
django-textvariation Documentation

Release 0.1

Corey Oordt

July 08, 2011

CONTENTS

1	Goals of Text Variations	1
2	Installation	3
2.1	Dependencies	3
3	Getting Started	5
3.1	What are Variations?	5
3.2	Specifying Variations	8
3.3	Configuring Models with Variations	9
3.4	Setting Up Search	9
4	Variations and URLs	11
4.1	Ways of Handling Variations in URLs	11
4.2	Defining the URL patterns	12
4.3	TextVariationMiddleware	12
4.4	Requested Variation, Returned Variation and URLs	13
5	Templates	15
5.1	Context Processors	15
5.2	Canonical URLs	16
6	Editing One Dimension	17
7	Reference	19
7.1	Settings	19
7.2	Implementation Notes	21
7.3	Management Commands	23
7.4	Template Tags	23
7.5	TextVariationMiddleware	24
7.6	Object Methods	24
8	Items for Future Versions	25
9	Indices and tables	27

GOALS OF TEXT VARIATIONS

Django Text Variations grew out of a need for certain types of content to have multiple versions of text based on specified variation types.

In short, text variations needs to accomplish:

1. There can be multiple types of variations (e.g. language, audience, text difficulty)
2. There could be multiple fields on a model which require text variations.
3. There could be a maximum number of variations of content equal to the multiplication of the number of each variation type. (e.g. text difficulty levels x audiences x languages)
4. Variation types are configurable on a per-project (site) implementation.
5. Any of the variation types, or the variations within that type, could be added to or removed from at any time. (e.g. deciding on going from one language to multiple languages, or going from two language variations to three)
6. There should be a way to handle addressing the variation in a url.
7. Not all the variations will be used for each item.
8. There should be a default variation. If an addressed variation does not exist, it should use the default variation, or a variation algorithmically specified using available variations (graceful fallback).
9. Invalidation of variations should happen when the default content is changed.

INSTALLATION

Installation is easy using `pip` or `easy_install`.

```
pip install django-textvariation
```

or

```
easy_install django-textvariation
```

2.1 Dependencies

None.

GETTING STARTED

3.1 What are Variations?

Textual variations are common in current websites, but are usually only implemented in one dimension: language. Django Text Variations allows you to have text (potentially) vary on multiple dimensions such as language, text difficulty, and audience.

In Django Text Variations, a variation is a specific combination of variants from each dimension.

3.1.1 Dimensions

A dimension is one set of ways in which text may vary, such as language. Each dimension has specific ways in which it could vary, such as English and Spanish, called variants.

In further examples, we'll assume that we need text to (potentially) vary on three dimensions.

- **Audiences** with three variants: Adult, Teen, Child
- **Text Difficulty** with three variants: Low, Standard, Advanced
- **Language** with four variants: English, Spanish, Mexican Spanish, French

3.1.2 Defaults and Graceful Fallback

In many cases, possibly most, you don't need text variations for every possible combination of dimensions. In the case when a specific combination is requested but does not exist for that object, another variation is delivered. The variation delivered is based on the initial request, the priority of the dimensions, and the available variations.

Django Text Variations is designed to allow for a default, which is the initial text provided, and is assigned to a combination based on the configuration of the dimensions. In the case where no other variations are provided, the default is always delivered.

Within each dimension, there may be a preference in which alternates are resolved. In the case of a language dimension, you may have an object missing a Mexican Spanish variation look first for a Spanish variation, and finally use default English variation. A missing French variation would simply look for an English variation.

Mexican Spanish
+> Spanish
+> English

French
+> English

3.1.3 Configuration of Dimensions and Fallback

Each text variation dimension defines a list of variants. Each variant has a “code” or “slug”, a name, and a fall back variant. One variant should not have a fall back. This is the default variant for the dimension.

Note: If a dimension contains multiple variants without a fall back variant, strange results may occur.

Because there are multiple ways to find the next potential variation, if the requested variation is missing, you specify the priority of the dimensions.

