RELEASE NÃO OFICIAL

```
<?php
    define ('NET_using', 0);
    include 'cleeasy.inc.php';
    cleeasy::cores ('vermelho');
    cleeasy::updown (10, 10,
    'cleeasy:cli programing easy!'."\n", 1);
?>
```

O QUE É?

cleeasy é uma classe que disponibiliza funções para animação de strings e para facilitar o suporte a rede de aplicativos já prontos, que funcione via linha de comandos.

Transformar um aplicativo já pronto sem suporte a rede, em um com suporte a rede pode ser uma tarefa bastante simples, porque apenas algumas funções são substituídas. Vejamos exemplo bastante simples, da transformação de um aplicativo:

1 - Aplicação sem utilizar a classe cleeasy:
primeiro.php
```
<?php
    echo "Ola mundo\n\r";
    exit(0);
?>
```

2 - Aplicação utilizando a classe cleeasy, sem suporte a rede
segundo.php
```
<?php
    define ('NET_using', 0);
    cleeasy::echo_ ("Ola mundo\n\r");
    cleeasy::sair ();
?>
```

3 - Aplicação utilizando a classe cleeasy, com suporte a rede
terceiro.php
```
<?php
    cleeasy::echo_ ("Ola mundo\n\r");
    cleeasy::sair ();
?>
```

Como podemos ver, a mudança é bastante simples. Para executarmos o **primeiro.php**, na linha de comandos executamos:

```
$php primeiro.php
Ola mundo
$
```

Para executarmos o **segundo.php**, na linha de comandos executamos:

```
$php segundo.php
Ola mundo
$
```

Para executarmos o **terceiro.php**, na linha de comandos executamos:

```
$php caller.php
```

E após isso, acessamos, via telnet, o IP configurado na PORTA especificada, no arquivo **cleeasy.inc.php**, nas seguintes linhas:

```
define ('NET_ip', '127.0.0.1');    // endereço ip que receberá conexões
define ('NET_porta', '10001');     // porta que ficará o aplicativo
```

Veremos nas próximas linhas uma explicação disso tudo.

COMO FUNCIONA

A **figura 01** tenta explicar o funcionamento do cleeasy:

