

# Improving the Editor Experience Through Validation

Scott Cranfill (he/him)  
Wagtail Space US 2024

Made with Podium

# Who am I?

- Full Stack Web Developer at NASA's Jet Propulsion Laboratory
- Resident of Rochester, New York
- Member of the Wagtail Core Team and Accessibility Team



BLM ROC logo credit: [www.dunwoode.design](http://www.dunwoode.design)

Made with Podium

# What is validation?

# Validation in Django

The process of checking that something is acceptable before saving it to the database.

- Can apply to form fields, model fields, and model instances
- As simple as required fields
- As complex as the relationship between multiple fields on a model instance
- Happens automatically when attempting to save a model instance

# A simple example

```
from django.core import validators
from django.db import models

class Card(models.Model):
    title = models.CharField(null=False, blank=False)
    desc = models.TextField(null=False, blank=False)
    link_text = models.CharField(null=True, blank=True)
    link_url = models.CharField(
        null=True,
        blank=True,
        validators=[validators.URLValidator]
    )
```

# Django's four-step validation

1. Validate the model fields – `Model.clean_fields()`
2. Validate the model as a whole – `Model.clean()`
3. Validate the field uniqueness – `Model.validate_unique()`
4. Validate the constraints – `Model.validate_constraints()`

<https://docs.djangoproject.com/en/4.2/ref/models/instances/#validating-objects>

# 1. Validate the model fields

```
from django.core import validators
from django.db import models

class Card(models.Model):
    title = models.CharField(null=False, blank=False)
    desc = models.TextField(null=False, blank=False)
    link_text = models.CharField(null=True, blank=True)
    link_url = models.CharField(
        null=True,
        blank=True,
        validators=[validators.URLValidator]
    )
```

# Custom field validation functions

```
class ConferencePhoneValidator(validators.RegexValidator):  
    """  
    Only allow valid characters for conference phone numbers.  
    - Only numeric, "+", "-", and "#" characters are allowed.  
    """  
    regex = r'^[0-9+,#]+$'  
    message = "Enter a valid conference phone number. The three  
groups of numbers in this example are a Zoom phone number,  
meeting code, and password:  
+19294362866,,2151234215#,,#,,12341234#"  
    flags = 0
```



## 2. Validate the model as a whole

```
class Model(AltersData, metaclass=ModelBase):  
    # ...  
  
    def clean(self):  
        """  
        Hook for doing any extra model-wide validation  
        after clean() has been called on every field  
        by self.clean_fields. ...  
        """  
        pass
```

<https://github.com/django/django/blob/main/django/db/models/base.py#L1338-L1345>

# Wagtail's `Page.clean()`

```
class Page(AbstractPage, ...):  
    # ...  
  
    def clean(self):  
        super().clean()  
        parent = self.get_parent()  
  
        if not Page._slug_is_available(self.slug, parent, self):  
            raise ValidationError({  
                "slug": "Slug is already in use within parent."  
            })
```

[https://github.com/wagtail/wagtail/blob/main/wagtail/models/\\_\\_init\\_\\_.py#L1455-L1466](https://github.com/wagtail/wagtail/blob/main/wagtail/models/__init__.py#L1455-L1466)

**The secret sauce:**  
`super().clean()`

# Validating your own page models

```
from django.core.exceptions import ValidationError
from django.db import models
from wagtail.models import Page

class EventPage(Page):
    start_date = models.DateTimeField()
    end_date = models.DateTimeField()

    def clean(self):
        super().clean()
        if self.end_date < self.start_date:
            raise ValidationError({
                "end_date": "End date can't be before start."
            })
```

# Enforcing specific slugs

```
class AuthorBioPage(Page):
    text = models.RichTextField()

    # Can only be created under AuthorPages
    parent_page_types = ["AuthorPage"]
    # Only 1 may exist under a given parent
    max_count_per_parent = 1

    content_panels = Page.content_panels + [
        FieldPanel("text"),
    ]
```

# Enforcing specific slugs

```
def clean(self):  
    super().clean()  
    self.slug = "bio"
```

# Related fields

# Related fields

```
class Mission(Page)
    launch_date_status = models.CharField(
        choices=LaunchDateStatus.choices,
        default=LaunchDateStatus.NONE,
        max_length=25
    )
    launch_date = models.DateField(
        null=True,
        blank=True,
        help_text="This can be left blank for TBD or unknown
launch dates"
    )
```