So, in our example dimensions, we could define them as:

Dimension	Name	Code	Fallback
Audience	Adult	ad	<i>None</i>
	Teen	tn	Adult
	Child	cd	Teen
	Low	l	Standard
Text Difficulty	Standard	s	<i>None</i>
	Advanced	a	Standard
	English	en	<i>None</i>
	Español	es	English
Language	Español mexicano	es-mx	Español
	Français	fr	English

And set the dimension priority as:

1. Language
2. Audience
3. Text Difficulty

How the priority determines the select of the variation is covered in the next section.

3.1.4 How a variation is determined

When a requested variation is not available, Django Text Variation looks for the next available variation by falling back through each variant of each dimension in reverse order of priority. In this way, if language is the highest priority, look for a variation that is the fall back of the other dimensions, but within the selected language before falling back on the next available language variation.

Here is an example:

1. A new user to the page gets the default variation (**Adult/Standard/English**)

2. This user clicks a ‘Ver en Español mexicano’ (‘view in Mexican Spanish’) link. Variations are looked for in the following order:

#	Audience	Text Difficulty	Language
1	Adult	Standard	Español mexicano
2	Adult	Standard	Español
3	Adult	Standard	English

In this case, the only variation from the default is language, so that is the only dimension that changes.

3. Next the user clicks on a ‘niños edición’ (‘kid’s edition’) link. Variations are looked for in the following order:

#	Audience	Text Difficulty	Language
1	Child	Standard	Español mexicano
2	Teen	Standard	Español mexicano
3	Adult	Standard	Español mexicano
4	Child	Standard	Español
5	Teen	Standard	Español
6	Adult	Standard	Español
7	Child	Standard	English
8	Teen	Standard	English
9	Adult	Standard	English

Now there are two dimension deviations from the default variation. Since the higher priority of these two is the Language dimension, each variant of the Audience dimension is substituted before trying each variant on the Language dimension.

4. Next the user clicks on a ‘texto menor dificultad’ (lower text difficulty) link. Variations are looked for in the following order:

#	Audience	Text Difficulty	Language
1	Child	Low	Español mexicano
2	Child	Standard	Español mexicano
3	Teen	Low	Español mexicano
4	Teen	Standard	Español mexicano
5	Adult	Low	Español mexicano
6	Adult	Standard	Español mexicano
7	Child	Low	Español
8	Child	Standard	Español
9	Teen	Low	Español
10	Teen	Standard	Español
11	Adult	Low	Español
12	Adult	Standard	Español
13	Child	Low	English
14	Child	Standard	English
15	Teen	Low	English
16	Teen	Standard	English
17	Adult	Low	English
18	Adult	Standard	English

The user has requested a variation in which all three dimensions are different from the default. Using the dimension priority, Text Difficulty variants are substituted before altering the audience or Language.

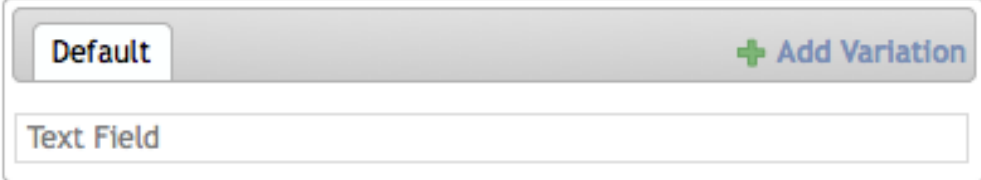
This fall back methodology allows a content producer to specify a variation that applies for only one

dimension. The most specific variation relating to the requested variation is returned.

3.2 Specifying Variations

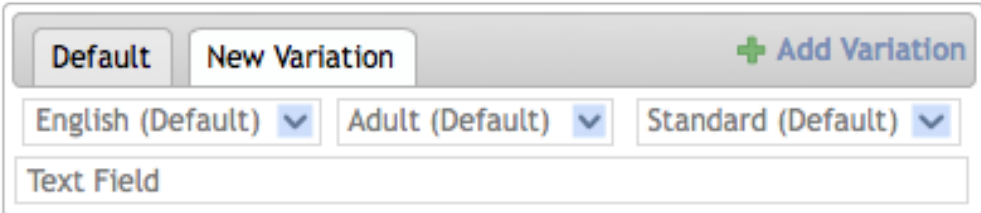
Initially, a field with variations will show a tabbed interface with a single tab and a button to add a variation.