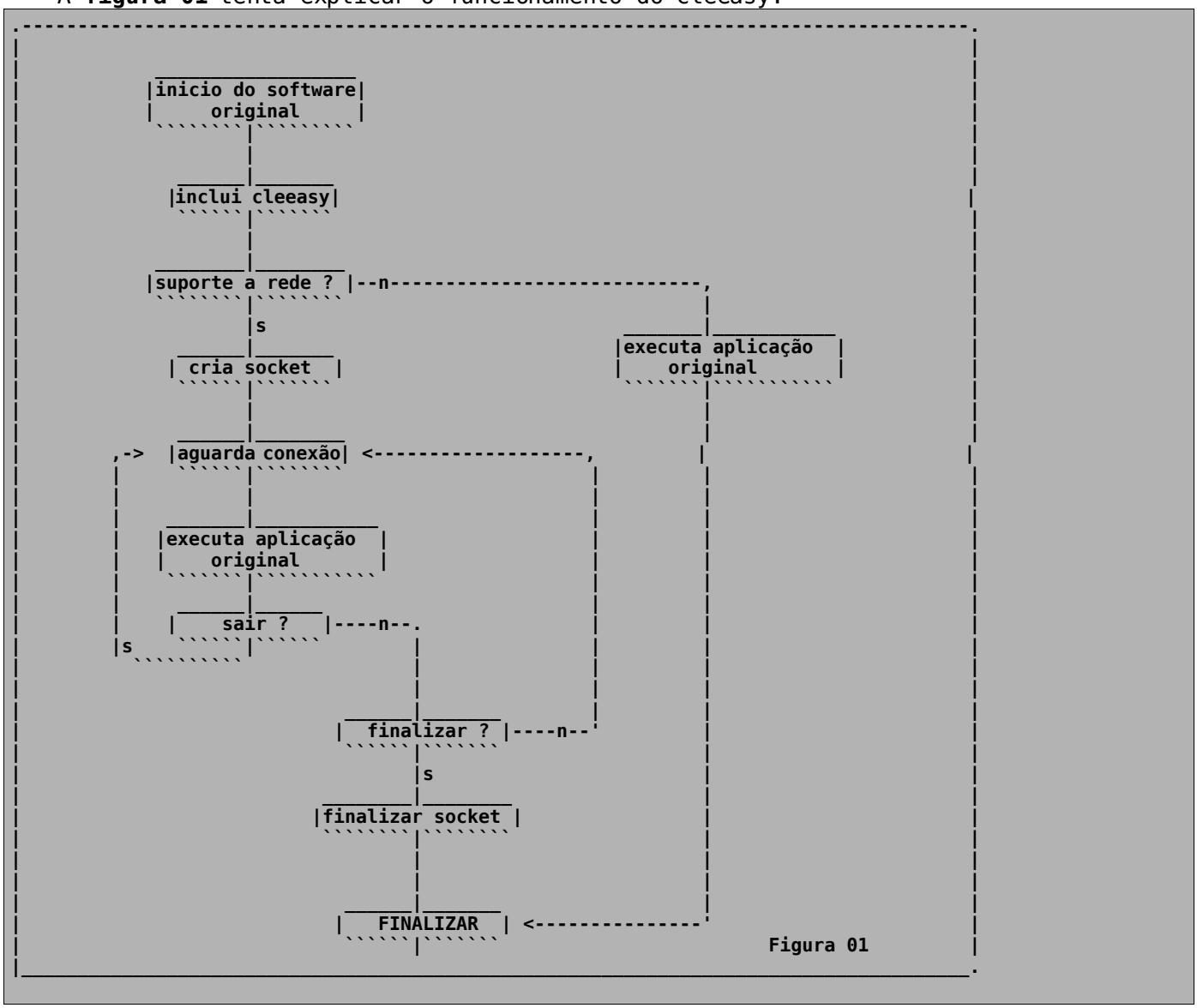

Figura 01

O Funcionamento é bastante simples, vamos a implementação de um aplicativo então, para um melhor entendimento de todo o processo.

PRIMEIRO EXEMPLO DE USO

Vamos criar um aplicativo que exiba a data e a hora, e após isso seja finalizado.

data_hora.php

```
<?php
      echo date("D M j H:i:s Y\r\n");
      exit(0);
?>
```

```
      Executando ele, temos:
      $ php data_hora.php
      Tue Aug 10 17:14:39 2004
      $
```

Agora, podemos fazer um aplicativo que utilize a classe cleeasy, mas que somente exiba as mesmas informações que foram exibidas pelo arquivo **data_hora.php**, ainda sem suporte a rede:

data_hora_cleeasy.php

```
<?php
      define ('NET_using', 0);
      include 'cleeasy.inc.php';

      cleeasy::echo_ (date("D M j H:i:s Y\r\n"));
      cleeasy::sair ();
?>
```

Executando ele, temos:

```
      $ php data_hora_cleeasy.php
      Tue Aug 10 17:21:32 2004
      $
```

Agora vamos a parte interessante: Transformar o arquivo ***data_hora_cleeasy.php*** em um servidor de data. Primeiro reescrevemos o arquivo ***data_hora_cleeasy.php***:

data_hora_net.php

```
<?php
      cleeasy::echo_ (date("D M j H:i:s Y\r\n"));
      cleeasy::sair ();
?>
```

Note que agora o arquivo **cleeasy.inc.php** não foi incluído. Isto é feito posteriormente, por um outro script.

Após reescrevermos o arquivo, devemos configurar algumas informações no arquivo **cleeasy.inc.php**:

```
define ('NET_ip', '127.0.0.1');
Endereço IP que o servidor vai funcionar;

define ('NET_porta', '10003');
Porta que o servidor vai escutar por conexões;

define ('NET_timeout', 10);
Tempo antes de o servidor ser finalizado, por falta de alguém conectar-se;

define ('NET_aplic', 'data_hora_net.php');
Path para o aplicativo que receberá o suporte a rede;

define ('NET_aplic_nome', 'DataServer');
Nome do aplicativo que receberá o suporte a rede;
```

Agora, para executarmos o arquivo **data_hora_net.php**, devemos chamar o arquivo **caller.php,** que inclui a classe cleeasy, lê as configurações e após isso executa o aplicativo original:

```
$ php caller.php
,----------------------------------------
| Servidor inicializado...
|
| Porta           : 10003
| Endereco ip     : 127.0.0.1
| Aplicativo      : DataServer
| Status          : Aguardando conexao
`----------------------------------------
```

