django-memento-framework Documentation

Release 0.0.1

Ben Welsh

December 01, 2015

A set for helpers for Django web sites to enable the Memento framework for time-based access.

CHAPTER 1

How does it work?

Django's class-based views are used to retrieve archived data and format it according to Memento's rules.

The generic views in this package can be used to quickly:

- Publish a TimeMap that lists all of the archived versions of each URL in your archive
- Host a TimeGate that handles requests that include a URL and a timestamp, redirecting to the detail view for the nearest archive
- Enrich detail views in the archive to include the extra metadata required by the Memento system

CHAPTER 2

Documentation

2.1 Getting started

2.1.1 Installation

Before you begin, you should have a Django project created and configured.

Install our library from PyPI, like so:

```
$ pip install django-memento-framework
```

Edit your `settings.py` and add this app to your `INSTALLED_APPS` list.

```
INSTALLED_APPS = (
    # ...
    # other apps would be above this of course
    # ...
    'memento',
)
```

2.1.2 Example project

For the purposes of this introduction, imagine a Django project that archives a list of URLs each hour. A simple set of models could look something like this.

```
from django.db import models


class Page(models.Model):
    """
    A web page that is archived each hour.
    """
    url = models.URLField()


class ArchivedHTML(model.Model):
    """
    A HTML snapshot of a single web page.
    """
    page = models.ForeignKey(Page)
    timestamp = models.DateTimeField()
    html = models.TextField()

    def get_absolute_url(self):
        return '/%s/' % self.pk
```

We won't go into all the code you'd need for this archive to actually work, but I bet you can imagine the sort of thing you'd need. Imagine it's there and it's been archiving URLs every hour for a few years.

2.1.3 TimeMaps

A good first step is create a TimeMap that will index all of the archives in your database. This is much like the sitemaps created for traditional search engines, but expanded to include datetime metadata.

This library includes a generic view that will publish a queryset in Memento's link format, that can be optionally paginated. It is designed to emulate Django's built-in feed framework and operates in much the same way.

You could start by create a pattern in `urls.py` that will accept a web page's URL with the goal of returning all the snapshots for that resource in our example archive.

```
url(
    r'^timemap/(?P<url>.*)$',
    views.ExampleTimemapLinkList(),
    name="timemap-archivedhtml"
),
```

That URL now needs to connect to a view named `ExampleTimemapLinkList` so let's go make it in your `views.py` file. It will inherit from our custom class and include configuration to pull the `Page` object from the database that matches the submitted URL and convert a list of all its archived HTML into a Timemap.

```
from memento.timemap import TimemapLinkList


class ExampleTimemapLinkList(TimemapLinkList):
    paginate_by = 1000

    def get_object(self, request, url):
        """
        Take the URL from the request and check if we have it in our Page model.
        """
        return get_object_or_404(Page, url=url)

    def get_original_url(self, obj):
        """
        Return the original URL being archived from the Page model after its been pulled.
        """
        return obj.url

    def memento_list(self, obj):
        """
        Pull a queryset set of all the ArchivedHTML records of the Page object.
        """
        return ArchivedHTML.objects.filter(page=obj)

    def memento_datetime(self, item):
        """
        Return the datetime when the archived was saved from each ArchivedURL object.
        """
        return item.timestamp
```

Now if you fired up your test server and visited http://localhost:8000/timemap/link/http://example.com/ You'd get a paginated index of all the archives saved from the page `example.com`.

2.1.4 TimeGate

The natural next step is to create the other pillar of the Memento system, a TimeGate. It is a view that when given a request that includes a URL and a timestamp will redirect to the detail page for the nearest archive.