Headline:



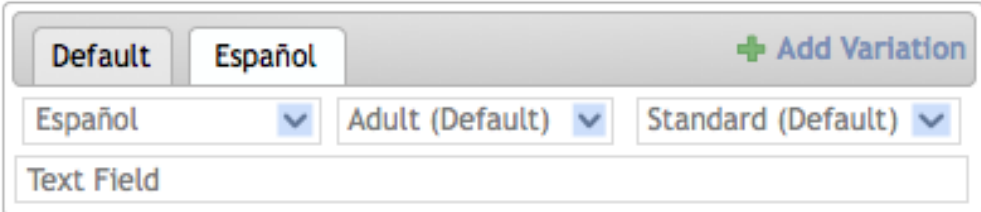
Clicking on this link creates a new tab with selection boxes for each dimension.

Headline:



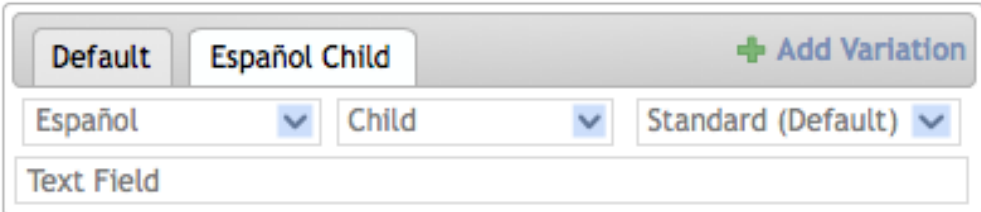
Changing one of the dimensions changes the name of the tab.

Headline:



Changing another dimensions changes the name of the tab again.

Headline:



3.2.1 Declaring Something Inappropriate

In some cases you may want to specifically exclude variants of a dimension from returning content. Content that is inappropriate for children is a good example.

To exclude content, simply create an empty variation. If you changed the Audience dimension to “Child” and left the field blank, then the template could properly display something like “This content is inappropriate for children.”

Headline:

Default
Child
+ Add Variation

English (Default)
Child
Standard (Default)

3.3 Configuring Models with Variations

3.3.1 Variations

3.4 Setting Up Search

VARIATIONS AND URLS

URLs are important. Content should exist at one and only one URL. Each variation in content needs to have a unique URL. Django Text Variation tries to be as flexible as possible by using middleware to check configured patterns as they are requested.

4.1 Ways of Handling Variations in URLs

A great resource for my research was [this blog post](#) regarding methods for handling multilingual websites. He defines nine ways of accomplishing language variations in content, many of which are not generic enough for use here.

First and foremost, [query parameters are not recommended](#), and will not be supported.

Ultimately, there are three ways specify a unique variation:

- Domain
- Path segment
- Path parameters or other delimited suffix

You can use any combination of these methods to specify your variation.

4.1.1 Domain

The domain difference could be at the host level, such as `dim1.example.com` or at a different top-level domain such as `example.es`. Top-level domains are probably only useful for specifying language variations.

Note: You can only vary on one dimension within a domain. That means you can vary on language (`en.example.com`) or audience (`kids.example.com`), but not both (`en.kids.example.com`) within the domain.

4.1.2 Path Segment

A common variation, especially for language specification, is to have the variant prefix the path, such as `www.example.com/en/` or `www.example.com/es/`. However, encoding the variant within the path (`www.example.com/blog/en/`) is also supported.

Note: Putting the variant within the path may make it difficult to re-create variation URLs of items. If the item's URL is either prefixed or suffixed with dimension information, the variation URL is easily created.

4.1.3 Path Parameters or Delimited Suffix

Path parameters are defined by [Section 3.3 of RFC 3986](#) as semicolon (;) and comma (,). One benefit of the semicolon is support within Python's `urlparse` module. It is also possible to use a dot (.) delimiter if you wish, although it is less common and not recommended.

4.2 Defining the URL patterns

4.2.1 Assigning a variant to a domain

Domain variations are defined within each variant's dictionary. Each variant for each dimension has several options, one of which is domain. If domain is not specified, the default domain is assumed.

