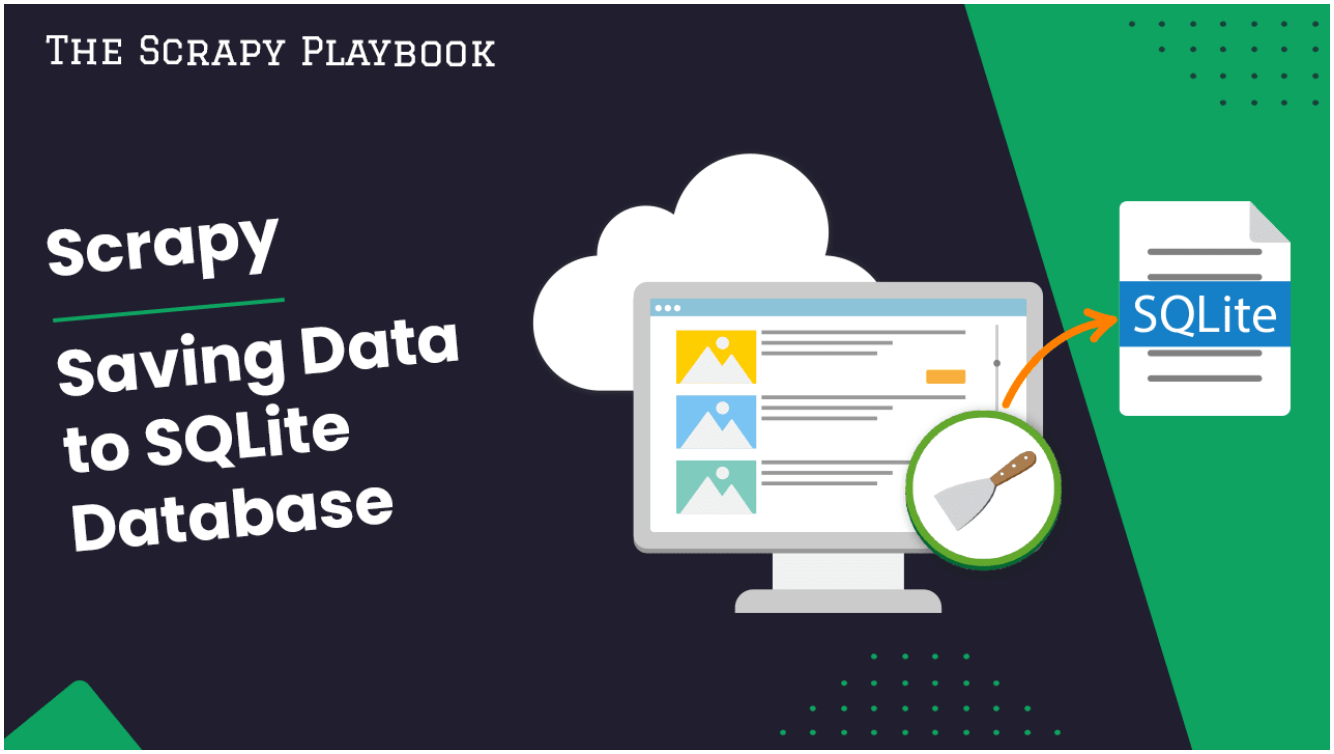

Saving Scraped Data To SQLite Database With Scrapy Pipelines

One of the most common tasks every scraping project needs to do is to save the data we scrape. There are numerous options we can choose from when it comes to saving data, however, using SQLite is one of the best when you have a small project.

In this guide, we will go through how to save our data to a SQLite3 database using Scrapy pipelines:

First, let's go over what are **Scrapy Item Pipelines**.

Need help scraping the web?

Then check out **ScrapeOps**, the complete toolkit for web scraping.

Proxy Manager

Scraper Monitoring

Job Scheduling

What Are Scrapy Item Pipelines?

Item Pipelines are Scrapy's way of process data scraped by spiders.

After an item has been scraped by a spider, it is sent to the Item Pipeline which processes it through a sequence of steps that can be configured to clean and process the scraped data before ultimately saving it somewhere.

You can use Item Pipelines to:

- Clean HTML data
- Validate scraped data
- Checking for and removing duplicate data
- Storing the data in database

For the purpose of this guide, we're going to focus on using Item Pipelines to store data in a SQLite database.

Saving Data To A SQLite Database

The first step is that we need to open our `pipelines.py` file and set up our pipeline.

When you open your `pipelines.py` file, the default file should look like this:

```
# pipelines.py

from itemadapter import ItemAdapter

class SqliteDemoPipeline:
    def process_item(self, item, spider):
        return item
```

Now we will configure this empty pipeline to store our data.

Note: For this guide I created a Scrapy project called **sqlite_demo** (thus the default pipeline is `SqliteDemoPipeline`), and am use this spider:

```
# spiders/quotes.py

import scrapy
from sqlite_demo.items import QuoteItem


class QuotesSpider(scrapy.Spider):
        name = 'quotes'

        def start_requests(self):
                url = 'https://quotes.toscrape.com/'
                yield scrapy.Request(url, callback=self.parse)

        def parse(self, response):
                quote_item = QuoteItem()
                for quote in response.css('div.quote'):
                        quote_item['text'] = quote.css('span.text::text').get()
                        quote_item['author'] = quote.css('small.author::text').get()
                        quote_item['tags'] = quote.css('div.tags a.tag::text').getall()
                        yield quote_item
```

And the Item:

```
# items.py

from scrapy.item import Item, Field

class QuoteItem(Item):
    text = Field()
```

```
    tags = Field()
    author = Field()
```

---

1. Create SQLite Database & Table

SQLite comes with Python, so you don't need to install anything to use it in your Scrapy projects.

