django-pagination - pagination without modifying views

This fork notes

This fork was created from ericflo's django-pagination to fix i18n support and to add Russian translation from issue http://code.google.com/p/django-pagination/issues/detail?id=45. Also I outputed orifginal instructions to this Readme file.

Installation

This fork can be easily installed with pip:

```
pip install -e git+git://github.com/alekseyr/django-pagination.git#egg=django-pagination
```

Usage

`django-pagination` allows for easy Digg-style pagination without modifying your views.

There are really 5 steps to setting it up with your projects (not including installation, which is covered in INSTALL.txt in this same directory.)

1. List this application in the `INSTALLED_APPS` portion of your settings file. Your settings file might look something like:

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

2. Install the pagination middleware. Your settings file might look something like:

   ```
   MIDDLEWARE_CLASSES = (
       # ...
       'pagination.middleware.PaginationMiddleware',
   )
   ```

3. If it's not already added in your setup, add the request context processor. Note that context processors are set by default implicitly, so to set them explicitly, you need to copy and paste this code into your under the value TEMPLATE_CONTEXT_PROCESSORS:

   ```
   ("django.core.context_processors.auth",
   "django.core.context_processors.debug",
   "django.core.context_processors.i18n",
   "django.core.context_processors.media",
   "django.core.context_processors.request")
   ```

4. Add this line at the top of your template to load the pagination tags:

   ```
   {% load pagination_tags %}
   ```

5. Decide on a variable that you would like to paginate, and use the autopaginate tag on that variable before iterating over it. This could take one of two forms (using the canonical `object_list` as an

example variable):

```
{% autopaginate object_list %}
```

This assumes that you would like to have the default 20 results per page. If you would like to specify your own amount of results per page, you can specify that like so:

```
{% autopaginate object_list 10 %}
```

Note that this replaces `object_list` with the list for the current page, so you can iterate over the `object_list` like you normally would.

6. Now you want to display the current page and the available pages, so somewhere after having used autopaginate, use the paginate inclusion tag:

```
{% paginate %}
```

This does not take any arguments, but does assume that you have already called autopaginate, so make sure to do so first.

That's it! You have now paginated `object_list` and given users of the site a way to navigate between the different pages--all without touching your views.

Optional Settings

In django-pagination, there are no required settings. There are, however, a small set of optional settings useful for changing the default behavior of the pagination tags. Here's an overview:

`PAGINATION_DEFAULT_PAGINATION`

The default amount of items to show on a page if no number is specified.

`PAGINATION_DEFAULT_WINDOW`

The number of items to the left and to the right of the current page to display (accounting for ellipses).

`PAGINATION_DEFAULT_ORPHANS`

The number of orphans allowed. According to the Django documentation, orphans are defined as:

```
The minimum number of items allowed on the last page, defaults to zero.
```

`PAGINATION_INVALID_PAGE_RAISES_404`

Determines whether an invalid page raises an `Http404` or just sets the `invalid_page` context variable. `True` does the former and `False` does the latter.