O aplicativo vai então ficar aguardando por uma conexão. Para testar-mos, devemos acessar o endereço **127.0.0.1** na porta **10003**:

```
$ telnet 127.0.0.1 10003
,----------------------------------------
| Recebendo conexao.
|
| Endereco remoto : 127.0.0.1
| Porta remota    : 32781
| Status          : Conectado!
`----------------------------------------
 Iniciando DataServer
```

Ao conectar-se ao servidor, uma mensagem informando que a conexão ocorreu é exibida, tanto no servidor quanto no cliente. Após alguns segundos, a execução continua:

```
Tue Aug 10 17:44:22 2004
Connection closed by foreign host.
$
```

A data é exibida e a conexão é encerrada. Toda informação enviada ao cliente, é exibida também no servidor.

Após a desconexão do cliente, o servidor já estará pronto para uma nova conexão. Caso o objetivo fosse criar um servidor que, após uma conexão fosse finalizado, poderíamos efetuar as seguintes mudanças no arquivo**data_hora_net.php**:

```
data_hora_net2.php
<?php
      cleeasy::echo_ (date("D M j H:i:s Y\r\n"));
      cleeasy::fim ();
?>
```

Desta forma, o servidor seria one-shot-server, ou seja, é inicializado, aguarda por uma conexão e, após esta ocorrer, o aplicativo é executado e o servidor é finalizado (verifique na figura 1).

SEGUNDO EXEMPLO DE USO

Vamos criar agora um aplicativo que pergunte qual o nome de quem está se conectando e exiba-o ao contrário:

```
seu_nome.php
<?php
    echo "Qual o seu nome ? ";
    $ask = fgets (STDIN, 1024);
    echo strrev ($ask)."\n\r";
   exit(0);
?>
```

Executando, temos:

```
$ php seu_nome.php
Qual o seu nome ? ano_nimous

suomin_ona
$
```

O que o script faz é ler 1024 bytes de STDIN (Um stream já aberto para o stdin. Isto economiza ter de abri-lo com **$stdin = fopen('php://stdin', 'r');**) e exibir a string reversa.

Vamos transformar este aplicativo em um servidor, com as seguintes modificações:

```
seu_nome_net.php
<?php
    cleeasy::echo_ ("Qual o seu nome ? ");
    $ask = cleeasy::fgets_ (1024);
    cleeasy::echo_ (strrev ($ask)."\n\r");
    cleeasy::sair();
?>
```

Após reescrevermos o arquivo, devemos configurar algumas informações no arquivo **cleeasy.inc.php**:

```
define ('NET_ip', '127.0.0.1');
Endereço IP que o servidor vai funcionar;

define ('NET_porta', '10003');
Porta que o servidor vai escutar por conexões;

define ('NET_timeout', 10);
Tempo antes de o servidor ser finalizado, por falta de alguém conectar-se;


define ('NET_aplic', 'seu_nome_net.php');
Path para o aplicativo que receberá o suporte a rede;

define ('NET_aplic_nome', 'Seu Nome ???');
Nome do aplicativo que receberá o suporte a rede;
```

Agora, para executarmos o arquivo **seu_nome_net.php**, devemos chamar o arquivo **caller.php,** como exposto anteriormente:

```
$php caller.php
,----------------------------------------
| Servidor inicializado...
|
| Porta           : 10003
| Endereco ip     : 127.0.0.1
| Aplicativo      : Seu nome??
| Status          : Aguardando conexao
`----------------------------------------
```

O aplicativo vai então ficar aguardando por uma conexão. Para testar-mos, devemos acessar o endereço **127.0.0.1** na porta **10003**:

```
$ telnet 127.0.0.1 10003
,----------------------------------------
| Recebendo conexao.
|
| Endereco remoto : 127.0.0.1
| Porta remota    : 32781
| Status          : Conectado!
`----------------------------------------
 Iniciando Seu nome??
```

Ao conectar-se ao servidor, uma mensagem informando que a conexão ocorreu é exibida, tanto no servidor quanto no cliente. Após alguns segundos, a execução continua:

```
Qual o seu nome ? ano_nimous
suomin_ona
Connection closed by foreign host.
$
```

Nota-se a facilidade de transformar aplicativos que só rodem localmente em servidores.

