I Socket in PHP

(manuale PHP)

Introduzione

Questa estensione implementa una interfaccia a basso livello verso i socket, fornendo la possibilità di agire sia come server sia come client.
Per l'utilizzo di queste funzioni, è importante ricordare che molte di esse hanno il medesimo nome della loro controparte in C, ma spesso hanno dichiarazioni differenti. Ricordarsi di leggere la descrizione per evitare confusione.

Gestione degli errori nei socket

L'estensione dei socket è stata scritta per fornire una pratica interfaccia ai potenti socket BSD. Si noti che le funzioni girano correttamente sia su sistemi Unix sia su sistemi Win32. Quasi tutte le funzioni dei socket possono fallire in certe situazioni e quindi generare un messaggio di tipo E_WARNING per descrivere l'errore. Qualche volta ciò non accade secondo le aspettative dello sviluppatore. Ad esempio la funzione socket_read() improvvisamente può generare un messaggio di E_WARNING a causa di una interruzione inaspettata della connessione. E' prassi comune sopprimere i messaggi di warning con l'operatore @ ed intercettare il codice dell'errore utilizzando la funzione socket_last_error(). Si può anche richiamare la funzione socket_strerror() con questo codice di errore per avere un testo descrittivo del problema. Per maggiori dettagli vedere la descrizione delle funzioni.
Nota: I messaggi E_WARNING generati dal modulo dei socket sono in inglese, mentre i messaggi di errore recuperati dipendono dalle correnti impostazioni locali (LC_MESSAGES):
Warning - socket_bind() unable to bind address [98]: Die Adresse wird bereits verwendet

Funzioni socket di PHP

- **socket_accept**: Accetta una connessione su un socket
- **socket_bind**: Bind di un nome ad un socket
- **socket_clear_error**: Azzera gli erorri di un socket, oppure l'ultimo codice d'errore
- **socket_close**: Chiude una risorsa di tipo socket
- **socket_connect**: Inizia una connessione su un socket
- **socket_create_listen**: Apre un socket per accettare connessioni su una porta
- **socket_create_pair**: Crea una coppia di socket non distinguibili e li memorizza in una matrice
- **socket_create:** Crea un socket (punto terminale di una comunicazione).
- **socket_get_option**: Ottiene le opzioni per un socket
- **socket_getpeername**: Interroga il lato remoto di un dato socket per ottenere o la combinazione host/porta od un percorso Unix, in base al tipo di socket
- **socket_getsockname**: Interroga il lato locale di un dato socket e restituisce o la combinazione host/porta oppure un percorso Unix in base al tipo di socket
- **socket_last_error:** Restituisce l'ultimo errore su un socket.
- **socket_listen:** Attende una richiesta di connessione su un socket
- **socket_read**: Legge fino ad un massimo di byte predefiniti da un socket
- **socket_recv**: Riceve i dati da un socket collegato
- **socket_recvfrom**: Riceve i dati da un socket, che sia connesso o meno
- **socket_select**: Esegue la system call select() su un set di socket con un dato timeout
- **socket_send**: Invia i dati ad un socket collegato
- **socket_sendto**: Invia un messaggio ad un socket, a prescindere che sia connesso o meno
- **socket_set_block**: Sets blocking mode on a socket resource
- **socket_set_nonblock**: Attiva la modalità "nonblocking" per il descrittore di file fd
- **socket_set_option**: Valorizza le opzioni per un socket

- **socket_shutdown**: Chiude un socket in ricezione, in invio o in entrambi i sensi
- **socket_strerror**: Restituisce una stringa con la descrizione dell'errore.
- **socket_write**: Scrive su un socket.

Esempi

Esempio 1. Esempio di programma con i socket: semplice server TCP/IP

Questo esempio illustra un semplice server. Occorre modificare le variabili address e port per adeguarle ai parametri della macchina su cui sarà eseguito. Ci si può connettere al server con un comando simile a telnet 192.168.1.53 10000 (dove l'indirizzo e la porta devono essere uguali a quanto indicato nel setup). Qualsiasi lettera sarà digitata, verrà visualizzata sul server e sul client. Per disconnettersi, digitare 'quit'.
Nelle applicazioni di domotica normalmente non dovremo creare dei server.

```
#!/usr/local/bin/php -q
<?php
error_reporting(E_ALL);

/* Si indica allo script di non uscire mentre attende una connessione */
set_time_limit(0);

/* Abilita lo scarico dell'output così si è in grado di vedere cosa passa
 * non appena arrivano i dati al server. */
ob_implicit_flush();

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)) < 0) {
  echo "socket_create() fallito: motivo: " . socket_strerror($sock) . "\n";
}

if (($ret = socket_bind($sock, $address, $port)) < 0) {
  echo "socket_bind() fallito: motivo: " . socket_strerror($ret) . "\n";
}

if (($ret = socket_listen($sock, 5)) < 0) {
  echo "socket_listen() fallito: motivo: " . socket_strerror($ret) . "\n";
}

do {
  if (($msgsock = socket_accept($sock)) < 0) {
     echo "socket_accept() fallito: motivo: " . socket_strerror ($msgsock) . "\n";
     break;
  }
  /* Invio delle istruzioni */
  $msg = "\nBenvenuti al server di test in PHP. \n" .
     "Per uscire, digitare 'quit'. Per chiudere il server digitare 'shutdown'.\n";
  socket_write($msgsock, $msg, strlen($msg));

  do {
     if (FALSE === ($buf = socket_read($msgsock, 2048, PHP_NORMAL_READ))) {
        echo "socket_read() fallito: motivo: " . socket_strerror($ret) . "\n";
        break 2;
```

```
    }
    if (!$buf = trim($buf)) {
        continue;
    }
    if ($buf == 'quit') {
        break;
    }
    if ($buf == 'shutdown') {
        socket_close($msgsock);
        break 2;
    }
    $talkback = "PHP: Testo scritto '$buf'.\n";
    socket_write($msgsock, $talkback, strlen $talkback));
    echo "$buf\n";
  } while (true);
  socket_close($msgsock);
} while (true);

socket_close($sock);
?>
```

