.. \_transformations:

Transformations

===============

.. \_transformationsintro:

Introduction

++++++++++++

To enable transformations, you have to setup the ``column\_info``

table and the proper directives. Please see the :ref:`config` on how to do so.

You can apply different transformations to the contents of each

column. The transformation will take the content of each column and

transform it with certain rules defined in the selected

transformation.

Say you have a column 'filename' which contains a filename. Normally

you would see in phpMyAdmin only this filename. Using transformations

you can transform that filename into a HTML link, so you can click

inside of the phpMyAdmin structure on the column's link and will see

the file displayed in a new browser window. Using transformation

options you can also specify strings to append/prepend to a string or

the format you want the output stored in.

For a general overview of all available transformations and their

options, you can consult your \*<www.your-host.com>/<your-install-

dir>/transformation\\_overview.php\* installation.

For a tutorial on how to effectively use transformations, see our

`Link section <http://www.phpmyadmin.net/home\_page/docs.php>`\_ on the

official phpMyAdmin homepage.

.. \_transformationshowto:

Usage

+++++

Go to your \*tbl\\_structure.php\* page (i.e. reached through clicking on

the 'Structure' link for a table). There click on "Change" (or change

icon) and there you will see three new fields at the end of the line.

They are called 'MIME-type', 'Browser transformation' and

'Transformation options'.

\* The field 'MIME-type' is a drop-down field. Select the MIME-type that

corresponds to the column's contents. Please note that transformations

are inactive as long as no MIME-type is selected.

\* The field 'Browser transformation' is a drop-down field. You can

choose from a hopefully growing amount of pre-defined transformations.

See below for information on how to build your own transformation.

There are global transformations and mimetype-bound transformations.

Global transformations can be used for any mimetype. They will take

the mimetype, if necessary, into regard. Mimetype-bound

transformations usually only operate on a certain mimetype. There are

transformations which operate on the main mimetype (like 'image'),

which will most likely take the subtype into regard, and those who

only operate on a specific subtype (like 'image/jpeg'). You can use

transformations on mimetypes for which the function was not defined

for. There is no security check for you selected the right

transformation, so take care of what the output will be like.

\* The field 'Transformation options' is a free-type textfield. You have

to enter transform-function specific options here. Usually the

transforms can operate with default options, but it is generally a

good idea to look up the overview to see which options are necessary.

Much like the ENUM/SET-Fields, you have to split up several options

using the format 'a','b','c',...(NOTE THE MISSING BLANKS). This is

because internally the options will be parsed as an array, leaving the

first value the first element in the array, and so forth. If you want

to specify a MIME character set you can define it in the

transformation\\_options. You have to put that outside of the pre-

defined options of the specific mime-transform, as the last value of

the set. Use the format "'; charset=XXX'". If you use a transform, for

which you can specify 2 options and you want to append a character

set, enter "'first parameter','second parameter','charset=us-ascii'".

You can, however use the defaults for the parameters: "'','','charset

=us-ascii'".

.. \_transformationsfiles:

File structure

++++++++++++++

All specific transformations for mimetypes are defined through class

files in the directory 'libraries/plugins/transformations/'. Each of

them extends a certain transformation abstract class declared in

libraries/plugins/transformations/abstract.

They are stored in files to ease up customization and easy adding of

new transformations.

Because the user cannot enter own mimetypes, it is kept sure that

transformations always work. It makes no sense to apply a

transformation to a mimetype the transform-function doesn't know to

handle.

There is a file called '\*transformations.lib.php\*' that provides some

basic functions which can be included by any other transform function.

The file name convention is ``[Mimetype]\_[Subtype]\_[Transformation

Name].class.php``, while the abtract class that it extends has the

name ``[Transformation Name]TransformationsPlugin``. All of the

methods that have to be implemented by a transformations plug-in are:

#. getMIMEType() and getMIMESubtype() in the main class;

#. getName(), getInfo() and applyTransformation() in the abstract class

it extends.

The getMIMEType(), getMIMESubtype() and getName() methods return the

name of the MIME type, MIME Subtype and transformation accordingly.

getInfo() returns the transformation's description and possible

options it may receive and applyTransformation() is the method that

does the actual work of the transformation plug-in.

Please see the libraries/plugins/transformations/TEMPLATE and

libraries/plugins/transformations/TEMPLATE\\_ABSTRACT files for adding

your own transformation plug-in. You can also generate a new

transformation plug-in (with or without the abstract transformation

class), by using

:file:`libraries/plugins/transformations/generator\_plugin.sh` or

:file:`libraries/plugins/transformations/generator\_main\_class.sh`.

The applyTransformation() method always gets passed three variables:

#. \*\*$buffer\*\* - Contains the text inside of the column. This is the

text, you want to transform.

#. \*\*$options\*\* - Contains any user-passed options to a transform

function as an array.

#. \*\*$meta\*\* - Contains an object with information about your column. The

data is drawn from the output of the `mysql\\_fetch\\_field()

<http://www.php.net/mysql\_fetch\_field>`\_ function. This means, all

object properties described on the `manual page

<http://www.php.net/mysql\_fetch\_field>`\_ are available in this

variable and can be used to transform a column accordingly to

unsigned/zerofill/not\\_null/... properties. The $meta->mimetype

variable contains the original MIME-type of the column (i.e.

'text/plain', 'image/jpeg' etc.)