JaphService Dokumentation

31. Okt. 2008, 13:46

Inhaltsverzeichnis

|| Zur JaphService Hauptseite ||

Dies ist die Dokumentation zum JaphService

in ständiger Bearbeitung!

Stand: 27.10.2008
Autoren: Tim Wahrendorff
--Tim Wahrendorff (HIS) 16:55, 28. Okt. 2008 (CET)

1 Introduction

JaphService is a PHP:SOAP based Webservices application, allowing users to do several things with Limesurvey from an other application. Start predefined surveys with varying titles, welcometexts, Admins and Endurls. Insert token, add predefined groups to your coresurvey etc. More features to come.

Now it is possible to bind Basic LimeSurvey Controlls into your webshop or other applications e.g. to evaluate customer satisfaction or else.

Be aware that this piece of code is written to bind Limesurvey into University software my Company is coding. So there could be comments in the code or documentation, which have to do with my Companys Software or which are just a little bit confusing when you have nothing to do with Universitys.

Development of JaphService has just begun, so post any comments, wishes or bugs to "wahrendorff@his.de" or to the LimeSurveyCommunity.

1.1 Files included in JaphService

- japh.config.php (configuration variables)
- japh.server.php (main SOAP:Server functions)
- japh.helper.php (class japhHelper, class with some functions, making japh work.)
- japh.wsdl (wsdl definition for SOAP messages)
- japh.testclient.php (to test the enviroment and functionality on the fly)

These files are deleted, because the class japhHelper does everything and more than they did.

- ~~japh.function.php~~
- ~~japh.importsurvey.php (based on: importsurvey.php 4454 2008-03-13 23:34:58Z c_schmitz)~~
- ~~japh.activate.php (based on: activate.php 4374 2008-02-29 00:39:04Z c_schmitz)~~

1.2 Directorys included

- \studiply_mod (for the exported modules, name groups with more than one question "mod_*desiredname*.csv") and groups with one question anything than "mod_" in the beginning.
- \studiply (for the core surveys .csv's)

1.3 Requirements

- LimeSurvey > v1.70 recommended
- PHP >= 5.0.0 with PHP:SOAP Extension enabled
- libXML installed
- MySQL, PostgreSQL or MSSQL

- (optional) OpenSSL
- (optional) a Certificate for better Security

2 Installation

2.1 How to install JaphService

- create a Dir "japh" in your limesurvey admin dir.
- copy the included files to the new japh directory.

2.2 How to configure JaphService

- In the japh.config.php file, you can set all needed Variables for the JaphService in Section Variables.
  - ~~set the $relativeurl to your limesurvey installation server.~~
  - set the dirs, where the .csv files to import can be found. You will have to export some survey you want to import and activate with JaphService first.
  - (optional) change $sJaphSeparator = "[your separator]"; in japh.config.php if you wish to separate tokens for the sInsertToken function with any other separator than default, which is a comma.
  - (optional) change $sDatasetSeperator= "[your separator]"; in japh.config.php if you wish to separate Datasets for the sInsertParticipants function with any other separator than default, which are two colons.
  - (optional) change $sDatafieldSeperator= "[your separator]"; in japh.config.php if you wish to separate Datafields for the sInsertParticipants function with any other separator than default, which is a semicolon.

- You will need to modify the last block in japh.wsdl

  ```
  <soap:address location="http://127.0.0.1/limesurvey/admin/japh.server.php" />
  ```

  to fit your installation of LimeSurvey. CAUTION: for testing on one machine localhost or 127.0.0.1 is ok, for real use, this has to be the IP or uri of the webserver where the (japh.server.php) file is located. HINT: for testing on two machines: Imaging how your proxy works. In some environments you will get 404, because of proxy configuration.

