djangocms-equation Documentation

Release 0.2.4

Sebastian Weigand

Oct 03, 2021

FOR USERS:

CHAPTER

ONE

DJANGOCMS-EQUATION

DjangoCMS plugin to write equations, utilizing KaTeX

1.1 Features

- Enables the use of LaTeX for equations with django-cms
- Live editing of LaTeX Code, via KaTeX
- Out of the box support for mhchem
- Configurable allowing of copying of equation LaTeX code

1.2 Installation

Install the plugin from PyPi

```
$ pip install djangocms-equation
```

Add the plugin to the installed apps in the `settings.py` of your django-cms project.

```
"INSTALLED_APPS": [..., "djangocms_equation"]
```

For the Equations to be properly displayed in `djangocms-text-ckeditor`, while edit them, you need to add the css file to the allowed files of ckeditor. To do this simply add the following lines to your `settings.py` of your django-cms project.

```
CKEDITOR_SETTINGS = {
    "contentsCss": ["/static/djangocms_equation/css/change_form_template.css"]
}
```

Note:

The equations might not be rendered properly in ckeditor-windows, when they are added the first time. This can be fixed by saving the text plugin or having another equation on the page.

To allow copying of equations LaTeX code, add the following line to your `settings.py`.

```
"KATEX_EQUATION_SETTINGS" = {"allow_copy": True}
```

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install djangocms-equation, run this command in your terminal:

```
$ pip install djangocms-equation
```

This is the preferred method to install djangocms-equation, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for djangocms-equation can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/s-weigand/djangocms-equation
```

Or download the tarball:

```
$ curl  -OL https://github.com/s-weigand/djangocms-equation/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

CREDITS

3.1 Development Lead

- Sebastian Weigand <s.weigand.phy@gmail.com>

3.2 Contributors

None yet. Why not be the first?

CHAPTER

FOUR

HISTORY

4.1 0.2.4 (2021-05-11)

- Upgraded KaTeX to 0.13.9
- Updated javascript dependencies to fix security issues

4.2 0.2.3 (2021-05-04)

- Upgraded KaTeX to 0.13.5

4.3 0.2.2 (2021-03-29)

- Added official python 3.9 support
- Upgraded KaTeX to 0.13.0

4.4 0.2.1 (2020-08-09)

- Added official python 3.8 support

4.5 0.2.0 (2020-08-09)

- Upgraded KaTeX to 0.12.0
- Dropped support for Django 2.0

4.6 0.1.1 (2020-02-20)

- Added project urls and removed conda badge

4.7 0.1.0 (2020-02-20)

- First release on PyPI.

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/s-weigand/djangocms-equation/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

djangocms-equation could always use more documentation, whether as part of the official djangocms-equation docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/s-weigand/djangocms-equation/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up `djangocms_equation` for local development.

1. Fork the `djangocms_equation` repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/djangocms_equation.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv djangocms_equation
$ cd djangocms-equation/
$ pip install -r requirements_dev.txt
$ pip install -e .
```

4. Install the `pre-commit` hooks, for quality assurance:

```
$ pre-commit install && pre-commit install -t pre-push
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

6. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox and docker-compose:

```
$ docker-compose up -d
$ tox
```

   Docker compose is needed for the integration tests, which use selenium and the selenium docker images.

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6 and 3.7. Check https://github.com/s-weigand/djangocms-equation/actions and make sure that the tests pass for all supported Python versions.

Note: Due to racing conditions in the integration tests, which I couldn't completely eliminate, the CI might fail for some tests. In this case just write a comment, so I know to restart the test suite.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_djangocms_equation
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Github actions will then deploy to PyPI if tests pass.

CHAPTER

SIX

PYTHON SOURCE DOCUMENTATION

Documentation of the Python source code. The auto generated Documentation mainly contains methods and attributes which are inherited from the corresponding classes of `django-cms`, but since the actual source code is very short you can just click the `[source]` link on top of each class. The auto generated documentation of the inherited classes is given to give insight on the actual finished objects.

cms_plugins.EquationPlugin	Implementation of the actual plugin.
forms.EquationForm	Inputs which are used in the Equation editing dialog.
models.EquationPluginModel	Database model of saved Equations.

6.1 EquationPlugin

class EquationPlugin(*model=None*, *admin_site=None*)

Implementation of the actual plugin.

Attributes Summary

actions
actions_on_bottom
actions_on_top
actions_selection_counter
add_form_template
admin_preview
allow_children
autocomplete_fields
cache
cache_child_classes

continues on next page

Table 2 – continued from previous page

cache_parent_classes
change_form_template
change_list_template
child_classes
date_hierarchy
delete_confirmation_template
delete_selected_confirmation_template
disable_child_plugins
exclude
fields
fieldsets
filter_horizontal
filter_vertical
formfield_overrides
inlines
list_display
list_display_links
list_editable
list_filter
list_max_show_all
list_per_page
list_select_related
media
module
name

continues on next page

Table 2 – continued from previous page

object_history_template
opts
ordering
page_only
parent_classes
plugin_urls
popup_response_template
prepopulated_fields
preserve_filters
radio_fields
raw_id_fields
readonly_fields
render_plugin
render_template
require_parent
save_as
save_as_continue
save_on_top
search_fields
show_full_result_count
sortable_by
system
text_editor_preview
text_enabled
urls

continues on next page

Table 2 – continued from previous page

value	
view_on_site	

Methods Summary

action_checkbox	A list_display column containing a checkbox widget.
add_view	
autocomplete_view	
change_view	
changeform_view	
changelist_view	The 'change list' admin view for this model.
check	
construct_change_message	Construct a JSON structure describing changes from a changed object.
delete_model	Given a model instance delete it from the database.
delete_queryset	Given a queryset, delete it from the database.
delete_view	
formfield_for_choice_field	Get a form Field for a database Field that has declared choices.
formfield_for_dbfield	Hook for specifying the form Field instance for a given database Field instance.
formfield_for_foreignkey	Get a form Field for a ForeignKey.
formfield_for_manytomany	Get a form Field for a ManyToManyField.
get_action	Return a given action from a parameter, which can either be a callable, or the name of a method on the ModelAdmin.
get_action_choices	Return a list of choices for use in a form object.
get_actions	Return a dictionary mapping the names of all actions for this ModelAdmin to a tuple of (callable, name, description) for each action.
get_autocomplete_fields	Return a list of ForeignKey and/or ManyToMany fields which should use an autocomplete widget.
get_cache_expiration	Provides hints to the placeholder, and in turn to the page for determining the appropriate Cache-Control headers to add to the HTTPResponse object.
get_changeform_initial_data	Get the initial form data from the request's GET params.
get_changelist	Return the ChangeList class for use on the changelist page.
get_changelist_form	Return a Form class for use in the Formset on the changelist page.

continues on next page

Table 3 – continued from previous page

get_changelist_formset	Return a FormSet class for use on the changelist page if list_editable is used.
get_changelist_instance	Return a *ChangeList* instance based on *request*.
get_child_class_overrides	Returns a list of plugin types that are allowed as children of this plugin.
get_child_classes	Returns a list of plugin types that can be added as children to this plugin.
get_child_plugin_candidates	Returns a list of all plugin classes that will be considered when fetching all available child classes for this plugin.
get_deleted_objects	Hook for customizing the delete process for the delete view and the "delete selected" action.
get_empty_change_form_text	Returns the text displayed to the user when editing a plugin that requires no configuration.
get_empty_value_display	Return the empty_value_display set on ModelAdmin or AdminSite.
get_exclude	Hook for specifying exclude.
get_extra_placeholder_menu_items	
get_extra_plugin_menu_items	
get_field_queryset	If the ModelAdmin specifies ordering, the queryset should respect that ordering.
get_fields	Hook for specifying fields.
get_fieldsets	Same as from base class except if there are no fields, show an info message.
get_form	Return a Form class for use in the admin add view.
get_formsets_with_inlines	Yield formsets and the corresponding inlines.
get_inline_formsets	
get_inline_instances	
get_list_display	Return a sequence containing the fields to be displayed on the changelist.
get_list_display_links	Return a sequence containing the fields to be displayed as links on the changelist.
get_list_filter	Return a sequence containing the fields to be displayed as filters in the right sidebar of the changelist page.
get_list_select_related	Return a list of fields to add to the select_related() part of the changelist items query.
get_model_perms	Return a dict of all perms for this model.
get_object	Return an instance matching the field and value provided, the primary key is used if no field is provided.
get_ordering	Hook for specifying field ordering.
get_paginator	
get_parent_classes	
get_plugin_urls	Return URL patterns for which the plugin wants to register views for.

continues on next page

Table 3 – continued from previous page

get_prepopulated_fields	Hook for specifying custom prepopulated fields.
get_preserved_filters	Return the preserved filters querystring.
get_queryset	Return a QuerySet of all model instances that can be edited by the admin site.
get_readonly_fields	Hook for specifying custom readonly fields.
get_render_queryset	
get_require_parent	
get_search_fields	Return a sequence containing the fields to be searched whenever somebody submits a search query.
get_search_results	Return a tuple containing a queryset to implement the search and a boolean indicating if the results may contain duplicates.
get_sortable_by	Hook for specifying which fields can be sorted in the changelist.
get_urls	
get_vary_cache_on	Provides hints to the placeholder, and in turn to the page for determining VARY headers for the response.
get_view_on_site_url	
has_add_permission	Return True if the given request has permission to add an object.
has_change_permission	Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.
has_delete_permission	Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.
has_module_permission	Return True if the given request has any permission in the given app label.
has_view_or_change_permission	
has_view_permission	Return True if the given request has permission to view the given Django model instance.
history_view	The 'history' admin view for this model.
icon_alt	Return the alt text for the shown icon.
icon_src	Return the path to an icon which is shown in the text editor.
is_in_text_editor	Check if the plugin was added to a text plugin.
log_addition	Log that an object has been successfully added.
log_change	Log that an object has been successfully changed.
log_deletion	Log that an object will be deleted.
lookup_allowed	
message_user	Send a message to the user.
render	Render the Plugin with self.render_template and the data in instance.
render_change_form	We just need the popup interface here

continues on next page

Table 3 – continued from previous page

render_close_frame	
render_delete_form	
requires_parent_plugin	
response_action	Handle an admin action.
response_add	Determine the HttpResponse for the add_view stage.
response_change	Determine the HttpResponse for the change_view stage.
response_delete	Determine the HttpResponse for the delete_view stage.
response_post_save_add	Figure out where to redirect after the 'Save' button has been pressed when adding a new object.
response_post_save_change	Figure out where to redirect after the 'Save' button has been pressed when editing an existing object.
save_form	Given a ModelForm return an unsaved instance.
save_formset	Given an inline formset save it to the database.
save_model	Override original method, and add some attributes to obj This have to be made, because if object is newly created, he must know where he lives.
save_related	Given the `HttpRequest`, the parent `ModelForm` instance, the list of inline formsets and a boolean value based on whether the parent is being added or changed, save the related objects to the database.
to_field_allowed	Return True if the model associated with this admin should be allowed to be referenced by the specified field.

6.1.1 action_checkbox

EquationPlugin.**action_checkbox**(*obj*)

A list_display column containing a checkbox widget.

6.1.2 add_view

EquationPlugin.**add_view**(*request*, *form_url=''*, *extra_context=None*)

6.1.3 autocomplete_view

EquationPlugin.**autocomplete_view**(*request*)

6.1.4 change_view

EquationPlugin.**change_view**(*request*, *object_id*, *form_url=''*, *extra_context=None*)

6.1.5 changeform_view

EquationPlugin.**changeform_view**(*request*, *object_id=None*, *form_url=''*, *extra_context=None*)

6.1.6 changelist_view

EquationPlugin.**changelist_view**(*request*, *extra_context=None*)
The 'change list' admin view for this model.

6.1.7 check

EquationPlugin.**check**(***kwargs*)

6.1.8 construct_change_message

EquationPlugin.**construct_change_message**(*request*, *form*, *formsets*, *add=False*)
Construct a JSON structure describing changes from a changed object.

6.1.9 delete_model

EquationPlugin.**delete_model**(*request*, *obj*)
Given a model instance delete it from the database.

6.1.10 delete_queryset

EquationPlugin.**delete_queryset**(*request*, *queryset*)
Given a queryset, delete it from the database.

6.1.11 delete_view

EquationPlugin.**delete_view**(*request*, *object_id*, *extra_context=None*)

6.1.12 formfield_for_choice_field

EquationPlugin.**formfield_for_choice_field**(*db_field*, *request*, ***kwargs*)
Get a form Field for a database Field that has declared choices.

6.1.13 formfield_for_dbfield

EquationPlugin.**formfield_for_dbfield**(*db_field*, *request*, ***kwargs*)
Hook for specifying the form Field instance for a given database Field instance.

If kwargs are given, they're passed to the form Field's constructor.

6.1.14 formfield_for_foreignkey

EquationPlugin.**formfield_for_foreignkey**(*db_field*, *request*, ***kwargs*)
Get a form Field for a ForeignKey.

6.1.15 formfield_for_manytomany

EquationPlugin.**formfield_for_manytomany**(*db_field*, *request*, ***kwargs*)
Get a form Field for a ManyToManyField.

6.1.16 get_action

EquationPlugin.**get_action**(*action*)
Return a given action from a parameter, which can either be a callable, or the name of a method on the ModelAdmin. Return is a tuple of (callable, name, description).

6.1.17 get_action_choices

EquationPlugin.**get_action_choices**(*request*, *default_choices=[('', '---------')]*)
Return a list of choices for use in a form object. Each choice is a tuple (name, description).

6.1.18 get_actions

EquationPlugin.**get_actions**(*request*)
Return a dictionary mapping the names of all actions for this ModelAdmin to a tuple of (callable, name, description) for each action.

6.1.19 get_autocomplete_fields

EquationPlugin.**get_autocomplete_fields**(*request*)
Return a list of ForeignKey and/or ManyToMany fields which should use an autocomplete widget.

6.1.20 get_cache_expiration

EquationPlugin.**get_cache_expiration**(*request*, *instance*, *placeholder*)

Provides hints to the placeholder, and in turn to the page for determining the appropriate Cache-Control headers to add to the HTTPResponse object.

Must return one of:

- None: This means the placeholder and the page will not even consider this plugin when calculating the page expiration;
- A TZ-aware *datetime* of a specific date and time in the future when this plugin's content expires;
- A *datetime.timedelta* instance indicating how long, relative to the response timestamp that the content can be cached;
- An integer number of seconds that this plugin's content can be cached.

There are constants are defined in *cms.constants* that may be helpful:

- *EXPIRE_NOW*
- *MAX_EXPIRATION_TTL*

An integer value of 0 (zero) or *EXPIRE_NOW* effectively means "do not cache". Negative values will be treated as *EXPIRE_NOW*. Values exceeding the value *MAX_EXPIRATION_TTL* will be set to that value.

Negative *timedelta* values or those greater than *MAX_EXPIRATION_TTL* will also be ranged in the same manner.

Similarly, *datetime* values earlier than now will be treated as *EXPIRE_NOW*. Values greater than *MAX_EXPIRATION_TTL* seconds in the future will be treated as *MAX_EXPIRATION_TTL* seconds in the future.

6.1.21 get_changeform_initial_data

EquationPlugin.**get_changeform_initial_data**(*request*)

Get the initial form data from the request's GET params.

6.1.22 get_changelist

EquationPlugin.**get_changelist**(*request*, ***kwargs*)

Return the ChangeList class for use on the changelist page.

6.1.23 get_changelist_form

EquationPlugin.**get_changelist_form**(*request*, ***kwargs*)

Return a Form class for use in the Formset on the changelist page.

6.1.24 get_changelist_formset

EquationPlugin.**get_changelist_formset**(*request*, ***kwargs*)
Return a FormSet class for use on the changelist page if list_editable is used.

6.1.25 get_changelist_instance

EquationPlugin.**get_changelist_instance**(*request*)
Return a *ChangeList* instance based on *request*. May raise *IncorrectLookupParameters*.

6.1.26 get_child_class_overrides

classmethod EquationPlugin.**get_child_class_overrides**(*slot*, *page*)
Returns a list of plugin types that are allowed as children of this plugin.

6.1.27 get_child_classes

classmethod EquationPlugin.**get_child_classes**(*slot*, *page*, *instance=None*)
Returns a list of plugin types that can be added as children to this plugin.

6.1.28 get_child_plugin_candidates

classmethod EquationPlugin.**get_child_plugin_candidates**(*slot*, *page*)
Returns a list of all plugin classes that will be considered when fetching all available child classes for this plugin.

6.1.29 get_deleted_objects

EquationPlugin.**get_deleted_objects**(*objs*, *request*)
Hook for customizing the delete process for the delete view and the "delete selected" action.

6.1.30 get_empty_change_form_text

classmethod EquationPlugin.**get_empty_change_form_text**(*obj=None*)
Returns the text displayed to the user when editing a plugin that requires no configuration.

6.1.31 get_empty_value_display

EquationPlugin.**get_empty_value_display**()
Return the empty_value_display set on ModelAdmin or AdminSite.

6.1.32 get_exclude

EquationPlugin.**get_exclude**(*request*, *obj=None*)
Hook for specifying exclude.

6.1.33 get_extra_placeholder_menu_items

classmethod EquationPlugin.**get_extra_placeholder_menu_items**(*request*, *placeholder*)

6.1.34 get_extra_plugin_menu_items

classmethod EquationPlugin.**get_extra_plugin_menu_items**(*request*, *plugin*)

6.1.35 get_field_queryset

EquationPlugin.**get_field_queryset**(*db*, *db_field*, *request*)
If the ModelAdmin specifies ordering, the queryset should respect that ordering. Otherwise don't specify the queryset, let the field decide (return None in that case).

6.1.36 get_fields

EquationPlugin.**get_fields**(*request*, *obj=None*)
Hook for specifying fields.

6.1.37 get_fieldsets

EquationPlugin.**get_fieldsets**(*request*, *obj=None*)
Same as from base class except if there are no fields, show an info message.

6.1.38 get_form

EquationPlugin.**get_form**(*request*, *obj=None*, *change=False*, ***kwargs*)
Return a Form class for use in the admin add view. This is used by add_view and change_view.

6.1.39 get_formsets_with_inlines

EquationPlugin.**get_formsets_with_inlines**(*request*, *obj=None*)
Yield formsets and the corresponding inlines.

6.1.40 get_inline_formsets

EquationPlugin.**get_inline_formsets**(*request*, *formsets*, *inline_instances*, *obj=None*)

6.1.41 get_inline_instances

EquationPlugin.**get_inline_instances**(*request*, *obj=None*)

6.1.42 get_list_display

EquationPlugin.**get_list_display**(*request*)
Return a sequence containing the fields to be displayed on the changelist.

6.1.43 get_list_display_links

EquationPlugin.**get_list_display_links**(*request*, *list_display*)
Return a sequence containing the fields to be displayed as links on the changelist. The list_display parameter is the list of fields returned by get_list_display().

6.1.44 get_list_filter

EquationPlugin.**get_list_filter**(*request*)
Return a sequence containing the fields to be displayed as filters in the right sidebar of the changelist page.

6.1.45 get_list_select_related

EquationPlugin.**get_list_select_related**(*request*)
Return a list of fields to add to the select_related() part of the changelist items query.

6.1.46 get_model_perms

EquationPlugin.**get_model_perms**(*request*)
Return a dict of all perms for this model. This dict has the keys `add`, `change`, `delete`, and `view` mapping to the True/False for each of those actions.

6.1.47 get_object

EquationPlugin.**get_object**(*request*, *object_id*, *from_field=None*)
Return an instance matching the field and value provided, the primary key is used if no field is provided. Return `None` if no match is found or the object_id fails validation.

6.1.48 get_ordering

EquationPlugin.**get_ordering**(*request*)
Hook for specifying field ordering.

6.1.49 get_paginator

EquationPlugin.**get_paginator**(*request*, *queryset*, *per_page*, *orphans=0*, *allow_empty_first_page=True*)

6.1.50 get_parent_classes

classmethod EquationPlugin.**get_parent_classes**(*slot*, *page*, *instance=None*)

6.1.51 get_plugin_urls

EquationPlugin.**get_plugin_urls**()
Return URL patterns for which the plugin wants to register views for.

6.1.52 get_prepopulated_fields

EquationPlugin.**get_prepopulated_fields**(*request*, *obj=None*)
Hook for specifying custom prepopulated fields.

6.1.53 get_preserved_filters

EquationPlugin.**get_preserved_filters**(*request*)
Return the preserved filters querystring.

6.1.54 get_queryset

EquationPlugin.**get_queryset**(*request*)
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

6.1.55 get_readonly_fields

EquationPlugin.**get_readonly_fields**(*request*, *obj=None*)
Hook for specifying custom readonly fields.

6.1.56 get_render_queryset

classmethod EquationPlugin.**get_render_queryset**()

6.1.57 get_require_parent

classmethod EquationPlugin.**get_require_parent**(*slot*, *page*)

6.1.58 get_search_fields

EquationPlugin.**get_search_fields**(*request*)
Return a sequence containing the fields to be searched whenever somebody submits a search query.

6.1.59 get_search_results

EquationPlugin.**get_search_results**(*request*, *queryset*, *search_term*)
Return a tuple containing a queryset to implement the search and a boolean indicating if the results may contain duplicates.

6.1.60 get_sortable_by

EquationPlugin.**get_sortable_by**(*request*)
Hook for specifying which fields can be sorted in the changelist.

6.1.61 get_urls

EquationPlugin.**get_urls**()

6.1.62 get_vary_cache_on

EquationPlugin.**get_vary_cache_on**(*request*, *instance*, *placeholder*)
Provides hints to the placeholder, and in turn to the page for determining VARY headers for the response.

Must return one of:

- None (default),
- String of a case-sensitive header name, or
- iterable of case-sensitive header names.

NOTE: This only makes sense to use with caching. If this plugin has `cache = False` or plugin.get_cache_expiration(...) returns 0, get_vary_cache_on() will have no effect.

6.1.63 get_view_on_site_url

EquationPlugin.**get_view_on_site_url**(*obj=None*)

6.1.64 has_add_permission

EquationPlugin.**has_add_permission**(*request*)

Return True if the given request has permission to add an object. Can be overridden by the user in subclasses.

6.1.65 has_change_permission

EquationPlugin.**has_change_permission**(*request*, *obj=None*)

Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to change the *obj* model instance. If *obj* is None, this should return True if the given request has permission to change *any* object of the given type.

6.1.66 has_delete_permission

EquationPlugin.**has_delete_permission**(*request*, *obj=None*)

Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to delete the *obj* model instance. If *obj* is None, this should return True if the given request has permission to delete *any* object of the given type.

6.1.67 has_module_permission

EquationPlugin.**has_module_permission**(*request*)

Return True if the given request has any permission in the given app label.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to view the module on the admin index page and access the module's index page. Overriding it does not restrict access to the add, change or delete views. Use *ModelAdmin.has_(add|change|delete)_permission* for that.

6.1.68 has_view_or_change_permission

EquationPlugin.**has_view_or_change_permission**(*request*, *obj=None*)

6.1.69 has_view_permission

EquationPlugin.**has_view_permission**(*request*, *obj=None*)

Return True if the given request has permission to view the given Django model instance. The default implementation doesn't examine the *obj* parameter.

If overridden by the user in subclasses, it should return True if the given request has permission to view the *obj* model instance. If *obj* is None, it should return True if the request has permission to view any object of the given type.

6.1.70 history_view

EquationPlugin.**history_view**(*request*, *object_id*, *extra_context=None*)

The 'history' admin view for this model.

6.1.71 icon_alt

EquationPlugin.**icon_alt**(*instance*)

Return the alt text for the shown icon.

This is used in django-cms==3.4 only.

Parameters **instance** (EquationPluginModel) – Instance of the plugins Model

Returns Path to the icon.

Return type str

See also:

icon_src

6.1.72 icon_src

EquationPlugin.**icon_src**(*instance*)

Return the path to an icon which is shown in the text editor.

This is used in django-cms==3.4 only.

Parameters **instance** (EquationPluginModel) – Instance of the plugins Model

Returns Path to the icon.

Return type str

6.1.73 is_in_text_editor

EquationPlugin.**is_in_text_editor**(*instance*)

Check if the plugin was added to a text plugin.

Parameters **instance** (EquationPluginModel) – Instance of the plugins Model

Returns True if the plugin was added to a text plugin ('djangocms-text-ckeditor') or False if it was added to page as stand alone element.

Return type bool

6.1.74 log_addition

EquationPlugin.**log_addition**(*request*, *obj*, *bypass=None*)
Log that an object has been successfully added.

The default implementation creates an admin LogEntry object.

6.1.75 log_change

EquationPlugin.**log_change**(*request*, *obj*, *message*, *bypass=None*)
Log that an object has been successfully changed.

The default implementation creates an admin LogEntry object.

6.1.76 log_deletion

EquationPlugin.**log_deletion**(*request*, *obj*, *object_repr*, *bypass=None*)
Log that an object will be deleted. Note that this method must be called before the deletion.

The default implementation creates an admin LogEntry object.

6.1.77 lookup_allowed

EquationPlugin.**lookup_allowed**(*lookup*, *value*)

6.1.78 message_user

EquationPlugin.**message_user**(*request*, *message*, *level=20*, *extra_tags=''*, *fail_silently=False*)
Send a message to the user. The default implementation posts a message using the django.contrib.messages backend.

Exposes almost the same API as messages.add_message(), but accepts the positional arguments in a different order to maintain backwards compatibility. For convenience, it accepts the *level* argument as a string rather than the usual level number.

6.1.79 render

EquationPlugin.**render**(*context*, *instance*, *placeholder*)
Render the Plugin with self.render_template and the data in instance.

Parameters

- **context** (*dict*) – [description]
- **instance** (EquationPluginModel) – Instance of the plugins Model
- **placeholder** (*str*) – [description]

Returns [description]

Return type dict

6.1.80 render_change_form

EquationPlugin.**render_change_form**(*request*, *context*, *add=False*, *change=False*, *form_url=''*, *obj=None*)

We just need the popup interface here

6.1.81 render_close_frame

EquationPlugin.**render_close_frame**(*request*, *obj*, *extra_context=None*)

6.1.82 render_delete_form

EquationPlugin.**render_delete_form**(*request*, *context*)

6.1.83 requires_parent_plugin

classmethod EquationPlugin.**requires_parent_plugin**(*slot*, *page*)

6.1.84 response_action

EquationPlugin.**response_action**(*request*, *queryset*)

Handle an admin action. This is called if a request is POSTed to the changelist; it returns an HttpResponse if the action was handled, and None otherwise.

6.1.85 response_add

EquationPlugin.**response_add**(*request*, *obj*, ***kwargs*)

Determine the HttpResponse for the add_view stage.

6.1.86 response_change

EquationPlugin.**response_change**(*request*, *obj*)

Determine the HttpResponse for the change_view stage.

6.1.87 response_delete

EquationPlugin.**response_delete**(*request*, *obj_display*, *obj_id*)

Determine the HttpResponse for the delete_view stage.

6.1.88 response_post_save_add

EquationPlugin.**response_post_save_add**(*request*, *obj*)
Figure out where to redirect after the 'Save' button has been pressed when adding a new object.

6.1.89 response_post_save_change

EquationPlugin.**response_post_save_change**(*request*, *obj*)
Figure out where to redirect after the 'Save' button has been pressed when editing an existing object.

6.1.90 save_form

EquationPlugin.**save_form**(*request*, *form*, *change*)
Given a ModelForm return an unsaved instance. `change` is True if the object is being changed, and False if it's being added.

6.1.91 save_formset

EquationPlugin.**save_formset**(*request*, *form*, *formset*, *change*)
Given an inline formset save it to the database.

6.1.92 save_model

EquationPlugin.**save_model**(*request*, *obj*, *form*, *change*)
Override original method, and add some attributes to obj This have to be made, because if object is newly created, he must know where he lives.

6.1.93 save_related

EquationPlugin.**save_related**(*request*, *form*, *formsets*, *change*)
Given the `HttpRequest`, the parent `ModelForm` instance, the list of inline formsets and a boolean value based on whether the parent is being added or changed, save the related objects to the database. Note that at this point save_form() and save_model() have already been called.

6.1.94 to_field_allowed

EquationPlugin.**to_field_allowed**(*request*, *to_field*)
Return True if the model associated with this admin should be allowed to be referenced by the specified field.

Methods Documentation

action_checkbox(*obj*)
A list_display column containing a checkbox widget.

add_view(*request*, *form_url=''*, *extra_context=None*)

autocomplete_view(*request*)

change_view(*request*, *object_id*, *form_url=''*, *extra_context=None*)

changeform_view(*request*, *object_id=None*, *form_url=''*, *extra_context=None*)

changelist_view(*request*, *extra_context=None*)
The 'change list' admin view for this model.

check(***kwargs*)

construct_change_message(*request*, *form*, *formsets*, *add=False*)
Construct a JSON structure describing changes from a changed object.

delete_model(*request*, *obj*)
Given a model instance delete it from the database.

delete_queryset(*request*, *queryset*)
Given a queryset, delete it from the database.

delete_view(*request*, *object_id*, *extra_context=None*)

formfield_for_choice_field(*db_field*, *request*, ***kwargs*)
Get a form Field for a database Field that has declared choices.

formfield_for_dbfield(*db_field*, *request*, ***kwargs*)
Hook for specifying the form Field instance for a given database Field instance.

If kwargs are given, they're passed to the form Field's constructor.

formfield_for_foreignkey(*db_field*, *request*, ***kwargs*)
Get a form Field for a ForeignKey.

formfield_for_manytomany(*db_field*, *request*, ***kwargs*)
Get a form Field for a ManyToManyField.

get_action(*action*)
Return a given action from a parameter, which can either be a callable, or the name of a method on the ModelAdmin. Return is a tuple of (callable, name, description).

get_action_choices(*request*, *default_choices=[('', '---------')]*)
Return a list of choices for use in a form object. Each choice is a tuple (name, description).

get_actions(*request*)
Return a dictionary mapping the names of all actions for this ModelAdmin to a tuple of (callable, name, description) for each action.

get_autocomplete_fields(*request*)
Return a list of ForeignKey and/or ManyToMany fields which should use an autocomplete widget.

get_cache_expiration(*request*, *instance*, *placeholder*)
Provides hints to the placeholder, and in turn to the page for determining the appropriate Cache-Control headers to add to the HTTPResponse object.

Must return one of:

- None: This means the placeholder and the page will not even consider this plugin when calculating the page expiration;

- A TZ-aware *datetime* of a specific date and time in the future when this plugin's content expires;
- A *datetime.timedelta* instance indicating how long, relative to the response timestamp that the content can be cached;
- An integer number of seconds that this plugin's content can be cached.

There are constants are defined in *cms.constants* that may be helpful:

- *EXPIRE_NOW*
- *MAX_EXPIRATION_TTL*

An integer value of 0 (zero) or *EXPIRE_NOW* effectively means "do not cache". Negative values will be treated as *EXPIRE_NOW*. Values exceeding the value *MAX_EXPIRATION_TTL* will be set to that value.

Negative *timedelta* values or those greater than *MAX_EXPIRATION_TTL* will also be ranged in the same manner.

Similarly, *datetime* values earlier than now will be treated as *EXPIRE_NOW*. Values greater than *MAX_EXPIRATION_TTL* seconds in the future will be treated as *MAX_EXPIRATION_TTL* seconds in the future.

get_changeform_initial_data(*request*)
Get the initial form data from the request's GET params.

get_changelist(*request*, ***kwargs*)
Return the ChangeList class for use on the changelist page.

get_changelist_form(*request*, ***kwargs*)
Return a Form class for use in the Formset on the changelist page.

get_changelist_formset(*request*, ***kwargs*)
Return a FormSet class for use on the changelist page if list_editable is used.

get_changelist_instance(*request*)
Return a *ChangeList* instance based on *request*. May raise *IncorrectLookupParameters*.

classmethod get_child_class_overrides(*slot*, *page*)
Returns a list of plugin types that are allowed as children of this plugin.

classmethod get_child_classes(*slot*, *page*, *instance=None*)
Returns a list of plugin types that can be added as children to this plugin.

classmethod get_child_plugin_candidates(*slot*, *page*)
Returns a list of all plugin classes that will be considered when fetching all available child classes for this plugin.

get_deleted_objects(*objs*, *request*)
Hook for customizing the delete process for the delete view and the "delete selected" action.

classmethod get_empty_change_form_text(*obj=None*)
Returns the text displayed to the user when editing a plugin that requires no configuration.

get_empty_value_display()
Return the empty_value_display set on ModelAdmin or AdminSite.

get_exclude(*request*, *obj=None*)
Hook for specifying exclude.

classmethod get_extra_placeholder_menu_items(*request*, *placeholder*)

classmethod get_extra_plugin_menu_items(*request*, *plugin*)

get_field_queryset(*db*, *db_field*, *request*)
If the ModelAdmin specifies ordering, the queryset should respect that ordering. Otherwise don't specify the queryset, let the field decide (return None in that case).

get_fields(*request*, *obj=None*)
Hook for specifying fields.

get_fieldsets(*request*, *obj=None*)
Same as from base class except if there are no fields, show an info message.

get_form(*request*, *obj=None*, *change=False*, ***kwargs*)
Return a Form class for use in the admin add view. This is used by add_view and change_view.

get_formsets_with_inlines(*request*, *obj=None*)
Yield formsets and the corresponding inlines.

get_inline_formsets(*request*, *formsets*, *inline_instances*, *obj=None*)

get_inline_instances(*request*, *obj=None*)

get_list_display(*request*)
Return a sequence containing the fields to be displayed on the changelist.

get_list_display_links(*request*, *list_display*)
Return a sequence containing the fields to be displayed as links on the changelist. The list_display parameter is the list of fields returned by get_list_display().

get_list_filter(*request*)
Return a sequence containing the fields to be displayed as filters in the right sidebar of the changelist page.

get_list_select_related(*request*)
Return a list of fields to add to the select_related() part of the changelist items query.

get_model_perms(*request*)
Return a dict of all perms for this model. This dict has the keys `add`, `change`, `delete`, and `view` mapping to the True/False for each of those actions.

get_object(*request*, *object_id*, *from_field=None*)
Return an instance matching the field and value provided, the primary key is used if no field is provided. Return `None` if no match is found or the object_id fails validation.

get_ordering(*request*)
Hook for specifying field ordering.

get_paginator(*request*, *queryset*, *per_page*, *orphans=0*, *allow_empty_first_page=True*)

classmethod get_parent_classes(*slot*, *page*, *instance=None*)

get_plugin_urls()
Return URL patterns for which the plugin wants to register views for.

get_prepopulated_fields(*request*, *obj=None*)
Hook for specifying custom prepopulated fields.

get_preserved_filters(*request*)
Return the preserved filters querystring.

get_queryset(*request*)
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

get_readonly_fields(*request*, *obj=None*)
Hook for specifying custom readonly fields.

classmethod get_render_queryset()

classmethod get_require_parent(*slot*, *page*)

get_search_fields(*request*)
Return a sequence containing the fields to be searched whenever somebody submits a search query.

get_search_results(*request*, *queryset*, *search_term*)
Return a tuple containing a queryset to implement the search and a boolean indicating if the results may contain duplicates.

get_sortable_by(*request*)
Hook for specifying which fields can be sorted in the changelist.

get_urls()

get_vary_cache_on(*request*, *instance*, *placeholder*)
Provides hints to the placeholder, and in turn to the page for determining VARY headers for the response.

Must return one of:

- None (default),
- String of a case-sensitive header name, or
- iterable of case-sensitive header names.

NOTE: This only makes sense to use with caching. If this plugin has `cache = False` or plugin.get_cache_expiration(...) returns 0, get_vary_cache_on() will have no effect.

get_view_on_site_url(*obj=None*)

has_add_permission(*request*)
Return True if the given request has permission to add an object. Can be overridden by the user in subclasses.

has_change_permission(*request*, *obj=None*)
Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to change the *obj* model instance. If *obj* is None, this should return True if the given request has permission to change *any* object of the given type.

has_delete_permission(*request*, *obj=None*)
Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to delete the *obj* model instance. If *obj* is None, this should return True if the given request has permission to delete *any* object of the given type.

has_module_permission(*request*)
Return True if the given request has any permission in the given app label.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to view the module on the admin index page and access the module's index page. Overriding it does not restrict access to the add, change or delete views. Use *ModelAdmin.has_(add|change|delete)_permission* for that.

has_view_or_change_permission(*request*, *obj=None*)

has_view_permission(*request*, *obj=None*)
Return True if the given request has permission to view the given Django model instance. The default implementation doesn't examine the *obj* parameter.

If overridden by the user in subclasses, it should return True if the given request has permission to view the *obj* model instance. If *obj* is None, it should return True if the request has permission to view any object of the given type.

history_view(*request*, *object_id*, *extra_context=None*)
The 'history' admin view for this model.

icon_alt(*instance*)
Return the alt text for the shown icon.

This is used in django-cms==3.4 only.

Parameters **instance** (EquationPluginModel) – Instance of the plugins Model

Returns Path to the icon.

Return type str

See also:

icon_src

icon_src(*instance*)
Return the path to an icon which is shown in the text editor.

This is used in django-cms==3.4 only.

Parameters **instance** (EquationPluginModel) – Instance of the plugins Model

Returns Path to the icon.

Return type str

is_in_text_editor(*instance*)
Check if the plugin was added to a text plugin.

Parameters **instance** (EquationPluginModel) – Instance of the plugins Model

Returns True if the plugin was added to a text plugin ('djangocms-text-ckeditor') or False if it was added to page as stand alone element.

Return type bool

log_addition(*request*, *obj*, *bypass=None*)
Log that an object has been successfully added.

The default implementation creates an admin LogEntry object.

log_change(*request*, *obj*, *message*, *bypass=None*)
Log that an object has been successfully changed.

The default implementation creates an admin LogEntry object.

log_deletion(*request*, *obj*, *object_repr*, *bypass=None*)
Log that an object will be deleted. Note that this method must be called before the deletion.

The default implementation creates an admin LogEntry object.

lookup_allowed(*lookup*, *value*)

message_user(*request*, *message*, *level=20*, *extra_tags=''*, *fail_silently=False*)
Send a message to the user. The default implementation posts a message using the django.contrib.messages backend.

Exposes almost the same API as messages.add_message(), but accepts the positional arguments in a different order to maintain backwards compatibility. For convenience, it accepts the *level* argument as a string rather than the usual level number.

render(*context*, *instance*, *placeholder*)
Render the Plugin with self.render_template and the data in instance.

Parameters

- **context** (*dict*) – [description]
- **instance** (EquationPluginModel) – Instance of the plugins Model
- **placeholder** (*str*) – [description]

Returns [description]

Return type dict

render_change_form(*request*, *context*, *add=False*, *change=False*, *form_url=''*, *obj=None*)
We just need the popup interface here

render_close_frame(*request*, *obj*, *extra_context=None*)

render_delete_form(*request*, *context*)

classmethod requires_parent_plugin(*slot*, *page*)

response_action(*request*, *queryset*)
Handle an admin action. This is called if a request is POSTed to the changelist; it returns an HttpResponse if the action was handled, and None otherwise.

response_add(*request*, *obj*, ***kwargs*)
Determine the HttpResponse for the add_view stage.

response_change(*request*, *obj*)
Determine the HttpResponse for the change_view stage.

response_delete(*request*, *obj_display*, *obj_id*)
Determine the HttpResponse for the delete_view stage.

response_post_save_add(*request*, *obj*)
Figure out where to redirect after the 'Save' button has been pressed when adding a new object.

response_post_save_change(*request*, *obj*)
Figure out where to redirect after the 'Save' button has been pressed when editing an existing object.

save_form(*request*, *form*, *change*)
Given a ModelForm return an unsaved instance. `change` is True if the object is being changed, and False if it's being added.

save_formset(*request*, *form*, *formset*, *change*)
Given an inline formset save it to the database.

save_model(*request*, *obj*, *form*, *change*)
Override original method, and add some attributes to obj This have to be made, because if object is newly created, he must know where he lives.

save_related(*request*, *form*, *formsets*, *change*)
Given the `HttpRequest`, the parent `ModelForm` instance, the list of inline formsets and a boolean value based on whether the parent is being added or changed, save the related objects to the database. Note that at this point save_form() and save_model() have already been called.

to_field_allowed(*request*, *to_field*)
Return True if the model associated with this admin should be allowed to be referenced by the specified field.

6.2 EquationForm

class EquationForm(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)
Inputs which are used in the Equation editing dialog.

Attributes Summary

base_fields	
changed_data	
declared_fields	
default_renderer	
errors	Return an ErrorDict for the data provided for the form.
field_order	
media	Return all media required to render the widgets on this form.
prefix	
use_required_attribute	

Methods Summary

add_error	Update the content of *self._errors*.
add_initial_prefix	Add a 'initial' prefix for checking dynamic initial values.
add_prefix	Return the field name with a prefix appended, if this Form has a prefix set.
as_p	Return this form rendered as HTML <p>s.
as_table	Return this form rendered as HTML <tr>s -- excluding the <table></table>.
as_ul	Return this form rendered as HTML s -- excluding the .
clean	Hook for doing any extra form-wide cleaning after Field.clean() has been called on every field.

continues on next page

Table 5 – continued from previous page

full_clean	Clean all of self.data and populate self._errors and self.cleaned_data.
get_initial_for_field	Return initial data for field on form.
has_changed	Return True if data differs from initial.
has_error	
hidden_fields	Return a list of all the BoundField objects that are hidden fields.
is_multipart	Return True if the form needs to be multipart-encoded, i.e. it has FileInput, or False otherwise.
is_valid	Return True if the form has no errors, or False otherwise.
non_field_errors	Return an ErrorList of errors that aren't associated with a particular field -- i.e., from Form.clean().
order_fields	Rearrange the fields according to field_order.
save	Save this form's self.instance object if commit=True.
validate_unique	Call the instance's validate_unique() method and update the form's validation errors if any were raised.
visible_fields	Return a list of BoundField objects that aren't hidden fields.

6.2.1 add_error

EquationForm.**add_error**(*field*, *error*)

Update the content of *self._errors*.

The *field* argument is the name of the field to which the errors should be added. If it's None, treat the errors as NON_FIELD_ERRORS.

The *error* argument can be a single error, a list of errors, or a dictionary that maps field names to lists of errors. An "error" can be either a simple string or an instance of ValidationError with its message attribute set and a "list or dictionary" can be an actual *list* or *dict* or an instance of ValidationError with its *error_list* or *error_dict* attribute set.

If *error* is a dictionary, the *field* argument *must* be None and errors will be added to the fields that correspond to the keys of the dictionary.

6.2.2 add_initial_prefix

EquationForm.**add_initial_prefix**(*field_name*)
Add a 'initial' prefix for checking dynamic initial values.

6.2.3 add_prefix

EquationForm.**add_prefix**(*field_name*)
Return the field name with a prefix appended, if this Form has a prefix set.

Subclasses may wish to override.

6.2.4 as_p

EquationForm.**as_p**()
Return this form rendered as HTML <p>s.

6.2.5 as_table

EquationForm.**as_table**()
Return this form rendered as HTML <tr>s – excluding the <table></table>.

6.2.6 as_ul

EquationForm.**as_ul**()
Return this form rendered as HTML s – excluding the .

6.2.7 clean

EquationForm.**clean**()
Hook for doing any extra form-wide cleaning after Field.clean() has been called on every field. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field named '__all__'.

6.2.8 full_clean

EquationForm.**full_clean**()
Clean all of self.data and populate self._errors and self.cleaned_data.

6.2.9 get_initial_for_field

EquationForm.**get_initial_for_field**(*field*, *field_name*)
Return initial data for field on form. Use initial data from the form or the field, in that order. Evaluate callable values.

6.2.10 has_changed

EquationForm.**has_changed**()
Return True if data differs from initial.

6.2.11 has_error

EquationForm.**has_error**(*field*, *code=None*)

6.2.12 hidden_fields

EquationForm.**hidden_fields**()
Return a list of all the BoundField objects that are hidden fields. Useful for manual form layout in templates.

6.2.13 is_multipart

EquationForm.**is_multipart**()
Return True if the form needs to be multipart-encoded, i.e. it has FileInput, or False otherwise.

6.2.14 is_valid

EquationForm.**is_valid**()
Return True if the form has no errors, or False otherwise.

6.2.15 non_field_errors

EquationForm.**non_field_errors**()
Return an ErrorList of errors that aren't associated with a particular field – i.e., from Form.clean(). Return an empty ErrorList if there are none.

6.2.16 order_fields

EquationForm.**order_fields**(*field_order*)
Rearrange the fields according to field_order.

field_order is a list of field names specifying the order. Append fields not included in the list in the default order for backward compatibility with subclasses not overriding field_order. If field_order is None, keep all fields in the order defined in the class. Ignore unknown fields in field_order to allow disabling fields in form subclasses without redefining ordering.

6.2.17 save

EquationForm.**save**(*commit=True*)
Save this form's self.instance object if commit=True. Otherwise, add a save_m2m() method to the form which can be called after the instance is saved manually at a later time. Return the model instance.

6.2.18 validate_unique

EquationForm.**validate_unique**()
Call the instance's validate_unique() method and update the form's validation errors if any were raised.

6.2.19 visible_fields

EquationForm.**visible_fields**()
Return a list of BoundField objects that aren't hidden fields. The opposite of the hidden_fields() method.

Methods Documentation

add_error(*field*, *error*)
Update the content of *self._errors*.

The *field* argument is the name of the field to which the errors should be added. If it's None, treat the errors as NON_FIELD_ERRORS.

The *error* argument can be a single error, a list of errors, or a dictionary that maps field names to lists of errors. An "error" can be either a simple string or an instance of ValidationError with its message attribute set and a "list or dictionary" can be an actual *list* or *dict* or an instance of ValidationError with its *error_list* or *error_dict* attribute set.

If *error* is a dictionary, the *field* argument *must* be None and errors will be added to the fields that correspond to the keys of the dictionary.

add_initial_prefix(*field_name*)
Add a 'initial' prefix for checking dynamic initial values.

add_prefix(*field_name*)
Return the field name with a prefix appended, if this Form has a prefix set.

Subclasses may wish to override.

as_p()
Return this form rendered as HTML <p>s.

as_table()
Return this form rendered as HTML <tr>s – excluding the <table></table>.

as_ul()
Return this form rendered as HTML s – excluding the .

clean()
Hook for doing any extra form-wide cleaning after Field.clean() has been called on every field. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field named '__all__'.

full_clean()
Clean all of self.data and populate self._errors and self.cleaned_data.

get_initial_for_field(*field*, *field_name*)
Return initial data for field on form. Use initial data from the form or the field, in that order. Evaluate callable values.

has_changed()
Return True if data differs from initial.

has_error(*field*, *code=None*)

hidden_fields()
Return a list of all the BoundField objects that are hidden fields. Useful for manual form layout in templates.

is_multipart()
Return True if the form needs to be multipart-encoded, i.e. it has FileInput, or False otherwise.

is_valid()
Return True if the form has no errors, or False otherwise.

non_field_errors()
Return an ErrorList of errors that aren't associated with a particular field – i.e., from Form.clean(). Return an empty ErrorList if there are none.

order_fields(*field_order*)
Rearrange the fields according to field_order.

field_order is a list of field names specifying the order. Append fields not included in the list in the default order for backward compatibility with subclasses not overriding field_order. If field_order is None, keep all fields in the order defined in the class. Ignore unknown fields in field_order to allow disabling fields in form subclasses without redefining ordering.

save(*commit=True*)
Save this form's self.instance object if commit=True. Otherwise, add a save_m2m() method to the form which can be called after the instance is saved manually at a later time. Return the model instance.

validate_unique()
Call the instance's validate_unique() method and update the form's validation errors if any were raised.

visible_fields()
Return a list of BoundField objects that aren't hidden fields. The opposite of the hidden_fields() method.

6.3 EquationPluginModel

class EquationPluginModel(**args*, ***kwargs*)
Database model of saved Equations.

Attributes Summary

alias_reference	Accessor to the related objects manager on the reverse side of a many-to-one relation.
alphabet	
changed_date	A wrapper for a deferred-loading field.
child_plugin_instances	

continues on next page

Table 6 – continued from previous page

cms_aliasplugin	Accessor to the related object on the reverse side of a one-to-one relation.
cms_placeholderreference	Accessor to the related object on the reverse side of a one-to-one relation.
cmsplugin_ptr	Accessor to the related object on the forward side of a one-to-one relation.
cmsplugin_ptr_id	A wrapper for a deferred-loading field.
cmsplugin_set	Accessor to the related objects manager on the reverse side of a many-to-one relation.
creation_date	A wrapper for a deferred-loading field.
depth	A wrapper for a deferred-loading field.
djangocms_equation_equationpluginmodel	Accessor to the related object on the reverse side of a one-to-one relation.
djangocms_text_ckeditor_text	Accessor to the related object on the reverse side of a one-to-one relation.
font_size_unit	Value of the font-size with size value font_size_value.
font_size_value	Value of the font-size with unit font_size_unit.
gap	
id	A wrapper for a deferred-loading field.
is_inline	If it should be displayed inline or be stand alone.
language	A wrapper for a deferred-loading field.
node_order_by	
numchild	A wrapper for a deferred-loading field.
numconv_obj_	
objects	
page	
parent	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
parent_id	A wrapper for a deferred-loading field.
path	A wrapper for a deferred-loading field.
pk	
placeholder	Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
placeholder_id	A wrapper for a deferred-loading field.
plugin_type	A wrapper for a deferred-loading field.
position	A wrapper for a deferred-loading field.
steplen	
tex_code	Latex code of the equation.

Methods Summary

add_child	Adds a child to the node.
add_root	Adds a root node to the tree.
add_sibling	Adds a new node as a sibling to the current node object.
check	
clean	Hook for doing any extra model-wide validation after clean() has been called on every field by self.clean_fields.
clean_fields	Clean all fields and raise a ValidationError containing a dict of all validation errors if any occur.
copy_plugin	Copy this plugin and return the new plugin.
copy_relations	Handle copying of any relations attached to this plugin.
date_error_message	
delete	Removes a node and all it's descendants.
dump_bulk	Dumps a tree branch to a python data structure.
find_problems	Checks for problems in the tree structure, problems can occur when:
fix_tree	Fixes the plugin tree by first calling treebeard fix_tree and the recalculating the correct position property for each plugin.
from_db	
full_clean	Call clean_fields(), clean(), and validate_unique() on the model.
get_action_urls	
get_add_url	
get_ancestors	**returns** A queryset containing the current node object's ancestors,
get_annotated_list	Gets an annotated list from a tree branch.
get_annotated_list_qs	Gets an annotated list from a queryset.
get_bound_plugin	Returns an instance of the plugin model configured for this plugin type.
get_breadcrumb	
get_breadcrumb_json	
get_children	**returns** A queryset of all the node's children

continues on next page

Table 7 – continued from previous page

get_children_count	**returns** The number the node's children, calculated in the most
get_copy_url	
get_database_vendor	returns the supported database vendor used by a treebeard model when performing read (select) or write (update, insert, delete) operations.
get_deferred_fields	Return a set containing names of deferred fields for this instance.
get_delete_url	
get_depth	**returns** the depth (level) of the node
get_descendant_count	**returns** the number of descendants of a node.
get_descendants	**returns** A queryset of all the node's descendants as DFS, doesn't
get_descendants_group_count	Helper for a very common case: get a group of siblings and the number of *descendants* in every sibling.
get_edit_url	
get_first_child	**returns**
get_first_root_node	**returns**
get_first_sibling	**returns**
get_font_size_unit_display	
get_foreign_keys	Get foreign keys and models they refer to, so we can pre-process the data for load_bulk
get_instance_icon_alt	Get alt text for instance's icon
get_instance_icon_src	Get src URL for instance's icon
get_last_child	**returns**
get_last_root_node	**returns**

continues on next page

Table 7 – continued from previous page

get_last_sibling	**returns**
get_media_path	
get_move_url	
get_next_by_changed_date	
get_next_by_creation_date	
get_next_sibling	**returns** The next node's sibling, or None if it was the rightmost
get_parent	**returns** the parent node of the current node object.
get_plugin_class	
get_plugin_class_instance	
get_plugin_info	
get_plugin_instance	Given a plugin instance (usually as a CMSPluginBase), this method returns a tuple containing: instance - The instance AS THE APPROPRIATE SUBCLASS OF CMSPluginBase and not necessarily just 'self', which is often just a CMSPluginBase, plugin - the associated plugin class instance (subclass of CMSPlugin)
get_plugin_name	
get_position_in_placeholder	1 based position!
get_prev_sibling	**returns** The previous node's sibling, or None if it was the leftmost
get_previous_by_changed_date	
get_previous_by_creation_date	
get_root	**returns** the root node for the current node object.
get_root_nodes	**returns** A queryset containing the root nodes in the tree.

continues on next page

Table 7 – continued from previous page

get_short_description	
get_siblings	**returns** A queryset of all the node's siblings, including the node
get_sorted_pos_queryset	**returns** A queryset of the nodes that must be moved
get_tree	**returns**
is_child_of	**returns** `True` is the node if a child of another node given as an
is_descendant_of	**returns** `True` if the node is a descendant of another node given
is_leaf	**returns** True if the node is a leaf node (else, returns False)
is_root	**returns** True if the node is a root node (else, returns False)
is_sibling_of	**returns** `True` if the node is a sibling of another node given as an
load_bulk	Loads a list/dictionary structure to the tree.
move	Moves the current node and all it's descendants to a new position relative to another node.
notify_on_autoadd	Method called when we auto add this plugin via default_plugins in CMS_PLACEHOLDER_CONF.
notify_on_autoadd_children	Method called when we auto add children to this plugin via default_plugins/<plugin>/children in CMS_PLACEHOLDER_CONF.
num_children	
numconv_obj	
post_copy	Handle more advanced cases (eg Text Plugins) after the original is copied
prepare_database_save	
refresh_from_db	Reload field values from the database.

continues on next page

Table 7 – continued from previous page

reload	
save	Save the current instance.
save_base	Handle the parts of saving which should be done only once per save, yet need to be done in raw saves, too.
serializable_value	Return the value of the field name for this instance.
set_base_attr	
unique_error_message	
update	
validate_unique	Check unique constraints on the model and raise ValidationError if any failed.

6.3.1 add_child

EquationPluginModel.**add_child**(***kwargs*)

Adds a child to the node.

This method saves the node in database. The object is populated as if via:

` obj = self.__class__(**kwargs) `

Raises **PathOverflow** – when no more child nodes can be added

6.3.2 add_root

classmethod EquationPluginModel.**add_root**(***kwargs*)

Adds a root node to the tree.

This method saves the node in database. The object is populated as if via:

` obj = cls(**kwargs) `

Raises **PathOverflow** – when no more root objects can be added

6.3.3 add_sibling

EquationPluginModel.**add_sibling**(*pos=None*, ***kwargs*)

Adds a new node as a sibling to the current node object.

This method saves the node in database. The object is populated as if via:

` obj = self.__class__(**kwargs) `

Raises **PathOverflow** – when the library can't make room for the node's new position

6.3.4 check

classmethod EquationPluginModel.**check**(***kwargs*)

6.3.5 clean

EquationPluginModel.**clean**()

Hook for doing any extra model-wide validation after clean() has been called on every field by self.clean_fields. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field defined by NON_FIELD_ERRORS.

6.3.6 clean_fields

EquationPluginModel.**clean_fields**(*exclude=None*)

Clean all fields and raise a ValidationError containing a dict of all validation errors if any occur.

6.3.7 copy_plugin

EquationPluginModel.**copy_plugin**(*target_placeholder*, *target_language*, *parent_cache*, *no_signals=False*)

Copy this plugin and return the new plugin.

The logic of this method is the following:

get a new generic plugin instance # assign the position in the plugin tree # save it to let mptt/treebeard calculate the tree attributes # then get a copy of the current plugin instance # assign to it the id of the generic plugin instance above;

this will effectively change the generic plugin created above into a concrete one

 # copy the tree related attributes from the generic plugin to the concrete one

save the concrete plugin # trigger the copy relations # return the generic plugin instance

This copy logic is required because we don't know what the fields of the real plugin are. By getting another instance of it at step 4 and then overwriting its ID at step 5, the ORM will copy the custom fields for us.

6.3.8 copy_relations

EquationPluginModel.**copy_relations**(*old_instance*)

Handle copying of any relations attached to this plugin. Custom plugins have to do this themselves!

6.3.9 date_error_message

EquationPluginModel.**date_error_message**(*lookup_type*, *field_name*, *unique_for*)

6.3.10 delete

EquationPluginModel.**delete**(*no_mp=False*, **args*, ***kwargs*)
Removes a node and all it's descendants.

6.3.11 dump_bulk

classmethod EquationPluginModel.**dump_bulk**(*parent=None*, *keep_ids=True*)
Dumps a tree branch to a python data structure.

6.3.12 find_problems

classmethod EquationPluginModel.**find_problems**()
Checks for problems in the tree structure, problems can occur when:

1. your code breaks and you get incomplete transactions (always use transactions!)
2. changing the `steplen` value in a model (you must *dump_bulk()* first, change `steplen` and then *load_bulk()*

Returns

A tuple of five lists:

1. a list of ids of nodes with characters not found in the `alphabet`
2. a list of ids of nodes when a wrong `path` length according to `steplen`
3. a list of ids of orphaned nodes
4. a list of ids of nodes with the wrong depth value for their path
5. a list of ids nodes that report a wrong number of children

6.3.13 fix_tree

classmethod EquationPluginModel.**fix_tree**(*destructive=False*)
Fixes the plugin tree by first calling treebeard fix_tree and the recalculating the correct position property for each plugin.

6.3.14 from_db

classmethod EquationPluginModel.**from_db**(*db*, *field_names*, *values*)

6.3.15 full_clean

EquationPluginModel.**full_clean**(*exclude=None*, *validate_unique=True*)

Call clean_fields(), clean(), and validate_unique() on the model. Raise a ValidationError for any errors that occur.

6.3.16 get_action_urls

EquationPluginModel.**get_action_urls**(*js_compat=True*)

6.3.17 get_add_url

EquationPluginModel.**get_add_url**()

6.3.18 get_ancestors

EquationPluginModel.**get_ancestors**()

Returns A queryset containing the current node object's ancestors, starting by the root node and descending to the parent.

6.3.19 get_annotated_list

classmethod EquationPluginModel.**get_annotated_list**(*parent=None*, *max_depth=None*)

Gets an annotated list from a tree branch.

Parameters

- **parent** – The node whose descendants will be annotated. The node itself will be included in the list. If not given, the entire tree will be annotated.
- **max_depth** – Optionally limit to specified depth

6.3.20 get_annotated_list_qs

classmethod EquationPluginModel.**get_annotated_list_qs**(*qs*)

Gets an annotated list from a queryset.

6.3.21 get_bound_plugin

EquationPluginModel.**get_bound_plugin**()
Returns an instance of the plugin model configured for this plugin type.

6.3.22 get_breadcrumb

EquationPluginModel.**get_breadcrumb**()

6.3.23 get_breadcrumb_json

EquationPluginModel.**get_breadcrumb_json**()

6.3.24 get_children

EquationPluginModel.**get_children**()

Returns A queryset of all the node's children

6.3.25 get_children_count

EquationPluginModel.**get_children_count**()

Returns The number the node's children, calculated in the most efficient possible way.

6.3.26 get_copy_url

EquationPluginModel.**get_copy_url**()

6.3.27 get_database_vendor

classmethod EquationPluginModel.**get_database_vendor**(*action*)
returns the supported database vendor used by a treebeard model when performing read (select) or write (update, insert, delete) operations.

Parameters **action** – *read* or *write*

Returns postgresql, mysql or sqlite

6.3.28 get_deferred_fields

EquationPluginModel.**get_deferred_fields**()
Return a set containing names of deferred fields for this instance.

6.3.29 get_delete_url

EquationPluginModel.**get_delete_url**()

6.3.30 get_depth

EquationPluginModel.**get_depth**()

Returns the depth (level) of the node

6.3.31 get_descendant_count

EquationPluginModel.**get_descendant_count**()

Returns the number of descendants of a node.

6.3.32 get_descendants

EquationPluginModel.**get_descendants**()

Returns A queryset of all the node's descendants as DFS, doesn't include the node itself

6.3.33 get_descendants_group_count

classmethod EquationPluginModel.**get_descendants_group_count**(*parent=None*)
Helper for a very common case: get a group of siblings and the number of *descendants* in every sibling.

6.3.34 get_edit_url

EquationPluginModel.**get_edit_url**()

6.3.35 get_first_child

EquationPluginModel.**get_first_child**()

Returns The leftmost node's child, or None if it has no children.

6.3.36 get_first_root_node

classmethod EquationPluginModel.**get_first_root_node**()

Returns The first root node in the tree or None if it is empty.

6.3.37 get_first_sibling

EquationPluginModel.**get_first_sibling**()

Returns The leftmost node's sibling, can return the node itself if it was the leftmost sibling.

6.3.38 get_font_size_unit_display

EquationPluginModel.**get_font_size_unit_display**(*, *field=<django.db.models.fields.CharField: font_size_unit>*)

6.3.39 get_foreign_keys

classmethod EquationPluginModel.**get_foreign_keys**()
Get foreign keys and models they refer to, so we can pre-process the data for load_bulk

6.3.40 get_instance_icon_alt

EquationPluginModel.**get_instance_icon_alt**()
Get alt text for instance's icon

6.3.41 get_instance_icon_src

EquationPluginModel.**get_instance_icon_src**()
Get src URL for instance's icon

6.3.42 get_last_child

EquationPluginModel.**get_last_child**()

Returns The rightmost node's child, or None if it has no children.

6.3.43 get_last_root_node

classmethod EquationPluginModel.**get_last_root_node**()

Returns The last root node in the tree or None if it is empty.

6.3.44 get_last_sibling

EquationPluginModel.**get_last_sibling**()

Returns The rightmost node's sibling, can return the node itself if it was the rightmost sibling.

6.3.45 get_media_path

EquationPluginModel.**get_media_path**(*filename*)

6.3.46 get_move_url

EquationPluginModel.**get_move_url**()

6.3.47 get_next_by_changed_date

EquationPluginModel.**get_next_by_changed_date**(*, *field=<django.db.models.fields.DateTimeField: changed_date>*, *is_next=True*, ***kwargs*)

6.3.48 get_next_by_creation_date

EquationPluginModel.**get_next_by_creation_date**(*, *field=<django.db.models.fields.DateTimeField: creation_date>*, *is_next=True*, ***kwargs*)

6.3.49 get_next_sibling

EquationPluginModel.**get_next_sibling**()

Returns The next node's sibling, or None if it was the rightmost sibling.

6.3.50 get_parent

EquationPluginModel.**get_parent**(*update=False*)

Returns the parent node of the current node object. Caches the result in the object itself to help in loops.

6.3.51 get_plugin_class

EquationPluginModel.**get_plugin_class**()

6.3.52 get_plugin_class_instance

EquationPluginModel.**get_plugin_class_instance**(*admin=None*)

6.3.53 get_plugin_info

EquationPluginModel.**get_plugin_info**(*children=None*, *parents=None*)

6.3.54 get_plugin_instance

EquationPluginModel.**get_plugin_instance**(*admin=None*)

Given a plugin instance (usually as a CMSPluginBase), this method returns a tuple containing:

instance - The instance AS THE APPROPRIATE SUBCLASS OF CMSPluginBase and not necessarily just 'self', which is often just a CMSPluginBase,

plugin - the associated plugin class instance (subclass of CMSPlugin)

6.3.55 get_plugin_name

EquationPluginModel.**get_plugin_name**()

6.3.56 get_position_in_placeholder

EquationPluginModel.**get_position_in_placeholder**()

1 based position!

6.3.57 get_prev_sibling

EquationPluginModel.**get_prev_sibling**()

Returns The previous node's sibling, or None if it was the leftmost sibling.

6.3.58 get_previous_by_changed_date

EquationPluginModel.**get_previous_by_changed_date**(*, *field=<django.db.models.fields.DateTimeField: changed_date>*, *is_next=False*, ***kwargs*)

6.3.59 get_previous_by_creation_date

EquationPluginModel.**get_previous_by_creation_date**(*, *field=<django.db.models.fields.DateTimeField: creation_date>*, *is_next=False*, ***kwargs*)

6.3.60 get_root

EquationPluginModel.**get_root**()

Returns the root node for the current node object.

6.3.61 get_root_nodes

classmethod EquationPluginModel.**get_root_nodes**()

Returns A queryset containing the root nodes in the tree.

6.3.62 get_short_description

EquationPluginModel.**get_short_description**()

6.3.63 get_siblings

EquationPluginModel.**get_siblings**()

Returns A queryset of all the node's siblings, including the node itself.

6.3.64 get_sorted_pos_queryset

EquationPluginModel.**get_sorted_pos_queryset**(*siblings*, *newobj*)

Returns A queryset of the nodes that must be moved

to the right. Called only for Node models with `node_order_by`

This function is based on _insertion_target_filters from django-mptt (BSD licensed) by Jonathan Buchanan: https://github.com/django-mptt/django-mptt/blob/0.3.0/mptt/signals.py

6.3.65 get_tree

classmethod EquationPluginModel.**get_tree**(*parent=None*)

Returns A *queryset* of nodes ordered as DFS, including the parent. If no parent is given, the entire tree is returned.

6.3.66 is_child_of

EquationPluginModel.**is_child_of**(*node*)

Returns `True` is the node if a child of another node given as an argument, else, returns `False`

6.3.67 is_descendant_of

EquationPluginModel.**is_descendant_of**(*node*)

Returns `True` if the node is a descendant of another node given as an argument, else, returns `False`

6.3.68 is_leaf

EquationPluginModel.**is_leaf**()

Returns True if the node is a leaf node (else, returns False)

6.3.69 is_root

EquationPluginModel.**is_root**()

Returns True if the node is a root node (else, returns False)

6.3.70 is_sibling_of

EquationPluginModel.**is_sibling_of**(*node*)

Returns `True` if the node is a sibling of another node given as an argument, else, returns `False`

6.3.71 load_bulk

classmethod EquationPluginModel.**load_bulk**(*bulk_data*, *parent=None*, *keep_ids=False*)

Loads a list/dictionary structure to the tree.

Parameters

- **bulk_data** – The data that will be loaded, the structure is a list of dictionaries with 2 keys:
  - `data`: will store arguments that will be passed for object creation, and
  - `children`: a list of dictionaries, each one has it's own `data` and `children` keys (a recursive structure)
- **parent** – The node that will receive the structure as children, if not specified the first level of the structure will be loaded as root nodes
- **keep_ids** – If enabled, loads the nodes with the same primary keys that are given in the structure. Will error if there are nodes without primary key info or if the primary keys are already used.

Returns A list of the added node ids.

6.3.72 move

EquationPluginModel.**move**(*target*, *pos=None*)

Moves the current node and all it's descendants to a new position relative to another node.

Raises **PathOverflow** – when the library can't make room for the node's new position

6.3.73 notify_on_autoadd

EquationPluginModel.**notify_on_autoadd**(*request*, *conf*)

Method called when we auto add this plugin via default_plugins in CMS_PLACEHOLDER_CONF. Some specific plugins may have some special stuff to do when they are auto added.

6.3.74 notify_on_autoadd_children

EquationPluginModel.**notify_on_autoadd_children**(*request*, *conf*, *children*)

Method called when we auto add children to this plugin via default_plugins/<plugin>/children in CMS_PLACEHOLDER_CONF. Some specific plugins may have some special stuff to do when we add children to them. ie : TextPlugin must update its content to add HTML tags to be able to see his children in WYSIWYG.

6.3.75 num_children

EquationPluginModel.**num_children**()

6.3.76 numconv_obj

classmethod EquationPluginModel.**numconv_obj**()

6.3.77 post_copy

EquationPluginModel.**post_copy**(*old_instance*, *new_old_ziplist*)
Handle more advanced cases (eg Text Plugins) after the original is copied

6.3.78 prepare_database_save

EquationPluginModel.**prepare_database_save**(*field*)

6.3.79 refresh_from_db

EquationPluginModel.**refresh_from_db**(**args*, ***kwargs*)
Reload field values from the database.

By default, the reloading happens from the database this instance was loaded from, or by the read router if this instance wasn't loaded from any database. The using parameter will override the default.

Fields can be used to specify which fields to reload. The fields should be an iterable of field attnames. If fields is None, then all non-deferred fields are reloaded.

When accessing deferred fields of an instance, the deferred loading of the field will call this method.

6.3.80 reload

EquationPluginModel.**reload**()

6.3.81 save

EquationPluginModel.**save**(*no_signals=False*, **args*, ***kwargs*)
Save the current instance. Override this in a subclass if you want to control the saving process.

The 'force_insert' and 'force_update' parameters can be used to insist that the "save" must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

6.3.82 save_base

EquationPluginModel.**save_base**(*raw=False*, *force_insert=False*, *force_update=False*, *using=None*, *update_fields=None*)

Handle the parts of saving which should be done only once per save, yet need to be done in raw saves, too. This includes some sanity checks and signal sending.

The 'raw' argument is telling save_base not to save any parent models and not to do any changes to the values before save. This is used by fixture loading.

6.3.83 serializable_value

EquationPluginModel.**serializable_value**(*field_name*)

Return the value of the field name for this instance. If the field is a foreign key, return the id value instead of the object. If there's no Field object with this name on the model, return the model attribute's value.

Used to serialize a field's value (in the serializer, or form output, for example). Normally, you would just access the attribute directly and not use this method.

6.3.84 set_base_attr

EquationPluginModel.**set_base_attr**(*plugin*)

6.3.85 unique_error_message

EquationPluginModel.**unique_error_message**(*model_class*, *unique_check*)

6.3.86 update

EquationPluginModel.**update**(*refresh=False*, ***fields*)

6.3.87 validate_unique

EquationPluginModel.**validate_unique**(*exclude=None*)

Check unique constraints on the model and raise ValidationError if any failed.

Methods Documentation

add_child(***kwargs*)

Adds a child to the node.

This method saves the node in database. The object is populated as if via:

` obj = self.__class__(**kwargs) `

Raises **PathOverflow** – when no more child nodes can be added

classmethod add_root(***kwargs*)

Adds a root node to the tree.

This method saves the node in database. The object is populated as if via:

` obj = cls(**kwargs) `

Raises `PathOverflow` – when no more root objects can be added

`add_sibling`(*pos=None*, ***kwargs*)
Adds a new node as a sibling to the current node object.

This method saves the node in database. The object is populated as if via:

` obj = self.__class__(**kwargs) `

Raises `PathOverflow` – when the library can't make room for the node's new position

`classmethod check`(***kwargs*)

`clean`()
Hook for doing any extra model-wide validation after clean() has been called on every field by self.clean_fields. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field defined by NON_FIELD_ERRORS.

