nider Documentation

Release 0.4.0

Vladyslav Ovchynnykov

Sep 26, 2017

Contents

Nider is an approach to make generation of text based images simple yet flexible.

CHAPTER 1

Installation

Stable release

To install nider, run this command in your terminal:

```
$ pip install nider
```

This is the preferred method to install nider, 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.

From sources

The sources for nider can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/pythad/nider
```

Or download the tarball:

```
$ curl  -OL https://github.com/pythad/nider/tarball/master
```

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

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

CHAPTER 2

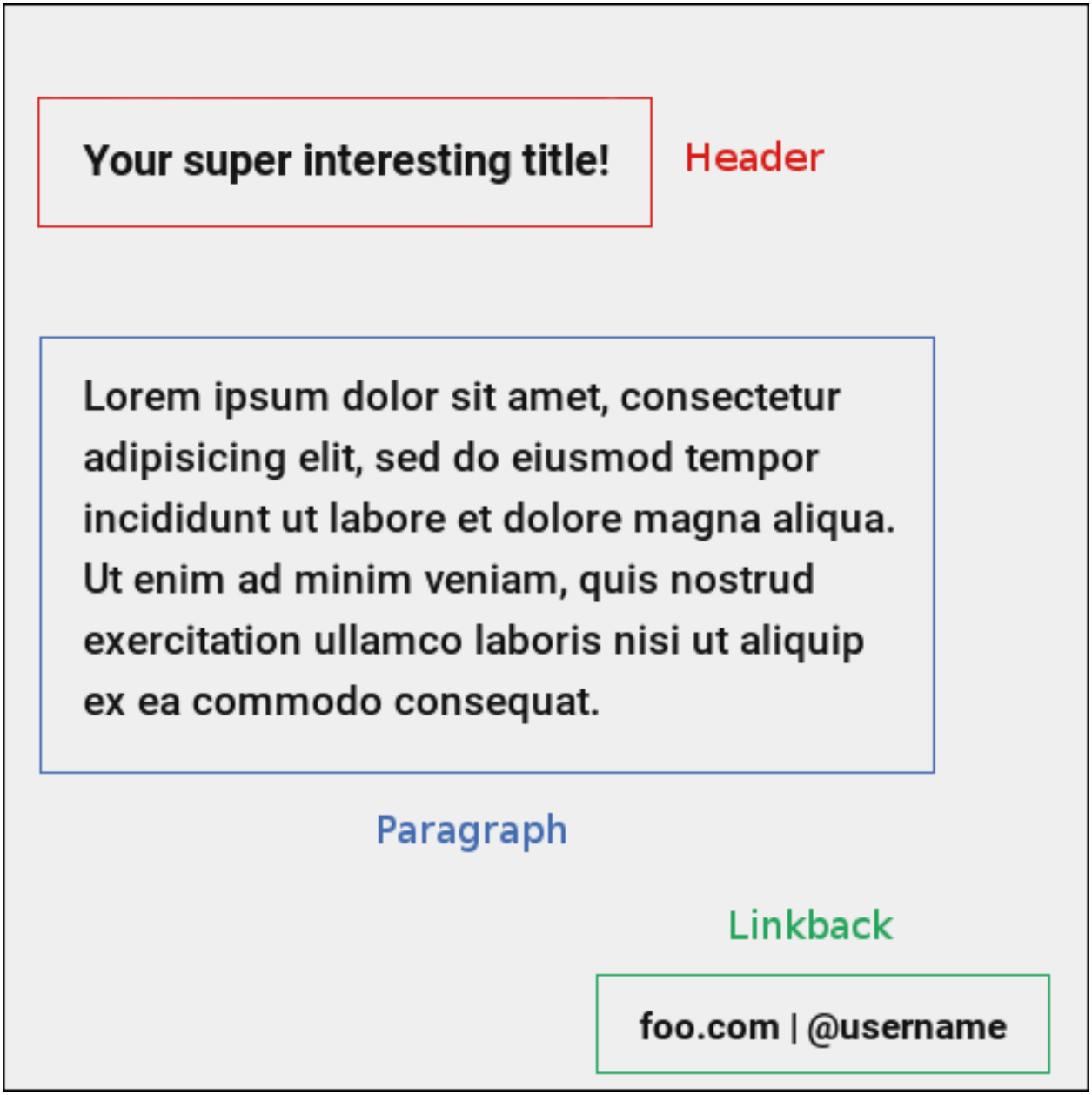
Usage

This article is a tutorial for `nider` package and at the same time it is a full reference of all `nider` models and possibilities.

Image units

There are three main units each `nider.Image` can consist of:

- header
- paragraph
- linkback



Each of the units is represented by a class in `nider.models`:

- *nider.models.Header*
- *nider.models.Paragraph*
- *nider.models.Linkback*

nider.models.Header

class nider.models.**Header**(*text*, *font=None*, *text_width=21*, *line_padding=6*, *color=None*, *outline=None*, *align='center'*)

Class that represents a header used in images

Parameters

- **text** (*str*) – text used for the unit.
- **font** (nider.core.Font) – font object that represents text's font.
- **text_width** (*int*) – units's text width - number of characters in a line.
- **line_padding** (*int*) – unit's line padding - padding (in pixels) between the lines.
- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names.
- **outline** (nider.core.Outline) – outline object that represents text's outline.
- **align** (*'left' or 'center' or 'right'*) – side with respect to which the text will be aligned.

Raises

- *nider.exceptions.InvalidAlignException* – if ‘‘align‘ is not supported by nider.
- AttributeError – if text_width < 0.
- *nider.exceptions.DefaultFontWarning* – if font.path is None.
- *nider.exceptions.FontNotFoundWarning* – if font.path does not exist.

Example