FUNÇÕES DISPONIBILIZADAS PELAS CLASSE CLEEASY

Aqui temos uma descrição de todas as funções disponibilizadas pelas classe cleeasy.

Primeiramente, vejamos as funções de exibição de strings animadas, manipulação de cores e cursor:

cleeasy::cores ($cor)

Define a cor do texto como **$cor**. Caso não exista a cor **$cor**,será definida como preto. Exemplo:

```
<?php
      include 'cleeasy.inc.php';          // inclui classe
      define ('NET_using', 0);            // retira suporte a rede
      cleeasy::cores ('vermelho');        // define a cor como vermelho
      cleeasy::echoxy (3, 5, 'Ola mundo');// exibe a string 'Ola mundo' na linha
                                          // 3, iniciando pela coluna 5
      cleeasy::sair ();                   // finaliza aplicativo
?>
```

cleeasy::echoxy ($linha, $coluna, $string)

Exibe string na posição **$linha**, **$coluna**. Exemplo:

```
<?php
      include 'cleeasy.inc.php';          // inclui classe
      define ('NET_using', 0);            // retira suporte a rede
      cleeasy::cores ('vermelho');        // define a cor como vermelho
      cleeasy::echoxy (3, 5, 'Ola mundo');// exibe a string 'Ola mundo' na linha
                                          // 3, iniciando pela coluna 5
      cleeasy::sair ();                   // finaliza aplicativo
?>
```

cleeasy::echo_ ($string)

Exibe string na posição atual do cursor. Exemplo:

```
<?php
      include 'cleeasy.inc.php';          // inclui classe
      define ('NET_using', 0);            // retira suporte a rede
      cleeasy::goto (3, 5);               // coloca cursor na linha 3 coluna 5
      cleeasy::echo_ ('Ola mundo');       // exibe a string 'Ola mundo' ( como
                                          // cursor foi colocado na linha 3,
                                          // coluna 5, a string sera exibida nesta
                                          // linha
      cleeasy::sair ();                   // finaliza aplicativo
?>
```

cleeasy::fgets_ ($tamanho)

Le **$tamanho** caracteres da entrada padrão. Exemplo:

```
<?php
      include 'cleeasy.inc.php';          // inclui classe
      define ('NET_using', 0);            // retira suporte a rede
      cleeasy::echo_ ('Digite seu nome: ');
      $nome = cleeasy::fgets (30)         // le 30 caracteres e armazena em $nome
      cleeasy::echo_ ("Ola $nome !!!\n"); // exibe $nome
      cleeasy::sair ();                   // finaliza aplicativo
?>
```

cleeasy::chargen ()
Retorna um caracter aleatório. Exemplo:

```
<?php
      include 'cleeasy.inc.php';          // inclui classe
      define ('NET_using', 0);            // retira suporte a rede
      $char = cleeasy::chargen ();        // gera um caracter...
      cleeasy::echo_ ($char);             // ... e o exibe.
      cleeasy::sair ();                   // finaliza aplicativo
?>
```

cleeasy::typewriter ($linha, $coluna, $string, $reverse = 0, $speed = 5)
Exibe a string **$string,** caracter por caracter, na linha **$linha** e iniciando pela coluna **$coluna**. Caso **$reverse** seja diferente de 0, inicia a exibição da string pelo final. A velocidade **$speed** pode ser um valor entre 1 (mais rápido) e 10 (mais lento), sendo padrão o valor 5. Veja exemplo no arquivo **exemplos_strings.php.**

cleeasy::randon_string ($linha, $coluna, $string, $speed = 5)
Exibe a string **$string** na linha **$linha**, iniciando pela coluna **$coluna**. A velocidade **$speed** pode ser um valor entre 1 (mais rápido) e 10 (mais lento). Primeiro é exibida uma string aleatória do mesmo comprimento de **$string** na posição indicada. Após isso, cada caracter vai sendo substituído pelo referente a string naquela posição. Veja exemplo no arquivo **exemplos_strings.php.**

cleeasy::vertical ($linha, $coluna, $string, $reverse = 0, $speed = 0)
Exibe a string **$string** na vertical, iniciando pela linha **$linha**, na coluna **$coluna**. Caso **$reverse** seja diferente de 0, começa a exibição pelo final. A velocidade **$speed** pode ser um valor entre 1 (mais rápido) e 10 (mais lento).Veja exemplo no arquivo **exemplos_strings.php.**