# Related fields

```
from django.db.models import TextChoices

class LaunchDateStatus(TextChoices):
    KNOWN = "known", "Known launch date (display full launch date)"
    TBD = "tbd", "TBD launch date (display TBD text)"
    TBD_YEAR_ONLY = "tbd_year_only", "TBD launch date (display year only)"
    TBD_MONTH_AND_YEAR_ONLY = "tbd_month_and_year_only", "TBD launch date (display year and month only)"
    NONE = "none", "No launch date (nothing will display for launch date)"
```

# Related fields

```
class Mission(Page)
    launch_date_status = models.CharField(
        choices=LaunchDateStatus.choices,
        default=LaunchDateStatus.NONE,
        max_length=25
    )
    launch_date = models.DateField(
        null=True,
        blank=True,
        help_text="This can be left blank for TBD or unknown
launch dates"
    )
```

# Related fields

```
def clean(self):
    super().clean()
    launch_date_required = self.launch_date_status in [
        LaunchDateStatus.KNOWN,
        LaunchDateStatus.TBD_YEAR_ONLY,
        LaunchDateStatus.TBD_MONTH_AND_YEAR_ONLY,
    ]

    if launch_date_required and not self.launch_date:
        raise ValidationError({
            "launch_date": "Launch date is required for the
launch date status selected."
        })
```

# Raising multiple validation errors

# Raising multiple validation errors

```
def clean(self):
    super().clean()
    errors = defaultdict(list)

    if self.show_clock and not self.start_date_time:
        errors["show_clock"].append('Add a start date/time or
untick "Show clock".')
    if launch_date_required and not self.launch_date:
        errors["launch_date"].append("Launch date is required
for the launch date status selected.")

    if any(errors.values()):
        raise ValidationError(errors)
```

# StreamField Blocks

Made with Podium

# Link blocks

```
class LinkBlock(StructBlock):  
    page = PageChooserBlock(required=False)  
    document = DocumentChooserBlock(required=False)  
    url = URLBlock(required=False)  
    text = CharBlock(required=False, help_text="Required for  
URLs. If not entered for a page or document link, the text  
will default to their title.")
```

[https://docs.wagtail.org/en/stable/advanced\\_topics/customisation/streamfield\\_blocks.html](https://docs.wagtail.org/en/stable/advanced_topics/customisation/streamfield_blocks.html)

# Basics of block validation

```
def clean(self, value):  
    result = super().clean(value)  
  
    # do your validation by inspecting the result dict  
  
    return result
```

[https://docs.wagtail.org/en/stable/advanced\\_topics/streamfield\\_validation.html](https://docs.wagtail.org/en/stable/advanced_topics/streamfield_validation.html)



# Validating our LinkBlock

```
def clean(self, value):
    result = super().clean(value)

    if not (
        result["page"]
        or result["document"]
        or result["url"]
    ):
        raise ValidationError("Page, document, or URL must be
set to determine link destination.")

    # to be continued
```

# Validating our LinkBlock

```
def clean(self, value):  
    # continued from previous slide  
  
    if (  
        (result["page"] and result["document"])  
        or (result["page"] and result["url"])  
        or (result["document"] and result["url"])  
    ):  
        raise ValidationError("Please set only one type of  
link (page, document, or URL).")  
  
    # to be continued
```

# Validating our LinkBlock

```
def clean(self, value):  
    # continued from previous slides  
  
    if result["url"] and not result["text":  
        raise ValidationError("Text required for URL links.")  
  
    return result
```

# Image blocks

```
class ImageBlock(StructBlock):  
    image = ImageChooserBlock()  
    alt_text = CharBlock(required=False)  
    decorative = BooleanBlock(required=False)
```


# Validating our ImageBlock

```
def clean(self, value):
    result = super().clean(value)

    if result["alt_text"] and result["decorative"]:
        raise StructBlockValidationError(
            block_errors={
                "alt_text": ValidationError("Marking an image
as decorative will override alt text entered here. Empty this
field or uncheck the decorative box.")
            }
        )


    return result
```

# Heading hierarchy



**Warnings**Issues found **1**

**Incorrect heading hierarchy. Avoid skipping levels.****1**

 `.hidden-md-down > .block-heading_block > h3`

# A simple heading block

```
class HeadingBlock(StructBlock):  
    heading_text = CharBlock(required=True)  
    size = ChoiceBlock(  
        choices=[  
            ("", "Select a header size"),  
            ("h2", "H2"),  
            ("h3", "H3"),  
            ("h4", "H4"),  
        ],  
        blank=True,  
        required=False,  
    )
```

# Setting up a StreamBlock

```
class BaseStreamBlock(StreamBlock):  
    heading_block = HeadingBlock()  
    rich_text_block = RichTextBlock(  
        features=["bold", "italic", "link", "ul", "ol"]  
    )  
    image_block = ImageBlock()  
  
class NewsPage(Page):  
    body = StreamField(BaseStreamBlock())
```



# Validating our BaseStreamBlock

```
class BaseStreamBlock(StreamBlock):  
    # ...  
  
    def clean(value):  
        result = super().clean(value)  
  
        headings = [  
            # tuples of block index and heading level  
            (0, 1) # mock H1 block at index 0  
        ]  
        errors = {}  
  
        # to be continued
```

# Validating our BaseStreamBlock

```
def clean(value):  
    # continued from previous slide  
  
    # first iterate through all blocks in the StreamBlock  
    for i in range(0, len(result)):  
        # if a block is of type "heading?"  
        if result[i].block_type == "heading":  
            # convert size string to integer  
            level = int(result[i].value.get("size")[-1:])  
            # append tuple of block index and heading level to  
list  
                headings.append((i, level))  
  
    # to be continued
```

# Validating our BaseStreamBlock

```
def clean(value):
    # continued from previous slides

    # now iterate through list of headings, starting with
    # second heading to skip over the mock H1 heading block
    for i in range(1, len(headings)):
        # compare its level to the previous heading's level
        if int(headings[i][1]) - int(headings[i-1][1]) > 1:
            # if the difference is more than 1, add an error
            # to the errors dict with its original index
            errors[headings[i][0]] = ValidationError("Incorrect
heading hierarchy. Avoid skipping levels.")

    # to be continued
```

# Validating our BaseStreamBlock

```
def clean(value):  
    # continued from previous slides  
  
    if errors:  
        raise StreamBlockValidationError(block_errors=errors)  
  
    return result
```

# It works in site settings, too!

```
@register_setting
class SiteWideAlertSettings(BaseSiteSetting):
    enable_sitewide_alert = models.BooleanField(
        default=False
    )
    alert_text = RichTextField(
        blank=True,
        features=["h2", "h3", "bold", "italic", "link"],
    )
```

[https://github.com/wagtail/wagtail.org/blob/main/wagtailio/sitewide\\_alert/models.py#L15-L27](https://github.com/wagtail/wagtail.org/blob/main/wagtailio/sitewide_alert/models.py#L15-L27)

# It works in site settings, too!

```
def clean(self):  
    if self.enable_sitewide_alert and not self.alert_text:  
        raise ValidationError(  
            {  
                "alert_text": ValidationError(  
                    "To enable the sitewide alert, please  
specify the alert text."  
                ),  
            }  
        )
```

# Some other things I've seen

# Some other things I've seen

- Ensuring items entered into a list are sorted in the correct order



# Some other things I've seen

- Ensuring items entered into a list are sorted in the correct order
- Dynamically setting the title field based on other data

# Some other things I've seen

- Ensuring items entered into a list are sorted in the correct order
- Dynamically setting the title field based on other data
- Automatically updating the slug if the title changes

# Further reading

[docs.djangoproject.com/en/4.2/ref  
/models/instances/#validating-objects](https://docs.djangoproject.com/en/4.2/ref/models/instances/#validating-objects)

[docs.wagtail.org/en/stable/advanced\\_topics/streamfield\\_validation.html](https://docs.wagtail.org/en/stable/advanced_topics/streamfield_validation.html)

# Thank you!

<https://github.com/Scotchester/wsus-validation>