Lab Website Documentation

Release 0.1.0

Dave Bridges

September 29, 2012

CONTENTS

Contents:

CHAPTER

ONE

PAPERS

This application will store and display the relevant `Publication` objects and associated data.

1.1 Current Functionality

- Includes and displays papers from the lab or other interesting papers.
- Papers are marked up with microdata markup from http://schema.org.
- There is an API estabished which serves some `Publication` information in json or xml format. See `papers.api` for details.
- There will be included comment threads at each paper served by Disqus.
- API's from Altmetric and PLOS are used to also display altmetrics for these papers.
- Papers will also link to author profiles (see the `personnel` app).
- It is possible to tweet, like (through facebook) or +1 (through google plus) a paper, though the functionality of these are not yet refined.

1.2 Longer Term Goals

- Another idea is to have hidden discussions of papers, or papers which are not publicly displayed and require a login.
- Incorporate Mendeley, TotalImpact and PubMedCentral APIs.
- Potentially convert to a facebook app with custom actions and objects.
- Markup with opengraph tags and incorporate twitter cards.
- The `Publication` objects are manually entered but I hope to have these be automatically be generated from CrossRef or Mendeley APIs

1.3 Source Code Documentation

1.3.1 Models

This file is the model configuration file for the :mod‘papers‘ app.

There are two models in this app, Publication and AuthorDetails

class papers.models.**AuthorDetails**(**args*, ***kwargs*)
This is a group of authors for a specific paper.

Because each Publication has a list of authors and the order matters, the authors are listed in this linked model. The authors are defined by the Person model class, which is also the UserProfile class. This model has a ManyToMany link with a paper as well as marks for order, and whether an author is a corresponding or equally contributing author.

class papers.models.**Publication**(**args*, ***kwargs*)
This model covers Publication objects of several types.

The publication fields are based on Mendeley and PubMed fields. For the author, there is a ManyToMany link to a group of authors with the order and other details, see AuthorDetails.

doi_link()
This turns the DOI into a link.

full_pmcid()
Converts the integer to a full PMCID

get_absolute_url(**moreargs*, ***morekwargs*)
the permalink for a paper detail page is **/papers/<title_slug>**

save(**args*, ***kwargs*)
The title is slugified upon saving into title_slug.

1.3.2 Views

This app contains the views for the :mod‘papers‘ app.

There are three views for this app, LaboratoryPaperList, InterestingPaperList and PaperDetailView

class papers.views.**InterestingPaperList**(***kwargs*)
This class generates the view for interesting-papers located at **/papers/interesting**.

This is filtered based on whether the Publication is marked as interesting_paper = True.

get_context_data(***kwargs*)
This method adds to the context the paper-list-type = interesting.

class papers.views.**LaboratoryPaperList**(***kwargs*)
This class generates the view for laboratory-papers located at **/papers**.

This is filtered based on whether the Publication is marked as laboratory_paper = True.

get_context_data(***kwargs*)
This method adds to the context the paper-list-type = interesting.

class papers.views.**PaperCreate**(***kwargs*)
This view is for creating a new Publication.

It requires the permissions to create a new paper and is found at the url **/paper/new**.

model
alias of Publication

class papers.views.**PaperDelete**(***kwargs*)
This view is for deleting a Publication.

It requires the permissions to delete a paper and is found at the url **/paper/<slug>/delete**.

model
alias of Publication

class papers.views.**PaperDetailView**(***kwargs*)
This class generates the view for paper-details located at **/papers/<title_slug>**.

model
alias of Publication

render_to_response(*context*, ***kwargs*)
The render_to_response for this view is over-ridden to add the api_keys context processor.

class papers.views.**PaperUpdate**(***kwargs*)
This view is for updating a Publication.

It requires the permissions to update a paper and is found at the url **/paper/<slug>/edit**.

model
alias of Publication

1.3.3 URLS

This package has the url encodings for the papers app.

1.3.4 Tests

This package contains the unit tests for the papers app.