```
from nider.core import Font
from nider.core import Outline

from nider.models import Header


header = Header(text='Your super interesting title!',
                font=Font('/home/me/.local/share/fonts/Roboto/Roboto-Bold.ttf', 30),
                text_width=40,
                align='left',
                color='#ededed',
                outline=Oultine(2, '#222')
                )
```

nider.models.Paragraph

This class has the same attributes and behaviour as *nider.models.Header*.

class nider.models.**Paragraph**(*text*, *font=None*, *text_width=21*, *line_padding=6*, *color=None*, *outline=None*, *align='center'*)

Class that represents a paragraph used in images

Parameters

- **text** (*str*) – text used for the unit.
- **font** (nider.core.Font) – font object that represents text's font.
- **text_width** (*int*) – units's text width - number of characters in a line.
- **line_padding** (*int*) – unit's line padding - padding (in pixels) between the lines.

- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names.
- **outline** (nider.core.Outline) – outline object that represents text's outline.
- **align** (*'left' or 'center' or 'right'*) – side with respect to which the text will be aligned.

Raises

- *nider.exceptions.InvalidAlignException* – if ‘‘align‘ is not supported by nider.
- AttributeError – if text_width < 0.
- *nider.exceptions.DefaultFontWarning* – if font.path is None.
- *nider.exceptions.FontNotFoundWarning* – if font.path does not exist.

Example

```
from nider.core import Font
from nider.core import Outline

from nider.models import Paragraph


para = Paragraph(text='Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed␣
↪do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim␣
↪veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo␣
↪consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum␣
↪dolore eu fugiat nulla pariatur.',
                 font=Font('/home/me/.local/share/fonts/Roboto/Roboto-Bold.ttf', 30),
                 text_width=65,
                 align='left',
                 color='#ededed'
                 outline=Oultine(1, '#000')
                 )
```

nider.models.Linkback

class nider.models.**Linkback**(*text*, *font=None*, *color=None*, *outline=None*, *align='right'*, *bottom_padding=20*)

Class that represents a linkback used in images

Parameters

- **text** (*str*) – text used for the unit.
- **font** (nider.core.Font) – font object that represents text's font.
- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names
- **outline** (nider.core.Outline) – outline object that represents text's outline.
- **align** (*'left' or 'center' or 'right'*) – side with respect to which the text will be aligned.

- **bottom_padding** (*int*) – linkback's bottom padding - padding (in pixels) between the bottom of the image and the linkback itself.

Raises

- *nider.exceptions.InvalidAlignException* – if `align` is not supported by nider.
- *nider.exceptions.DefaultFontWarning* – if `font.path` is `None`.
- *nider.exceptions.FontNotFoundWarning* – if `font.path` does not exist.

Example

