

---

# **djangocms-equation Documentation**

*Release 0.2.1*

**Sebastian Weigand**

**Nov 23, 2020**



## FOR USERS:

<b>1</b>	<b>djangoCMS-equation</b>	<b>1</b>
1.1	Features . . . . .	1
1.2	Installation . . . . .	1
1.3	Credits . . . . .	2
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Stable release . . . . .	3
2.2	From sources . . . . .	3
<b>3</b>	<b>Credits</b>	<b>5</b>
3.1	Development Lead . . . . .	5
3.2	Contributors . . . . .	5
<b>4</b>	<b>History</b>	<b>7</b>
4.1	0.2.1 (2020-08-09) . . . . .	7
4.2	0.2.0 (2020-08-09) . . . . .	7
4.3	0.1.1 (2020-02-20) . . . . .	7
4.4	0.1.0 (2020-02-20) . . . . .	7
<b>5</b>	<b>Contributing</b>	<b>9</b>
5.1	Types of Contributions . . . . .	9
5.2	Get Started! . . . . .	10
5.3	Pull Request Guidelines . . . . .	11
5.4	Tips . . . . .	11
5.5	Deploying . . . . .	11
<b>6</b>	<b>Python Source Documentation</b>	<b>13</b>
6.1	EquationPlugin . . . . .	13
6.2	EquationForm . . . . .	37
6.3	EquationPluginModel . . . . .	42
<b>7</b>	<b>Indices and tables</b>	<b>67</b>
	<b>Index</b>	<b>69</b>