2.3 How to enable OpenSSL / https Support

- JaphService supports SSL over http resp. https.
- Your Webserver needs to support and enable OpenSSL (See your Webserver Manual)
- There are two locations in the code, where you *force the webservice to use https*

  1. On the client side, when you call the SOAP-Server over wsdl. For **https** do it like that:

```
new SoapClient("https://localhost/limesurvey/admin/japh.wsdl", array("soap_version" => SOAP_1_1,
                "trace" => 1));
```

  for **http** like that:

```
new SoapClient("http://localhost/limesurvey/admin/japh.wsdl", array("soap_version" => SOAP_1_1,
                "trace" => 1));
```

Note that the client will only get the wsdl-definition over https, to get the SOAP Messages over https, set the uri in the wsdl file, like below...

2. Here is the most important Step to SSL. At the End of the japh.wsdl file, last block, same procedure:

```
<port name="JaphPort" binding="tns:JaphBinding">
  <soap:address location="https://localhost/limesurvey/admin/japh.server.php" />
</port>
```

or for normal http mode:

```
<port name="JaphPort" binding="tns:JaphBinding">
  <soap:address location="http://localhost/limesurvey/admin/japh.server.php" />
</port>
```

~~3. (Appendix:) In **japh.server.php** we "header" japh.function.php. On some Apache-installations, header elements must have a absolute URI, first we only had a relative URI. So by now you have to change the URI in japh.server.php ($headerURL) from http://... to https://... , too. (**Line 71**)~~

~~`$headerURL="https://".$_SERVER['HTTP_HOST'].$relativeurl."/admin/";`~~

That's it! all request, response and fault messages will be transported over https...

3 How to use JaphService

The JaphService offers some functions:

(m) means: Mandatory Parameter

3.1 sCreateSurvey

- String sCreateSurvey($sUser, $sPass, $iVid, $sVtit , $sVbes, $sVwel, $sMail, $sName, $sUrl, $sUbes, $sVtyp)
  - **Purpose**: Creates a new survey. It imports an exported survey from any LS v1.xx. and changes some fields (in the current Version it changes SurveyTitle, SurveyDescription and SurveyWelcomeText, adminemail, adminname, url and url_description to the string which was given by the client.)
  - **Parameters**:
    - (m) string USER, this have to be an existing Useraccount with appropriate rights
    - (m) string PASSWORD, this has to match the password for the account.
    - (m) int $iVid, SurveyID desired ID for the Survey, make sure it doesn't already exist in LimeSurvey
    - (m) string $sVtit, SurveyTitle, should be the (Nr +)Name of the Event to be evaluated
    - (m) string $sVbes, Survey Description, This should be String put together with veransttyp + doztitle + dozname
    - string $sVwel, Survey Welcometext, A text shown before the survey starts. If not given it says: "Herzlich Willkommen zur Evaluation von "$sVtit"
    - string $sMail, Admin Email, Email Adress of the responsible person for this survey
    - string $sName, Name of the responsible person for this survey
    - string $sUrl, Url of the Link, shown on the last page of survey.
    - string $sUbes, Description String of the URL above. ($sUbes)
    - string $sVtyp, The type of Event that should be evaluated. Use in combination with sAvailableModules with $mode='core' to get all available "types"
  - **Returns**:

        - if everything went fine, it returns the SurveyID from the created survey
    - **Possible Faults**:
        - throw new SoapFault("Authentication: ", "User or password wrong");
        - throw new SoapFault("Database: ", "SurveyID already exists");
        - throw new SoapFault("Server: ", "Mandatory Parameters missing");
        - throw new SoapFault("Server: ", "Import went wrong somehow");
    - **Note**:
        - Goto line 714 in japh.helper.php to see what can be done and customized while importing.
        - If you want Limesurvey to give your Survey a random Number if the given ID exist (because you do not need a specific identification for your surveys), just go into the function sCreateSurvey in japh.server.php and delete or comment out the if($japhHelper->surveyExists($iVid)) clause. In Result, Limesurvey will give your survey a random ID when the given ID exists.

