spyglass Documentation

Release 0.1

Alex Mavrogiannis, Greg Lyras

January 24, 2016

Contents

Spyglass provides real-time content notifications to a Django application by distributed crawling.

Contents:

CHAPTER 1

Getting Started

Spyglass is a ready to use app that you can use to monitor new content across several predefined websites.

After configuring spyglass, you may run several instances of crawlie that will connect to your application's web server through a RESTful API provided by django-tastypie.

This tutorial assumes you have a basic understanding of Django.

1.1 News Spyglass

Throughout this tutorial we will be referencing an example configuration called News Spyglass.

It's a news monitoring service that receives queries from users containing:

- Search Terms String
- A Site name from a selection box
- A Persistent choice box that when selected the query won't be completed after one notification.

Additionally, News Spyglass has one model, which will be the one populated by spyglass:

```
class NewsStory(models.Model):
    headline = models.CharField(max_length=200, blank=False)
    category = models.CharField(max_length=200, blank=False)
    subtitle = models.CharField(max_length=200, blank=False)

    def __unicode__(self):
        return self.headline

    class Meta:
        verbose_name_plural = "newsstories"
```

The source code for News Spyglass can be found at the example folder of the github repo.

1.2 Installation

first off, install the requirements. They can also be found in the pip-friendly spyglass/requirements.txt:

- Python 2.7+
- Django 1.4.3

- django-tastypie 0.9.11+

You can install them all with `pip install -r requirements.txt`

1.3 Configuration

1. Add `spyglass` to `INSTALLED_APPS` in your `settings.py`.
2. Add `METAMODEL='<path-to-your-model>'` in `settings.py` for the metamodel. In our example, it's `METAMODEL=core.models.NewsStory`

   The metamodel is a model of your application that will be populated when new results are found by the crawlers. All of its fields will later have to correspond to XPaths on each given Site.
3. **In your project's urls.py, add `import spyglass.urls` and** `url(r'^', include(spyglass.urls))` to add the spyglass urls to the root level.

   You can include spyglass' urls at any level but you'll have to edit the crawler's serverconf appropriately.
4. Sync the database with `./manage.py syncdb`
5. Add the sites to be crawled to the spyglass.Site model usign django's admin panel, direct database access or fixtures. For example:

```
[
{
    "model": "spyglass.Site",
    "pk": 1,
    "fields": {
        "name":"Example",
        "url":"http://example.com",
        "poll_time": 5
    }
},
{
    "model": "spyglass.Site",
    "pk": 2,
    "fields": {
        "name":"Example2",
        "url":"http://example2.com",
        "poll_time": 0
    }
},
]

The ``poll_time`` field indicates a time interval (in minutes) between each
visit by any crawler. Sites with poll_time set to 0 will be crawled in a
round-robin fashion.
```

6. For each Site, add a DataField entry for each field of the metamodel, and a corresponding XPath. For example:

```
[
{
    "model": "spyglass.DataField",
    "pk": 1,
    "fields": {
        "site": 1,
        "field_name": "Headline",
        "xpath": "/html/body/div[4]/div/div/div[2]/div/div/h2/a"
```

```
        }
    },
    {
        "model": "spyglass.DataField",
        "pk": 2,
        "fields": {
            "site": 1,
            "field_name": "Category",
            "xpath": "/html/body/div[4]/div/div/div[2]/div/div/div/a"
        }
    },
    {
        "model": "spyglass.DataField",
        "pk": 3,
        "fields": {
            "site": 1,
            "field_name": "Subtitle",
            "xpath": "/html/body/div[4]/div/div/div[2]/div/div/div/p"
        }
    }
    ]
```

Currently, there is no way to exclude fields of the metamodel from this process.

CHAPTER 2

Features

CHAPTER 3

Examples

CHAPTER 4

Indices

- genindex
- search