## 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.

## 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
```



**CREDITS**

### **3.1 Development Lead**

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

### **3.2 Contributors**

None yet. Why not be the first?



## HISTORY

### 4.1 0.2.1 (2020-08-09)

- Added official python 3.8 support

### 4.2 0.2.0 (2020-08-09)

- Upgraded KaTeX to 0.12.0
- Dropped support for Django 2.0

### 4.3 0.1.1 (2020-02-20)

- Added project urls and removed conda badge

### 4.4 0.1.0 (2020-02-20)

- First release on PyPI.



## 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.



## 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.

---

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

---

### 6.1 EquationPlugin

**class EquationPlugin** (*model=None, admin\_site=None*)  
Implementation of the actual plugin.

#### Attributes Summary

---

<code>actions</code>
<code>actions_on_bottom</code>
<code>actions_on_top</code>
<code>actions_selection_counter</code>
<code>add_form_template</code>
<code>admin_preview</code>
<code>allow_children</code>
<code>autocomplete_fields</code>
<code>cache</code>
<code>cache_child_classes</code>
<code>cache_parent_classes</code>
<code>change_form_template</code>
<code>change_list_template</code>
<code>child_classes</code>
<code>date_hierarchy</code>
<code>delete_confirmation_template</code>
<code>delete_selected_confirmation_template</code>
<code>disable_child_plugins</code>
<code>exclude</code>
<code>fields</code>

---

continues on next page

Table 2 – continued from previous page

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
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
value
view_on_site

## Methods Summary

<i>action_checkbox</i>	A <i>list_display</i> column containing a checkbox widget.
<i>add_view</i>	
<i>autocomplete_view</i>	
<i>change_view</i>	
<i>changeform_view</i>	
<i>changelist_view</i>	The ‘change list’ admin view for this model.
<i>check</i>	
<i>construct_change_message</i>	Construct a JSON structure describing changes from a changed object.
<i>delete_model</i>	Given a model instance delete it from the database.
<i>delete_queryset</i>	Given a queryset, delete it from the database.
<i>delete_view</i>	
<i>formfield_for_choice_field</i>	Get a form Field for a database Field that has declared choices.
<i>formfield_for_dbfield</i>	Hook for specifying the form Field instance for a given database Field instance.
<i>formfield_for_foreignkey</i>	Get a form Field for a ForeignKey.
<i>formfield_for_manytomany</i>	Get a form Field for a ManyToManyField.
<i>get_action</i>	Return a given action from a parameter, which can either be a callable, or the name of a method on the ModelAdmin.
<i>get_action_choices</i>	Return a list of choices for use in a form object.
<i>get_actions</i>	Return a dictionary mapping the names of all actions for this ModelAdmin to a tuple of (callable, name, description) for each action.
<i>get_autocomplete_fields</i>	Return a list of ForeignKey and/or ManyToMany fields which should use an autocomplete widget.
<i>get_cache_expiration</i>	Provides hints to the placeholder, and in turn to the page for determining the appropriate Cache-Control headers to add to the HTTPResponse object.
<i>get_changeform_initial_data</i>	Get the initial form data from the request’s GET params.
<i>get_changelist</i>	Return the ChangeList class for use on the changelist page.
<i>get_changelist_form</i>	Return a Form class for use in the Formset on the changelist page.
<i>get_changelist_formset</i>	Return a FormSet class for use on the changelist page if <i>list_editable</i> is used.
<i>get_changelist_instance</i>	Return a <i>ChangeList</i> instance based on <i>request</i> .
<i>get_child_class_overrides</i>	Returns a list of plugin types that are allowed as children of this plugin.
<i>get_child_classes</i>	Returns a list of plugin types that can be added as children to this plugin.
<i>get_child_plugin_candidates</i>	Returns a list of all plugin classes that will be considered when fetching all available child classes for this plugin.
<i>get_deleted_objects</i>	Hook for customizing the delete process for the delete view and the “delete selected” action.

continues on next page

Table 3 – continued from previous page

<code>get_empty_change_form_text</code>	Returns the text displayed to the user when editing a plugin that requires no configuration.
<code>get_empty_value_display</code>	Return the <code>empty_value_display</code> set on <code>ModelAdmin</code> or <code>AdminSite</code> .
<code>get_exclude</code>	Hook for specifying exclude.
<code>get_extra_placeholder_menu_items</code>	
<code>get_extra_plugin_menu_items</code>	
<code>get_field_queryset</code>	If the <code>ModelAdmin</code> specifies ordering, the <code>queryset</code> should respect that ordering.
<code>get_fields</code>	Hook for specifying fields.
<code>get_fieldsets</code>	Same as from base class except if there are no fields, show an info message.
<code>get_form</code>	Return a <code>Form</code> class for use in the admin add view.
<code>get_formsets_with_inlines</code>	Yield formsets and the corresponding inlines.
<code>get_inline_formsets</code>	
<code>get_inline_instances</code>	
<code>get_list_display</code>	Return a sequence containing the fields to be displayed on the changelist.
<code>get_list_display_links</code>	Return a sequence containing the fields to be displayed as links on the changelist.
<code>get_list_filter</code>	Return a sequence containing the fields to be displayed as filters in the right sidebar of the changelist page.
<code>get_list_select_related</code>	Return a list of fields to add to the <code>select_related()</code> part of the changelist items query.
<code>get_model_perms</code>	Return a dict of all perms for this model.
<code>get_object</code>	Return an instance matching the field and value provided, the primary key is used if no field is provided.
<code>get_ordering</code>	Hook for specifying field ordering.
<code>get_paginator</code>	
<code>get_parent_classes</code>	
<code>get_plugin_urls</code>	Return URL patterns for which the plugin wants to register views for.
<code>get_prepopulated_fields</code>	Hook for specifying custom prepopulated fields.
<code>get_preserved_filters</code>	Return the preserved filters querystring.
<code>get_queryset</code>	Return a <code>QuerySet</code> of all model instances that can be edited by the admin site.
<code>get_readonly_fields</code>	Hook for specifying custom readonly fields.
<code>get_render_queryset</code>	
<code>get_require_parent</code>	
<code>get_search_fields</code>	Return a sequence containing the fields to be searched whenever somebody submits a search query.
<code>get_search_results</code>	Return a tuple containing a <code>queryset</code> to implement the search and a boolean indicating if the results may contain duplicates.
<code>get_sortable_by</code>	Hook for specifying which fields can be sorted in the changelist.
<code>get_urls</code>	

continues on next page

Table 3 – continued from previous page

<i>get_vary_cache_on</i>	Provides hints to the placeholder, and in turn to the page for determining VARY headers for the response.
<i>get_view_on_site_url</i>	
<i>has_add_permission</i>	Return True if the given request has permission to add an object.
<i>has_change_permission</i>	Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the <i>obj</i> parameter.
<i>has_delete_permission</i>	Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the <i>obj</i> parameter.
<i>has_module_permission</i>	Return True if the given request has any permission in the given app label.
<i>has_view_or_change_permission</i>	
<i>has_view_permission</i>	Return True if the given request has permission to view the given Django model instance.
<i>history_view</i>	The 'history' admin view for this model.
<i>icon_alt</i>	Return the alt text for the shown icon.
<i>icon_src</i>	Return the path to an icon which is shown in the text editor.
<i>is_in_text_editor</i>	Check if the plugin was added to a text plugin.
<i>log_addition</i>	Log that an object has been successfully added.
<i>log_change</i>	Log that an object has been successfully changed.
<i>log_deletion</i>	Log that an object will be deleted.
<i>lookup_allowed</i>	
<i>message_user</i>	Send a message to the user.
<i>render</i>	Render the Plugin with <i>self.render_template</i> and the data in instance.
<i>render_change_form</i>	We just need the popup interface here
<i>render_close_frame</i>	
<i>render_delete_form</i>	
<i>requires_parent_plugin</i>	
<i>response_action</i>	Handle an admin action.
<i>response_add</i>	Determine the HttpResponse for the <i>add_view</i> stage.
<i>response_change</i>	Determine the HttpResponse for the <i>change_view</i> stage.
<i>response_delete</i>	Determine the HttpResponse for the <i>delete_view</i> stage.
<i>response_post_save_add</i>	Figure out where to redirect after the 'Save' button has been pressed when adding a new object.
<i>response_post_save_change</i>	Figure out where to redirect after the 'Save' button has been pressed when editing an existing object.
<i>save_form</i>	Given a <i>ModelForm</i> return an unsaved instance.
<i>save_formset</i>	Given an inline formset save it to the database.
<i>save_model</i>	Override original method, and add some attributes to <i>obj</i> This have to be made, because if object is newly created, he must know where he lives.

continues on next page

Table 3 – continued from previous page

<code>save_related</code>	Given the <code>HttpRequest</code> , the parent <code>ModelForm</code> 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.
<code>to_field_allowed</code>	Return <code>True</code> 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 `change-list-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 ('`django-cms-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 ('django-cms-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, ren-
derer=None)
```