```
from nider.core import Font
from nider.core import Outline

from nider.models import Linkback


linkback = Linkback(text='foo.com | @username',
                    font=Font('/home/me/.local/share/fonts/Roboto/Roboto-Bold.ttf',␣
↪30),
                    color='#ededed',
                    outline=Oultine(2, '#000')
                    )
```

Note: Parameters `color` and `outline.color` are optional for any unit. They can be generated automatically by `nider`. `nider` analyzes background color of either a texture or of an image and chooses an opposite one to it. So if your image in mainly dark , white text color will be auto generated and set. The same applies to outline color.

Although it's a nice feature for backgrounds you have no control over, we'd recommend to provide colors explicitly.

Image content

In order to aggregate all of the units together you need to create an instance of *nider.models.Content* class.

nider.models.Content

class nider.models.**Content**(*paragraph=None*, *header=None*, *linkback=None*, *watermark=None*, *padding=45*)

Class that aggregates different units into a sigle object

Parameters

- **paragraph** (nider.models.Paragraph) – paragraph used for in the content.
- **header** (nider.models.Header) – header used for in the content.
- **linkback** (nider.models.Linkback) – linkback used for in the content.
- **watermark** (nider.models.Watermark) – watermark used for in the content.
- **padding** (*int*) – content's padding - padding (in pixels) between the units.

Raises *nider.exceptions.ImageGeneratorException* – if neither of `paragraph`, `header` or `linkback` is provided.

Note: `padding` is taken into account only if image is to get resized. If size allows content to fit freely, pre-calculated paddings will be used.

Note: Content has to consist at least of one unit: header, paragraph or linkback.

Example

```
from nider.models import Content
from nider.models import Linkback
from nider.models import Paragraph


para = Paragraph(...)

linkback = Linkback(...)

content = Content(para, linkback=linkback, padding=60)
```

Initializing an image

After the content is prepared it's the right time to initialize an image. In `nider` a basic image is represented by `nider.models.Image`.

nider.models.Image

class `nider.models.`**Image**(*content*, *fullpath*, *width=1080*, *height=1080*, *title=None*, *description=None*)

Base class for a text based image

Parameters

- **content** (*nider.models.Content*) – content object that has units to be rendered.
- **fullpath** (*str*) – path where the image has to be saved.
- **width** (*int*) – width of the image to be generated.
- **height** (*int*) – height of the image to be generated.
- **title** (*str*) – title of the image. Serves as metadata for latter rendering in html. May be used as alt text of the image. If no title is provided `content.header.text` will be set as the value.
- **description** (*str*) – description of the image. Serves as metadata for latter rendering in html. May be used as description text of the image. If no description is provided `content.paragraph.text` will be set as the value.

Raises

- *nider.exceptions.ImageGeneratorException* – if the current user doesn't have sufficient permissions to create the file at passed `fullpath`.
- `AttributeError` – if width <= 0 or height <= 0.

Example

```
from nider.models import Content
from nider.models import Image


content = Content(...)

img = Image(content,
            fullpath='example.png',
            width=500,
            height=500
            )
```

Social media images

`nider` comes with some pre-built models that can be used to generate images for some social networks. These are subclasses of `nider.models.Image` with changed size.

Instagram

- *nider.models.InstagramSquarePost* - 1080x1080 image
- *nider.models.InstagramPortraitPost* - 1080x1350 image
- *nider.models.InstagramLandscapePost* - 1080x566 image

Facebook

- *nider.models.FacebookSquarePost* - 470x470 image
- *nider.models.FacebookLandscapePost* - 1024x512 image

Twitter

- *nider.models.TwitterPost* - 1024x512 image
- *nider.models.TwitterLargeCard* - 506x506 image

I highly recommend reading this post if you are curious about what are the right image sizes for social media images.

Drawing on the image

Having an instance of `nider.models.Image` we are ready to create a real image.

`nider` comes with 3 options of drawing your image:

- `Image.draw_on_texture` - draws preinitialized image and its attributes on a texture.

Note: You don't need to create textured images by pasting texture mulpitle times in Photoshop or Gimp. `nider` takes care of filling image of any size with textrure you privide.

- `Image.draw_on_bg` - Draws preinitialized image and its attributes on a colored background. nider uses a color you provide to fill the image and then draws the content.
- `Image.draw_on_image` - Draws preinitialized image and its attributes on an image. Content will be drawn directly on the image you provide.

`Image.draw_on_texture`

`Image.`**`draw_on_texture`**(*texture_path=None*)

Draws preinitialized image and its attributes on a texture

If `texture_path` is set to `None`, random texture from `nider/textures` will be taken.

Parameters **`texture_path`** (*`str`*) – path of the texture to use.

Raises

- `FileNotFoundError` – if texture file at path `texture_path` cannot be found.
- *`nider.exceptions.ImageSizeFixedWarning`* – if the image size has to be adjusted to the provided content's size because the content takes too much space.

Example

```
from nider.models import Content
from nider.models import Image


content = Content(...)

img = Image(content,
            fullpath='example.png',
            width=500,
            height=500
            )

img.draw_on_texture('example_texture.png')
```

Check the full example here .

`nider` comes with a huge bundle of textures. As for now you need to copy them to your machine if you want to use any of them.

Image.draw_on_bg

Image.**draw_on_bg**(*bgcolor=None*)
Draws preinitialized image and its attributes on a colored background

If bgcolor is set to None, random nider.colors.colormap.FLAT_UI_COLORS will be taken.

Parameters **bgcolor** (*str*) – string that represents a background color. Must be compatible with PIL.ImageColor color names

Raises *nider.exceptions.ImageSizeFixedWarning* – if the image size has to be adjusted to the provided content's size because the content takes too much space.

Example

```
from nider.models import Content
from nider.models import Image


content = Content(...)

img = Image(content,
            fullpath='example.png',
            width=500,
            height=500
            )

img.draw_on_bg('#efefef')
```

Check the full example here .

Image.draw_on_image

Image.**draw_on_image**(*image_path*, *image_enhancements=None*, *image_filters=None*)
Draws preinitialized image and its attributes on an image

Parameters

- **image_path** (*str*) – path of the image to draw on.
- **image_enhancements** (*itarable*) – itarable of tuples, each containing a class from PIL.ImageEnhance that will be applied and factor - a floating point value controlling the enhancement. Check documentation of PIL.ImageEnhance for more info about availabe enhancements.
- **image_filters** (*itarable*) – itarable of filters from PIL.ImageFilter that will be applied. Check documentation of PIL.ImageFilter for more info about availabe filters.

Raises FileNotFoundError – if image file at path image_path cannot be found.

Examples

```
from nider.models import Content
from nider.models import Image
```

```
content = Content(...)

img = Image(content,
            fullpath='example.png',
            width=500,
            height=500
            )

img.draw_on_image('example_bg.jpg')
```

Using filters and enhancements:

```
img.draw_on_image('example_bg.jpg',
                  image_enhancements=((ImageEnhance.Contrast, 0.75),
                                      (ImageEnhance.Brightness, 0.5)),
                  image_filters=((ImageFilter.BLUR),),
                  )
```

Check the full example here .

That's it. After any of draw methods has been called and successfully completed the new image will be saved to `Image.fullpath`.

Adding watermarks

`nider` comes with built-in support for adding watermarks to your images.

First of all you need to create an instanse of *nider.models.Watermark* class.

class `nider.models.`**Watermark**(*text*, *font*, *color=None*, *outline=None*, *cross=True*, *rotate_angle=None*, *opacity=0.25*)

Class that represents a watermark used in images

Parameters

- **text** (*str*) – text to use.
- **font** (`nider.core.Font`) – font object that represents text's font.
- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names
- **outline** (`nider.core.Outline`) – outline object that represents text's outline.
- **cross** (*bool*) – boolean flag that indicates whether watermark has to be drawn with a cross.
- **rotate_angle** (*int*) – angle to which watermark's text has to be rotated.
- **opacity** (*0 <= float <= 1*) – opacity level of the watermark (applies to both the text and the cross).

Raises

- *nider.exceptions.ImageGeneratorException* – if `font` is the default font.
- *nider.exceptions.DefaultFontWarning* – if `font.path` is `None`.

- *nider.exceptions.FontNotFoundWarning* – if `font.path` does not exist.

Note: Watermark tries to takes all available width of the image so providing `font.size` has no affect on the watermark.

Example

```
watermark = Watermark(text='COPYRIGHT',
                      font=Font('/home/me/.local/share/fonts/Roboto/Roboto-Bold.ttf'),
                      color='#111',
                      cross=True,
                      rotate_angle=-45,
                      opacity=0.35
                      )
```

After this you can either add watermark to you `Content` instance and draw watermark on `nider` generated images:

