

Application socket client/serveur en PHP

Partie I – Utilisation des flux ("streams" en anglais) pour gérer les sockets

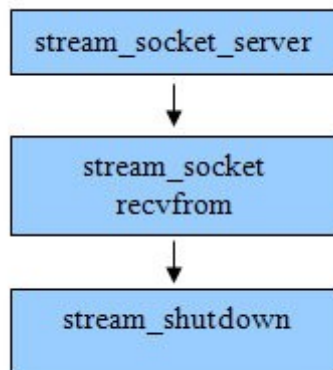
I – Définition d'un flux

Un flux est une ressource qui présente des capacités de flux : c'est-à-dire que ces objets peuvent être lus ou recevoir des écritures de manière linéaire, et dispose aussi de moyen d'accéder à des positions arbitraires dans le flux.

Source : <http://fr.php.net/manual/fr/intro.stream.php>

II – Création d'un serveur simple

Notre serveur permettrait de recevoir un simple message envoyé à partir de notre client. Pour cet exemple, nous allons utiliser les sockets en mode non connecté, soit UDP (User Datagram Protocol), les données sont donc envoyées sans connexion préalable. On va structurer notre programme en plusieurs fonctions sur ce principe :



Dans un fichier function.inc.php

Fonction de création d'un flux stream socket.

```
function creation_stream_socket_server($addr, $port)
{
    /* Fonction : création d'une socket en mode non connecté (UDP) */

    /* Initialisation des variables */
    $errno=0;
    $errstr='';
}
```

```

if(empty($addr) OR empty($port))
{
    echo 'Adresse ou port vide.<br />';
    exit(1);
}
$socket = stream_socket_server(UDP'://'.$addr.':'.$port, , $errstr,
STREAM_SERVER_BIND);

if (!$socket)
{
    echo 'La socket n\'a pas été créée.';
    exit(2);
}
return $socket;
}

```

UDP'://.\$addr.':.\$port est le \$local_port, c'est-à-dire l'adresse à laquelle la socket va écouter sur la machine ainsi que son port, c'est ici que l'on définit le transport soit UDP soit TCP.

\$errno et **\$errstr** sont les variables servant à récupérer les erreurs pour savoir pourquoi le descripteur du flux stream socket n'a pas été renvoyé.

On ne s'en sert pas ici, ces variables sont uniquement présente pour définir le flags suivant.

On met le flags à **STREAM_SERVER_BIND** pour utiliser le flux en mode UDP.

Cette fonction renvoi un descripteur de socket (servant pour la suite), ici on le place dans \$socket et on retourne \$socket;

Une fois la connexion effectuée, on passe à la réception des données.

```

function reception_donnees($socket)
{
    /* Réception d'un message, la limite du message est ici en octets,
soit 1.5ko */
    echo 'Message reçu : ' . stream_socket_recvfrom($socket, 1500) .
'<br />';
}

```

La fonction de réception des données n'a besoin que du descripteur des sockets.

Ici on fait un simple affichage du message reçu.

Lors de l'utilisation de cette fonction, elle est dîtes en mode bloquant, c'est-à-dire qu'elle attend qu'un flux soit reçu pour se terminer, si aucun message n'est reçu au bout d'un certain temps, la socket se fermera d'elle-même (pour modifier ceci il faut régler le timeout grâce à cette fonction : **stream_set_timeout()**).

On peut utiliser la fonction **stream_set_blocking()**, dans laquelle on peut spécifier si on configure en mode bloquant ou non.

Le problème est que si on utilise ceci, PHP renverra la page au serveur http, et le serveur renverra la page sans que le message soit reçu ou non.

Une solution serait d'utiliser un code JavaScript qui rechargerait la page, nous n'étudierons pas ceci ici.

Une fois la réception du ou des messages, il faut fermer le flux.

```

function close_stream_socket($socket)
{
    if($socket)
    {
        stream_socket_shutdown ($socket, STREAM_SHUT_RDWR);
    }
}

```

On arrête toutes les transmissions en réception ou en émission avec l'option **STREAM_SHUT_RDWR**.

Création de la page serveur.php

```
<?php
include('function.inc.php'); // Inclusion des fonctions que l'on vient de
créer
$addr = '127.0.0.1'; // Adresse d'écoute du flux
$port = '62000'; // Port d'écouter du flux

$socket = creation_stream_socket_server($addr, $port);
reception_donnees($socket);

close_socket($socket);
?>
```

III – Création d'un client simple

Toujours sur le même mode de connexion (UDP).

On va désormais appliquer ce principe (la méthode est sensiblement la même).

On ouvre notre flux stream socket, on envoi les données et on ferme le flux.

Toujours dans le fichier function.inc.php

```
function creation_stream_socket_client($addr, $port)
{
    /* fonction : création d'une socket en mode non connecté (UDP */
    /* Initialisation des variables */
    $errno=0;
    $errstr='';

    if (empty($addr) OR empty($port))
    {
        echo 'Adresse ou port vide.<br />';
        exit(1);
    }

    $socket = stream_socket_client(UDP:'//'.$addr.':'.$port, , $errstr,
    STREAM_SERVER_BIND);

    if (!$socket)
    {
        echo 'La socket n\'a pas été créée.';
        exit(2);
    }
    return $socket;
}
```

Une fois la connexion effectuée, on passe à la renvoyé des données.

```
function envoi_donnees($socket, $donnees)
{
    fwrite($socket, $donnees);
}
```

La fonction de fermeture est la même que celle du serveur.

Création de la page client.php

```
<?php
include('function.inc.php'); // Inclusion des fonctions que l'on vient de
créer
$addr = '127.0.0.1'; // Adresse du serveur
$port = '62000'; // Port du serveur

$socket = creation_stream_socket_server($addr, $port);
$donnees = 'Bonjour je vous envoi un message !';
envoi_donnees($socket, $donnees);

close_socket($socket);
?>
```

Et voilà, nous avons créé une simple application client/serveur.

Références :

- <http://fr.php.net/manual/fr/ref.stream.php>