WebMathematics Interactive 2

Documentation for the developers

This document contains information about **WebMathematics Interactive, version 2. (WMI2)**, for programmers and the maintainers of it.

1. What is WMI2?

WebMathematics Interactive 2 (WMI2) is a web application primarily designed for mathematics education of the age group 14–21. In version 2, its main functionality is solving particular mathematical problems and displaying their formulas on a worksheet. It is like a computer algebra system (CAS), but with an easy accessibility on the web and support for educational purposes. Actually, the computation in WMI2 is done by Maxima, a console-based computer algebra system, so WMI2 is only a front-end for *Maxima* (and perhaps other computer algebra systems later). WMI2 has a previous version, WebMathematics Interactive, in which other functionalities can be found for education, and WMI2 is open for similar new functionalities, too.

2. How does it work?

WMI2 is started by one sitting before a computer and starting a web browser program, and then writing the address of the installment place of WMI2 in the location bar. When the application starts, it tries to find out some information about the user based on IP address and previous cookie values. For example, language settings and location city approximations are tried to be found out this way.

When everything is loaded in the page, users can see a calculator-looking box on the left, and a worksheet on the right, and some other things. Then users can begin manipulating the page, which happens with the support of the *JavaScript* and *Ajax* technology, which make it possible to refresh parts of the page without reloading the entire page. When a user presses the buttons of the calculator, or types in the text field of the calculator, the JavaScript based calculator takes the entered formula string, and sends it with Ajax technology to the server, which processes the formula string, and makes a formula picture out of it, and sends back the address of the picture to the client in the Ajax response. The client displays the picture as a part of the worksheet located on the right. So this way the entered formula is visualized at almost every calculator button clicks. This is the formula editor function of WMI2.

The method of the formula picture generating on the server takes place as follows: At first the input formula is converted to *TEX* format by the program *formconv*. Then a TEX file is made out of this TEX format formula, which is saved in the cache. The next step is the conversion of the TEX file to *DVI* format by the program *LATEX*, and then to *PNG* format by the programs *dvips* and *convert*. These programs, and also formconv are running on Linux, and they are started by the *PHP* external program calling mechanism (*exec*). The PNG file is saved in the cache, too.

Here we should tell some words about the *formula cache*. The pictures which are generated for the worksheet, are saved in the `cache/` directory for later access. This is particularly useful when a whole class of students are using WMI2 on a lesson, and solve the same problem. The filenames for the formulas are stored in the database of WMI2, so if a formula is to be visualized, then it should be checked, if it is in the database already or not. If it is in the database, then the picture file (or other data file) is supposed to already exist, and so its address (name) can be sent back to the client in the Ajax response, and no LATEX call is needed. This method for acceleration is in WMI since its first versions.

So we have seen the basic method for generating a formula picture with the calculator looking box on the left. But WMI2 is capable for not only displaying formulas, but for executing operations on them. Normally, there are some greenish blue buttons on the right of the calculator, and these are called *Maxima buttons*, or *operation buttons*. When a formula is entered with the mechanism described before, and an operation button is pressed (supposing in a right context), then there is another type of request for an output picture. If the pressed button was a Maxima button, the request is processed on the server as follows: At first, the formula is not converted to TEX format, but to Maxima format. Then a *Maxima* call is made out of this Maxima format formula by inserting it (or its parts) into the Maxima calling string for the Maxima button which was pressed. When the call is prepared, the cache should be checked. If the formula is not in the cache already, then Maxima is called with the prepared call. Maxima can return with TEX format formula, this TEX format formula is filtered out from the Maxima output, and converted to PNG with the method described above, and displayed. Maxima can also return with string format formula, this formula is returned to the client, too, and saved in the page in a hidden place in order to reuse it when the user clicks on the output picture.

Not only Maxima buttons, but at least one other type of buttons can be operation button, this type is the *gnuplot* button type, which starts evaluation executing in a similar fashion. The common in these evaluations is that a formula of an exercise is displayed first on the worksheet on the right, and then, after the operation, the exercise formula remains staying there, but a new picture is generated below it for the output, optionally with some text. After the new picture is displayed, one can enter new exercises below it, because the user is assumed to prefer seeing the former computations and the newer ones on one worksheet.

So in version 2, WMI supports worksheets for longer or shorter computational procedures, with messages coming from the application. The worksheet can be saved as HTML, or printed, or a new worksheet can be started. These functions are at the bottom of the page on the right.

