METRIX INTERNET DESIGN GMBH

XML Socket Server zur Kommunikation mit Flash

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

Stephan Schmidt & Gerd Schaufelberger

Inhalt

- Flash als Client
- Einführung in Sockets
- Socket Funktionen in PHP 4.2
- Generische Server Klasse "patServer"
- Socket Funktionen in Flash MX
- XML Protokoll zur Kommunikation
- Architektur des XML Servers
- Beispielsapplikation "patTicTacToe"

Gerd Schaufelberger

- Software Entwickler bei Epygi Labs DE
- Student der Informatik
- Gründungsmitglied der PHP Application Tools http://www.php-tools.net
- Autor von patSysinfo, patUser und anderen Open Source Tools

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

Ist HTML schlecht?

Nein, aber...

- entwickelt zur Präsentation von Textinformationen
- Nicht zum Entwickeln von Applikation gedacht
- Ursprünglich nur Links und Formulare zur Interaktion
- verwendet (berechtigterweise) HTTP

Flash vs HTML

HTML

- Inkompatibilitäten zwischen verschiedenen Browsern
- Unterschiedliche Darstellung auf verschiedenen Betriebssystemen

Flash

- Unabhängig von Browser und Betriebssystem, nur Plugin muss installiert sein

» 1:0 für Flash

Flash vs HTML

HTML

- Ursprünglich zur Textaufbereitung entwickelt
- Kaum Multimediafähigkeiten
- Nicht interaktiv

Flash

- Kombination von Grafik und Text
- Multimediafähig (MP3, Video,...)
- Interaktion möglich

» 2:0 für Flash

Flash vs HTML

HTML

- Übertragung durch HTTP
  - » Verbindungslos
  - » Emulation einer Verbindung durch Sessions

Flash

- Eingebettet in HTML
  - » Übertragung durch HTTP
- Aber: Flash bietet Möglichkeiten für ständige Serververbindungen

» 3:0 für Flash

Weitere Vorteile von Flash

- Einbetten von Schriftarten
- Skalierbare Vektorgrafiken
- Erzeugen von ausführbaren Dateien (Desktop Applikation)
- MP3 Unterstützung
- DOMXML-Support
- Interaktion ohne Request an den Server
- Graphische IDE

Dauerhafte Serververbindung

Wichtigstes Feature für diese Session:

» Ständige Verbindung zum Server durch Sockets

Einführung in Sockets

- Socket := Endpunkt einer Netzwerkverbindung
- Abstraktionslayer für Netzwerkkommunikation
- Unabhängig von
  - » Programmiersprache
  - » Betriebssystem

  auf der Gegenstelle
- Endpunkte können auf beliebigen Rechnern liegen

Verbindungslose Sockets

- Paketorientiert
- Jedes Paket enthält Absender- und Zieladresse
- Verschiedene Pakete können auf unterschiedlichen Wegen zum Ziel gelangen
- Pakete können sich überholen
- Beispiel: UDP (User Datagram Protocol)

Verbindungsorientierte Sockets

- Verbindungsorientiert
- Verbindungsaufbau am Anfang und Verbindungsabbau am Ende
  » Overhead
- Route wird beim Verbindungsaufbau festgelegt
- Daten werden ohne weitere Meta-Informationen übertragen
  » kompensiert Overhead
- Beispiel: TCP

Socket Funktionen in PHP 4.2

- Socket Support bereits seit PHP 4.1
- Bisher instabil und häufigen Änderungen der API unterworfen
- Kompilieren mit **`--enable-sockets`**
- Support für Verbindungslose und Verbindungsorientierte Sockets
- nicht fsockopen()

» Zur Kommunikation mit Flash sind nur verbindungsorientierte Sockets interessant.