3.2 sActivateSurvey

- String **sActivateSurvey**(string USER, string PASSWORD, int SurveyID, date start, date end)
    - **Purpose**: Activates an existing survey and set start and enddate optional
    - **Parameters**:
        - (m) string USER, this have to be an existing Useraccount with appropriate rights
        - (m) string PASSWORD, this has to match the password for the account.
        - (m) int SurveyID, the SurveyID for the Survey to activate.
        - date start, date for the Survey to start. 2008-10-22 means it starts exactly at 2008-10-22 00:00:00 Servertime
        - date end, date for the Survey to end. 2008-10-22 means it ends exactly at 2008-10-22 23:59:59 Servertime. Nobody can access nor complete the Survey only a second later.
    - **Returns**:
        - if everything went fine, it returns the SurveyID from the activated survey
    - **Possible Faults**:
        - throw new SoapFault("Authentication: ", "User or password wrong");
        - throw new SoapFault("Database: ", "Survey you want to activate does not exists");
        - throw new SoapFault("Server: ", "Mandatory Parameters missing");
        - throw new SoapFault("Server: ", "Activation went wrong somehow");
    - **Note**:
        - You can set startDate and endDate after activation if needed. Just run the function again with desired parameters.

3.3 sImportGroup

- String **sImportGroup**(string USER, string PASSWORD, int SurveyID, String Module)
    - **Purpose**: Imports an exported group file (.csv) to the Survey given by SurveyID
    - **Parameters**:
        - (m) string USER, this have to be an existing Useraccount with appropriate rights
        - (m) string PASSWORD, this has to match the password for the account.
        - (m) int SurveyID, the SurveyID
        - string Module, the name of the Module you want to add to the Survey. Use sAvailableModules to get the Modules available.
    - **Returns**:
        - if everything went fine, it returns the String "Import OK"
    - **Possible Faults**:
        - throw new SoapFault("Authentication: ", "User or password wrong");

        - throw new SoapFault("Server: ", "Survey Module $sMod does not exist");
        - throw new SoapFault("Server: ", "Group does not support Surveys Baselanguage ($langcode)");
    - **Note**:
        - It imports only exported groups as .csv which lay under \studiply_mod or any other dir given in japh.config.php
        - It imports only the files beginning with "mod_" ending with ".csv"
        - The Param Module must only contain the "name of the Module": the String between "mod_" and ".csv"
        - The baselanguage of the group to import, have to match the baselanguage of the coresurvey.

3.4 sImportQuestion

- String **sImportGroup**(string USER, string PASSWORD, int SurveyID, String Module, String questionTitle, String questionText, String questionHelp)
    - **Purpose**: Imports an exported group file (.csv) to the Survey given by SurveyID and changes the Title, question and helptext of the last Question. This is meant for importing groups with just one question (Longtext Questions)
    - **Parameters**:
        - (m) string USER, this have to be an existing Useraccount with appropriate rights
        - (m) string PASSWORD, this has to match the password for the account.
        - (m) int SurveyID, the SurveyID
        - String Module, the name of the Module you want to add to the Survey. It should be "Freitext" by default.
        - String questionTitle, Name or Categorie of the Question
        - String questionText, The Question itself
        - String questionHelp, A littel Help text for better understanding of the Question
    - **Returns**:
        - if everything went fine, it returns the String "OK"
    - **Possible Faults**:
        - throw new SoapFault("Authentication: ", "User or password wrong");
        - throw new SoapFault("Server: ", "Group does not support Surveys Baselanguage ($langcode)");
    - **Note**:
        - It imports only exported groups as .csv which lay under \studiply_mod or any dir given in japh.config.php
        - It is meant for importing a group with only one Longtext Question and changing the Question and Helptext.
        - The baselanguage of the group to import, have to match the baselanguage of the coresurvey.
        - Be sure to use an user with the surveys baselanguage as default language, because this language is used to fill the language fields for answers and questions while importing.