Inputs which are used in the Equation editing dialog.

### Attributes Summary

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

## Methods Summary

<code>add_error</code>	Update the content of <code>self._errors</code> .
<code>add_initial_prefix</code>	Add a ‘initial’ prefix for checking dynamic initial values.
<code>add_prefix</code>	Return the field name with a prefix appended, if this Form has a prefix set.
<code>as_p</code>	Return this form rendered as HTML <code>&lt;p&gt;s</code> .
<code>as_table</code>	Return this form rendered as HTML <code>&lt;tr&gt;s</code> – excluding the <code>&lt;table&gt;&lt;/table&gt;</code> .
<code>as_ul</code>	Return this form rendered as HTML <code>&lt;li&gt;s</code> – excluding the <code>&lt;ul&gt;&lt;/ul&gt;</code> .
<code>clean</code>	Hook for doing any extra form-wide cleaning after <code>Field.clean()</code> has been called on every field.
<code>full_clean</code>	Clean all of <code>self.data</code> and populate <code>self._errors</code> and <code>self.cleaned_data</code> .
<code>get_initial_for_field</code>	Return initial data for field on form.
<code>has_changed</code>	Return True if data differs from initial.
<code>has_error</code>	
<code>hidden_fields</code>	Return a list of all the <code>BoundField</code> objects that are hidden fields.
<code>is_multipart</code>	Return True if the form needs to be multipart-encoded, i.e. it has <code>FileInput</code> , or False otherwise.
<code>is_valid</code>	Return True if the form has no errors, or False otherwise.
<code>non_field_errors</code>	Return an <code>ErrorList</code> of errors that aren’t associated with a particular field – i.e., from <code>Form.clean()</code> .
<code>order_fields</code>	Rearrange the fields according to <code>field_order</code> .
<code>save</code>	Save this form’s <code>self.instance</code> object if <code>commit=True</code> .
<code>validate_unique</code>	Call the instance’s <code>validate_unique()</code> method and update the form’s validation errors if any were raised.
<code>visible_fields</code>	Return a list of <code>BoundField</code> 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 `<li>s` – excluding the `<ul></ul>`.