Erzeugen des Sockets

```
$fd  = socket_create( AF_INET, SOCK_STREAM, SOL_TCP );
if( $fd )
  die( "Socket konnte nicht erstellt werden" );
```

Parameter:

- Protokollfamilie (AF_INET)
- Protkolltyp (SOCK_STREAM, SOCK_DGRAM)
- Protokollname (SOL_TCP)

Rückgabe:

» File Descriptor

Socket binden und abhören

```
if( !socket_bind( $fd, "myServer.myDomain.net", 9090 )
  die( "Konnte Socket nicht an Adresse binden." );
```

Parameter:

- File Descriptor
- Adresse
- Port

```
if( !socket_listen( $fd, 10 ) )
  die( "Konnte nicht auf Port hören" );
```

Parameter:

- File Descriptor
- Backlog

Verbindungen annehmen

```
$newFd  = socket_accept( $fd );
```

- Wartet auf eingehende Verbindung
- Skript hält an, bis Verbindung eingeht
- Erzeugt Socket mit identischen Eigenschaften
- Gibt File Descriptor für den erzeugten Socket zurück

» neuer File Descriptor identifiziert die Verbindung
und muss bei weiteren Funktionen verwendet werden.

Daten senden

```
$text = "Hallo Client!\n";
socket_write( $newFd, $text, strlen( $text ) );
```

- Länge der Daten muss als dritter Parameter übergeben werden
- Daten kommen automatisch bei der Gegenstelle an
- Gibt **`false`** zurück, falls Verbindung abgebrochen ist

Daten empfangen

```
$buf = socket_read( $newFd, 2048 );
```

Parameter:

- File Descriptor
- Anzahl an Bytes, die gelesen werden
- Lesemodus (optional)
  - » Eine Zeile
  - » Binary-safe

Rückgabe:

- » Gelesene Daten

Talkback Server (Main Loop)

```
while( true )
{
  if( false === ( $buf = socket_read( $newFd, 2048 ) ) )
  {
    echo "Fehler beim Lesen."\n";
    break 2;
  }
  if( !$buf = trim( $buf ) )
    continue;
  if( $buf == '/quit' )
    break;

  $antwort = "Sie sagten: '$buf'.\n";
  socket_write( $newFd, $antwort, strlen( $antwort ) );
}
```

Verbindungen schließen

```
socket_close( $newFd );
socket_close( $fd );
```

- Verbindungsabbau
- Entfernt den Socket aus dem System
- Gegenstelle registriert Abbau automatisch

Wichtig: Beide Sockets schließen.

Hinweise

- Skript läuft in einer Endlosschleife
  » Zeitlimit beachten
- Nach Möglichkeit CLI Version von PHP verwenden
  » #!/usr/bin/php -q
- Serverprozess von der Shell abhängen
  » nohup <Skriptname> &

Keine Kommunikation!

Problem:

- Nur ein Client möglich
- Server wird nach Beenden der Verbindung gestoppt

Lösung:

- Port mit socket_set_opt() wieder freigeben
- socket_select() statt socket_accept()

Keine Lust alles zu programmieren?

» patServer

patServer

- generische Serverklasse in PHP
- Abstraktion der Socket Funktionalität
  » Kaum Hintergrundwissen nötig
- Download von http://www.php-tools.net
- Ereignisbasiert
  » Callback Funktionen werden aufgerufen
- Eigene Serverklasse einfach von patServer ableiten
  » z.B: HTTP Server, Chat Server,...

Ereignisbehandlung

Mögliche Ereignisse und Handler:

- Serverstart » **`onStart()`**
- Server wird gestoppt » **`onShutdown()`**
- Verbindungsaufbau » **`onConnect()`**
- Verbindungsaufbau verweigert » **`onConnectionRefused()`**
- Empfang von Daten » **`onDataReceived()`**
- Verbindungsabbau » **`onClose()`**

