# Ordered lists in django

Asked 15 years, 11 months ago  Modified 15 years, 11 months ago  Viewed 5k times

13

i have very simple problem. I need to create model, that represent element of ordered list. This model can be implemented like this:

```
class Item(models.Model):
    data = models.TextField()
    order = models.IntegerField()
```

or like this:

```
class Item(models.Model):
    data = models.TextField()
    next = models.ForeignKey('self')
```

What way is preferred? What drawbacks have each solution?

python  django  django-models

Share  Improve this question  Follow

asked Jan 9, 2009 at 14:06

Evgeny Lazin

9,403 ● 6 ● 51 ● 86

Hi! I am having the same trouble with the same problem. One solution I thought of is using the model in your `solution 1`. But instead of moving everything, I only changed the value of the `reordered item`. For example I want to move an Item in between the items with order 9 and 10, I will just change the order of the item to 9.5. Do you think this can be bad for long tables? – Miguel Ike Sep 1, 2015 at 7:18

## 3 Answers

Sorted by: Highest score (default)

21

Essentially, the second solution you propose is a linked list. Linked list implemented at the database level are usually not a good idea. To retrieve a list of `n` elements, you will need `n` database access (or use complicated queries). Performance wise, retrieving a list in O(n) is awfully not efficient.

In regular code, linked list are used to get better insert performance compared to arrays (no need to move all elements around). In your database, updating all

elements is not that complicated in only 2 queries :

```
UPDATE item.order = item.order + 1 FROM item WHERE order > 3
INSERT INTO item (order, ...) VALUES (3, ...)
```

I remember seeing a reuseable app that implemented all that and a nice admin interface, but I cant find it right now ...

To summarize, definitly use solution #1 and stay away from solution #2 unless you have a very very good reason not to !

Share
Improve this answer
Follow

edited Jan 9, 2009 at 15:26

answered Jan 9, 2009 at 14:17

Guillaume
18.8k ● 8 ● 55 ● 76

> Two more libraries which help to manage the "order" field on a Django model are pypi.python.org/pypi/django-positions and pypi.python.org/pypi/django-ordered-model. Django positions looks better at a glance. – Nathan Jan 22, 2014 at 1:40

---

6

That depends on what you want to do.

The first one seems better to make a single query in the database and get all data in the correct order

The second one seems better to insert an element between two existing elements (because in the first one you'd have to change a lot of items if the numbers are sequential)

I'd use the first one, because it seems to fit better a database table, which is how django stores model data behind the hood.

Share
Improve this answer
Follow

edited Jan 9, 2009 at 14:20

answered Jan 9, 2009 at 14:12

nosklo
222k ● 58 ● 296 ● 297

---

-6

There is another solution.

```
class Item(models.Model):
    data = models.TextField()
```

You can just pickle or marshal Python list into the data field and the load it up. This one is good for updating and reading, but not for searching e.g. fetching all lists that

contain a specific item.

Share Improve this answer Follow

answered Jan 9, 2009 at 15:22

Seb

**17.7k** ● 7 ● 39 ● 27