4.2.2 Dimensions Within the Path

The middleware checks if the requested path fits a defined set of regular expressions in `TEXT_VARIATIONS['URL_REGEXES']`. To make the regular expressions easier to define, there are two shortcuts available.

{<dimension_name>} This string will substitute a named group with every variant as an option. So `{language}` could result in `(?P<language>en|es|es-mx|fr)`

{path} This string will substitute a group to catch any character: `(.*)`

4.2.3 Examples

Assuming two dimensions: 'language' and 'audience' with variants 'en', 'es', 'es-mx', 'fr' and 'cd', 'tn', 'ad' respectively.

#	RegEx Pattern	Result
1	<code>{language}/{path};{audience}</code>	<code>(?P<language>en es es-mx fr)/(.*);(?P<audience>cd/tn/ad)</code>
2	<code>{path}/{language}/{path};{audience}</code>	<code>(.*)/(?P<language>en es es-mx fr)/(.*);(?P<audience>cd/tn/ad)</code>
3	<code>{language}/{path}\.{audience}</code>	<code>(?P<language>en es es-mx fr)/(.*)\.(?P<audience>cd/tn/ad)</code>

4.3 TextVariationMiddleware

The middleware modifies the request

- Matches request path against the configured regular expressions
- If it matches, extract the dimensions and set into `request.META['TEXT_VARIATIONS']`
- Join the unnamed groups in the matching regular expression with `"/"`

- Set `request.path`, `request.path_info` and `request.META['PATH_INFO']` to the resulting value

See [TextVariationMiddleware](#) for more detailed information.

4.4 Requested Variation, Returned Variation and URLs

- URLs without dimension information serve the default variation
- URLs should either contain no variation information, or contain information for every dimension.
- If a variant of one dimension is requested, the default variants of any other dimensions should be requested.
- If a requested variation does not exist, a fallback variation is returned.
- The Text Variation context processor will create a `text_variations` variable with the variations returned for each field. If there are multiple variable fields in a model, you should pick one field to populate a `<link rel="canonical" href="" />` tag.
- The template could provide an alert on the page that the requested variation was not available and the substituted variation was.

TEMPLATES

At the top of the page, you need to call a template tag to get the proper variation fit for the object

```
{% get_text_variations object %}
```

Default usage: get the best variation fit for the requested variation:

```
{{ object.headline_variation }}
```

Testing the requested variation/dimension

```
{% if text_variations.requested.language.code == 'es' %}
```

Displaying information about the requested variation

```
<p>You are viewing the
{{ text_variations.requested.audience.name }}
version of the site in
{{ text_variation.requested.language.name }}.</p>
```

testing each field

```
{% if text_variations.headline.language.code == 'es' %}
```

Testing if a field is not the requested variation

```
{% if text_variations.headline == text_variations.requested %}
```

Does a specific variation exist for a field?

```
{% if object.text_variations.headline.en.ad %}
```

get specific variation of a field

```
{% get_variation object headline language=es %}
```

5.1 Context Processors

```
text_variations = {
    'requested': {'dim1': {'code': 'c2', 'name': 'Code2'}},
    'field1': {'dim1': {'code': 'co', 'name': 'Code1'}},
}
```

```
'field2': {'dim1': {'code': 'c2', 'name': 'Code2'},}
```

5.2 Canonical URLs

When the returned variation does not match the requested variation

If there are multiple fields on a model with variations, each could have different set defined variations, you will have to pick one field to define the canonical variation, such as blog entry content.

EDITING ONE DIMENSION

In some cases the person who enters the content will not be the one who applies other variations. For example if the default language was English, there may be a person(s) who translate the content previously entered.

For these people, there is a different view on the content to make creating the variations easier.