```
from nider.models import Content
from nider.models import Image
from nider.models import Watermark


watermark = Watermark(...)

content = Content(..., watermark=watermark)

img = Image(content,
            fullpath='example.png',
            width=500,
            height=500
            )

img.draw_on_bg('#efefef')
```

or you can add a watermark to an existing image using *nider.tools.add_watermark()*:

nider.tools.**add_watermark**(*image_path*, *watermark*, *new_path=None*, *image_enhancements=None*, *image_filters=None*)

Function to add watermarks to images

Parameters

- **image_path** (*str*) – path of the image to which watermark has to be added.
- **watermark** (`nider.models.Watermark`) – watermark object.
- **new_path** (*str*) – path where the image has to be saved. **If set to None (default option), initial image will be overwritten.**
- **image_enhancements** (*itarable*) – itarable of tuples, each containing a class from `PIL.ImageEnhance` that will be applied and factor - a floating point value controlling the enhancement. Check documentation of `PIL.ImageEnhance` for more info about availabe enhancements.

- **image_filters** (*itarable*) – itarable of filters from `PIL.ImageFilter` that will be applied. Check documentation of `PIL.ImageFilter` for more info about availabe filters.

Raises

- `FileNotFoundError` – if image file at path `image_path` cannot be found.
- *nider.exceptions.ImageGeneratorException* – if the current user doesn't have sufficient permissions to create the file at passed `new_path`.

Example