3. The database structure

WMI2 is developed by using *Wekker*, a general web based application generator and running environment. Because of this, the input file structure of the database is so that every file contains information about only one table, and that file is for that table. There are table creator files which contain SQL create table statements, and table loader files, which contain the data which is to be contained in the database. We describe the database table definitions of WMI2 here file by file.

3.1. Texts

_LANGUAGE(id, langnameen, langnameown, flagiconfile, available)

A row of this table defines a language that can be used in WMI2. The id is a field that is contained in every table in WMI2, its type is *serial unique*. The langnameen field is a textual identifier for the language, in order to reference it from the application code. The langnameown field contains the name of the language in that language. The flagiconfile field contains the name of the file which serves as the flag of the language in order to be clicked on it. available is a boolean field, and it is true if one can change to this language during the use of WMI2. If it is false, then WMI2 may contain texts that are in that language, but will not display these texts.

I18NTEXT(id, mainid, _language, text)

This table is a solution for internationalizing the texts in WMI2. There are two ids in this table, one is the id for the database row, which identifies the text absolutely, and the other is the mainid, which identifies the meaning of the text, so that a text would be identified with the mainid and the _language fields. The mainid itself is the same as one of the ids of the rows that has that mainid. The _language field is a reference to the id field of the _LANGUAGE table. The field named text contains the internationalized text itself.

MESSAGES(id, handle, textid)

This is for those messages which should be internationalized and able to be referenced by a textual id or handle. This is particularly needed when a message should be loaded directly from the program code, and not as a part of other database structure. The handle field is a textual identifie, and the field named textid is a reference to the I18NTEXT table and its id (in fact, we use it as a mainid in most cases).

3.2. Calculator

BUTTONTYPE(id, name, description, textcolor, bgcolor)

This table is for the types of the buttons in the calculator, for helping the determination of their colors and behaviour. The name field is a textual name of the type, as referenced by the JavaScript calculator. The description field can contain a reference to an internationalizable text as a guide. The textcolor and bgcolor fields contain information about the coloring of this type of buttons.

BUTTONGROUP(id, name, buttonrow, description)

The BUTTONGROUP table was used to name the categories of the buttons for mouseover effects on the calculator, and now it is used to describe the groups which are coming up on mouseover in one or more specific layouts. The name field is a textual identifier for the programmers, buttonrow is a number meaning how many buttons should be displayed in a row in that group, and description is a reference to an internationalizable text meaning nothing much now.

CALCBUTTON(id, name, _type, changestring, _group, grouporder, picture, description, html_text, solve_text)

This is a very important table, because this table defines the buttons of the calculator in WMI2. The name field is the title of the button, it appears on the button as text when no other display options are specified. The field named _type is a reference to a row of the BUTTONTYPE table, it defines the basic behaviour of the button. The very important changestring field is a text which contains information about what to do exactly when the button is pressed. _group is a deprecated field which once contained the information of the assignment of buttons to groups. The grouporder field is also deprecated, this was for determining the position of the button in the group of it. If there is an image for the button, then the picture field contains the file name of that image. The description field of the CALCBUTTON table contains the text which goes into the title attribute of the button, and comes up on the mouseover event. The field named html_text can contain a html text label of the button, if specified (in this case the name field is not used for this purpose). If the button is an operation button (of the _type *maxima* or *gnuplot*) then the solve_text field can contain the text written in the worksheet before the formula, in the case of successful return.

CALCLAYOUT(id, name, description, introductory_text)

A calculator layout is the layout of the buttons of the calculator at a moment. WMI2 loads this from the database, and this table is for naming these layouts. The name field is a textual name for the layout, description contains the text which appears when WMI2 starts with that layout, and introductory_text contains the text which appears on the start of every exercise.

BUTTONLAYOUT(id, layout, button, buttonorder [sorrend], groupid, grouporder)

Important! The buttonorder field is substituted with the sorrend field, because in Wekker, the web 2.0 based application generator which we use as the database manager of WMI2, the ordering of the rows is solved by the introducing a new field named sorrend, a Hungarian word meaning "order".

This table is intended to determine the place of the buttons in the layouts, and in the groups which are coming up when we move the mouse over a button of them in a specific layout, or in some layouts. This table contains only integer, reference and serial field types. A row of this table means the following:

The button field contains a reference to a row of the CALCBUTTON table, because a row in the BUTTONLAYOUT table is primarily for placing the buttons.

