METRIX
INTERNET DESIGN GMBH

IPC mit PHP4

International PHP Conference 2002
06/11/2002, Frankfurt-Mörfelden

Stephan Schmidt

Inhalt

- Was ist IPC?
- Anwendungen in der PHP Welt
- PHP ohne IPC
- Kommunikationsarten
- PHP mit IPC Support
- Shared Memory mit PHP
- Semaphore mit PHP
- Beispielsapplikation
- Fallstricke
- Message Queues mit PHP

Stephan Schmidt

- Web Application Developer bei der Metrix Internet Design GmbH in Karlsruhe
- PHP seit 1999
- Gründungsmitglied der PHP Application Tools www.php-tools.net
- Autor von patTemplate, patXMLRenderer, patUser und anderen Open Source Klassen
- Regelmäßiger Autor des PHP Magazins

Was ist IPC?

- Inter-Process-Communication
- Gleichzeitig laufende Prozesse kommunizieren miteinander
  - » Austausch beliebiger Daten
- Verschiedene Kommunikationsmedien
  - » Sockets
  - » Shared Memory
  - » Dateisystem
- Verschiedene Protokolle, z.B. WDDX, serialize(),...

Anwendungen in der PHP Welt

- Webapplikationen
  - » Web-Chats
  - » Messenger
- Desktop-Anwendungen
  - » PHP-GTK
- „Remote Procedure Calls“ ohne „Remote“
  - » Ständig laufendes Skript führt Arbeiten aus und erhält Anweisungen von anderen Skripten.

PHP ohne IPC

- Standardfall von Webanwendung
- Jeder Prozess läuft in einer „Sandbox“
  - » Jeder Prozess läuft für sich
  - » PHP ist im Webserver gekapselt
  - » Weiß nichts über andere Prozesse
- Verhalten ist gewünscht
  - » Sicherheit
  - » Stabilität

PHP entwickelt sich weiter, IPC kann auch eine Rolle spielen.

Bisher keine Kommunikation?

Kommunikation zwischen verschiedenen Prozessen findet fast in jeder Webanwendung statt:

- Gästebuch, bzw. Forum
  » Eingabe von Daten, Ausgabe von Daten

- Mehrseitige Formularabläufe
  » Weitergabe von Daten über Session

Unterschiede zu IPC?

» Art der Kommunikation

Kommunikationsarten

- **Zeitversetzte Kommunikation**
  - » hohe Latenzzeit („Briefverkehr”)
  - » Prozesse laufen nicht parallel
  - » Nachricht bleibt längere Zeit im Medium
  - » Textdateien oder Datenbank als Medium
- **Zeitnahe (unmittelbare) Kommunikation**
  - » kurze Latenzzeit („Telefongespräch“)
  - » Prozesse laufen gleichzeitig
  - » Nachricht bleibt nur kurz im Medium
  - » Schnelles Medium erforderlich

» Verwenden eines schnellen Mediums

Shared Memory

- Gemeinsam genutzte Speichersegmente
- Lese- und Schreibzugriff möglich
- Zugriff unabhängig von
  - » Prozess
  - » Skript / Applikation
  - » Benutzer (kann eingeschränkt werden)
  - » Programmiersprache
- schneller Zugriff möglich

» Perfektes Medium für IPC

SHM Support in PHP

- Keine Unterstützung auf Windows Systemen
- Extension seit PHP 3.0.6 verfügbar
- Kompilieren mit **`--enable-sysvshm`** und **`--enable-sysvsem`**
- System V Messages ab PHP 4.3 verfübgar, kompilieren mit **`--enable-sysvmsg`**

Shared Memory Funktionen

- Anlegen eines neuen Segments oder Anbinden eines bestehenden Segments
  - » Indetifizierung durch eindeutige Kennung
  - » Rückgabe eines Handles
- Lesen und Schreiben von Werten unter Verwendung des Handles
- Lösen der Anbindung
- Bei Bedarf Löschen des Segments

Erzeugen bzw. Anbinden eines Segments

```
$shmId = shm_attach( 2002, 1024 );
   if( !$shmId )
     die( "Zugriff auf Segment nicht möglich." );
```

Max. drei Parameter:

