

## The PHP GraphDB Client

- Introduction
- Feature List
- First Steps
  - Source of supply
  - Embed into projects
  - Usage
- Walkthrough the DemoExample
  - The Demoproject
  - The visualization of the “QueryResult”
- The code documentation
- Get involved at github

## Introduction

This document describes the PHP Client for the sones GraphDB database.

There are already a big number of communication interfaces to sones GraphDB. Now there is a new interface available: The PHP GraphDB Client. This library gives you the ability to establish a connection to the database, send GQL-Queries and parse the result. It is very easy to use and to include in existing projects.

Everyone who wants to use the GraphDB as data basis in PHP-Applications, needs a suitable data transmission interface between the visualization layer and the persistence layer. This is precisely purpose of our PHP GraphDB Client. It performs all tasks to deliver and parse the graph out of the database.

The Client uses the already known REST Port of the data storage. To set up a connection, there is only the host and the credentials necessary. In this way it is possible to create any numbers of clients to talk to any numbers of GraphDB instances.

Every GQL-Query gets a result. The PHP GraphDB Client is able to parse the “QueryResult” and represents the graph in vertex- and edge-objects. So it is very easy to gather the requested information and get direct access.

Beyond this, the Client is part of the Open Source Edition (OSE) of our GraphDB and you can take a look at the operation of the library and add or manipulate functionality.

## Feature List

There are a lot of features already implemented, these include the REST Client which handles the communication to the service and the provision of the credentials. Furthermore there is the result parser, which iterates over the result document and returns known objects to display a propertygraph in a PHP environment. Additionally, there are the features to handle some metadata about the request process and the actual result.

The initial release also contains some missing features, like traverse methods or anonymous functions. Probably these functions will be added in later releases.

Here a short summary of features and missing functionality:

### Features:

- possibility to easily send GQL-Queries to the service
- parsing methods to create a QueryResult out of the XML-Response
- API to handle vertices, edges and some result meta data
- take one client to one instance of sones GraphDB Service
- simple connection to a GraphDB REST service based on given URI and credentials

### Missing functionality:

#### General

- BasicTypes of C# are not supported, because of PHP

#### IVertex

- anonymous functions
- traverse methods
- link/unlink

#### GraphDSCClient

- QueryResultActions
- ObjectRevisionID
- The pattern doesn't exactly parse the timestamp (GraphDB uses C# DateTime.ParseExact()-Method with 7 most significant digits of seconds fraction)

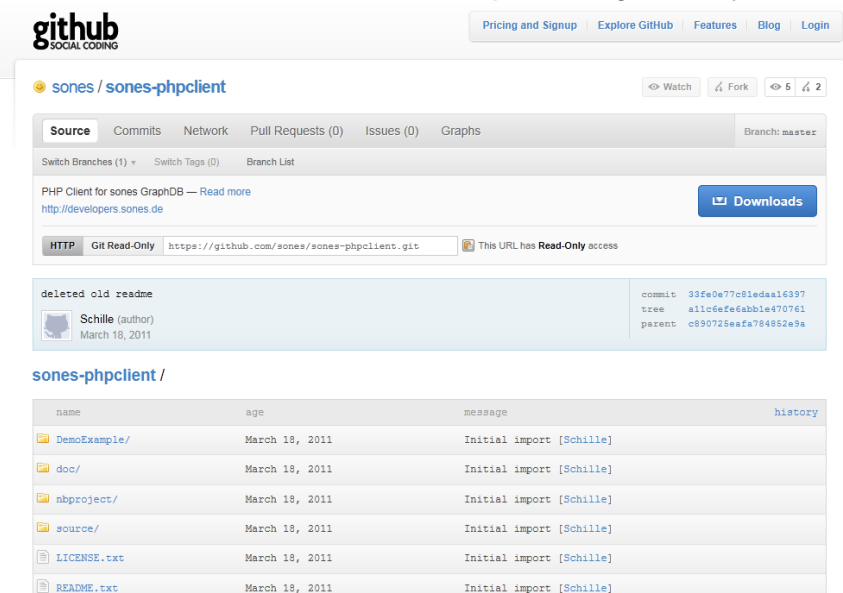
There are still some Warnings, caused by the "Include\_once..." in the Library.

## First Steps

### Source of supply

Since the PHP GraphDB Client was released, it is reachable at "github". You can download it for free at: [www.github.com/sones/sones-phpclient](https://www.github.com/sones/sones-phpclient). Just click on "Download" and get the source. You can choose between \*.zip or \*.tar.gz. After you extracted the code, you are able

to use the client.









deleted old readme

Schille (author)  
March 18, 2011

commit 33fe0e77c81edaa16397  
tree a11c6efe6abb1e470761  
parent c890725eafa784852e9a

sones-phpclient /

name	age	message	history
 DemoExample/	March 18, 2011	Initial import [Schille]	
 doc/	March 18, 2011	Initial import [Schille]	
 nbproject/	March 18, 2011	Initial import [Schille]	
 source/	March 18, 2011	Initial import [Schille]	
 LICENSE.txt	March 18, 2011	Initial import [Schille]	
 README.txt	March 18, 2011	Initial import [Schille]	

### Embed into projects

The only folder you really need is the "source".

There are all important files located. Just copy the source folder into your project and start using the client.

To use the connector in your source, you simply have to include

“GraphDBClient.php”, creating a “include” statement:

```
<?php
```

```
include_once "../source/sones/GraphDBClient.php";
```

```
?>
```

**Important:** Before the first test, make sure there is a GraphDB instance running. Without running a database it is impossible to reach a query result, of course.  
Let's shift to the usage.

## Usage

Well, you got and included the source. Now it's time to look at the usage.

Since you included the GraphDBClient.php it is possible to create an instance – object of it. Just type the following line:

```
<?php
```

```
$myGraphClient = new GraphDBClient($myHost, $myUsername, $myPassword, $myPort);
```

```
?>
```

Now “\$myGraphClient” is an instance of GraphDBClient. The parameterlist is:

- *\$myHost* = the Host, where the GraphDB is running. The address could be a name or ipaddress. If you are running the OSE(v 1.1) of GraphDB, you can use “localhost”.
- *\$myUsername* = the username for the login. This is the same like in the WebShell. In the OSE(v 1.1) of GraphDB it is “test”.
- *\$myPassword* = the password for the login. This is the same like in the WebShell. In the OSE(v 1.1) of GraphDB it is “test”.
- *\$myPort* = the port number. This is a standard parameter with 9975.

The host and credentials will be stored in the object, to reuse the connection configuration. Of course, you create for each GraphDB one instance, to have a representation of the existing systems.

Now, let's fire some GQL-Statements. It works like this:

```
<?php
```

```
$myQueryResult = $myGraphClient->InsertQuery("CREATE VERTEX Car");
```

```
?>
```

In this Way, you can talk to the database, using InsertQuery(\$myGQLQuery);

The return value is an instance of “QueryResult.php”. This query result contains all information about the last query. It is really useful to evaluate the query status, whether it was successful or faulty! The most important public methods of QueryResult are:

- getVertices() – returns an array of vertices from the result
- getErrors() – returns all occurred errors, empty if none occurred
- getWarnings() – returns all occurred warnings, empty if none occurred
- getDuration() – returns the duration time
- getQueryString() – returns the query, which was executed

There are some other functions to evaluate the result.

As you have seen, it is really easy to use the PHP GraphDB Client. For visualization examples, take a glance at the DemoExample.

## Walkthrough the DemoExample

### The Demoproject

The Demoproject consists of 6 PHP-Webpages. It is constructed like a Wizard. The DemoExample uses the OSE of the GraphDB, because of that the parameter are already set. The standard port number is the 9975 (set to standard because of the OSE). The DemoExample shows the regular flow in a database application.

First of all, there is an input form for host and credential information. The second step is, to insert some information into the GraphDB.

**Important:** If you want to use the DemoExample you have to set "max\_execution\_time" to false (max\_execution\_time = false) in the php.ini. The progress of storing data takes a while and overstepped my maximum execution time and ended with an error.

In the following steps are several statements to interact with the database. . The output of the DemoExample is splitted in two exhibitions. On the one hand is the parsed output of the database, and on the other the source code, which was executed. There is just one additional class which generates the visualized output of the GraphDB.

The DemoExample stores the PHP client in the SESSION variable. This is useful, because you don't need to create always a new client:

```
<?php
```

```
$myGraphClient = new GraphDBClient($_SESSION["Host"], $_SESSION["Username"],$_SESSION["Password"]);
```

```
$_SESSION["Client"] = serialize($myGraphClient);
```

```
?>
```

### The visualization of the "QueryResult"

The DemoExample contains a class named "DemoExample". This class has two functions:

- printQueryResult(\$myResult) – this function takes the "QueryResult" object and generates a simple output
- printVertex(\$myVertex, \$depth) – this function takes a vertex object and a specified depth. The depth is a factor to represent the eisthesis

This two functions handles the simple output in the DemoExample, just bei adding \$demo as object:

```
<?php
```

```
    $result = $myGraphClient->InsertQuery("From User SELECT UUID, *");
```

```
    $demo = new DemoExample();
```

```
    $demo->printQueryResult($result);
```

```
?>
```

## **The code documentation**

The code documentation is located in the doc folder. It was generated with the “phpDocumentor” and contains a simple code documentation. Just open the index.php and you’re able to browse through the whole code documentation.

## **Get involved at github**

The complete OSE (Open Source Edition) of our sones GraphDB is provided at: <http://www.github.com/sones>. At this location, there are the database and some other projects available.