3.5 sImportMatrix

- String **sImportGroup**(string USER, string PASSWORD, int SurveyID, String Module, String questionText, String questionHelp, String questionItems)
    - **Purpose**: Imports an exported group file (.csv) to the Survey given by SurveyID and changes the question, helptext and gives Items to the 5 Scale Matrix. This is meant for importing groups with just one 5 Scale Matrix Question.
    - **Parameters**:

        - (m) string USER, this have to be an existing Useraccount with appropriate rights
        - (m) string PASSWORD, this has to match the password for the account.
        - (m) int SurveyID, the SurveyID
        - string Module, the name of the Module you want to add to the Survey. It should be "Matrix" by default.
        - String questionText, the question or text to the Matrix
        - String questionHelp, the help text
        - String questionItems, the Items that should be rated separated by commas
    - **Returns**:
        - if everything went fine, it returns the String "OK"
    - **Possible Faults**:
        - throw new SoapFault("Authentication: ", "User or password wrong");
        - throw new SoapFault("Server: ", "Group does not support Surveys Baselanguage ($langcode)");
    - **Note**:
        - It imports only exported groups as .csv which lay under docs\demosurveys\studiply_mod
        - It is meant for importing a group with only one 5 Scale Matrix Question and changing the Question and Helptext and add Items.
        - Be sure to use an user with the surveys baselanguage as default, because this language is used to fill the language fields for answers and questions while importing.

3.6 sAvailableModules

- String **sImportGroup**(string USER, string PASSWORD, string mode)
    - **Purpose**: Returning all the modules in /studiply_mod seperated by comma in one String
    - **Parameters**:
        - (m) string USER, this have to be an existing Useraccount with appropriate rights
        - (m) string PASSWORD, this has to match the password for the account.
        - String mode, (optional) "mod" for the available modules and "core" for the coreSurveys. Default, when not given is "mod"
    - **Returns**:
        - if everything went fine, it returns a String with all Mods in /studiply_mod or any other Dir given in japh.config.php instead, separated by comma.
        - in mode "core", it returns a String with all Surveys in /studiply or any other Dir given in japh.config.php instead, separated by comma.
    - **Possible Faults**:
        - throw new SoapFault("Authentication: ", "User or password wrong");
    - **Note**:
        - It only gives back the Name of the files in the $modDir, beginning with mod_. If there would be 4 files in the folder (named: mod_BWL.csv, mod_BIO.xls, freitext.csv, mod_INF.csv) it would give back "BWL" and "INF".
        - In core mode it gives back all .csv files in $coreDir

3.7 sInsertToken

- String **sInsertToken**(string USER, string PASSWORD, int SurveyID, string Token)
    - **Purpose**: Creates and populates a table [dbprefix]token_[SurveyID] and thus "closes" the survey. It is possible to add tokens to the existing table.
    - **Parameters**:
        - (m) int SurveyID, ID of the survey to close with a token table
        - string Token, a string with comma separated tokens. There is a possibility to set any other separator in config.php. Max. tested length of a string was 500,000 tokens, that

makes a String of 5,500,000 Chars, this is just the limit of the apache webserver on my notebook (Core2Duo 1,9MHz, 1GB Ram) with default Resource Limits (60,60,16). I recommend to send max. 100,000 tokens over the webservice. Repeat for more tokens.
    - (m) string USER, this have to be an existing Admin-, Superadmin- or Useraccount with the rights to create and activate Surveys
    - (m) string PASSWORD, this has to match the password for the account.
  - **Returns**:
    - if everything went ok, it returns the String: "[X] Token given, [X] Token inserted"
  - **Possible Faults**:
    - throw new SoapFault("Authentication: ", "User or password wrong");
    - throw new SoapFault("Database: ", "Survey does not exists");
    - throw new SoapFault("Server: ", "Token could not be inserted");
  - **Note**:
    - Make sure on the client side that no token appears twice in the table. There is no check on this and won't be in the future. What happens if there are two same tokens? 2 participants will get the same token, one will fill out the form for both, and the other will get a message: "your token have already participated"
    - If there is no token table, it will create one and add the tokens.
    - If the Tokentable already exists, it just adds the given tokens to the existing table.