If the layout field is not null, then it contains a reference to a row of the CALCLAYOUT table. In this case the row of the BUTTONLAYOUT table defines data that is associated only with that specific layout. We still have two main possibilities in this case:

- If the buttonorder field (which is substituted by the sorrend field) is not null, then this row of the BUTTONLAYOUT table should define one out of the 40 buttons which are initially visible when the layout loads. In this case, the groupid field may optionally contain a reference to a row of the BUTTONGROUP table, and if so, it may contain an integer which describes the position of the button in the group.
- If the buttonorder field is null, then the row of the BUTTONLAYOUT table defines a button place which belongs to a button group specific to the given layout, and having a main button which lies on the layout when the layout loads. So in this case the groupid and grouporder fields must be filled for both the button and its main button.

If the layout field is null, then the row of the BUTTONLAYOUT table defines a button place which is not associated strictly with any layouts, but is associated with a group that can be included in any layouts. In this case, the grouporder and groupid fields must also be filled, but this should not be done for the main button of the group, because in other layouts other main buttons can be defined. Instead of it, the main button should be defined as not belonging to the layout (but the button), too.

3.3. Formula cache

APPLICATION(id, handle, outputmime, idcode, maxsize, description)

A row of this table defines a type of entry in the table for the cache. The textual handle field tells us the type of the input formula, one of the fields which the cached file is identified by. The textual outputmime field tells us something similar to the mime type of the output file that is stored in the cache. And there is a textual idcode field by which an entry in the APPLICATION table is identified. The maxsize field may contain the maximal size of files of this this type of request, but actually it is not implemented this way yet. The description field may contain reference to the I18NTEXT table, but it is not used yet.

REQUEST(id, application, parameter, formula, filename, sent, received)

This is used to store the entries of the cache. The application field is a reference to the formerly described APPLICATION table, it defines a use case of the REQUEST table. A row of the APPLICATION table often suggests an application (or more) to launch in order to get the output formula, so the parameter field of the REQUEST table is a field containing the parameters which the application is to be started with. The formula field contains the formula, one of the fields which the cached file is identified by. The filename field is a text describing the name of the cached file. The sent and the received fields are timestamps, sent means 'sent the file to the client', and received means 'received the file from server'.

We have seen the tables that are fundamental for the formula manipulation of WMI2, but some other tables are necessary for storing the data of the users of WMI2.

3.4. User related

FELHASZNALO(id, vezeteknev, keresztnev, jelszo, jelszo_ervenyes_tol, jelszo_ervenyes_ig, teljes_nev, azonosito, elsodleges_telefon, email, honlap, cim, kis_foto_fajl, nagy_foto_fajl, geoip_szelesseg, geoip_hosszusag, rovid_leiras, hosszu_leiras, varosnev)

The name of this table means 'user' in Hungarian. The reason why this table is in Hungarian is because in Wekker, the application generator program used with WMI2, there was a table named so, and that was altered to a table of WMI2. The textual vezeteknev, keresztnev and teljes_nev fields mean last name, first name and full name, respectively. The jelszo field means password, and the jelszo_ervenyes_tol field means password_operates_since, and the jelszo_ervenyes_ig field means password_operates_up_to. The latter two are timestamps. The azonosito field means identifier, and it is a user name to log in Wekker, with the former defined password.

The following are the fields which are in the FELHASZNALO table especially for WMI2. This is because WMI2 has a subsystem for advertisements. The elsodleges_telefon field means primary phone, email means email, the honlap field means website, and cim means address. These fields are filled implicitly. The kis_foto_fajl field means little_photo_file, and nagy_foto_fajl means big_photo_file. These are file names. The geoip_szelesseg field means geoip_latitude, the geoip_hosszusag means geoip_longitude, these are longitude and latitude data of the place where a teacher lives, and these are used in connection with the *GeoLite City* software in order to approximate the distance from a teacher to the student. The rovid_leiras and hosszu_leiras fields means short_description, long_description, respectively. The varosnev field means city name. These are filled implicitly.

CATEGORY(id, short_description, long_description, calclayout, parent)

This table is for the advertisement system of WMI2, too. The teachers can choose categories of what to teach. The short_description and long_description fields are filled implicitly. The calclayout field connects the category with a layout, and makes it possible to associate teachers to layouts. It is a reference to the CALCLAYOUT table. The parent field is a reference to the CATEGORY table itself, and makes it possible to make subcategories.