It contains view and model tests for each model, grouped together. Contains the two model tests:

- PublicationModelTests
- AuthorDetailsModelTests

The API tests:

- PublicationResourceTests

And the view tests:

- PublicationViewTests

class papers.tests.**AuthorDetailsModelTests**(*methodName='runTest'*)
This class tests varios aspects of the AuthorDetails model.

setUp()
Instantiate the test client. Creates a test user.

tearDown()
Depopulate created model instances from test database.

test_authordetail_unicode()
This tests that the unicode representaton of an AuthorDetails object is correct.

test_create_new_authordetail_all()
This test creates a AuthorDetails with the required information only.

test_create_new_authordetail_minimum()
This test creates a AuthorDetails with the required information only.

class papers.tests.**PublicationModelTests**(*methodName='runTest'*)
This class tests various aspects of the Publication model.

setUp()
Instantiate the test client. Creates a test user.

tearDown()
Depopulate created model instances from test database.

test_create_new_paper_minimum()
This test creates a Publication with the required information only.

test_full_pmcid()
This tests that a correct full PMCID can be generated for a Publication.

test_paper_absolute_url()
This tests the title_slug field of a Publication.

test_paper_doi_link()
This tests the title_slug field of a Publication.

test_paper_title_slug()
This tests the title_slug field of a Publication.

test_paper_unicode()
This tests the unicode representation of a Publication.

class papers.tests.**PublicationResourceTests**(*methodName='runTest'*)
This class tests varios aspects of the PublicationResource API model.

api_publication_detail_test()
This tests that the API correctly renders a particular Publication objects.

api_publication_list_test()
This tests that the API correctly renders a list of Publication objects.

setUp()
Instantiate the test client. Creates a test user.

tearDown()
Depopulate created model instances from test database.

class papers.tests.**PublicationViewTests**(*methodName='runTest'*)
This class tests the views for Publication objects.

setUp()
Instantiate the test client. Creates a test user.

tearDown()
Depopulate created model instances from test database.

test_interesting_papers_list()
This tests the interesting-papers view ensuring that templates are loaded correctly.

This view uses a user with superuser permissions so does not test the permission levels for this view.

test_lab_papers_list()
This tests the laboratory-papers view ensuring that templates are loaded correctly.

This view uses a user with superuser permissions so does not test the permission levels for this view.

test_publication_view()
This tests the paper-details view, ensuring that templates are loaded correctly.

This view uses a user with superuser permissions so does not test the permission levels for this view.

test_publication_view_create()
This tests the paper-new view, ensuring that templates are loaded correctly.

This view uses a user with superuser permissions so does not test the permission levels for this view.

test_publication_view_delete()
This tests the paper-delete view, ensuring that templates are loaded correctly.

This view uses a user with superuser permissions so does not test the permission levels for this view.

test_publication_view_edit()
This tests the paper-edit view, ensuring that templates are loaded correctly.

This view uses a user with superuser permissions so does not test the permission levels for this view.

1.3.5 Admin

This package sets up the admin interface for the papers app.

class papers.admin.**AuthorDetailsAdmin**(*model*, *admin_site*)
The AuthorDetails model admin is the default.

class papers.admin.**PublicationAdmin**(*model*, *admin_site*)
The Publication model admin is the default.

1.3.6 API

This package controls API access to the papers app.

Overview

The API for the papers application provides data on publications. The data can be provided as either a group of publications or as a single publication. Only GET requests are accepted. These urls are served at the endpoint **/api/v1/publications/**, and depends on your server url. For these examples we will presume that you can reach this endpoint at **http://yourserver.org/api/v1/publications/**. Currently for all requests, no authentication is required. The entire API schema is available at:

```
http://yourserver.org/api/v1/publications/schema/?format=xml
http://yourserver.org/api/v1/publications/schema/?format=json
```

Sample Code

Either group requests or single publication requests can be served depending on if the primary key is provided. The request URI has several parts including the servername, the api version (currently v1) then the item type (publications). There must be a trailing slash before the request parameters (which are after a **?** sign and separated by a **&** sign).