3.8 sInsertParticipants

- String **sInsertParticipants**(string USER, string PASSWORD, int SurveyID, string ParticipantData)
  - **Purpose**: Creates and populates a table [dbprefix]token_[SurveyID] and thus "closes" the survey. This is the counterpart of sInsertToken. You will give the Participants Data and the function will create tokens automatically for them.
  - **Parameters**:
    - (m) int SurveyID, ID of the survey to close with a token table
    - string ParticipantData, The Data have to be in the right Syntax, it's like this: `FIRSTNAME;LASTNAME;EMAIL;[ATTRIB1];[ATTRIB2]::FIRSTNAME;LASTNAME;EMAIL;[ATTRIB1];[ATTRIB2]` seperate the Datasets with double colons (::) and the fields with semicolon (;). The attrib1 and attrib2 field are optional, you can just leave them out if not used. Note that the attrib field will get empty and not NULL, by now there where no complications, inform us, if you see some difficulties in not setting them NULL. Max. successfully tested length of a string was 3,000,000 Chars, this is just the limit of the apache webserver on my notebook (CoreDuo 1,9MHz, 1GB Ram) with default Resource Limits (60,60,16). I recommend to send max. 10,000 Datasets over the webservice(~1,000,000 chars). Repeat for more Datasets.
    - (m) string USER, this have to be an existing Admin-, Superadmin- or Useraccount with the rights to create and activate Surveys
    - (m) string PASSWORD, this has to match the password for the account.
  - **Returns**:
    - if everything went ok, it returns the String: "[X] Datasets given, [X] rows inserted. ";
  - **Possible Faults**:
    - throw new SoapFault("Authentication: ", "User or password wrong");
    - throw new SoapFault("Server: ", "No SurveyId given");
    - throw new SoapFault("Database: ", "Survey does not exists");
  - **Note**:
    - default tokenlength is 5 chars, if you want to change the length, goto japh.server.php, line 407
    - If the token table does not exists, it will be created, and if it does, the data will be added.

3.9 sTokenReturn

- list(int iVid, string return)**sTokenReturn**(string USER, string PASSWORD, int SurveyID)
  - **Purpose**: Gives back all Tokens, which did not complete the Survey, in a String seperated by commas.
  - **Parameters**:
    - (m) int SurveyID, the ID for the Survey from which you want get the unused tokens
    - (m) string USER, this have to be an existing Admin-, Superadmin- or Useraccount with the rights to create and activate Surveys
    - (m) string PASSWORD, this has to match the password for the account.
  - **Returns**:
    - Returns a list, behaving like an array, containing two Values: first the SurveyID, second a string of all tokens seperated by comma. Notice that the keys of the array are as the defined names of the message part in the wsdl... array(['iVid']=>SurveyID ['return']=>StringOfTokens) ...
  - **Possible Faults**:
    - throw new SoapFault("Authentication: ", "User or password wrong");
    - throw new SoapFault("Database: ", "Survey does not exists");
    - throw new SoapFault("Server: ", "No SurveyId given");
    - throw new SoapFault("Database: ", "Token table for this Survey does not exists");
    - throw new SoapFault("Database: ", "No unused Tokens found");

3.10 using the functions

You will need a SOAP- or at least RPC-Client to communicate with the JaphService Webservice. How to create such a Client depends on the programming language you use. The JaphService itself is written in PHP5 using the PHP:SOAP extension with SOAP 1.1.

In PHP you would initiate the Client-object and call a function from the Webservice with code like e.g.:

```
$wsdl="http://localhost/limesurvey/admin/japh.wsdl";
$client = new SoapClient($wsdl, array('soap_version' => SOAP_1_1,
                  'trace' => 1));
 try
         {
            $sReturn = $client->sInsertToken($user, $pass, $iVid, $sToken);
         }
         catch (SoapFault $fault)
         {
           $sOutput .= " <br/><br/>SOAP Error: ".$fault->faultcode." : ".$fault->faultstring;
         }
```

4 Testing

For testing purposes, we made a testclient in php. Therefore you need a webserver to run it. If you test on one machine you need nothing to do, you can go to http://localhost/limesurvey/admin/japh/japh.testclient.php in your browser to test the webservices functionality. Of Course this is very boring, because the apache talks with himself (and thats not the matter of a webservice).

The Testclient have two purposes:

1. It checks the Server(the one the testclient is running on!) for the PHP:SOAP Extension and installed libXML version and gives feedback if the enviroment does support SOAP via PHP:SOAP. This is very useful to check your Server and your Client towards PHP:SOAP compatibility. In addition to this, it will

check if the wsdl file you want to connect to exists, it does not validate the wsdl file.

2. You can test the functions of the webservice.

If you test the webservice in realistic enviroment (two machines) you have to customize three things

- the japh.testclient.php needs to be on the machine where the webservice is NOT running, but it needs a webserver... download xampp from apachefriends if you dont have one (caution for Linux Users: you have to compile PHP5 with SOAP and enable the extension, futhermore you have to install libXML2. Last thing can by done very easy via Synaptics or apt-get).
- this Line: `$wsdl="http://localhost/limesurvey/admin/japh.wsdl";` At the beginning of japh.testclient.php needs to fit the target webservices wsdl. localhost is always wrong, 127.0.0.1, too.
- the japh.wsdl you trigger with your testclient needs the information, where to find the script which initiates the SOAP Server. It is given in the last block of the japh.wsdl file on the Serverside.(see: "How to configure JaphService")

NOTE: There is a variable $limeUrl in the Configuration Block of the testclient. If you set this to your Limesurvey Installation Dir, the "test survey" link will be working...

5 Notes

- Security is enhanced: no action without permission (as good as Limesurveys Security itself --> very good)
- No headers or "Quick and Dirtys" anymore, no Scripts, just functions and a class.
- All configuration Variables are stored in japh.config.php by now
- There is a debug functionality, showing where the script runs through the code in an own log file.
- (**DONE**)~~It is planned to make an extra function to populate the token table with personal data about the participants, for people who want or need it.~~
- JaphService is part of LimeSurvey, therefore it stands under GNU/GPL License v2 or later.
- SOAP Services and therefore JaphService only support UTF-8 charsets.
- ~~By now, JaphService works with MySql only.~~
- ~~By now, JaphService works with MySql and Postgresql.~~
- By now, JaphService works with AdoDB (MySql,Postgresql, MSSQL) .
- JaphService is not fully tested and under steady development, use on your own risk, test the functions before using them!

6 Known Bugs (and workarounds)

- ~~the testclient doesn't support real UTF-8 charset, note that if you use special chars in the surveydecription, the string will be cut of after the first specialchar, by now, we do not know why. If you do not use a html form for datainput, it's working fine with specialchars and entities... Same bug for sInsertPaticipants Data...~~
- ~~the sInsertPaticipants and sInsertToken functions will create an empty field in the database if you set a seperator to the end or beginning of the String. Until this is fixed, simply do not do so...~~
  FIXED: sInsertPaticipants and sInsertToken function: Now, everything will went fine, even if you set a seperator in the beginnig or the end of the string. You should just note, that in case of an Seperator at the beginning or the end of the Datastring, the return Message will recognize one more given Dataset or Token as inserted into the db...
- There is a bug with Webservices in general: Sometimes (~ 1 of 10.000 Cases) you will get an error message like "function 'anyFunction' does not exist on the server" or similar, altough everything is fine. Just wait a few Moments, clear you cache, phone your mom (she will be happy to hear from you ;)) and try again.

- Webservice will not work if the "install" dir is not renamed or deleted in limesurvey -> "looks like we got no xml document" without errorlogs

7 SourceCode / Files

~~ALT:Media:JaphService 1.0 20080424 1500.zip~~
~~ALT:Media:JaphService_1.1_PgMySql_20080519_1102.zip~~

NEU: Media:Japh_v2.0_stable_2008-10-27.zip Tested and working fine. Version 2.0, what virtually means ~~nothing...~~ that we are supporting AdoDB now, everything is in functions or even classes, very modular and OO. More Parameters like: startdate, enddate, Admin Email, Admin Name, etc. Import and activation of Surveys are now two different Functions instead of one (so we can change some things in DB between these steps). Post any bugs or comments to tim.wahrendorff@gmail.com or the Limesurvey community

The included file "japh.testclient.php", which is not a functional part of JaphService, is meant for testing the Webservice... The testclient is PHP therefore it needs to be run on a webserver.