A good first step is to create an URL in `urls.py`.

```
url(
    r'^timegate/(?P<url>.*)$',
    views.ExampleTimeGateView.as_view(),
    name="timegate"
),
```

Next you need to create the view included there.

```
class ExampleTimeGateView(TimeGateView):
    model = ArchivedHTML
    url_field = 'page__url' # You can walk across ForeignKeys like normal
    datetime_field = 'timestamp'
    timemap_pattern_name = "timemap-archived" # The name of the timemap URL we created above
```

This will now return a 302 redirect when given a URL in the request with a header of *Accept-Datetime*. A good way to check out if this is working is to use cURL on the command line.

```
$ url -X HEAD -i http://localhost:8000/timegate/http://archivedsite.com/ --header "Accept-Datetime: F
HTTP/1.1 302 Moved Temporarily
Server: Apache/2.2.22 (Ubuntu)
Link: <http://archivedsite.com/>; rel="original", <http://localhost:800/timemap/link/http://archiveds
Location: http://www.example.com/screenshot/100/
Vary: accept-datetime
```

2.1.5 Memento detail view

The detail page for each snapshot in the archive should also be enriched to include Memento metadata in its response. This is done by using our extended version of Django's generic DetailView.

```
from memento.timegate import MementoDetailView


class ExampleMementoDetailView(MementoDetailView):
    model = ArchivedHTML
    datetime_field = 'timestamp'
    timemap_pattern_name = "timemap-archivedhtml"
    timegate_pattern_name = "timegate"

    def get_original_url(self, obj):
        return obj.page.url
```

That can be linked to a pattern in the `urls.py` file like any other Django detail view.

```
url(
    r'^archived-html/(?P<pk>\d+)/$',
    views.ExampleMementoDetailView.as_view(),
    name='archivedhtml-detail'
)
```

And that's it! Get all that going and your archive is ready to be indexed by http://mementoweb.org/ and integrated into Memento-based services.

2.2 Generic views

2.2.1 MementoDetailView

class `MementoDetailView`(*DetailView*)

Extends Django's generic DetailView to describe an archived resource.

A set of extra headers is added to the response. They are:

- The `Memento-Datetime` when the resource was archived.
- A `Link` that includes the URL of the original resource as well as the location of the TimeMap the publishes the directory of all versions archived by your site and a TimeGate where a datetime can be submitted to find the closest mementos for this resource.

`datetime_field`

A string attribute that is the name of the database field that contains the timestamp when the resource was archived. Default `'datetime'`.

`timemap_pattern_name`

The name of the URL pattern for this site's TimeMap that, given the original url, is able to reverse to return the location of the map that serves as the directory of all versions of this resource archived by your site. Optional.

`timegate_pattern_name`

The name of the URL pattern for this site's TimeGate that, given the original url, is able to reverse to return the location of the url where a datetime can be submitted to find the closest mementos for this resource. Optional.

`get_original_url`()

A method that, given the object being rendered by the view, will return the original URL of the archived resource.

Example myapp/views.py

```
from memento.timegate import MementoDetailView


class ExampleMementoDetailView(MementoDetailView):
    """
    A Memento-enabled detail page for a screenshot in my example
    archive.

    It is linked to a url that looks like something like:

        url(
            r'^screenshot/(?P<pk>\d+)/$',
            views.ExampleMementoDetailView.as_view(),
            name='screenshot-detail'
        )
    """
    model = Screenshot
    datetime_field = 'timestamp'
    timemap_pattern_name = "timemap-screenshot"
    timegate_pattern_name = "timegate-screenshot"

    def get_original_url(self, obj):
        return obj.site.url
```

Example response

```
$ curl -X HEAD -i http://www.example.com/screenshot/100/
HTTP/1.1 200 OK
Server: Apache/2.2.22 (Ubuntu)
Link: <http://archivedsite.com/>; rel="original", <http://www.example.com/timemap/link/http://ar
Memento-Datetime: Fri, 1 May 2015 00:00:01 GMT
```

2.2.2 TimemapLinkList

class TimemapLinkList (*object*)

Returns a queryset list in Memento's TimeMap link format. Can be optionally paginated. Designed to emulate Django's built-in feed framework.

paginate_by

An optional integer attribute that will trigger the pagination of the result set so that each page includes the provided number of objects.

get_object (*request*, *url*)

Returns the model object for the provided original URL. Required.

get_original_url (*obj*)

Returns the original URL that was archived given the model object. Required.

memento_list (*obj*)

Returns the queryset of archived resources associated with the submitted original URL given its object. Required.

memento_datetime (*item*)

Returns the timestamp of when an archived resource was retrieved given its object. Required.