```
from nider.models import Watermark

from nider.tools import add_watermark


watermark = Watermark(...)
add_watermark('path/to/my/img', watermark)
```

CHAPTER 3

nider package

nider.core module

class nider.core.**Font**(*path=None*, *size=18*)
Base class for text's font

Parameters

- **path** (*str*) – path to the font used in the object.
- **size** (*int*) – size of the font.

Raises

- *nider.exceptions.DefaultFontWarning* – if path is None.
- *nider.exceptions.FontNotFoundWarning* – if path does not exist.

class nider.core.**Outline**(*width=2*, *color=None*)
Base class for text's outline

Parameters

- **width** (*int*) – width of the stroke.
- **color** (*str*) – string that represents outline color. Must be compatible with PIL.ImageColor color names.

Warning: Due to PIL limitations - core library used for drawing, nider doesn't support 'true' outline. That is why high width outlines will look rather ugly and we don't recommend usign outlines with width > 3.

nider.models module

class nider.models.**Header**(*text*, *font=None*, *text_width=21*, *line_padding=6*, *color=None*, *outline=None*, *align='center'*)

Class that represents a header used in images

Parameters

- **text** (*str*) – text used for the unit.
- **font** (nider.core.Font) – font object that represents text's font.
- **text_width** (*int*) – units's text width - number of characters in a line.
- **line_padding** (*int*) – unit's line padding - padding (in pixels) between the lines.
- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names.
- **outline** (nider.core.Outline) – outline object that represents text's outline.
- **align** (*'left' or 'center' or 'right'*) – side with respect to which the text will be aligned.

Raises

- *nider.exceptions.InvalidAlignException* – if ‘‘align‘ is not supported by nider.
- AttributeError – if text_width < 0.
- *nider.exceptions.DefaultFontWarning* – if font.path is None.
- *nider.exceptions.FontNotFoundWarning* – if font.path does not exist.

class nider.models.**Paragraph**(*text*, *font=None*, *text_width=21*, *line_padding=6*, *color=None*, *outline=None*, *align='center'*)

Class that represents a paragraph used in images

Parameters

- **text** (*str*) – text used for the unit.
- **font** (nider.core.Font) – font object that represents text's font.
- **text_width** (*int*) – units's text width - number of characters in a line.
- **line_padding** (*int*) – unit's line padding - padding (in pixels) between the lines.
- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names.
- **outline** (nider.core.Outline) – outline object that represents text's outline.
- **align** (*'left' or 'center' or 'right'*) – side with respect to which the text will be aligned.

Raises

- *nider.exceptions.InvalidAlignException* – if ‘‘align‘ is not supported by nider.
- AttributeError – if text_width < 0.
- *nider.exceptions.DefaultFontWarning* – if font.path is None.
- *nider.exceptions.FontNotFoundWarning* – if font.path does not exist.

class nider.models.**Linkback**(*text*, *font=None*, *color=None*, *outline=None*, *align='right'*, *bottom_padding=20*)

Class that represents a linkback used in images

Parameters

- **text** (*str*) – text used for the unit.
- **font** (nider.core.Font) – font object that represents text's font.
- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names
- **outline** (nider.core.Outline) – outline object that represents text's outline.
- **align** (*'left' or 'center' or 'right'*) – side with respect to which the text will be aligned.
- **bottom_padding** (*int*) – linkback's bottom padding - padding (in pixels) between the bottom of the image and the linkback itself.

Raises

- *nider.exceptions.InvalidAlignException* – if `align` is not supported by nider.
- *nider.exceptions.DefaultFontWarning* – if `font.path` is `None`.
- *nider.exceptions.FontNotFoundWarning* – if `font.path` does not exist.

class nider.models.**Watermark**(*text*, *font*, *color=None*, *outline=None*, *cross=True*, *rotate_angle=None*, *opacity=0.25*)

Class that represents a watermark used in images

Parameters

- **text** (*str*) – text to use.
- **font** (nider.core.Font) – font object that represents text's font.
- **color** (*str*) – string that represents a color. Must be compatible with PIL.ImageColor color names
- **outline** (nider.core.Outline) – outline object that represents text's outline.
- **cross** (*bool*) – boolean flag that indicates whether watermark has to be drawn with a cross.
- **rotate_angle** (*int*) – angle to which watermark's text has to be rotated.
- **opacity**(*0 <= float <= 1*) – opacity level of the watermark (applies to both the text and the cross).

Raises

- *nider.exceptions.ImageGeneratorException* – if `font` is the default font.
- *nider.exceptions.DefaultFontWarning* – if `font.path` is `None`.
- *nider.exceptions.FontNotFoundWarning* – if `font.path` does not exist.

Note: Watermark tries to takes all available width of the image so providing `font.size` has no affect on the watermark.

---

class nider.models.**Content**(*paragraph=None*, *header=None*, *linkback=None*, *watermark=None*, *padding=45*)

Class that aggregates different units into a sigle object

Parameters

- **paragraph** (nider.models.Paragraph) – paragraph used for in the content.
- **header** (nider.models.Header) – header used for in the content.
- **linkback** (nider.models.Linkback) – linkback used for in the content.
- **watermark** (nider.models.Watermark) – watermark used for in the content.
- **padding** (*int*) – content's padding - padding (in pixels) between the units.

Raises *nider.exceptions.ImageGeneratorException* – if neither of `paragraph`, `header` or `linkback` is provided.

Note: `padding` is taken into account only if image is to get resized. If size allows content to fit freely, pre-calculated paddings will be used.

Note: Content has to consist at least of one unit: header, paragraph or linkback.

---

class nider.models.**Image**(*content*, *fullpath*, *width=1080*, *height=1080*, *title=None*, *description=None*)

Base class for a text based image

Parameters

- **content** (nider.models.Content) – content object that has units to be rendered.
