Thrift PHP Server

== Einrichtung ==
Wir gehen davon aus, dass Apache Thrift als Voraussetzung installiert ist. Sollte es noch nicht installiert sein, finden sich hier Anleitungen zur Einrichtung von Thrift und der PHP Extension.

- [wiki]Apache Thrift Installation[/wiki]
- [wiki]Thrift PHP Extension installieren[/wiki]

Wir verwenden die aktuellen Quellen aus dem Thrift Trunk, falls einige Klassen noch bei euch fehlen, dann gleicht diese bitte mit dem Beispiel Download am Ende dieser Anleitung ab.

== Service Definition ==
Zu Beginn brauchen müssen wir einen Thrift Service definieren. Wir werden in diesem Beispiel die Thrift Definition von Flume verwenden.
Der Flume Client ist ein normaler Thrift Client mit, dessen Schema im Thrift Format beschrieben wurde.
Ihr könnt also auch jeden anderen Thrift Service implementieren.

Quellcode

```
1. struct ThriftFlumeEvent {
2. 1: Timestamp timestamp,
3. 2: Priority priority,
4. 3: binary body,
5. 4: i64 nanos,
6. 5: string host,
7. 6: map<string,binary> fields
8. }
9. service ThriftFlumeEventServer {
10. oneway void append( 1:ThriftFlumeEvent evt ),
11. void close(),
12. }
```

Alles anzeigen

== Code generieren ==
Mit dem Thrift Compiler erzeugen wir nun Client- und Serverdateien und verschieben diese anschließend in das packages Verzeichnis.

Quellcode

```
1. thrift --gen php:server flume.thrift
2. mv gen-php/ thrift/packages
```

== Implementierung ==
Nun kommen wir zur Implementierung des Servercodes. Wir haben uns im Beispiel für den **TForkingServer** Server entschieden.
Ansonsten müssen wir nur das Interface implementieren. Wir werden den Server als Beispiel in der Konsole laufen lassen und die Ausgaben direkt mit Ausgaben monitorn können.

Wir haben den Server so implementiert, dass die angegebene Kategorie im Feld genutzt wird um die Methode zu bestimmen.

Quellcode

```
<?php
$GLOBALS['THRIFT_ROOT'] = './thrift';
require_once $GLOBALS['THRIFT_ROOT'] . '/Thrift.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/transport/TSocket.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/transport/TServerSocket.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/transport/TTransportFactory.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/protocol/TBinaryProtocol.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/server/TForkingServer.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/packages/flume/flume_types.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/packages/flume/ThriftFlumeEventServer.php';
/**
* thrift server to handle flume events
* @author Torben Brodt <www.easy-coding.de>
*/
class FlumeHandler implements ThriftFlumeEventServerIf {
/**
* public api call
* @implements ThriftFlumeEventServerIf
*/
public function append($evt) {
$cat = $evt->fields['cat'];
if(method_exists($this, $cat)) {
$this->$cat($evt);
} else {
echo 'unknown method: '.$cat;
}
}
/**
* public api call
* @implements ThriftFlumeEventServerIf
*/
public function close() {
echo "close";
}
/**
* for debug purpose
*/
protected function ping($evt) {
print_r($evt);
}
}
$handler = new FlumeHandler();
$processor = new ThriftFlumeEventServerProcessor($handler);
// will run server on port 9090
$transport = new TServerSocket();
$outputTransportFactory = $inputTransportFactory = new TTransportFactory($transport);
$outputProtocolFactory = $inputProtocolFactory = new TBinaryProtocolFactory();
$server = new TForkingServer(
$processor,
$transport,
$inputTransportFactory,
$outputTransportFactory,
$inputProtocolFactory,
$outputProtocolFactory
);
header('Content-Type: application/x-thrift');
```

```
67. $server->serve();
```

Alles anzeigen

== Beispiel: Thrift zu Thrift ==
Wir starten den Server nun mit

Quellcode

```
1. php -f server.php
```

Den Sender aus dem Tutorial "[wiki]Flume mit PHP[/wiki]" lassen wir direkt auf unser Thrift Service zeigen.
Dazu instanziieren wir den Socket mit Port 9090 und senden "ping" als Kategorie.