### 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 `<li>s` – excluding the `<ul></ul>`.

**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

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

continues on next page

Table 6 – continued from previous page

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

### Methods Summary

<code>add_child</code>	Adds a child to the node.
<code>add_root</code>	Adds a root node to the tree.
<code>add_sibling</code>	Adds a new node as a sibling to the current node object.
<code>check</code>	

continues on next page

Table 7 – continued from previous page

<i>clean</i>	Hook for doing any extra model-wide validation after <code>clean()</code> has been called on every field by <code>self.clean_fields</code> .
<i>clean_fields</i>	Clean all fields and raise a <code>ValidationError</code> containing a dict of all validation errors if any occur.
<i>copy_plugin</i>	Copy this plugin and return the new plugin.
<i>copy_relations</i>	Handle copying of any relations attached to this plugin.
<i>date_error_message</i>	
<i>delete</i>	Removes a node and all its descendants.
<i>dump_bulk</i>	Dumps a tree branch to a python data structure.
<i>find_problems</i>	Checks for problems in the tree structure, problems can occur when:
<i>fix_tree</i>	Fixes the plugin tree by first calling <code>treebeard.fix_tree</code> and the recalculating the correct position property for each plugin.
<i>from_db</i>	
<i>full_clean</i>	Call <code>clean_fields()</code> , <code>clean()</code> , and <code>validate_unique()</code> on the model.
<i>get_action_urls</i>	
<i>get_add_url</i>	
<i>get_ancestors</i>	<b>returns</b> A queryset containing the current node object's ancestors,
<i>get_annotated_list</i>	Gets an annotated list from a tree branch.
<i>get_annotated_list_qs</i>	Gets an annotated list from a queryset.
<i>get_bound_plugin</i>	Returns an instance of the plugin model configured for this plugin type.
<i>get_breadcrumb</i>	
<i>get_breadcrumb_json</i>	
<i>get_children</i>	<b>returns</b> A queryset of all the node's children
<i>get_children_count</i>	<b>returns</b> The number the node's children, calculated in the most
<i>get_copy_url</i>	
<i>get_database_vendor</i>	returns the supported database vendor used by a treebeard model when performing read (select) or write (update, insert, delete) operations.
<i>get_deferred_fields</i>	Return a set containing names of deferred fields for this instance.
<i>get_delete_url</i>	
<i>get_depth</i>	<b>returns</b> the depth (level) of the node

continues on next page

Table 7 – continued from previous page

<code>get_descendant_count</code>	<b>returns</b> the number of descendants of a node.
<code>get_descendants</code>	<b>returns</b> A queryset of all the node's descendants as DFS, doesn't
<code>get_descendants_group_count</code>	Helper for a very common case: get a group of siblings and the number of <i>descendants</i> in every sibling.
<code>get_edit_url</code>	
<code>get_first_child</code>	<b>returns</b>
<code>get_first_root_node</code>	<b>returns</b>
<code>get_first_sibling</code>	<b>returns</b>
<code>get_font_size_unit_display</code>	
<code>get_foreign_keys</code>	Get foreign keys and models they refer to, so we can pre-process the data for <code>load_bulk</code>
<code>get_instance_icon_alt</code>	Get alt text for instance's icon
<code>get_instance_icon_src</code>	Get src URL for instance's icon
<code>get_last_child</code>	<b>returns</b>
<code>get_last_root_node</code>	<b>returns</b>
<code>get_last_sibling</code>	<b>returns</b>
<code>get_media_path</code>	
<code>get_move_url</code>	
<code>get_next_by_changed_date</code>	
<code>get_next_by_creation_date</code>	
<code>get_next_sibling</code>	<b>returns</b> The next node's sibling, or None if it was the rightmost
<code>get_parent</code>	<b>returns</b> the parent node of the current node object.
<code>get_plugin_class</code>	
<code>get_plugin_class_instance</code>	
<code>get_plugin_info</code>	

continues on next page

Table 7 – continued from previous page

<i>get_plugin_instance</i>	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)
<i>get_plugin_name</i>	
<i>get_position_in_placeholder</i>	1 based position!
<i>get_prev_sibling</i>	<b>returns</b> The previous node's sibling, or None if it was the leftmost
<i>get_previous_by_changed_date</i>	
<i>get_previous_by_creation_date</i>	
<i>get_root</i>	<b>returns</b> the root node for the current node object.
<i>get_root_nodes</i>	<b>returns</b> A queryset containing the root nodes in the tree.
<i>get_short_description</i>	
<i>get_siblings</i>	<b>returns</b> A queryset of all the node's siblings, including the node
<i>get_sorted_pos_queryset</i>	<b>returns</b> A queryset of the nodes that must be moved
<i>get_tree</i>	<b>returns</b>
<i>is_child_of</i>	<b>returns</b> True is the node if a child of another node given as an
<i>is_descendant_of</i>	<b>returns</b> True if the node is a descendant of another node given
<i>is_leaf</i>	<b>returns</b> True if the node is a leaf node (else, returns False)
<i>is_root</i>	<b>returns</b> True if the node is a root node (else, returns False)

continues on next page

Table 7 – continued from previous page

<i>is_sibling_of</i>	<b>returns</b> True if the node is a sibling of another node given as an
<i>load_bulk</i>	Loads a list/dictionary structure to the tree.
<i>move</i>	Moves the current node and all its descendants to a new position relative to another node.
<i>notify_on_autoadd</i>	Method called when we auto add this plugin via default_plugins in CMS_PLACEHOLDER_CONF.
<i>notify_on_autoadd_children</i>	Method called when we auto add children to this plugin via default_plugins/<plugin>/children in CMS_PLACEHOLDER_CONF.
<i>num_children</i>	
<i>numconv_obj</i>	
<i>post_copy</i>	Handle more advanced cases (eg Text Plugins) after the original is copied
<i>prepare_database_save</i>	
<i>refresh_from_db</i>	Reload field values from the database.
<i>reload</i>	
<i>save</i>	Save the current instance.
<i>save_base</i>	Handle the parts of saving which should be done only once per save, yet need to be done in raw saves, too.
<i>serializable_value</i>	Return the value of the field name for this instance.
<i>set_base_attr</i>	
<i>unique_error_message</i>	
<i>update</i>	
<i>validate_unique</i>	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 id that are given in the structure. Will error if there are nodes without id info or if the ids 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 its 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 atnames. 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 id that are given in the structure. Will error if there are nodes without id info or if the ids 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 atnames. 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.



## INDICES AND TABLES

- genindex
- modindex
- search



## A

action\_checkbox() (*EquationPlugin method*), 31  
 add\_child() (*EquationPluginModel method*), 60  
 add\_error() (*EquationForm method*), 41  
 add\_initial\_prefix() (*EquationForm method*), 41  
 add\_prefix() (*EquationForm method*), 41  
 add\_root() (*EquationPluginModel class method*), 60  
 add\_sibling() (*EquationPluginModel method*), 60  
 add\_view() (*EquationPlugin method*), 31  
 as\_p() (*EquationForm method*), 41  
 as\_table() (*EquationForm method*), 41  
 as\_ul() (*EquationForm method*), 41  
 autocomplete\_view() (*EquationPlugin method*), 31

## C

change\_view() (*EquationPlugin method*), 31  
 changeform\_view() (*EquationPlugin method*), 31  
 changelist\_view() (*EquationPlugin method*), 31  
 check() (*EquationPlugin method*), 31  
 check() (*EquationPluginModel class method*), 60  
 clean() (*EquationForm method*), 41  
 clean() (*EquationPluginModel method*), 60  
 clean\_fields() (*EquationPluginModel method*), 60  
 construct\_change\_message() (*EquationPlugin method*), 31  
 copy\_plugin() (*EquationPluginModel method*), 60  
 copy\_relations() (*EquationPluginModel method*), 60

## D

date\_error\_message() (*EquationPluginModel method*), 61  
 delete() (*EquationPluginModel method*), 61  
 delete\_model() (*EquationPlugin method*), 31  
 delete\_queryset() (*EquationPlugin method*), 31  
 delete\_view() (*EquationPlugin method*), 31  
 dump\_bulk() (*EquationPluginModel class method*), 61

## E

EquationForm (*class in.djangocms\_equation.forms*), 37  
 EquationPlugin (*class in.djangocms\_equation.cms\_plugins*), 13  
 EquationPluginModel (*class in.djangocms\_equation.models*), 42

## F

find\_problems() (*EquationPluginModel class method*), 61  
 fix\_tree() (*EquationPluginModel class method*), 61  
 formfield\_for\_choice\_field() (*EquationPlugin method*), 31  
 formfield\_for\_dbfield() (*EquationPlugin method*), 31  
 formfield\_for\_foreignkey() (*EquationPlugin method*), 32  
 formfield\_for\_manytomany() (*EquationPlugin method*), 32  
 from\_db() (*EquationPluginModel class method*), 61  
 full\_clean() (*EquationForm method*), 41  
 full\_clean() (*EquationPluginModel method*), 61

## G

get\_action() (*EquationPlugin method*), 32  
 get\_action\_choices() (*EquationPlugin method*), 32  
 get\_action\_urls() (*EquationPluginModel method*), 61  
 get\_actions() (*EquationPlugin method*), 32  
 get\_add\_url() (*EquationPluginModel method*), 61  
 get\_ancestors() (*EquationPluginModel method*), 61  
 get\_annotated\_list() (*EquationPluginModel class method*), 61  
 get\_annotated\_list\_qs() (*EquationPluginModel class method*), 61  
 get\_autocomplete\_fields() (*EquationPlugin method*), 32  
 get\_bound\_plugin() (*EquationPluginModel method*), 61

[get\\_breadcrumb\(\)](#) (*EquationPluginModel method*), 62  
[get\\_breadcrumb\\_json\(\)](#) (*EquationPluginModel method*), 62  
[get\\_cache\\_expiration\(\)](#) (*EquationPlugin method*), 32  
[get\\_changeform\\_initial\\_data\(\)](#) (*Equation-Plugin method*), 32  
[get\\_changelist\(\)](#) (*EquationPlugin method*), 32  
[get\\_changelist\\_form\(\)](#) (*EquationPlugin method*), 32  
[get\\_changelist\\_formset\(\)](#) (*EquationPlugin method*), 32  
[get\\_changelist\\_instance\(\)](#) (*EquationPlugin method*), 33  
[get\\_child\\_class\\_overrides\(\)](#) (*EquationPlugin class method*), 33  
[get\\_child\\_classes\(\)](#) (*EquationPlugin class method*), 33  
[get\\_child\\_plugin\\_candidates\(\)](#) (*Equation-Plugin class method*), 33  
[get\\_children\(\)](#) (*EquationPluginModel method*), 62  
[get\\_children\\_count\(\)](#) (*EquationPluginModel method*), 62  
[get\\_copy\\_url\(\)](#) (*EquationPluginModel method*), 62  
[get\\_database\\_vendor\(\)](#) (*EquationPluginModel class method*), 62  
[get\\_deferred\\_fields\(\)](#) (*EquationPluginModel method*), 62  
[get\\_delete\\_url\(\)](#) (*EquationPluginModel method*), 62  
[get\\_deleted\\_objects\(\)](#) (*EquationPlugin method*), 33  
[get\\_depth\(\)](#) (*EquationPluginModel method*), 62  
[get\\_descendant\\_count\(\)](#) (*EquationPluginModel method*), 62  
[get\\_descendants\(\)](#) (*EquationPluginModel method*), 62  
[get\\_descendants\\_group\\_count\(\)](#) (*Equation-PluginModel class method*), 62  
[get\\_edit\\_url\(\)](#) (*EquationPluginModel method*), 62  
[get\\_empty\\_change\\_form\\_text\(\)](#) (*EquationPlugin class method*), 33  
[get\\_empty\\_value\\_display\(\)](#) (*EquationPlugin method*), 33  
[get\\_exclude\(\)](#) (*EquationPlugin method*), 33  
[get\\_extra\\_placeholder\\_menu\\_items\(\)](#) (*EquationPlugin class method*), 33  
[get\\_extra\\_plugin\\_menu\\_items\(\)](#) (*Equation-Plugin class method*), 33  
[get\\_field\\_queryset\(\)](#) (*EquationPlugin method*), 33  
[get\\_fields\(\)](#) (*EquationPlugin method*), 33  
[get\\_fieldsets\(\)](#) (*EquationPlugin method*), 33  
[get\\_first\\_child\(\)](#) (*EquationPluginModel method*), 62  
[get\\_first\\_root\\_node\(\)](#) (*EquationPluginModel class method*), 62  
[get\\_first\\_sibling\(\)](#) (*EquationPluginModel method*), 62  
[get\\_font\\_size\\_unit\\_display\(\)](#) (*Equation-PluginModel method*), 62  
[get\\_foreign\\_keys\(\)](#) (*EquationPluginModel class method*), 62  
[get\\_form\(\)](#) (*EquationPlugin method*), 33  
[get\\_formsets\\_with\\_inlines\(\)](#) (*EquationPlugin method*), 33  
[get\\_initial\\_for\\_field\(\)](#) (*EquationForm method*), 41  
[get\\_inline\\_formsets\(\)](#) (*EquationPlugin method*), 33  
[get\\_inline\\_instances\(\)](#) (*EquationPlugin method*), 33  
[get\\_instance\\_icon\\_alt\(\)](#) (*EquationPlugin-Model method*), 62  
[get\\_instance\\_icon\\_src\(\)](#) (*EquationPlugin-Model method*), 62  
[get\\_last\\_child\(\)](#) (*EquationPluginModel method*), 63  
[get\\_last\\_root\\_node\(\)](#) (*EquationPluginModel class method*), 63  
[get\\_last\\_sibling\(\)](#) (*EquationPluginModel method*), 63  
[get\\_list\\_display\(\)](#) (*EquationPlugin method*), 33  
[get\\_list\\_display\\_links\(\)](#) (*EquationPlugin method*), 33  
[get\\_list\\_filter\(\)](#) (*EquationPlugin method*), 33  
[get\\_list\\_select\\_related\(\)](#) (*EquationPlugin method*), 33  
[get\\_media\\_path\(\)](#) (*EquationPluginModel method*), 63  
[get\\_model\\_perms\(\)](#) (*EquationPlugin method*), 33  
[get\\_move\\_url\(\)](#) (*EquationPluginModel method*), 63  
[get\\_next\\_by\\_changed\\_date\(\)](#) (*EquationPlugin-Model method*), 63  
[get\\_next\\_by\\_creation\\_date\(\)](#) (*EquationPlugin-Model method*), 63  
[get\\_next\\_sibling\(\)](#) (*EquationPluginModel method*), 63  
[get\\_object\(\)](#) (*EquationPlugin method*), 34  
[get\\_ordering\(\)](#) (*EquationPlugin method*), 34  
[get\\_paginator\(\)](#) (*EquationPlugin method*), 34  
[get\\_parent\(\)](#) (*EquationPluginModel method*), 63  
[get\\_parent\\_classes\(\)](#) (*EquationPlugin class method*), 34  
[get\\_plugin\\_class\(\)](#) (*EquationPluginModel method*), 63  
[get\\_plugin\\_class\\_instance\(\)](#) (*EquationPlugin*

- inModel method*), 63
- `get_plugin_info()` (*EquationPluginModel method*), 63
- `get_plugin_instance()` (*EquationPluginModel method*), 63
- `get_plugin_name()` (*EquationPluginModel method*), 63
- `get_plugin_urls()` (*EquationPlugin method*), 34
- `get_position_in_placeholder()` (*Equation-PluginModel method*), 63
- `get_prepopulated_fields()` (*EquationPlugin method*), 34
- `get_preserved_filters()` (*EquationPlugin method*), 34
- `get_prev_sibling()` (*EquationPluginModel method*), 63
- `get_previous_by_changed_date()` (*Equation-PluginModel method*), 63
- `get_previous_by_creation_date()` (*EquationPluginModel method*), 63
- `get_queryset()` (*EquationPlugin method*), 34
- `get_readonly_fields()` (*EquationPlugin method*), 34
- `get_render_queryset()` (*EquationPlugin class method*), 34
- `get_require_parent()` (*EquationPlugin class method*), 34
- `get_root()` (*EquationPluginModel method*), 63
- `get_root_nodes()` (*EquationPluginModel class method*), 63
- `get_search_fields()` (*EquationPlugin method*), 34
- `get_search_results()` (*EquationPlugin method*), 34
- `get_short_description()` (*EquationPlugin-Model method*), 64
- `get_siblings()` (*EquationPluginModel method*), 64
- `get_sortable_by()` (*EquationPlugin method*), 34
- `get_sorted_pos_queryset()` (*EquationPlugin-Model method*), 64
- `get_tree()` (*EquationPluginModel class method*), 64
- `get_urls()` (*EquationPlugin method*), 34
- `get_vary_cache_on()` (*EquationPlugin method*), 34
- `get_view_on_site_url()` (*EquationPlugin method*), 34
- H**
- `has_add_permission()` (*EquationPlugin method*), 34
- `has_change_permission()` (*EquationPlugin method*), 35
- `has_changed()` (*EquationForm method*), 42
- `has_delete_permission()` (*EquationPlugin method*), 35
- `has_error()` (*EquationForm method*), 42
- `has_module_permission()` (*EquationPlugin method*), 35
- `has_view_or_change_permission()` (*EquationPlugin method*), 35
- `has_view_permission()` (*EquationPlugin method*), 35
- `hidden_fields()` (*EquationForm method*), 42
- `history_view()` (*EquationPlugin method*), 35
- I**
- `icon_alt()` (*EquationPlugin method*), 35
- `icon_src()` (*EquationPlugin method*), 35
- `is_child_of()` (*EquationPluginModel method*), 64
- `is_descendant_of()` (*EquationPluginModel method*), 64
- `is_in_text_editor()` (*EquationPlugin method*), 36
- `is_leaf()` (*EquationPluginModel method*), 64
- `is_multipart()` (*EquationForm method*), 42
- `is_root()` (*EquationPluginModel method*), 64
- `is_sibling_of()` (*EquationPluginModel method*), 64
- `is_valid()` (*EquationForm method*), 42
- L**
- `load_bulk()` (*EquationPluginModel class method*), 64
- `log_addition()` (*EquationPlugin method*), 36
- `log_change()` (*EquationPlugin method*), 36
- `log_deletion()` (*EquationPlugin method*), 36
- `lookup_allowed()` (*EquationPlugin method*), 36
- M**
- `message_user()` (*EquationPlugin method*), 36
- `move()` (*EquationPluginModel method*), 64
- N**
- `non_field_errors()` (*EquationForm method*), 42
- `notify_on_autoadd()` (*EquationPluginModel method*), 64
- `notify_on_autoadd_children()` (*Equation-PluginModel method*), 65
- `num_children()` (*EquationPluginModel method*), 65
- `numconv_obj()` (*EquationPluginModel class method*), 65
- O**
- `order_fields()` (*EquationForm method*), 42
- P**
- `post_copy()` (*EquationPluginModel method*), 65