For a collection of publications

For a collection of publications you can request:

```
http://yourserver.org/api/v1/publications/?format=json
```

This would return all publications in the database. This would return the following json response with two JSON objects, meta and objects. The meta object contains fields for the limit, next, offset, previous and total_count for the series of objects requested. The objects portion is an array of the returned publications. Note the id field of a publication. This is used for retrieving a single publication. Collections can also be filtered based on type or year:

```
http://yourserver.org/api/v1/publications/?format=json&year=2012
http://yourserver.org/api/v1/publications/?format=json&type=journal-article
http://yourserver.org/api/v1/publications/?format=json&type=journal-article&year=2012
http://yourserver.org/api/v1/publications/set/1;3/?format=json
```

The last example requests the publications with id numbers 1 and 3.

For a single publication

To retrieve a single publication you need to know the primary key of the object. This can be found from the id parameter of a collection (see above) or from the actual object page. You can retrieve details about a single article with a call such as:

```
http://yourserver.org/api/v1/publications/2/?format=json
```

In this case **2** is the primary key (or id field) of the publication in question.

Reference

Request Parameters

The following are the potential request variables. You must supply a format, but can also filter based on other parameters. By default 20 items are returned but you can increase this to all by setting limit=0.

Parameter	Potential Values
format	**json** or **xml**
year	**2008**
type	**journal-article** or **book-section**
laboratory_paper	**true** or **false**
limit	**0** for all, any other number

Response Values

The response (in either json or xml) provides the following fields for each object (or for the only object in the case of a single object request).

Field	Explanation	Sample Value
absolute_url	the url of the page on this site	/papers/tc10-is-regulated-by-caveolin-in-3t3-l1-adipocytes/
abstract	abstract or summary	some text...
date_added	data added to this database	2012-08-18
date_last_modified	last modified in database	2012-08-25
doi	digital object identifier	10.1371/journal.pone.0042451
id	the database id number	1
interesting_paper	whether the paper is marked as an interesting paper	false
issue	the issue of the journal	8
journal	the name of the journal	PLOS One
laboratory_paper	whether the paper is from this laboratory	true
mendeley_id	the mendeley id number for the paper	null
mendeley_url	the mendeley url for the paper	
pages	page range for the paper	e42451
pmcid	PubMed Central id number	null
pmid	PubMed id number	22900022
resource_uri	a link to the api for this publication	/api/v1/publications/1/
title	the title of the paper	TC10 Is Regulated by Caveolin in 3T3-L1 Adipocytes.
title_slug	slugified title of the paper	tc10-is-regulated-by-caveolin-in-3t3-l1-adipocytes
type	type of publication	journal-article
volume	volume of the article in a journal	7
year	publication year	2012

class papers.api.**PublicationResource**(*api_name=None*)

This generates the API resource for Publication objects.

It returns all publications in the database. Authors are currently not linked, as that would require an API to the personnel app.

class Meta

The API serves all Publication objects in the database..

object_class

alias of Publication

1.3.7 Sitemap

This package controls the sitemap for the papers app.

This sitemap will be generated at **/sitemap-papers.xml**

class papers.sitemap.**LabPublicationsSitemap**

This sitemap lists all Publication objects to be indexed.

It only includes papers from this laboratory (laboratory_papers=True)

items()

Filters Publication to show only laboratory papers.

lastmod(*paper*)

lastmod uses the last modification of the paper (not the comments).

1.3.8 Context Processors

This file contains context processors to pass api keys to templates as part of the papers app.

This is needed to properly render the PLOS API requests.

papers.context_processors.**api_keys**(*request*)

A context processor to add the a dictionary of api keys to the context.

If no accounts are specified then empty strings should be passed.

CHAPTER

TWO

INDICES AND TABLES

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

PYTHON MODULE INDEX

p

INDEX