pinax-ratings Documentation

Release 2.0.0

Pinax

Sep 27, 2017

Contents

Provides a site with user ratings of objects.

CHAPTER 1

Development

The source repository can be found at https://github.com/pinax/pinax-ratings

Contents

Installation

- To install pinax-ratings:

```
pip install pinax-ratings
```

- Add `pinax-ratings` to your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = (
    # other apps
    "pinax.ratings",
)
```

- See the list of *Settings* to modify pinax-ratings's default behavior and make adjustments for your website.
- Lastly you will want to add *pinax-ratings.urls* to your urls definition:

```
...
url(r"^ratings/", include("pinax.ratings.urls")),
...
```

- Optionally, if want to use the ratings category feature of *pinax-ratings* then you will need to add the *pinax-RATINGS_CATEGORY_CHOICES* setting in your *settings.py*:

```
PINAX_RATINGS_CATEGORY_CHOICES = {
    "app.Model": {
        "exposure": "How good is the exposure?",
        "framing": "How well was the photo framed?",
```

```
        "saturation": "How would you rate the saturation?"
    },
    "app.Model2": {
        "grammar": "Good grammar?",
        "complete": "Is the story complete?",
        "compelling": "Is the article compelling?"
    }
}
```

Usage

Integrating *pinax-ratings* into your project is just a matter of using a couple of template tags and wiring up a bit of javascript. The rating form is intended to function via AJAX and as such returns JSON.

Firstly, add load the template tags for *pinax-ratings*:

```
{% load pinax_ratings_tags %}
```

Then, if you want to display an overall rating average for an object you can set a context variable and display it:

```
{% overall_rating obj as the_overall_rating %}

<div class="overall_rating">{{ the_overall_rating }}</div>
```

Likewise for displaying a user's rating:

```
{% user_rating request.user obj as the_user_rating %}

<div class="user_rating">{{ the_user_rating }}</div>
```

If you want to add an AJAX form for allowing a user to set a rating, add the following in the appropriate location on your page:

```
<div id="user_rating"></div>
```

And then add this near the end of your HTML <*body*> to emit some Javascript libraries and hook up the ratings UI:

```
{% user_rating_js request.user obj %}
```

If you want to do any rating based on categories of ratings for an object or objects then you do the same as above but just use an optional argument on the tags:

```
{% overall_rating obj "accuracy" as category_rating %}

<div class="overall_rating category-accuracy">
    {{ category_rating }}
</div>
```

and:

```
{% user_rating request.user obj "accuracy" as category_rating %}

<div class="user_rating category-accuracy">
    {{ category_rating }}
</div>
```

and:

```
<div id="user_rating" class="category-accuracy"></div>

{% user_rating_js request.user obj "accuracy" %}
```

Settings

PINAX_RATINGS_NUM_OF_RATINGS

Default 5

Defines the number of different rating choices there will be.

PINAX_RATINGS_CATEGORY_CHOICES

Default *None*

Defines a dictionary of choices for different models for the application of ratings along different dimensions rather than just a single rating for an object.

It should follow the format of a dictionary of dictionaries. For example, think of the context of a website that allowed ratings of photographs and articles published by other users:

```
PINAX_RATINGS_CATEGORY_CHOICES = {
    "app.Model": {
        "exposure": "How good is the exposure?",
        "framing": "How well was the photo framed?",
        "saturation": "How would you rate the saturation?"
    },
    "app.Model2": {
        "grammar": "Good grammar?",
        "complete": "Is the story complete?",
        "compelling": "Is the article compelling?"
    }
}
```

Templates

pinax-ratings comes with one template that is a minimal snippet that gets rendered from the template tags for displaying the rating form.

_script.html

This is a snippet that renders the bundled Javascript and a simple AJAX posting and hooking up of a rating UI. This is optional and overridable by the site developer.

ChangeLog

2.0.0

- converted category on ratings.Rating and *ratings.OverallRating* models to be a CharField that is the actual category label rather than a runtime generated ID. _upgrading will require you manually update the database **values**_

1.0.0

- @@@ write change log

0.3

- renamed from agon_ratings to pinax-ratings

0.2.1

- added ability in overall_rating template tag to omit the category label and get an average rating without concern for category averages.
- added ability to get average rating over all categories for a particular user and particular object.

0.2

- added support for ratings to have categories instead of just a single rating for an object
- dropped natural language of template tags

Migrations

Added a category model and updated the unique index on both models:

```
ALTER TABLE "agon_ratings_overallrating" ADD COLUMN "category" int;
ALTER TABLE "agon_ratings_rating" ADD COLUMN "category" int;
CREATE UNIQUE INDEX "agon_ratings_overallrating_unq_object_id_content_type_id_
↪category_idx"
    ON "agon_ratings_overallrating" (object_id, content_type_id, category);
CREATE UNIQUE INDEX "agon_ratings_rating_unq_object_id_content_type_id_user_id_
↪category_idx"
    ON "agon_ratings_rating" (object_id, content_type_id, user_id, category);
ALTER TABLE "agon_ratings_rating" DROP CONSTRAINT
    IF EXISTS "agon_ratings_rating_object_id_content_type_id_user_id_key";
ALTER TABLE "agon_ratings_overallrating" DROP CONSTRAINT
    IF EXISTS "agon_ratings_overallrating_object_id_content_type_id_key";
```

0.1.2

- added a tag, *user_rating_url*, for getting the POST url for posting a rating
- changed *user_rate_form* and documented javascript wiring to a single *user_rating_js* inclusion tag that output all the javascript and removed the need for a form.

0.1

- initial release