Example myapp/feeds.py

```
from memento.timemap import TimemapLinkList


class ExampleTimemapLinkList(TimemapLinkList):
    """
    A Memento TimeMap that, given a URL, will return a list of archived objects for that page in

    It is linked to a url that looks like something like:

        url(
            r'^timemap/link/(?P<url>.*)$',
            feeds.ExampleTimemapLinkList(),
            name="timemap-screenshot"
        ),

    """
```

```
        paginate_by = 1000

        def get_object(self, request, url):
            return get_object_or_404(Site, url__startswith=url)

        def get_original_url(self, obj):
            return obj.url

        def memento_list(self, obj):
            return Screenshot.objects.filter(site=obj)

        def memento_datetime(self, item):
            return item.timestamp
```

Example response

```
    $ curl -i http://www.example.com/timemap/http://archivedsite.com/
    HTTP/1.0 200 OK
    Server: Apache/2.2.22 (Ubuntu)
    Content-Type: application/link-format; charset=utf-8

    <http://archivedsite.com/>;rel="original",
     <http://www.pastpages.org/timemap/link/http://archivedsite.com/>
       ; rel="self";type="application/link-format",
     <http://www.example.com/timemap/link/http://archivedsite.com?page=1>
       ; rel="timemap";type="application/link-format",
     <http://www.example.com/timemap/link/http://archivedsite.com/?page=2>
       ; rel="timemap";type="application/link-format",
     <http://www.example.com/timemap/link/http://archivedsite.com/?page=3>
       ; rel="timemap";type="application/link-format",
     <http://www.example.com/timemap/link/http://archivedsite.com/?page=4>
       ; rel="timemap";type="application/link-format"
```

2.2.3 TimeGateView

class **`TimeGateView`** (*RedirectView*)

Creates a TimeGate that handles a request with a 'Accept-Datetime' headers and returns a response that redirects to the corresponding Memento.

`model`

A Django database model where the object will be drawn with a `Model.objects.filter()` query. Optional. If you want to provide a more specific list, define the `queryset` attribute instead.

`queryset`

The list of objects that will be provided to the template. Can be any iterable of items, not just a Django queryset. Optional, but if this attribute is not defined the `model` attribute must be defined.

`datetime_field`

A string attribute that is the name of the database field that contains the timestamp when the resource was archived. Default `'datetime'`.

`url_kwarg`

The name for the keyword argument in the URL pattern that will be used to filter the queryset down to objects archived for the resource. Default `'url'`.

`url_field`

A string attribute that is the name of the database field that contains the original URL archived. Default `'url'`.

timemap_pattern_name
The name of the URL pattern for this site's TimeMap that, given the original url, is able to reverse to return the location of the map that serves as the directory of all versions of this resource archived by your site. Optional.

Example myapp/views.py

```
from memento.timegate import TimeGateView


class ExampleTimeGateView(TimeGateView):
    """
    A Memento TimeGate that, given a timestamp, will redirect to the detail page for a screensho

    It is linked to a url that looks like something like:

        url(
            r'^timegate/(?P<url>.*)$',
            views.ExampleTimeGateView.as_view(),
            name="timegate"
        ),

    """
    model = Screenshot
    url_field = 'site__url' # You can walk across ForeignKeys like normal
    datetime_field = 'timestamp'
    timemap_pattern_name = "timemap-screenshot"
```

Example response

```
$ curl -X HEAD -i http://www.example.com/timegate/http://archivedsite.com/ --header "Accept-Date
HTTP/1.1 302 Moved Temporarily
Server: Apache/2.2.22 (Ubuntu)
Link: <http://archivedsite.com/>; rel="original", <http://www.example.com/timemap/link/http://ar
Location: http://www.example.com/screenshot/100/
Vary: accept-datetime
```

CHAPTER 3

Other resources

- Code repository: github.com/pastpages/django-memento-framework
- Issues: github.com/pastpages/django-memento-framework/issues
- Packaging: pypi.python.org/pypi/django-memento-framework
- Testing: travis-ci.org/pastpages/django-memento-framework
- Coverage: coveralls.io/r/pastpages/django-memento-framework

Index