009-06-27T23:28:07+09:00,Sat,logout ,
```

図 2.6: 記録された worklog.csv の内容

- sample.cmd 内に書かれたコマンドがメニューアイテムとして登録されています。最初のコマンドから順に，1 から 9 までのキーボードニーモニックが設定されます。

  Alt+1，Alt+2，Alt+3，Alt+4 ..... Alt+9 で，それぞれのラベルの内容を実行できます。

- ヘルプ (H)
  - ヘルプ (C) ヘルプを表示します。(現在、準備中)
  - このアプリケーションについて (A)　バージョン等を表示します。

2.8　ボタンパネルの解説

ボタンパネルには，sample.cmd 内に書かれたコマンドがボタンとして登録されています。最初のコマンドから順に，1 から 9 までのキーボードニーモニックが設定されます。

Alt+1，Alt+2，Alt+3，Alt+4 ..... Alt+9 で，それぞれのラベルの内容を実行できます。

第3章　作業記録機能について

3.1　基本操作の記録

以下の作業時刻が，自動的に log ファイルに記録されます.

- 起動と終了の時刻
- 登録されたエディタでファイルを開いた時の時刻とそのファイル名
- make を実行した時の時刻とそのラベル名

3.2　コマンドの直接実行機能について

ボタンのリストの下にコマンド実行欄があります。ここに、コマンドを書き込み、実行ボタンを押してください。shell でコマンドを実行した時と同じように動作し，その実行時間が作業ログに書き込まれます。コマンドリストに登録していなかったコマンドを実行する時に利用してください.

3.3　コメント入力機能について

最下段にコメント欄があります。ここに、コメントを書き込み、追加ボタンを押してください。作業ログに、記録時間とともに、そのコメントが書き込まれます。以下のような内容を自由に書いてください。

- 変数名の変更
- バグの発生
- デバッグ内容
- プログラムが動いた。プログラムが動かなくなった。
- 誤って、ファイルを消してしまった
- 独り言 (お腹がすいた。早く帰りたいよーー)
- その他気がついたこと

これらは、自分自身を振り返るための記録となります。未来の自分へのメッセージのつもりで書いてください。

第4章　今後の開発

4.1　todo

- 記録したログファイルの解析ツールの開発
- ドキュメントの充実
- 実際の開発に利用してそのログの収集を行う.

4.2　ユーザの方へ

ソースリストはすべて公開しておりますので，必要に応じて，拡張および改良をしてください.

また，力量のある方は，解析ツールの開発をお願いします．データは，単純な CSV ファイルなので，さほど難しくないと思います.

4.3　おまけ

配布パッケージ内にある以下のファイルは、サンプルとして作成した cmd ファイルです。

表 4.1: サンプル cmd ファイル

ファイル名	用途
etrobocon.cmd	ET ロボコン用コマンドセット
mozilla.cmd	アプリケーションランチャーの例

戻る[1]

[1] ../index.html