FELHASZNALO_CATEGORY(id, felhasznalo, category)

This table is for connecting the FELHASZNALO and the CATEGORY tables. The felhasznalo field means user, and it is a reference to the FELHASZNALO table. The category field is a reference to the CATEGORY table.

SHOWED_AD(id, felhasznalo, category, showed, visitor_ip, visitor_hostname, visitor_geoip_city, visitor_geoip_latitude, visitor_geoip_longitude, clicked, visitor_distance)

In this table the teachers, whose advertisements are showed, are registered. The felhasznalo field is for the teacher, the category field is for the category, these are references. The showed field is a timestamp indicating when was the ad showed. The visitor_ip field is a text containing the IP address of the one who are seeing the data for the teacher. Fields with similar function are visitor_hostname, visitor_geoip_city, visitor_geoip_latitude, visitor_geoip_longitude, meaning the things which are in their name. The boolean clicked field shows if the visitor has clicked on the short description of the teacher in order to know more about the teacher. The visitor_distance is a field of the type float, meaning how far is the place where the visitor is using WMI2 from the location where the teaching would take place.

These were the database tables which are in use of version 2 of WMI.

4. The contents of source files

The contents of the PHP script files of the WMI2 program and the necessary files running them are described in this section. At first, let's see the JavaScript files located in the **js** directory.

ajax.js

This script is responsible for the Ajax communication between **main.php** (the server script) and the client. It has a general function for the communication using Ajax, and many special Ajax functions using it. Some special functions have meaning only if some of the other JavaScript files used in WMI2 are included in the page.

calc.js

This script is for the calculator management. It generates the HTML structure of the calculator in JavaScript with DOM, and handles the keyboard and mouse events associated with it, and uses the Ajax functions defined in ajax.js to communicate with the server, and uses the DOM functions of Javascript to access some parts of the HTML page.

wmiutils.js

This script is for that JavaScript functions of WMI2 which are not associated strictly with the calculator, but with the other parts of the page, for example the worksheet.

These three JavaScript files should be used together as included in the HTML of the **index1.php** file. There are other files which should be used together with them, these are the CSS files containing the styles of the page. These files are located in the css directory:

users1.css

This is the most important style file, because it defines the style of the main page.

calc1.css

This defines the style of the calculator.

ie1.css

This style file is for the case when the visitor uses Microsoft Internet Explorer.

popup.css

This style file is for a popup window containing a teacher.

teachers.css

This style is for the style of window of the database of all teachers.

worksheet.css

This style is for the popup window in which the worksheet is opened separately from the page.

These JavaScript and CSS files are the client-side program files used in WMI2. Let us see now that PHP files of the WMI2 program which are close to the client-side:

index.php

This is the index file, this usually starts when someone accesses the page the first time. It is pure HTML now in version 2. It defines a frameset with only one big frame, which occupies the whole page, the big frame is named **index1.php**.

index1.php

This file behaves as the index file of WMI2. The main JavaScript and CSS files are included in it, and it defines the main structure of the page elements. Its language is mixed PHP and HTML.

info.php

This file is the content of the popup window giving some information about WMI2, it is a mixed PHP and HTML file.

teachers.php

This is for the content of the popup window showing the database of all teachers in WMI2, it is mixed PHP and HTML, too.

error.php

This is a page containing the error message which comes up on error in WMI2. For now, this is pure PHP.

We have seen those PHP files which could be started by the user and give HTML output. But WMI2 is an application basicly using Ajax. For the Ajax communication, there is a server PHP script in WMI2, which serves the Ajax requests, and outputs not a whole HTML page, but miscellaneous data. This script is the following:

main.php

This is the main script of WMI2 which serves all the Ajax requests initiated by functions of the **ajax.js** script. This can return HTML elements, JavaScript and pure text, too. This processes the HTTP POST requests, and depending on the parameters got, tries to give that type of output which is requested. There are many PHP files (from the **../includes/** directory) included in this file. The language of this file is pure PHP.

We have seen those PHP files which start PHP programs used in WMI2, but these do not contain all the code necessary to run WMI2. There are other PHP files which are included in these files, and these files reside in the **includes** directory. Let us see the most often included files of these now:

config.php

This is a generated file, and it contains values of many default variables depending on the installation of WMI2.

session.php

This file contains code for the initialization of some session variables used in WMI2.

sql.php

This file contains the functions used for the connection to the SQL database, a function checking the SQL result, and some miscellaneous functions somehow related to the SQL database used with WMI2.