- Eindeutige Kennung des Segments
- Größe des Segments in Bytes, beschränkt durch Betriebssystem
- Berechtigungsmaske analog zum Dateisystem

Schreiben von Werten

```
define( "Ort", 1 );
shm_put_var( $shmId, Ort, "PHP Conference 2002" );
```

Drei Parameter:

- Handle, das von shm_attach() zurückgegeben wurde
- Eindeutige Kennung der Variablen
  - » Nur Integer Werte erlaubt
  - » Verwendung von Konstanten
- Wert der geschrieben werden soll

Lesen von Werten

```
$ort = shm_get_var( $shmId, Ort );
```

Zwei Parameter:

- Handle des Shared Memory Segments
- Eindeutige Kennung der Variablen

Löschen von Werten:

```
$ort = shm_remove_var( $shmId, Ort );
```

Beenden der Anbindung

```
shm_detach( $shmId );
```

- Handle des Shared Memory Segments als einzigen Parameter
- Daten bleiben im Speicher erhalten

Löschen des Speichersegments

```
shm_remove( 2002 );
```

Vorsicht: Kennung des Handles statt der Kennung übergeben!

Konkurrierende Zugriffe

- **Problem:**
  Mehr als ein Prozess greift auf das Segment zu
  » Korrupte Daten, falls ein Prozess schreibend zugreift (Daten werden während dem Lesen geändert)

- **Problemlösung:**
  Lockingmechanismus (analog zu **`lock()`**)
  » Nur ein Prozess greift auf Daten zu
  » Muss schnell sein (Testen und Setzen des Locks innerhalb eines Prozessortakts)

» Sempahore

Semaphore in PHP

- Handle der Semaphore mit **`sem_get()`**
  - » Erwartet eindeutige Kennung
  - » Gibt Handle zurück
- Anfordern der Semaphore mit **`sem_aquire()`**
  - » Erwartet Handle
  - » Blockiert Ausführung bis Semaphore frei ist
- Freigeben der Semaphore mit **`sem_release()`**
  - » Erwartet Handle
- Löschen der Semaphore mit **`sem_remove()`**
  - » Erwartet Handle

Beispiel

```
define( "Ort", 1 );
$shmKey   = 2002;
$semKey   = 2003;

$shmId    = shm_attach( $shmKey, 1024 );

$semId    = sem_get( $semKey );

sem_acquire( $semId );

shm_put_var( $shmId, Ort, "PHP Conference 2002" );

echo shm_get_var( $shmId, Ort );

sem_release( $semId );

shm_detach( $shmId );
```

patSHMC

- Kapselt shm_* und sem_* in einer Klasse
- Instanziierung erzeugt Speichersegment
- Methoden zum
  - » Lesen
  - » Schreiben
  - » Lock setzen
- Download unter http://www.php-tools.net