cleeasy::carregando ($linha, $coluna, $tempo)
Simula uma barra de progresso na linha **$linha**, iniciando pela coluna **$coluna**, durante **$tempo** segundos. Veja exemplo no arquivo **exemplos_strings.php.**

cleeasy::updown ($linha, $coluna, $string, $speed = 5)
Exibe a string **$string** na linha **$linha**, iniciando pela coluna **$coluna**. A velocidade **$speed** pode ser um valor entre 1 (mais rápido) e 10 (mais lento). Inicia-se caindo caracteres aleatórios 4 linhas acima e subindo caracteres aleatórios 4 linhas abaixo. Quando eles se encontram, exibe o caracter definitivo daquela posição. Veja exemplo no arquivo **exemplos_strings.php.**

cleeasy::limpar_linha ($linha, $inicio, $fim, $speed = 0)
Limpa linha **$linha**, iniciando em **$inicio** e terminando em **$fim**, na velocidade determinada por **$speed**. A velocidade **$speed** pode ser um valor entre 1 (mais rápido) e 10 (mais lento). Caso **$inicio** for maior que **$fim**, a operação será feita iniciando pelo fim. Veja exemplo no arquivo **exemplos_strings.php.**

As funções para dar suporte a rede são as seguintes:

cleeasy::sair ()
Sai do servidor, sem finaliza-lo. Util para desconectar usuários e manter o servidor em funcionamento.

cleeasy::fim ()
Finaliza o servidor e desconecta usuário, caso esteja conectado.

cleeasy::fgets_ ($tamanho)
Lê **$tamanho** caracteres da entrada padrão.

No caso de utilização de suporte a rede, para ler-mos algo vindo do cliente, podemos utilizar a função **fgets_()**. Para enviar algo ao cliente, podemos utilizar qualquer uma das funções de exibição de strings listadas acima (exatamente, podemos enviar strings animadas e coloridas ao cliente). Para um exemplo, verifique o arquivo **exemplos_strings_net.php** (para executa-lo, execute "**php caller.php**").

CONSTANTES DEFINIDAS PELA CLASSE CLEEASY QUE DEVEM SER CONFIGURADAS

define ('NET_ip', '127.0.0.1')
Endereço IP que o servidor vai funcionar;

define ('NET_porta', '10003')
Porta que o servidor vai escutar por conexões;

define ('NET_timeout', 10)
Tempo antes de o servidor ser finalizado, por falta de alguém conectar-se;

define ('NET_aplic', 'SCRIPT.php')
Path para o aplicativo que receberá o suporte a rede;

define ('NET_aplic_nome', 'SCRIPT NOME')
Nome do aplicativo que receberá o suporte a rede;

define ('NET_using', 0)
Informa se vai ou não dar suporte a rede na aplicação. A constante deve ser definida no arquivo que utiliza classe, somente se o suporte a rede não for utilizado.

SESSIONS UTILIZADAS PELA CLASSE CLEEASY, QUE NÃO DEVEM SER MODIFICADOS

$_SESSION['NET_socket']
Socket que está sendo utilizado pelo cliente. Criado somente após alguém conectar-se ao servidor.

$_SESSION['NET_main_sock']
Socket que o servidor utiliza. Criado após execução do script, caso não ocorra nenhuma falha.

LICENÇA, COPYRIGHT, ETC...

O script é este e faz isso que foi dito (talvez um pouco mais...) mas não tem nenhuma garantia. Caso você encontre algum problema (é só procurar), basta me avisar que aplicarei as devidas correções e colocarei os devidos créditos. Caso aumente a funcionalidade da classe, com a adição de novas funções, ficaria grato se me enviasse, para que em uma futura versão, esta funcionalidade fosse incluída. Em caso de dúvidas e sugestões, entre em contato comigo também. Em caso de reclamações, sugiro que você procure o PROCON de sua região...

Pretendo, na medida do possível adicionar mais features na classe, dar continuidade ao projeto. Mas no momento é isso!

Para informações sobre copyright, verifique o arquivo Licenca-pt-BR.txt, que deve ter sido incluído no pacote.

“tDs”, 2004