Now let's see the other include files:

formula.php

This include file is responsible for the conversion of formulas between various formats, including PNG image.

calc.php

This is reponsible for getting the elements of the calculator from the database, and the related things.

cas.php

This is responsible for executing a call to a computer algebra system.

plot.php

This is responsible for executing a call to function plotting software.

cache.php

This contains database functions responsible for saving generated files in the cache, these generated files are identified by the type of files, the parameters of generating, and the formulas of them.

adverts.php

In this file, there are functions related to the teacher advertisement system of WMI2.

log.php

Logging related functions are in this file.

install.php

This file tests if the requirements of WMI2 are met on the system.

text.php

This is a utility file, with only one function now. It is related to text processing.

These were the web-related program files of WMI2. There is a shell script, namely **timeout**, which is used in connection with the running of WMI2 as well, this provides that an application which WMI2 starts does not run forever, but stops after 10 seconds.

5. Some PHP functions

In this section there are those PHP functions which are considered more essential for documentation than others. The parameters of the documented functions do not contain dollar signs here. At first, some functions of **main.php** are documented here:

function calcact(value, density, usecursor)

This function does the conversion of the entered input formula to PNG file. It returns the relative path to the file from the web root directory. It checks if the formula is in the cache, and if not, converts the intuitive format formula to TEX format, makes a TEX file out of it, and converts the TEX to PNG all with suitable functions of WMI2. The value parameter means the entered formula, density means the size of the PNG file, and usecursor means whether the formula picture should contain a red square for the cursor or not.

function calcop(value, operation, density)

This function does the execution of the chosen operation on the entered input formula, and converts the result to a PNG format picture. It returns the relative path to the PNG file from the web root directory. It converts the intuitive format formula to Maxima format, prepares a call statement out of the formula and the operation on it, checks if the call is in the cache, and if not, executes the call with Maxima (or some other CAS later), and converts the returned TEX to a PNG format file. All of these are done with suitable functions of WMI2. The value parameter means the entered formula, operation means the Maxima (or other CAS) operation on it, and density means the size of the PNG file.

function calcopCAS(value, operation)

This function is similar to calcop, but it does not return a file, but a string containing the result in mixed Maxima/intuitive format (or other CAS later). So it does the operation given in the operation parameter on the formula given in the formula parameter. At first, it converts the formula to the suitable format (Maxima), and then makes a call out of the formula and the operation, and then executes the call with a suitable function of WMI2. The output is then filtered in order to be reusable in intuitive format.

function plotter(formula, xmin, xmax, ymin, ymax, grid, is3d, ispar)

This function is similar to calcop, too, but it does not make a call to a computer algebra system, but to a function plotting utility, named gnuplot. At first, it converts the formula given in the formula parameter to gnuplot format. Then checks if the formula with these parameters is in the cache, and if not, then makes a picture and saves it in the cache. The function returns the relative path to the PNG file which contains the plotting of the formula. All of these are done with suitable functions of WMI2.

Some important functions which are defined in miscallenous files:

function timeout(app)

This function is defined in **config.php**. It ensures that an application started by WMI2 does not run for long, but stops after 10 seconds. Its parameter is the full path of an executable, and returns the path of the timeout script called with the given executable. This timeout script is a Linux shell script, and the executable can be Maxima or formconv or some other applications.

function connect()

This function is defined in **sql.php**. It connects WMI2 to the PostgreSQL database.

function hasResult(result_resource)

This function is defined in **sql.php**. It checks if the result_resource returned by an SQL query points to a valid result or not.

function getmessage(handle, lang)

This function is defined in **sql.php**. It retrieves an internationalized message from the database in a given language referred by a given handle. The two parameters of it are the referring handle and the lang meaning language.

Some functions of **formula.php**:

function formconv(informula, informat, outformat, params)

This function calls the program *formconv* using the PHP *exec* function, in order to convert the input formula (informula) which is in format informat, to the format outformat with the parameters params. This function returns the output formula of formconv.

function isFormconvException(formconvreturn)

formconv can return not only valid formulas, but error messages, too. This function checks if formconv is returned with exception or not.

function maketex(informula, outfilename)

This function makes a TEX file with the name outfilename from the TEX format formula given in the parameter informula.

function tex2png(texfilename, pngfilename, texdensity)

This function makes a PNG file with the name pngfilename and with the size texdensity from a TEX file named as the parameter texfilename.