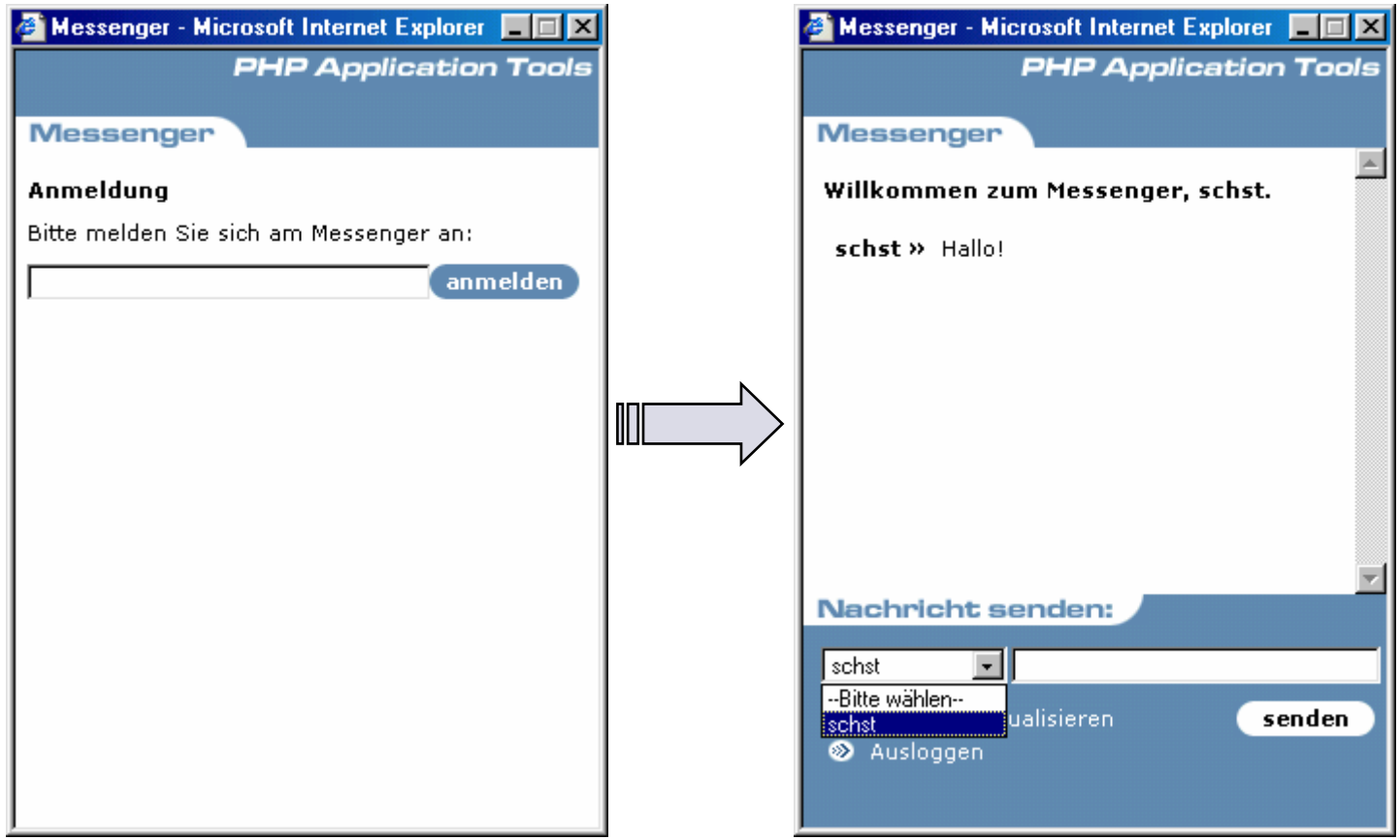
patSHMC Beispiel

```
define( "Ort", 1 );
require_once("include/patSHMC.php");

$shmKey       = 2002;
$semKey       = 2003;

$shm    = new patSHMC( $shmKey, 50000, $semKey );

$shm->lock( patEXCLUSIVE_LOCK );

$shm->put_var( Ort, "PHP Conference 2002" );

echo $shm->get_var( Ort );

$shm->lock( patRELEASE_LOCK );

$shm->detach();
```

Beispiel: Messenger

- Zeitnahe Kommunikation beliebig vieler Benutzer über Webbrowser
- Nachrichten haben genau einen Empfänger
- Login nur mit Nickname
  - » keine Authorisierung implementiert
- Zwei Frames
  - » Eingabe neuer Nachrichten
  - » Ausgabe empfangener Nachrichten

Beispiel: Messenger



Beispiel: Messenger

- Userliste
  - » Array im Shared Memory Segment
  - » Array-Key = User ID
  - » Array-Wert = Nickname
- Nachrichten
  - » Array im Shared Memory
  - » Array-Key = User ID des Empfängers
  - » Array-Wert = Array mit allen Nachrichten
  - » Schicken von Nachrichten mit **`array_push()`**
  - » Empfangen mit **`array_pop()`**

Daten im Speicher

```
Array
(
    [1] => schst
    [2] => argh
)
Array
(
    [1] => Array
        (
        )
    [2] => Array
        (
            [0] => Array
                (
                    [sender] => schst
                    [message] => Hallo!
                )
        )
)
```

Eingabeframe

- einfaches HTML Formular
- Empfänger und Nachricht werden via POST an ein PHP Skript geschickt
  - » Anbinden des Speichersegments
  - » Lock setzen
  - » Holen des Arrays mit allen Nachrichten
  - » Anfügen der neuen Nachricht an das Array
  - » Schreiben des Arrays in das Speichersegment
  - » Lock freigeben
  - » Anbindung lösen
  - » Formular wieder ausgeben

