
django-lazy-tags Documentation

Release 0.1

Grant McConnaughey

Sep 27, 2017

Contents

1	Installation	3
2	Usage	5
3	The lazy_tag decorator	7
4	Settings	9
5	Customizing the Loading Animation	11
6	Current Limitations	13

A Django app for loading template tags over AJAX after a page has loaded.

CHAPTER 1

Installation

1. Install via pip

```
pip install django-lazy-tags
```

2. Add to installed apps

```
INSTALLED_APPS = (  
    # ...  
    'lazy_tags',  
)
```

3. Add the lazy tags urls to your root urlconf.

```
urlpatterns = patterns('',  
    # ...  
    url(r'^lazy_tags/', include('lazy_tags.urls')),  
)
```


CHAPTER 2

Usage

First, load the `lazy_tags` library in your templates.

```
{% load lazy_tags %}
```

Then, call the `lazy_tag` template tag passing your tag name as the first parameter. The format is `tag_library.tag_name` where `tag_library` is what you would load at the top of the page (e.g. `my_tags`) and `tag_name` is the name of your template tag (e.g. `my_template_tag`). After the first argument to `lazy_tag` simply pass the rest of the args and kwargs just as you would pass them to your own tag.

This:

```
{% load my_tags %}

{% my_template_tag arg1 arg2 kw1='hello' kw2='world' %}
```

Becomes this:

```
{% load lazy_tags %}

{% lazy_tag 'my_tags.my_template_tag' arg1 arg2 kw1='hello' kw2='world' %}
```

After placing your template tags in the template you still need to specify where you would like the AJAX JavaScript to output to the page. That is what the `lazy_tags_js` tag is for:

```
{% block js-additional %}
    {% lazy_tags_js %}
{% endblock %}
```

This will spit out the JavaScript required to run the AJAX. The JavaScript changes depending on your `LAZY_TAGS_AJAX_JS` setting.

CHAPTER 3

The lazy_tag decorator

django-lazy-tags also includes a decorator that can be used on template tags that use `simple_tag`. When using the `lazy_tag` decorator you can use your template tags exactly the same as before and they will use AJAX.

```
from lazy_tags.decorators import lazy_tag

@register.simple_tag
@lazy_tag
def show_user(pk):
    user = User.objects.get(pk=pk)
    return render_to_string('user/show_user.html', {
        'user': user,
    })
```

There are a few caveats with this method. First, the decorator currently only works with tags that use `simple_tag`. Hopefully this will work with `inclusion_tag` in the future. Secondly, the `lazy_tag` decorator must come *after* the `simple_tag` decorator.

CHAPTER 4

Settings

LAZY_TAGS AJAX_JS The library to use to run AJAX. Options are 'javascript', 'jquery', or 'prototype'. Defaults to 'jquery'.

LAZY_TAGS_CACHE_TIMEOUT The timeout on each lazy tag cache. Defaults to 60 seconds (60 seconds).

LAZY_TAGS_ERROR_MESSAGE The error message to display if the AJAX request fails. Defaults to 'An error occurred.'

Customizing the Loading Animation

This is the default HTML on the page before the AJAX request completes:

```
<div id="{ tag_id }" class="lazy-tag">
  <div class="lazy-tag-spinner-container"
    style="width: 100%; text-align: center;">
    
  </div>
</div>
```

To customize the loading animation, override the `lazy-tag`, `lazy-tag-spinner-container`, or `lazy-tag-spinner` classes in your CSS.

Current Limitations

- Does not work with tags that take context.
- Template tag arguments must be serializable (str, unicode, int, float, etc.).