Some functions of **calc.php**:

function buttongroup(value, lang)

This function returns the button group which has the name that is given in the value parameter. The group is returned with messages in the given language. The language is given in the lang parameter.

function returnlayout(value, lang)

This function returns the calculator layout which has the name that is given in the value parameter. The layout is returned with messages in the given language. The language is given in the lang parameter.

Some functions of **cas.php**:

function maxima(call)

This function returns the result of a Maxima call in TEX format.

function maxima_string(call)

This function returns the result of a Maxima call in Maxima string format.

Some functions of **plot.php**:

function plotprogram(formula, xmin, xmax, ymin, ymax, grid, is3d, ispar)

This function makes a gnuplot program out of the given parameters: the formula, the boundaries and grid parameter.

function gnuplot(plotprogram, outputfile)

This function executes a gnuplot program and puts the result picture to the file outputfile.

Some functions of **cache.php**:

function isincache(application, parameter, formula)

This function checks if there is an entry in the cache database table with these three parameters: the application which makes the cached file, the parameter which the application is started with, and the formula which is transformed to the output file. If there is no such an entry, then it returns FALSE, else it returns the file name of the entry in the cache.

function puttocache(application, parameter, formula)

This function returns a file name which does not refer to any file yet, and saves this in the cache with these three parameters as handles: the application which makes the cached file, the parameter which the application is started with, and the formula which is transformed to the output file.

function clearfromcache(application, parameter, formula)

This function clears an entry from the database cache with the given parameters.

Some functions of **adverts.php**:

function compareGeoABigger(a, b)

This function is used by the PHP *ushort* function, and is called from the getFirstTeachers function. It compares two teachers based on the distance from the location of the visitor.

function getFirstTeachers(layout)

This function returns the teachers fitting into the category determined by the layout parameter. The return value is an array shorted by the distance from the visitor. The method is in the function compareGeoABigger.

6. Frequently asked questions

What can I help in the development of WMI2?

You can help in many areas.

- You can help by giving feedback about your experience using WMI2. So you can *report bugs* (program errors) you have experienced while you used WMI2, and *request new features* that you would like to be in this application. The list of bugs and feature requests which are already known to us are on the Flyspray installation associated to WMI2.
- If you can speak some language which is not fully supported in WMI2 yet, then you can help in the translation of WMI2 to another language.
- If you are a programmer, you can help in the development by joining to the developers, and making some programming tasks. You can even browse the CVS repository or download the current CVS tarball which is generated in every hour automatically.

How can I help in the development of WMI2?

Just write an email to the developers to the address **info@matek.hu**. For some types of tasks you should try WMI2 out: install it and modify it. If you want to add code, you may have to go to sf.net, register, learn using CVS, and be added to the developers' list by us. Or you can have a user id on our server as well, if it is needed for the development. (Note: the current CVS tree for version 2 is not hosted on SourceForge, but on our private server. This may change in the future.)

How to make WMI2 use my language?

In the database table I18NTEXT, there are the internationalized texts. If you want to translate a text, just add an entry to this table with the same mainid field as of the text to translate, and fill the _language field with the language id you have chosen. These ids are in the _LANGUAGE table, you can define new languages here, too. Please notify us if you have chosen to translate WMI2 to your language, because perhaps we would like to include it in the official WMI2 version, and if you won't notify us, then putting your code into our version, and putting our code in your version would be a difficult task subsequently.

How to create a new layout? How to create a new button? How to create a new operation button? How to create a new button text? How to create button groups?

All the questions above will be easily answered when the Wekker system is fully available to support external database editing. However, current version of WMI2 has no open source support for doing these tasks. Now the only way is to modify the database by hand. Please go to **backend/sql/wmi2os.sql** and try to figure out its structure and content.

WMI2 has a built-in advertisement system. What is it exactly?

To gain financial support for developing WMI2, we introduced this advertisement system since version 2.0. It requires the GeoLite package developed by MaxMind. This feature can be switched off by setting the **$adverts_on** variable to **FALSE** in includes/config.php. (You may also want to disable the **$developers_on** variable as well if you do not want to reach the Wekker database, if it is installed at all.)

Is there a mailing list for WMI2?

Yes, Google Groups hosts the "wmi2" group. You are encouraged to join us.