Verbindungsmanagement

- Jeder Client bekommt eine eindeutige ID zugewiesen, wird bei **`onConnect()`** übergeben
- Client ID wird bei allen Aktionen des Clients (**`onClose()`**, **`onConnectionRefused()`**, **`onDataReceived()`**) an den Handler übergeben
- Methoden zur Interaktion mit den Clients
  - » **`sendData( $clientId, $data )`**
  - » **`broadcastData( $data )`**
  - » **`closeConnection( $clientId )`**
  - » **`isConnected( $clientId )`**

Beispiel

```
class myServer extends patServer
{
   function onConnect( $clientId )
   {
      echo "Neue Verbindung mit der ID $clientId";
   }

   function onDataReceived( $clientId, $data )
   {
      $this->sendData( $clientId,
                       "Sie haben $data gesendet" );
   }
}
```

Starten des Servers

```
require_once( "include/patServer.php" );
require_once( "include/myServer.php" );
$server = new myServer( "myServer.myDomain.net", 9090 );
$server->setMaxClients( 50 );
$server->start();
```

- Hostname und Port werden beim Instanziieren angegeben
- Maximale Anzahl an parallelen Clients kann beschränkt werden

Die andere Seite

- Flash bietet **`XMLSocket`** Objekt
- Stellt verbindungsorientierte Verbindung her
- Daten über diese Verbindung werden im XML Format verschickt
  - » Ausgehende Daten werden serialisiert
  - » Eingehende Daten stehen als DOM-Baum zur Verfügung
- Arbeitet ereignisbasiert

Action Script

Erzeugen des XML Sockets:

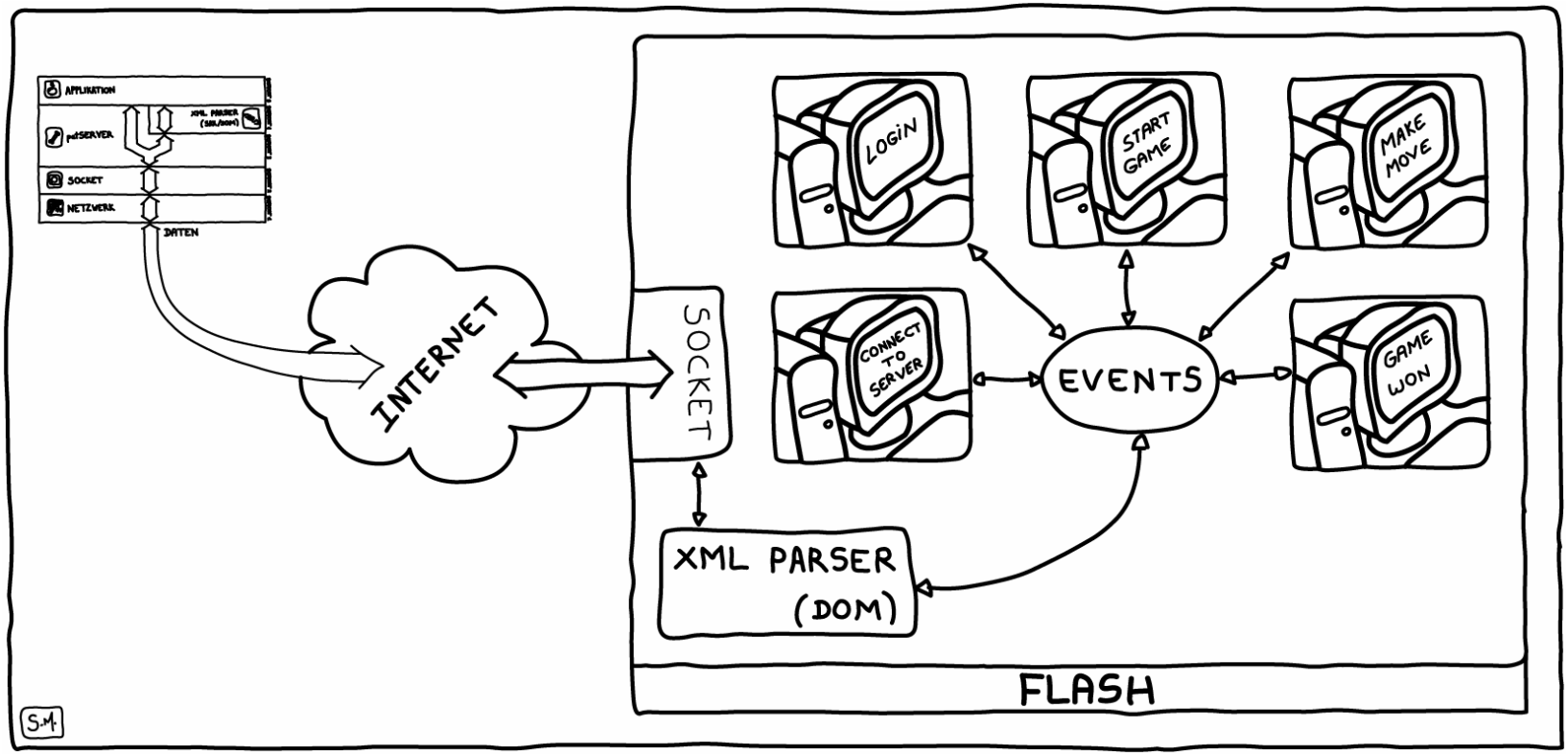
```
var socket = new XMLSocket();
socket.connect( "myServer.myDomain.net", 9090 );
```

