Использование сокетов

Использование сокетов в PHP

Открытие сокета

Для открывания сокета используется функция fsockopen():

```
int fsockopen (string hostname, int port [, int errno [, string errstr [, double timeout]]])
```

например

```
/* попытка соединения */
$smtp_conn = fsockopen(
   $host,    // имя или IP-адрес сервера
   $port,    // номер порта
   $errno,   // переменная, в которую можно записать номер ошибки
   $errstr,  // переменная, в которую можно записать описание ошибки
   $tval);   // максимальное время ожидания ответа

/* проверка, получилось ли подключиться */
if(empty($smtp_conn))
   echo "SMTP -> ERROR: Failed to connect to server: $errstr ($errno)";
```

Ведение диалога с сервером.

Для диалога с сервером используются функции для вывода в файл и чтения из файла:

```
int fwrite ( resource $handle , string $string [, int $length ] ) (другое имя этой функции - fputs)
string fgetss ( resource $handle [, int $length [, string $allowable_tags ]] )
string fgets ( resource $handle [, int $length ] )
```

Пример 1. Авторизация на SMTP сервере.

```
// чтение из сокета
   function get_lines($smtp_conn)
   {
     $data = "";
     while($str = @fgets($smtp_conn,515)) $data .= $str;
     return $data;
   }

// отправка запроса на авторизацию
   fputs($smtp_conn,"AUTH LOGIN\r\n");
   $reply= get_lines($smtp_conn);
   $code = substr($reply,0,3);
   if($code != 334) echo "SMTP -> ERROR: AUTH not accepted from server: " . $rply . $this->CRLF;

// Оправка имени пользователя
   fputs($smtp_conn, base64_encode($username) ."\r\n");
   $reply = get_lines($smtp_conn);
   $code = substr($reply,0,3);
   if($code != 334) echo "SMTP -> ERROR: Username not accepted from server: $reply \r\n";

// Отправка пароля
   fputs($smtp_conn, base64_encode($password) . "\r\n");
   $reply = get_lines($smtp_conn);
   $code = substr($reply,0,3);
   if($code != 235)  echo "SMTP -> ERROR: Password not accepted from server: " . $reply . "\r\n";
```

Пример 2. Загрузка веб-страницы

```
<?php
$fp = fsockopen("www.example.com", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br />\n";
} else {
    $out = "GET / HTTP/1.1\r\n";
    $out .= "Host: www.example.com\r\n";
    $out .= "Connection: Close\r\n\r\n";
    fwrite($fp, $out);
    while (!feof($fp)) {
        echo fgets($fp, 128);
    }
```

```
    fclose($fp);
}
?>
```

CURL

Простой пример

Самым мощной клиентской библиотекой для HTTP является CURL (Client URL function library). PHP поддерживает её, опираясь на системную библиотеку libcurl. Чтобы включить поддержку CURL, надо сконфигурировать PHP с ключом --with-curl=[location of curl libraries] (для ОС UNIX) или включить поддержку CURL в файле конфигурации(для ОС Цштвшці). Естественно перед этим вы должны установить библиотеки libcurl.

CURL содержит большинство функций, которые могум понадобиться вам для общения с удалённым сервером: отправка запросов POST и GET, поддержка SSL, авторизация HTTP, сессии и куки.

Чтобы начать сеанс связи, в curl используется функция curl_init(). Параметры сеанса устанавливаются функцией curl_setopt(). Установив все параметры, вы можете отправить запрос удалённому серверу, вызвав curl_exec().

Пример:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com');
curl_setopt($ch, CURLOPT_HEADER, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
$data = curl_exec();
curl_close($ch);
```

В этом примере мы сохранили указатель на сеанс CURL в переменной $ch. Тогда, используя функцию curl_setopt() мы установили URL, которому надо отправить запрос(http://www.example.com). Параметр CURLOPT_HEADER устанавливает, стоит ли возвращать заголовки HTTP вместе с основным текстом ответа. По умолчанию curl возвращает ответ на стандартный вывод (т.е. в обозреватель интернет). Чтобы отменить это, мы изменили параметр CURLOPT_RETURNTRANSFER. После установки параметров была вызвана функция curl_exec(), которая вернула текст ответа, полученный с сервера. Этот ответ мы сохранили в переменную $data.

Отправка данных

```
$phoneNumber = '4045551111';
$message = 'This message was generated by curl and php';
$curlPost = 'pNUMBER='  . urlencode($phoneNumber) . '&MESSAGE=' . urlencode($message) . '&SUBMIT=Send';
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com/sendSMS.php');
curl_setopt($ch, CURLOPT_HEADER, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $curlPost);
$data = curl_exec();
curl_close($ch);
```

За отправку данных методом POST отвечают строки

```
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $curlPost);
```

curl_setopt($ch, CURLOPT_POST, 1); показывает, что будет использоваться метод POST.
curl_setopt($ch, CURLOPT_POSTFIELDS, $curlPost); передаёт сеансу данные.

Сами данные содержаться в строке $curlPost и имеют точн такой же вид, как и данные , передаваемые в адресной строке:

```
$curlPost = 'pNUMBER='.urlencode($phoneNumber)
           .'&MESSAGE='.urlencode($message)
           .'&SUBMIT=Send';
```

Использование прокси-сервера

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com');
curl_setopt($ch, CURLOPT_HEADER, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HTTPPROXYTUNNEL, 1);
```

```
curl_setopt($ch, CURLOPT_PROXY, 'fakeproxy.com:1080');
curl_setopt($ch, CURLOPT_PROXYUSERPWD, 'user:password');
$data = curl_exec();
curl_close($ch);
```

Для использования прокси-сервера следует передать сеансу три параметра:

```
curl_setopt($ch, CURLOPT_HTTPPROXYTUNNEL, 1);
curl_setopt($ch, CURLOPT_PROXY, 'fakeproxy.com:1080');
curl_setopt($ch, CURLOPT_PROXYUSERPWD, 'user:password');
```

Использование SSL

Чтобы использовать SSL, просто впишите в URL правильное название протокола: https:// а не http://

Cookies

CURLOPT_COOKIE используется для установки cookies в текущем сеансе. CURLOPT_COOKIEJAR указывает, в какой файл надо сохранить полученные от сервера cookies. CURLOPT_COOKIEFILE указывает, в каком файле хранятся cookies.

HTTP authentication

Простая передача имени и пароля:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt(CURLOPT_USERPWD, '[username]:[password]')

$data = curl_exec($ch);
curl_close($ch);
```

Кроме CURLAUTH_BASIC можно ещё писать CURLAUTH_DIGEST, CURLAUTH_GSSNEGOTIATE, CURLAUTH_NTLM для выбора одной из доступных форм передачи имени и пароля.

Комбинация
CURLAUTH_BASIC|CURLAUTH_DIGEST|CURLAUTH_GSSNEGOTIATE|CURLAUTH_NTLM
называется CURLAUTH_ANY.

Комбинация **CURLAUTH_DIGEST|CURLAUTH_GSSNEGOTIATE|CURLAUTH_NTLM** называется CURLAUTH_ANYSAFE.

Выбрать любой способ:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_ANY);
curl_setopt(CURLOPT_USERPWD, '[username]:[password]')
$data = curl_exec($ch);
curl_close($ch);
```