REFERENCE

7.1 Settings

All configuration for Django Text Variations occurs within the `TEXT_VARIATIONS` setting variable. `TEXT_VARIATIONS` is a dictionary with the following keys:

- *`DIMENSIONS`*
- *`DIMENSION_PRIORITY`*
- *`MODELS`*
- *`DEFAULT_URL_REGEX`*
- *`URL_REGEX_MAP`*
- *`Example Settings`*

7.1.1 DIMENSIONS

7.1.2 DIMENSION_PRIORITY

7.1.3 MODELS

7.1.4 DEFAULT_URL_REGEX

7.1.5 URL_REGEX_MAP

7.1.6 Example Settings

```
TEXT_VARIATIONS = {
    'DIMENSIONS': {
        'audience': {
            'ad': {
                'name': u'Adult',
                'fallback': None,
            },
            'tn': {
                'name': u'Teen',
                'fallback': 'ad',
            },
        },
    },
}
```

```

    },
    'cd': {
        'name': u'Child',
        'fallback': 'tn',
    }
},
'text_difficulty': {
    'l': {
        'name': u'Low',
        'fallback': 's',
    },
    's': {
        'name': u'Standard',
        'fallback': None,
    },
    'a': {
        'name': u'Advanced',
        'fallback': 's',
    }
},
'language': {
    'en': {
        'name': u'English',
        'fallback': None,
    },
    'es': {
        'name': u'Español',
        'fallback': 'en',
        'domain': 'es.example.com',
    },
    'es-mx': {
        'name': u'Español mexicano',
        'fallback': 'es',
        'domain': 'es-mx.example.com',
    },
    'fr': {
        'name': u'Français',
        'fallback': 'en',
        'domain': 'fr.example.com',
    }
},
},
'DIMENSION_PRIORITY': ['language', 'audience', 'text_difficulty'],
'MODELS': {
    'app.model': ['field1', 'field2'],
    'app2.model': ['field1', ]
},
'DEFAULT_URL_REGEX': '{language}/{path};{audience},{text_difficulty}',
'URL_REGEX_MAP': {
    'app.model': '{path};{language},{audience},{text_difficulty}',
}
}

```


7.2 Implementation Notes

1. There could be potentially a large number of variations for an individual field.
2. We want to keep the number of database calls/accesses and data storage to a minimum.
3. We must track some metadata about the variations, such as: variations used, invalidated variations.

7.2.1 Methods

1. **Related table(s)** There are several variations:
 - One table for each model with variations (using a Foreign Key)
 - One table for all models with variations (using a Generic Foreign Key)
 - Use a JSON field for all text variations of an object
 - A separate row for each text variation
2. **Additional field(s) in the model** There are two variations:
 - Use a JSON field for all text variations of an object
 - A separate field for each potential text variation

7.2.2 Implementation

A combination of both methods will be used in a non-normalized way to provide flexibility and performance. A variation table for each model with variations will store each variation for each field. A JSON/serialized field on the model will store a cached version of the same information.

The fields in the variation table are:

field	Description
<model>	Foreign key to the row in the master table
field-name	Name of the field in which this variation applies
<name>	A field for each defined dimension to contain the dimension variant
is_invalid	A boolean flag to indicate the variation is currently invalid, as the default variation has changed. Declaring variations invalid instead of deleting them should be an option as to allow the site to continue to display the content.
value	The value of this variation for this field

The cache field contains a structure based on the current *DIMENSION_PRIORITY* settings.

Warning: If this setting changes the cache field for each item in every table will need to be recalculated. There is a management command to do this.

The cache is a hierarchical dictionary with each field at the top level and then each dimension hierarchically contained in order of priority.

```
{
  'field': {
    'dim1-variant1': {
```

```

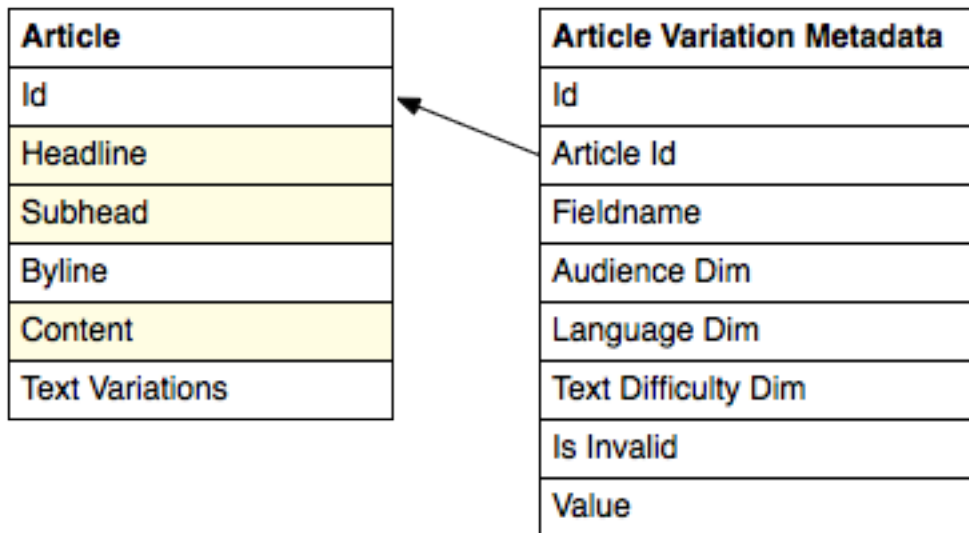
    'dim2-variant1': 'variation value',
    'dim2-variant2': 'variation value',
  }
}
}