Ausgabeframe

- **`set_time_limit()`** verhindert Abbruch nach 30 Sekunden
- register_shutdown_function() falls Skript abbricht
- Skript in Endlosschleife
  - » Lock setzen
  - » Array mit Nachrichten holen
  - » Nachrichten für User aus Array löschen
  - » Nachrichten ausgeben
  - » flush();
  - » Lock freigeben
  - » usleep(500000);

Fallstricke

- set_time_limit() kann nicht ausgeführt werden
- regelmäßig nicht sichtbare Daten schicken, um Timeout des Browsers zu verhindern
- Ausgabepuffer mit **`flush()`** leeren und **`output_buffering`** in **`php.ini`** deaktivieren
- Verbindungssteuerung mit **`connection_status()`** oder **`register_shutdown_function()`** nicht vergessen

Message Queues

- Verfügbar ab PHP 4.3
- Ermöglicht Senden von Nachrichten an einen anderen Prozess
- Kein Locking mehr nötig
- Nachrichten werden beim Lesen aus der Queue entfernt

Anlegen einer Queue

```
$queue  = msg_get_queue( 2002 );
```

- Erwartet eindeutige Kennung der Queue
- Optional kann eine Berechtigungsmaske übergeben werden
- Liefert Handle der Queue zurück

Senden einer Nachricht

```
msg_send( $queue, 1, "PHP Conference" );
```

Maximal 6 Parameter

- Handle der Queue
- Typ der Nachricht (Integer)
  » Empfänger ID beim Messenger
- Inhalt der Nachricht
- "serialize" Flag
- "blocking" Flag (zu große Nachrichten)
- Referenz zu Variable, in die ein eventueller Fehler gespeichert wird

Empfangen von Nachrichten

```
$success  =  msg_receive( $queue, 0, $msgType, 4096,
   $msg, true, MSG_IPC_NOWAIT, $error );
```

Parameter:

- Handle der Queue
- Typ der Nachricht, die gelesen werden soll
- Referenz auf eine Variable, in die der Typ der gelesenen Nachricht geschrieben wird
- maximale Länge der Nachricht
- Referenz auf eine Variable, in die die Nachricht geschrieben wird
- "deserialize" Flag
- Flags, z.B. MSG_NOERROR, MSG_EXCEPT
- Referenz auf eine Variable in die eine eventuelle Fehlermeldung geschrieben wird

Beispiel

```
$queue  =  msg_get_queue( 2002 );

msg_send( $queue, 1, "PHP Conference" );

echo     "<pre>";
print_r( msg_stat_queue( $queue ) );
echo     "</pre>";

$msg      = "";
$msgType  = 0;
$error    = 0;
$success  =      msg_receive( $queue, 1, $msgType, 4096, $msg, true,
   MSG_IPC_NOWAIT, $error );
if( $success )
  echo   $msg."<br />";
else
  echo $error."<br />";

msg_remove_queue( $queue );
```

Informationen zu einer Queue

stat_queue() liefert Informationen zu einer Nachrichten Queue:

```
Array
(
  [msg_perm.uid] => 33
  [msg_perm.gid] => 33
  [msg_perm.mode] => 438
  [msg_stime] => 1034647575
  [msg_rtime] => 1034524622
  [msg_ctime] => 1034523467
  [msg_qnum] => 1
  [msg_qbytes] => 16384
  [msg_lspid] => 9699
  [msg_lrpid] => 9301
)
```

Fazit

- IPC ist mit PHP leicht möglich
- Nicht alles im Shared Memory speichern
  - » keine sensitiven Daten (Passwörter)
  - » keine Daten, die länger gespeichert werden
- shm_* und sem_* Funktionen sind nach Kapselung einfach zu bedienen
- System V Messages sollten beim professionellen Einsatz auch gekapselt werden

Ende

Viele Dank für Ihre Aufmerksamkeit.

Weitere Informationen:

- http://www.php-tools.net
- schst@php-tools.net

Dank an:

Sebastian Mordziol, Gerd Schaufelberger, Stephan Eisler, Metrix Internet Design GmbH