`prepare_database_save()` (*EquationPluginModel method*), 65

## R

`refresh_from_db()` (*EquationPluginModel method*), 65

`reload()` (*EquationPluginModel method*), 65

`render()` (*EquationPlugin method*), 36

`render_change_form()` (*EquationPlugin method*), 36

`render_close_frame()` (*EquationPlugin method*), 36

`render_delete_form()` (*EquationPlugin method*), 36

`requires_parent_plugin()` (*EquationPlugin class method*), 36

`response_action()` (*EquationPlugin method*), 36

`response_add()` (*EquationPlugin method*), 36

`response_change()` (*EquationPlugin method*), 36

`response_delete()` (*EquationPlugin method*), 37

`response_post_save_add()` (*EquationPlugin method*), 37

`response_post_save_change()` (*EquationPlugin method*), 37

## S

`save()` (*EquationForm method*), 42

`save()` (*EquationPluginModel method*), 65

`save_base()` (*EquationPluginModel method*), 65

`save_form()` (*EquationPlugin method*), 37

`save_formset()` (*EquationPlugin method*), 37

`save_model()` (*EquationPlugin method*), 37

`save_related()` (*EquationPlugin method*), 37

`serializable_value()` (*EquationPluginModel method*), 65

`set_base_attr()` (*EquationPluginModel method*), 65

## T

`to_field_allowed()` (*EquationPlugin method*), 37

## U

`unique_error_message()` (*EquationPluginModel method*), 65

`update()` (*EquationPluginModel method*), 65

## V

`validate_unique()` (*EquationForm method*), 42

`validate_unique()` (*EquationPluginModel method*), 65

`visible_fields()` (*EquationForm method*), 42