Quellcode

```
1. $transport = new TSocket($host = 'localhost', $port = '9091');
2. ...
3. $client->append(...
4. 'fields' => array(
5. 'cat' => 'ping'
6. )
7. );
```

Sobald wir nun den Sender mit php -f sender.php aufrufen können wir in der Ausgabe des Servers sehen, dass der Event ankommt.

== Beispiel: Thrift über Flume zu Thrift ==
Wir werden die Ausgaben nun in einer lokalen Anwendung puffern und darüber zu unserem Thrift Service übertragen.
Dazu benötigen wir ein Flume Setup wie es hier erläutert wird: [wiki]Cloudera Flume Installation[/wiki]

Wir werden den Node über den Flume Master (localhost:35871/) wie folgt konfigurieren. Als Flume Sink verwenden wir den Server time.easy-coding.de wo der Flume Service läuft.
* rpcSource(9091)
* rpcSink("time.easy-coding.de", 9090)

Wir binden den Flume Node damit an einen lokalen Port 9091, der die Anfragen dann wiederum an den Remote Server weiterleitet.

== Beispiel: Strukturierte Daten von Thrift über Flume zu Thrift ==
Um strukturierte Daten auszutauschen erstellen wir uns ein eigenes Thrift Schema "vector.thrift".

Quellcode

```
1. namespace java easycoding.thrift
2. typedef i64 id
3. enum VectorTarget {
4. THREAD = 0,
5. WIKI = 1,
6. BOARD = 2,
7. BLOG = 3
8. }
9. enum Vector {
10. BROWSER = 0,
11. OS = 1,
12. GEO_REGION = 2
13. }
```

```
17. struct VectorSequence {
18. 1: map<VectorTarget, id> targets,
19. 2: map<Vector, string> env,
20. 3: string query
21. }
```

Alles anzeigen

Wir erzeugen die PHP Dateien und kopieren die Klassen in den packages Ordner

Quellcode

```
1. thrift --gen php vector.thrift
2. mv gen-php/ thrift/packages
```

Nun können wir mit folgendem Code das Event übertragen:

Quellcode

```
<?php
$GLOBALS['THRIFT_ROOT'] = './thrift';
require_once $GLOBALS['THRIFT_ROOT'] . '/Thrift.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/transport/TSocket.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/protocol/TBinaryProtocol.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/packages/flume/ThriftFlumeEventServer.php';
$transport = new TSocket($host = 'localhost', $port = '9091');
$protocol = new TBinaryProtocolAccelerated($transport);
$client = new ThriftFlumeEventServerClient($protocol, $protocol);
$transport->open();
$timestamp = intval(microtime(true) * 1000);
// log structed message
require_once $GLOBALS['THRIFT_ROOT'] . '/protocol/TBinarySerializer.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/packages/vector/vector_types.php';
$vector = new VectorSequence();
$vector->query = 'thrift server example';
$vector->targets = array(
VectorTarget::THREAD => 100,
VectorTarget::BOARD => 10
);
$vector->env = array(
Vector::BROWSER => 'Firefox 3.5',
Vector::OS => 'Ubuntu 10.10',
Vector::GEO_REGION => 'de-16', // for berlin
);
$body = TBinarySerializer::serialize($vector);
$client->append(new ThriftFlumeEvent(array(
'priority' => Priority::INFO,
'timestamp' => $timestamp,
'body' => $body,
'host' => 'server1',
'fields' => array(
'cat' => 'impression'
)
)));
$transport->close();
```

Alles anzeigen

Im Thrift Server Code fügen wir nun die Methode Impression ein, damit wir das Objekt nutzen können:

Quellcode

```
/**
* unserialize thrift object
*/
protected function impression($evt) {
require_once $GLOBALS['THRIFT_ROOT'] . '/packages/vector/vector_types.php';
require_once $GLOBALS['THRIFT_ROOT'] . '/protocol/TBinarySerializer.php';
$instance = TBinarySerializer::deserialize($evt->body, 'VectorSequence');
print_r($instance);
}
```

easy-coding.de/Attachment/1133…9763049e8bd278ce33210c7c2
== Download ==
Ihr könnt euch den gesamten Quelltext der Beispiele hier herunterladen: