

The PHP GraphDB(v2.0) 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(v2.0) database.

With the new GraphDB in version 2.0 are a handful clients available. This includes the completely new created 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 to pars the graph out of the database in a query result.

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 an 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 Community Edition of our GraphDB(v2.0) and you can take glance 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 enviroment. Additional, there are the feature to handle some metadata about the request process and the actual result.

Here a short summary of the features:

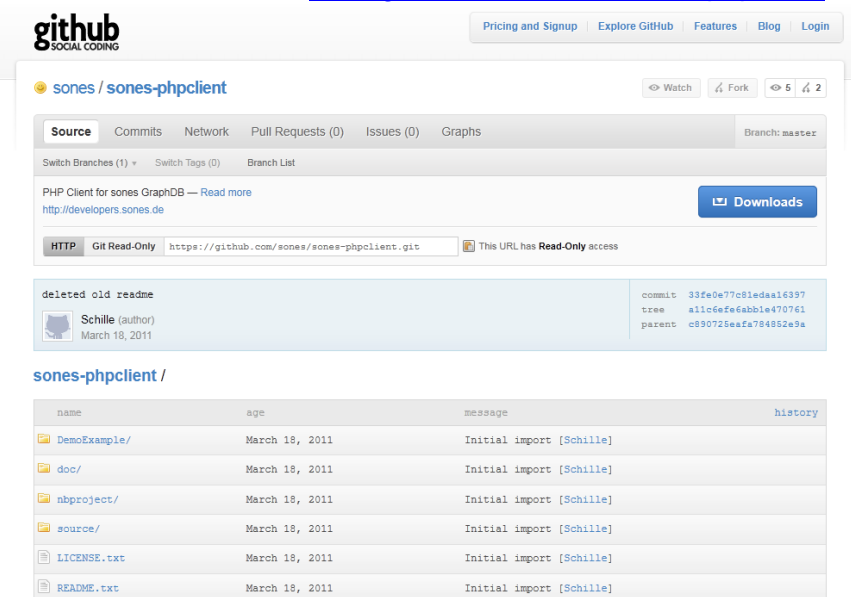
Features:

- possibility to easily send GQL-Queries to the service
- parsing methods to create a QueryResult out of the XML-Response
- API to handle vertices, single-edges and hyper-edges
- one client instance to one instance of sones GraphDB(v2.0) service
- simple connection to a GraphDB(v2.0) REST service based on given URI and credentials
- a validation against the QueryResultSchema.xsd
- full featured functionality about handling the query result

First Steps

Source of supply

Since the PHP GraphDB(v2.0) Client was released, it is reachable at “github”. You can download it for free at: [www.github.com/sones/sones-phpclient](https://github.com/sones/sones-phpclient). The “master” branch is



The screenshot shows the GitHub repository page for `sones/sones-phpclient`. The repository is in the `master` branch. Below the repository name, there is a 'Downloads' button. The commit history table shows the following data:

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

relevant. 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.

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

```
<?php
```

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

```
?>
```

Important: Before the first test, make sure there is a GraphDB(v2.0) 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 Community Edition of GraphDB, you can use “localhost”.
- *\$myUsername* = the username for the login. This is the same like in the WebShell. In the Community Edition of GraphDB it is “test”.
- *\$myPassword* = the password for the login. This is the same like in the WebShell. In the Community Edition 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->Query("CREATE VERTEX TYPE 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:

- `getVertexViewList()` – returns an array of all vertices from the result
- `getErrorMessage()` – returns all occurred errors, empty if none occurred
- `getDuration()` – returns the duration time
- `getQueryString()` – returns the query, which was executed
- `getResultType()` – returns the type of the query (currently Successful/ Failed)

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 Demo.

Walkthrough the DemoExample

The Demoproject

The Demoproject consists of just one PHP-Webpages. The Demo uses the Community Edition of the GraphDB(v2.0), because of that the parameter are already set. The standard port number is the 9975 (set to standard because of the Community Edition).

The Demo shows the regular flow in a database application.

First of all, there is an input form for the query string. May you want to insert some information into the GraphDB.

In the following, you are able to interact with the database. The output shows just the properties of the current vertex including the single-/ and hyper - edges.

The visualization of the “QueryResult”

The output is just generated by `var_dump()` through a iteration over the returned vertices of your query.



PHP - GraphDB(v2.0) Client Example

[\(c\) sones GmbH](#)

Just enter any GQL Statement below - for help click [sones Cheatsheet](#)

Properties of Vertex 0:

```
object(Property) [17]
  private 'id' => string 'UUID' (length=4)
  private 'type' => string 'Int64' (length=5)
  private 'value' => int -2147483648
```

```
object(Property) [16]
  private 'id' => string 'TYPE' (length=4)
  private 'type' => string 'RuntimeType' (length=11)
  private 'value' => string 'sones.GraphDB.TypeManagement.VertexType' (length=39)
```

```
object(Property) [18]
  private 'id' => string 'Name' (length=4)
  private 'type' => string 'String' (length=6)
  private 'value' => string 'BaseType' (length=8)
```

```
object(Property) [19]
  private 'id' => string 'Comment' (length=7)
  private 'type' => string 'String' (length=6)
  private 'value' => string 'BaseTypeComment' (length=15)
```

Properties of Vertex 1:

```
object(Property) [22]
  private 'id' => string 'UUID' (length=4)
  private 'type' => string 'Int64' (length=5)
  private 'value' => int -2147483648
```

```
object(Property) [21]
  private 'id' => string 'TYPE' (length=4)
  private 'type' => string 'RuntimeType' (length=11)
  private 'value' => string 'sones.GraphDB.TypeManagement.VertexType' (length=39)
```

```
object(Property) [23]
  private 'id' => string 'Name' (length=4)
  private 'type' => string 'String' (length=6)
  private 'value' => string 'VertexType' (length=10)
```

```
object(Property) [24]
  private 'id' => string 'Comment' (length=7)
  private 'type' => string 'String' (length=6)
  private 'value' => string 'VertexTypeComment' (length=17)
```

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 Community Edition of our sones GraphDB(v2.0) is provided at: <http://www.github.com/sones>. At this location, there are the database and some other projects available.