Setzen der Ereignishandler:

```
socket.onConnect = connectionEstablished;
socket.onClose   = connectionClosed;
socket.onXML     = xmlReceived;
```

Methoden zur Ereignisbehandlung müssen noch implementiert werden.

Ereignisbehandlung in Action Script



Senden von Daten

```
var command = new XML();
var request = command.createElement("request");
command.appendChild( request );
// ...Einfügen weiterer Elemente...
socket.send( command );
```

- Flash stellt DOM Funktionen zur Verfügung
- Daten werden beim Senden serialisiert, z.B.

  <request><type>pat</type></request>

XML Protokoll

- Client und Server müssen gemeinsame Sprache sprechen
  - » Protokoll
- "Flaches" XML Format genügt, da keine komplexen Strukturen übetragen werden
  - » Beliebig viele Key/Value Paare
- Protokollprimitiven:
  - » REQUEST
  - » RESPONSE
  - » FAULT

Beispiele

```
<request>
  <type>Typ der Anfrage</type>
  <param>....</param>
  <!-beliebig viele Parameter-->
</request>
```

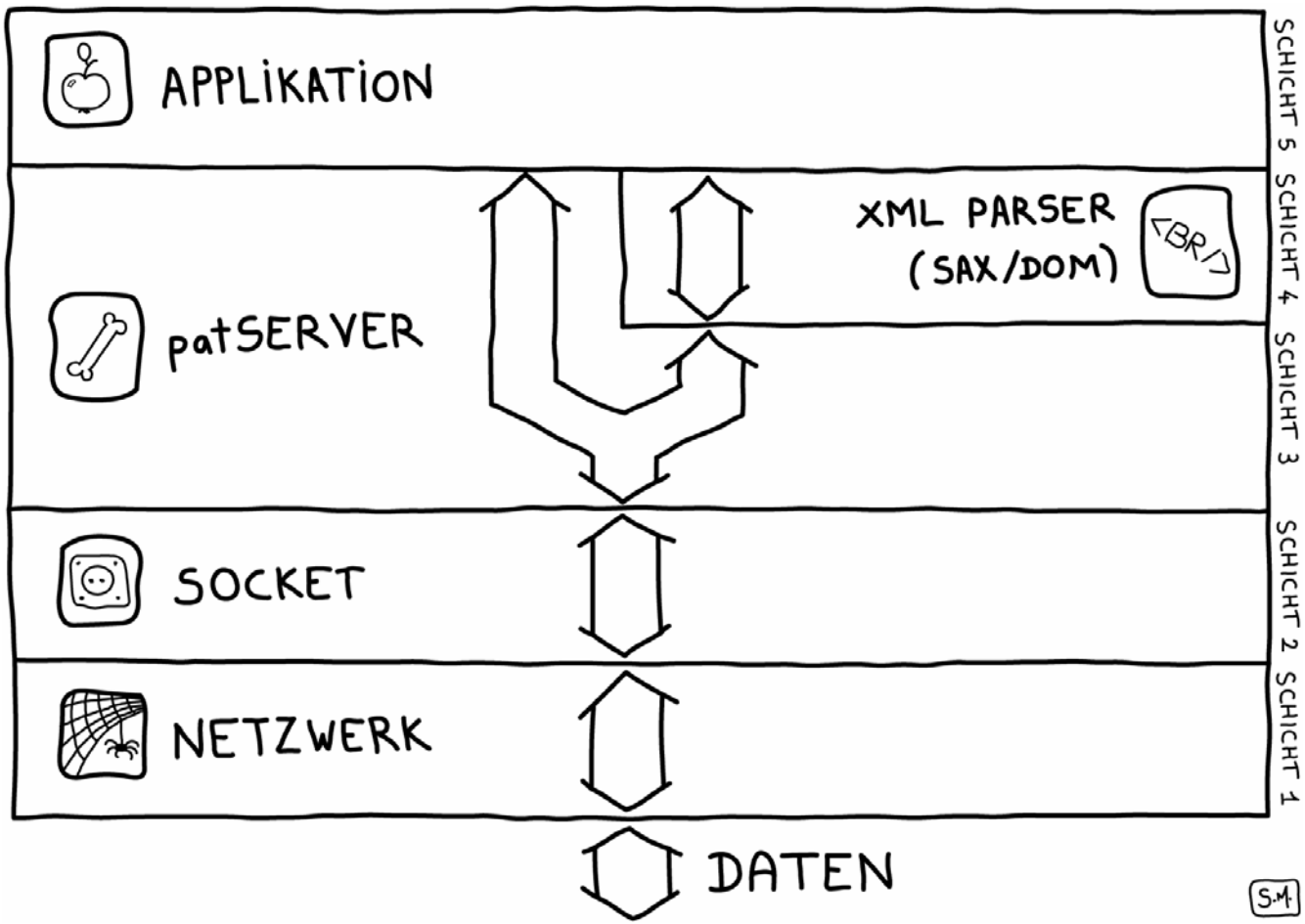
```
<response>
  <type>Typ der Antwort</type>
  <param>....</param>
  <!-beliebig viele Parameter-->
</response>
```

```
<fault>
  <type>Fehlertyp</type>
  <param>....</param>
  <!-beliebig viele Parameter-->
</fault>
```