- **fullpath** (*str*) – path where the image has to be saved.
- **width** (*int*) – width of the image to be generated.
- **height** (*int*) – height of the image to be generated.
- **title** (*str*) – title of the image. Serves as metadata for latter rendering in html. May be used as alt text of the image. If no title is provided `content.header.text` will be set as the value.
- **description** (*str*) – description of the image. Serves as metadata for latter rendering in html. May be used as description text of the image. If no description is provided `content.paragraph.text` will be set as the value.

Raises

- *nider.exceptions.ImageGeneratorException* – if the current user doesn't have sufficient permissions to create the file at passed `fullpath`.
- `AttributeError` – if width <= 0 or height <= 0.

draw_on_bg(*bgcolor=None*)

Draws preinitialized image and its attributes on a colored background

If `bgcolor` is set to `None`, random `nider.colors.colormap.FLAT_UI_COLORS` will be taken.

Parameters **bgcolor** (*str*) – string that represents a background color. Must be compatible with PIL.ImageColor color names

Raises *nider.exceptions.ImageSizeFixedWarning* – if the image size has to be adjusted to the provided content's size because the content takes too much space.

draw_on_image(*image_path*, *image_enhancements=None*, *image_filters=None*)
Draws preinitialized image and its attributes on an image

Parameters

- **image_path** (*str*) – path of the image to draw on.
- **image_enhancements** (*itarable*) – itarable of tuples, each containing a class from `PIL.ImageEnhance` that will be applied and factor - a floating point value controlling the enhancement. Check documentation of `PIL.ImageEnhance` for more info about availabe enhancements.
- **image_filters** (*itarable*) – itarable of filters from `PIL.ImageFilter` that will be applied. Check documentation of `PIL.ImageFilter` for more info about availabe filters.

Raises `FileNotFoundError` – if image file at path `image_path` cannot be found.

draw_on_texture(*texture_path=None*)
Draws preinitialized image and its attributes on a texture

If `texture_path` is set to `None`, random texture from `nider/textures` will be taken.

Parameters **texture_path** (*str*) – path of the texture to use.

Raises

- `FileNotFoundError` – if texture file at path `texture_path` cannot be found.
- *nider.exceptions.ImageSizeFixedWarning* – if the image size has to be adjusted to the provided content's size because the content takes too much space.

class nider.models.**FacebookSquarePost**(**args*, ***kwargs*)
Alias of *nider.models.Image* with `width=470` and `height=470`

class nider.models.**FacebookLandscapePost**(**args*, ***kwargs*)
Alias of *nider.models.Image* with `width=1024` and `height=512`

nider.models.**TwitterPost**
alias of *FacebookLandscapePost*

class nider.models.**TwitterLargeCard**(**args*, ***kwargs*)
Alias of *nider.models.Image* with `width=506` and `height=506`

class nider.models.**InstagramSquarePost**(**args*, ***kwargs*)
Alias of *nider.models.Image* with `width=1080` and `height=1080`

class nider.models.**InstagramPortraitPost**(**args*, ***kwargs*)
Alias of *nider.models.Image* with `width=1080` and `height=1350`

class nider.models.**InstagramLandscapePost**(**args*, ***kwargs*)
Alias of *nider.models.Image* with `width=1080` and `height=566`

nider.exceptions module

exception nider.exceptions.**AutoGeneratedUnitColorUsedWarning**(*unit*, *color_used*)
Warning raised when auto generated unit color was used

exception nider.exceptions.**AutoGeneratedUnitOutlinecolorUsedWarning**(*unit*, *color_used*)
Warning raised when auto generated unit's outline color was used

exception nider.exceptions.**DefaultFontWarning**
Warning raised when default font was used

exception nider.exceptions.**FontNotFoundException**(*fontpath_provided*)
Exception raised when font cannot be found

exception nider.exceptions.**FontNotFoundWarning**(*fontpath_provided*)
Warning raised when font cannot be found

exception nider.exceptions.**ImageGeneratorException**
Base class for exceptions raised by nider

exception nider.exceptions.**ImageGeneratorWarning**
Base class for warnings raised by nider

exception nider.exceptions.**ImageSizeFixedWarning**
Warning raised when the size of the image has to be adjusted to the provided content's size because the content takes much space

exception nider.exceptions.**InvalidAlignException**(*align_provided*, *available_aligns=None*)
Exception raised when align is not supported by nider

CHAPTER 4

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/pythad/nider/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/pythad/nider/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 :)

Get Started!

Ready to contribute? Here's how to set up *nider* for local development.

1. Fork the *nider* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/nider.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 nider
$ cd nider/
$ python setup.py develop
```

4. Create a branch for local development:

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

   Now you can make your changes locally.

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

```
$ flake8 nider tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. 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
```

7. Submit a pull request through the GitHub website.

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 3.4 and 3.5. Check https://travis-ci.org/pythad/nider/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest discover tests
```

CHAPTER 5

Credits

Development Lead

- Vladyslav Ovchynnykov <ovd4mail@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

0.1.0 (2017-07-27)

- First release on PyPI.

0.2.0 (2017-08-12)

- Added `PIL.ImageEnhance` and `PIL.ImageFilter` built-in support
- Enabled auto color generation for unit colors

0.3.0 (2017-08-17)

- Dropped shadow support for units
- Added outline support for units
- Made unit's font config as a separate class

0.4.0 (2017-09-14)

- Added ability to add watermarks to images

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index