`clean_fields`(*exclude=None*)
Clean all fields and raise a ValidationError containing a dict of all validation errors if any occur.

`copy_plugin`(*target_placeholder*, *target_language*, *parent_cache*, *no_signals=False*)
Copy this plugin and return the new plugin.

The logic of this method is the following:

get a new generic plugin instance # assign the position in the plugin tree # save it to let mptt/treebeard calculate the tree attributes # then get a copy of the current plugin instance # assign to it the id of the generic plugin instance above;

this will effectively change the generic plugin created above into a concrete one

 # copy the tree related attributes from the generic plugin to the concrete one

save the concrete plugin # trigger the copy relations # return the generic plugin instance

This copy logic is required because we don't know what the fields of the real plugin are. By getting another instance of it at step 4 and then overwriting its ID at step 5, the ORM will copy the custom fields for us.

`copy_relations`(*old_instance*)
Handle copying of any relations attached to this plugin. Custom plugins have to do this themselves!

`date_error_message`(*lookup_type*, *field_name*, *unique_for*)

`delete`(*no_mp=False*, **args*, ***kwargs*)
Removes a node and all it's descendants.

`classmethod dump_bulk`(*parent=None*, *keep_ids=True*)
Dumps a tree branch to a python data structure.

`classmethod find_problems`()
Checks for problems in the tree structure, problems can occur when:

1. your code breaks and you get incomplete transactions (always use transactions!)
2. changing the `steplen` value in a model (you must *dump_bulk()* first, change `steplen` and then *load_bulk()*

Returns

A tuple of five lists:

1. a list of ids of nodes with characters not found in the `alphabet`

2. a list of ids of nodes when a wrong `path` length according to `steplen`
3. a list of ids of orphaned nodes
4. a list of ids of nodes with the wrong depth value for their path
5. a list of ids nodes that report a wrong number of children

classmethod fix_tree(*destructive=False*)
Fixes the plugin tree by first calling treebeard fix_tree and the recalculating the correct position property for each plugin.

classmethod from_db(*db*, *field_names*, *values*)

full_clean(*exclude=None*, *validate_unique=True*)
Call clean_fields(), clean(), and validate_unique() on the model. Raise a ValidationError for any errors that occur.

get_action_urls(*js_compat=True*)

get_add_url()

get_ancestors()

Returns A queryset containing the current node object's ancestors, starting by the root node and descending to the parent.

classmethod get_annotated_list(*parent=None*, *max_depth=None*)
Gets an annotated list from a tree branch.

Parameters

- **parent** – The node whose descendants will be annotated. The node itself will be included in the list. If not given, the entire tree will be annotated.
- **max_depth** – Optionally limit to specified depth

classmethod get_annotated_list_qs(*qs*)
Gets an annotated list from a queryset.

get_bound_plugin()
Returns an instance of the plugin model configured for this plugin type.

get_breadcrumb()

get_breadcrumb_json()

get_children()

Returns A queryset of all the node's children

get_children_count()

Returns The number the node's children, calculated in the most

efficient possible way.

get_copy_url()

classmethod get_database_vendor(*action*)
returns the supported database vendor used by a treebeard model when performing read (select) or write (update, insert, delete) operations.

Parameters `action` – *read* or *write*

Returns postgresql, mysql or sqlite

`get_deferred_fields()`
Return a set containing names of deferred fields for this instance.

`get_delete_url()`

`get_depth()`

Returns the depth (level) of the node

`get_descendant_count()`

Returns the number of descendants of a node.

`get_descendants()`

Returns A queryset of all the node's descendants as DFS, doesn't include the node itself

`classmethod get_descendants_group_count(`*parent=None*`)`
Helper for a very common case: get a group of siblings and the number of *descendants* in every sibling.

`get_edit_url()`

`get_first_child()`

Returns The leftmost node's child, or None if it has no children.

`classmethod get_first_root_node()`

Returns The first root node in the tree or `None` if it is empty.

`get_first_sibling()`

Returns The leftmost node's sibling, can return the node itself if it was the leftmost sibling.

`get_font_size_unit_display(`*, field=<django.db.models.fields.CharField: font_size_unit>*`)`

`classmethod get_foreign_keys()`
Get foreign keys and models they refer to, so we can pre-process the data for load_bulk

`get_instance_icon_alt()`
Get alt text for instance's icon

`get_instance_icon_src()`
Get src URL for instance's icon

`get_last_child()`

Returns The rightmost node's child, or None if it has no children.

`classmethod get_last_root_node()`

Returns The last root node in the tree or `None` if it is empty.

`get_last_sibling()`

Returns The rightmost node's sibling, can return the node itself if it was the rightmost sibling.

get_media_path(*filename*)

get_move_url()

get_next_by_changed_date(*, *field=<django.db.models.fields.DateTimeField: changed_date>*, *is_next=True*, ***kwargs*)

get_next_by_creation_date(*, *field=<django.db.models.fields.DateTimeField: creation_date>*, *is_next=True*, ***kwargs*)

get_next_sibling()

Returns The next node's sibling, or None if it was the rightmost sibling.

get_parent(*update=False*)

Returns the parent node of the current node object. Caches the result in the object itself to help in loops.

get_plugin_class()

get_plugin_class_instance(*admin=None*)

get_plugin_info(*children=None*, *parents=None*)

get_plugin_instance(*admin=None*)
Given a plugin instance (usually as a CMSPluginBase), this method returns a tuple containing:

instance - The instance AS THE APPROPRIATE SUBCLASS OF CMSPluginBase and not necessarily just 'self', which is often just a CMSPluginBase,

plugin - the associated plugin class instance (subclass of CMSPlugin)

get_plugin_name()

get_position_in_placeholder()
1 based position!

get_prev_sibling()

Returns The previous node's sibling, or None if it was the leftmost sibling.

get_previous_by_changed_date(*, *field=<django.db.models.fields.DateTimeField: changed_date>*, *is_next=False*, ***kwargs*)

get_previous_by_creation_date(*, *field=<django.db.models.fields.DateTimeField: creation_date>*, *is_next=False*, ***kwargs*)

get_root()

Returns the root node for the current node object.

classmethod get_root_nodes()

Returns A queryset containing the root nodes in the tree.

get_short_description()

get_siblings()

Returns A queryset of all the node's siblings, including the node itself.

`get_sorted_pos_queryset`(*siblings*, *newobj*)

Returns A queryset of the nodes that must be moved

to the right. Called only for Node models with `node_order_by`

This function is based on _insertion_target_filters from django-mptt (BSD licensed) by Jonathan Buchanan: https://github.com/django-mptt/django-mptt/blob/0.3.0/mptt/signals.py

`classmethod get_tree`(*parent=None*)

Returns A *queryset* of nodes ordered as DFS, including the parent. If no parent is given, the entire tree is returned.

`is_child_of`(*node*)

Returns `True` is the node if a child of another node given as an argument, else, returns `False`

`is_descendant_of`(*node*)

Returns `True` if the node is a descendant of another node given as an argument, else, returns `False`

`is_leaf`()

Returns True if the node is a leaf node (else, returns False)

`is_root`()

Returns True if the node is a root node (else, returns False)

`is_sibling_of`(*node*)

Returns `True` if the node is a sibling of another node given as an argument, else, returns `False`

`classmethod load_bulk`(*bulk_data*, *parent=None*, *keep_ids=False*)

Loads a list/dictionary structure to the tree.

Parameters

- **`bulk_data`** – The data that will be loaded, the structure is a list of dictionaries with 2 keys:
  - `data`: will store arguments that will be passed for object creation, and
  - `children`: a list of dictionaries, each one has it's own `data` and `children` keys (a recursive structure)
- **`parent`** – The node that will receive the structure as children, if not specified the first level of the structure will be loaded as root nodes
- **`keep_ids`** – If enabled, loads the nodes with the same primary keys that are given in the structure. Will error if there are nodes without primary key info or if the primary keys are already used.

Returns A list of the added node ids.

move(*target*, *pos=None*)
Moves the current node and all it's descendants to a new position relative to another node.

Raises **PathOverflow** – when the library can't make room for the node's new position

notify_on_autoadd(*request*, *conf*)
Method called when we auto add this plugin via default_plugins in CMS_PLACEHOLDER_CONF. Some specific plugins may have some special stuff to do when they are auto added.

notify_on_autoadd_children(*request*, *conf*, *children*)
Method called when we auto add children to this plugin via default_plugins/<plugin>/children in CMS_PLACEHOLDER_CONF. Some specific plugins may have some special stuff to do when we add children to them. ie : TextPlugin must update its content to add HTML tags to be able to see his children in WYSIWYG.

num_children()

classmethod numconv_obj()

post_copy(*old_instance*, *new_old_ziplist*)
Handle more advanced cases (eg Text Plugins) after the original is copied

prepare_database_save(*field*)

refresh_from_db(**args*, ***kwargs*)
Reload field values from the database.

By default, the reloading happens from the database this instance was loaded from, or by the read router if this instance wasn't loaded from any database. The using parameter will override the default.

Fields can be used to specify which fields to reload. The fields should be an iterable of field attnames. If fields is None, then all non-deferred fields are reloaded.

When accessing deferred fields of an instance, the deferred loading of the field will call this method.

reload()

save(*no_signals=False*, **args*, ***kwargs*)
Save the current instance. Override this in a subclass if you want to control the saving process.

The 'force_insert' and 'force_update' parameters can be used to insist that the "save" must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

save_base(*raw=False*, *force_insert=False*, *force_update=False*, *using=None*, *update_fields=None*)
Handle the parts of saving which should be done only once per save, yet need to be done in raw saves, too. This includes some sanity checks and signal sending.

The 'raw' argument is telling save_base not to save any parent models and not to do any changes to the values before save. This is used by fixture loading.

serializable_value(*field_name*)
Return the value of the field name for this instance. If the field is a foreign key, return the id value instead of the object. If there's no Field object with this name on the model, return the model attribute's value.

Used to serialize a field's value (in the serializer, or form output, for example). Normally, you would just access the attribute directly and not use this method.

set_base_attr(*plugin*)

unique_error_message(*model_class*, *unique_check*)

update(*refresh=False*, ***fields*)

validate_unique(*exclude=None*)
Check unique constraints on the model and raise ValidationError if any failed.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

H

I

L

M

N

O

P

R

S

T

U

V