Esempio 2. Esempio di programma con i socket: semplice client TCP/IP

In questo esempio sarà illustrato un semplice client HTTP. Questo, molto semplicemente, si collega ad un server, invia una richiesta HEAD, visualizza la risposta ed esce.

```
<?php
error_reporting(E_ALL);

echo "<h2>Connessione TCP/IP </h2>\n";

/* Ottiene la porta per il servizio WWW. */
$service_port = getservbyname('www', 'tcp');

/* Ottiene l'indirizzo IP per il server cercato. */
$address = gethostbyname('www.php.net');

/* Crea un socket TCP/IP. */
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
if ($socket < 0) {
  echo "socket_create() fallito: motivo: " . socket_strerror($socket) . "\n";
} else {
  echo "OK.\n";
}

echo "Tentativo di connessione a '$address' sulla porta '$service_port'...";
$result = socket_connect($socket, $address, $service_port);
if ($result < 0) {
  echo "socket_connect() fallito.\nMotivo: ($result) " . socket_strerror($result) . "\n";
} else {
  echo "OK.\n";
}
$in = "HEAD / HTTP/1.1\r\n";
$in .= "Host: www.example.com\r\n";
```

```
$in .= "Connection: Close\r\n\r\n";
$out = '';

echo "Invio HTTP HEAD...";
socket_write($socket, $in, strlen ($in));
echo "OK.\n";

echo "Lettura della risposta:\n\n";
while ($out = socket_read($socket, 2048)) {
   echo $out;
}

echo "Chiusura del socket...";
socket_close($socket);
echo "OK.\n\n";
?>
```

Funzionalità socket di alto livello

int fsockopen (string hostname, int porta [, int errno [, string errstr [, float timeout]]])
Inizializza una connessione nel dominio Internet (AF_INET, usando TCP o UDP) o Unix (AF_UNIX). Per il dominio Internet, apre una connessione a un socket TCP verso l' hostname sulla porta port. hostname può essere in questo caso, sia un fully qualified domain name o un indirizzo IP. Per le connessioni UDP, è necessario specificare esplicitamente il protocollo, usando: 'udp://' come prefisso di hostname. Per il dominio Unix, hostname viene utilizzato come percorso verso il socket, in questo caso, porta deve essere impostato a 0. Il parametro opzionale timeout può essere usato per impostare un timeout in secondi per la chiamata di sistema connect.

A partire da PHP 4.3.0, se si è compilato con il supporto OpenSSL, si può prefissare hostname con 'ssl://' oppure 'tls://' per utilizzare una connessione client SSL o TLS su una connessione TCP/IP per connettersi all'host remoto.

fsockopen() restituisce un puntatore a file che può essere usato nelle altre funzioni orientate ai file (come fgets(), fgetss(), fputs(), fclose() e feof()).

Se la chiamata non ha successo, viene restituito FALSE e se gli argomenti opzionali errno e errstr sono presenti, vengono impostati a indicare l'errore a livello di sistema che è avvenuto nella chiamata alla funzione connect() del sistema operativo. Se il valore di errno restituito è 0 e la funzione restituisce FALSE, è un'indicazione che l'errore è avvenuto prima della chiamata di connect(). Questo è molto probabilmente legato ad un problema di inizializzazione del socket. Si noti che gli argomenti errno e errstr verranno sempre passati by reference.

A seconda dell'ambiente operativo, il dominio Unix o l'opzionale timeout della connect potrebbero non essere disponibili.

Il socket viene aperto di default in modo blocking. Si può passare al modo non-blocking usando socket_set_blocking().

Esempio 1. Esempio di fsockopen()

```
<?php
$fp = fsockopen ("www.php.net", 80, $errno, $errstr, 30);
if (!$fp) {
```

```
   echo "$errstr ($errno)<br>\n";
} else {
   fputs ($fp, "GET / HTTP/1.0\r\nHost: www.php.net\r\n\r\n");
   while (!feof($fp)) {
      echo fgets ($fp,128);
   }
   fclose ($fp);
}
?>
```

L'esempio seguente mostra come ottenere data e ora tramite il servizio UDP "daytime" (porta 13) della vostra stessa macchina.

Esempio 2. Uso di connessione UDP

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, $errno, $errstr);
if (!$fp) {
   echo "ERRORE: $errno - $errstr<br>\n";
} else {
   fwrite($fp,"\n");
   echo fread($fp, 26);
   fclose($fp);
}
?>
```

Nota: Il parametro timeout è stato introdotto nel PHP 3.0.9 e il supporto UDP è stato aggiunto nel PHP 4.