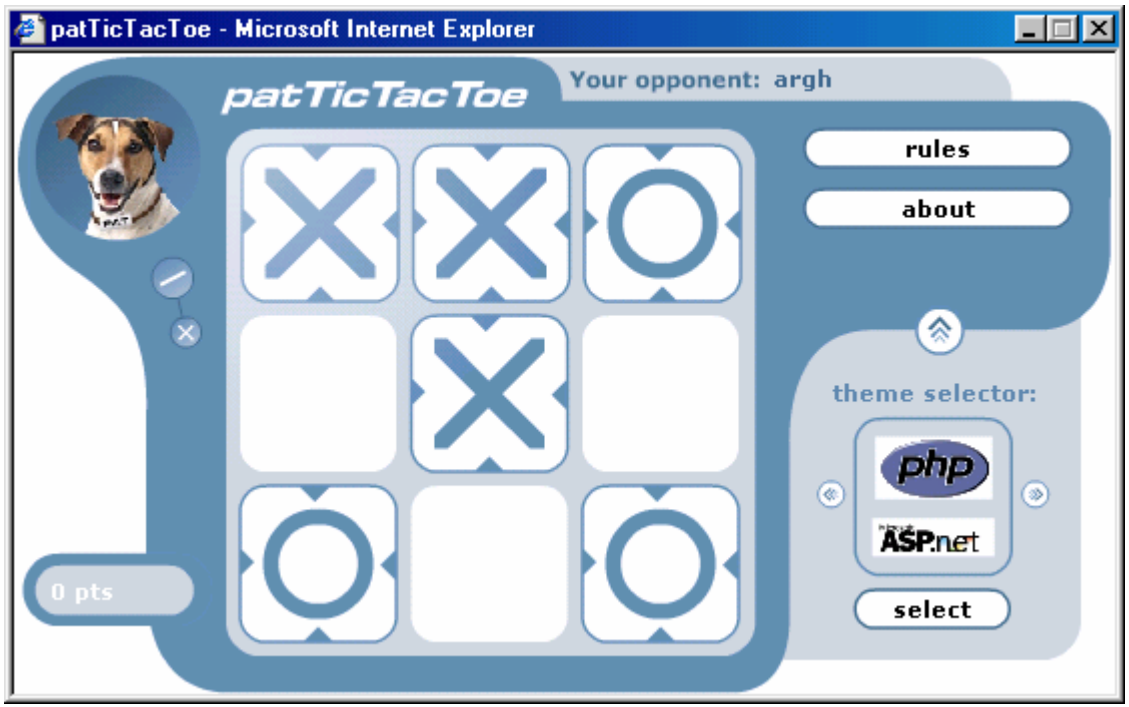
Architektur des Servers

- patServer Distribution enthält (bald) Basisklasse für XML Server zu Kommunikation mit Flash
- Nachrichten werden automatisch deserialisiert
  » Neuer Handler: onReceiveRequest()
- Ausgehende Nachrichten werden automatisch serialisiert und mit einem Nullbyte terminiert
  » Neue Methode: sendResponse()
- Momentan nur mit DOM, bald auch SAX Unterstützung

Schichtenmodell



Beispiel: patTicTacToe



Beispiel: patTicTacToe

- Beliebig viele Spieler über einen Server
- Für zwei "freie" Spieler wird ein Spiel gestartet
- Komplette Applikationslogik im Server
- Clients übernehmen Präsentationslogik
- Clients nehmen Usereingaben entgegen
- Logik auf Client und Server ist trivial

Protokollausschnitt

```
00:30:36 New connection ( 7 ) from 192.168.1.113 on port 1035

00:30:36 Received <request><name>schst</name>
                  <type>signon</type></request> from 7

00:30:36 sending: "<response><type>updateScore</type>
                   <score>0</score></response>" to: 7

00:30:36 Received <request><type>startGame</type></request>
                  from 7

00:30:39 sending: "<response><type>startGame</type>
                   <gameid>3</gameid><item>circle</item>
                   <opponent>argh</opponent></response>" to: 7

00:30:39 sending: "<response><type>makeMove</type></response>" to: 7

00:35:32 Received <request><xpos>2</xpos><ypos>0</ypos>
                  <type>makeMove</type></request> from 7

00:35:32 sending: "<response><type>updateField</type>
                   <xpos>2</xpos><ypos>0</ypos>
                   <item>circle</item></response>" to: 1

00:35:32 sending: "<response><type>makeMove</type></response>" to: 1
```

Fazit

- Nahezu unendliche Möglichkeiten
  - » Chats
  - » Virtuelle Welten
  - » Multiplayer-Games
  - » eCommerce Anwenungen
  - » Online-Beratung
- In PHP einfach zu realisieren
- Flash kann HTML in manchen Bereichen ablösen, allerdings nicht überall
  - » Zielgruppe ist wichtig

Ende

Viele Dank für Ihre Aufmerksamkeit.

Weitere Informationen und Folien:

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

Dank an:

Sebastian Mordziol, Stephan Eisler,
Metrix Internet Design GmbH und dem wunderbaren Publikum