```

7.2.3 Example Walkthrough

Assuming that we have three dimensions: audience, text difficulty, and language; a table called article with three variable fields: headline, subhead, and content; the table and metadata table might look like:

Note: See [Example Settings](#) for the configuration.



If you added a row into article, you might have

Id	Headline	Subhead	Byline	Content	Text Variations
1	First article		Carmen	Lots of ...	{'headline':{'en':{'ad'...

The Text Variations field would contain a serialized dictionary:

```

{
  'headline': {
    'en': { 'ad': { 's': 'First Article', }, }, },
  'subhead': {
    'en': { 'ad': { 's': '', }, }, },
  'content': {
    'en': { 'ad': { 's': 'Lots of ...', }, }, },
}

```

And the Audience Variation table contains

Id	Article Id	Field-name	Audience Dim	Language Dim	Text Difficulty Dim	Is Invalid	Value
1	1	headline	ad	en	s	False	First Article
2	1	subhead	ad	en	s	False	Lots of ...
3	1	content	ad	en	s	False	

If you add a spanish variation to the headline and content fields, the Text Variations field would contain:

```
{
  'headline': {
    'en': { 'ad': { 's': 'First Article', }, },
    'es': { 'ad': { 's': u'Artículo Primero', }, },
  },
  'subhead': {
    'en': { 'ad': { 's': '', }, },
  },
  'content': {
    'en': { 'ad': { 's': 'Lots of ...', }, },
    'es': { 'ad': { 's': 'Mucho ...', }, },
  },
}
```

And the Audience Variation table contains

Id	Article Id	Field-name	Audience Dim	Language Dim	Text Difficulty Dim	Is Invalid	Value
1	1	headline	ad	en	s	False	First Article
2	1	subhead	ad	en	s	False	Lots of ... Artículo Primero Mucho ...
3	1	content	ad	en	s	False	
4	1	headline	ad	es	s	False	
5	1	content	ad	es	s	False	

7.2.4 Schema Migrations

Adding/removing the JSON field

Adding/removing a metadata dimension field

7.3 Management Commands

7.4 Template Tags

```
{% variation_url object variation %}
```

returns a url of the object with variations applied to it

```
{% get_text_variations object %}
```

Using the request, or other method, if possible, set the object's <field>_variation attributes and possibly update the text_variations context variable

Required before using the content object.

```
{% get_variation object field dimension=val dimension2=val ... %}
```

Returns the best variation fit for the object's specified field

7.5 TextVariationMiddleware

Needs to be first in the list so it can properly modify the path information.

The middleware modifies the request

- Matches request path against the configured regular expressions
- If it matches, extract the dimensions and set into request.META['TEXT_VARIATIONS']
- Join the unnamed groups in the matching regular expression with "/"
- Set request.path, request.path_info and request.META['PATH_INFO'] to the resulting value

7.6 Object Methods

```
get_FIELD_variation (dimension=val, dimension2=val, ...)
```

pulls the variation for FIELD that fits the requested dimension values. Dimensions not specified are assumed to be the default. Will fall back to default variation if none exist.

ITEMS FOR FUTURE VERSIONS

- Configurable storage engine for variation storage, to allow for example:
 - hstore in PostgreSQL
 - segmentation of variations (e.g. a field for each language which stores variations)

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*