First, we're going to `import sqlite3` into our `pipelines.py` file, and create an `__init__` method that we will use to create our database and table.

```
# pipelines.py

import sqlite3

class SqliteDemoPipeline:

    def __init__(self):
        pass

    def process_item(self, item, spider):
        return item
```

Inside the `__init__` method, we will configure the pipeline to do the following everytime the pipeline gets activated by a spider:

1. Try to connect to our database `demo.db`, but if it doesn't exist create the database.
2. Create a cursor which we will use to execute SQL commands in the database.
3. Create a new table `quotes` with the columns `text`, `tags` and `author`, if one doesn't already exist in the database.

```
# pipelines.py

import sqlite3

class SqliteDemoPipeline:

    def __init__(self):
```

```
        ## Create/Connect to database
        self.con = sqlite3.connect('demo.db')

        ## Create cursor, used to execute commands
        self.cur = self.con.cursor()

        ## Create quotes table if none exists
        self.cur.execute("""
        CREATE TABLE IF NOT EXISTS quotes(
            text TEXT,
            tags TEXT,
            author TEXT
        )
        """)

    def process_item(self, item, spider):
        return item
```

---

2. Save Scraped Items Into Database

Next, we're going to use the `process_item` event inside in our Scrapy pipeline to store the data we scrape into our SQLite database.

The `process_item` will be activated everytime, a item is scraped by our spider so we need to configure the `process_item` method to insert the items data in the database.

```
# pipelines.py

import sqlite3

class SqliteDemoPipeline:

    def __init__(self):

        ## Create/Connect to database
        self.con = sqlite3.connect('demo.db')

        ## Create cursor, used to execute commands
        self.cur = self.con.cursor()
```

```
        ## Create quotes table if none exists
        self.cur.execute("""
        CREATE TABLE IF NOT EXISTS quotes(
            text TEXT,
            tags TEXT,
            author TEXT
        )
        """)


    def process_item(self, item, spider):

        ## Define insert statement
        self.cur.execute("""
            INSERT INTO quotes (text, tags, author) VALUES (?, ?, ?)
        """,
        (
            item['text'],
            str(item['tags']),
            item['author']
        ))

        ## Execute insert of data into database
        self.con.commit()
        return item
```

Here we first define our SQL insert statement and give it the data (note, I'm stringifying the tags value as it is an array).

Then we use the `self.con.commit()` command to insert the data.

---

3. Activate Our Item Pipeline

Finally, to activate our Item Pipeline we need to include it in our `settings.py` file:

```
# settings.py

ITEM_PIPELINES = {
    'sqlite_demo.pipelines.SqliteDemoPipeline': 300,
}
```

Now, when we run our **quotes** spider the **SqliteDemoPipeline** will store all the scraped items in the database.

Properties | Data | ER Diagram — demo.db — Tables — quotes

quotes — Enter a SQL expression to filter results (use Ctrl+Space)

	text	tags	author
1	"The world as we have created it is a process of our thinking. It cannot be changed without changing our th	['change', 'deep-thoughts', 'thinking', 'world']	Albert Einstein
2	"It is our choices, Harry, that show what we truly are, far more than our abilities."	['abilities', 'choices']	J.K. Rowling
3	"There are only two ways to live your life. One is as though nothing is a miracle. The other is as though even	['inspirational', 'life', 'live', 'miracle', 'miracles']	Albert Einstein
4	"The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid."	['aliteracy', 'books', 'classic', 'humor']	Jane Austen
5	"Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring	['be-yourself', 'inspirational']	Marilyn Monroe
6	"Try not to become a man of success. Rather become a man of value."	['adulthood', 'success', 'value']	Albert Einstein
7	"It is better to be hated for what you are than to be loved for what you are not."	['life', 'love']	André Gide
8	"I have not failed. I've just found 10,000 ways that won't work."	['edison', 'failure', 'inspirational', 'paraphrased']	Thomas A. Edison
9	"A woman is like a tea bag; you never know how strong it is until it's in hot water."	['misattributed-eleanor-roosevelt']	Eleanor Roosevelt
10	"A day without sunshine is like, you know, night."	['humor', 'obvious', 'simile']	Steve Martin
11	"The world as we have created it is a process of our thinking. It cannot be changed without changing our thi	['change', 'deep-thoughts', 'thinking', 'world']	Albert Einstein
12	"It is our choices, Harry, that show what we truly are, far more than our abilities."	['abilities', 'choices']	J.K. Rowling
13	"There are only two ways to live your life. One is as though nothing is a miracle. The other is as though even	['inspirational', 'life', 'live', 'miracle', 'miracles']	Albert Einstein
14	"The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid."	['aliteracy', 'books', 'classic', 'humor']	Jane Austen
15	"Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring	['be-yourself', 'inspirational']	Marilyn Monroe
16	"Try not to become a man of success. Rather become a man of value."	['adulthood', 'success', 'value']	Albert Einstein
17	"It is better to be hated for what you are than to be loved for what you are not."	['life', 'love']	André Gide
18	"I have not failed. I've just found 10,000 ways that won't work."	['edison', 'failure', 'inspirational', 'paraphrased']	Thomas A. Edison
19	"A woman is like a tea bag; you never know how strong it is until it's in hot water."	['misattributed-eleanor-roosevelt']	Eleanor Roosevelt
20	"A day without sunshine is like, you know, night."	['humor', 'obvious', 'simile']	Steve Martin

If you don't have a SQL database viewer you can use DB Browser for SQLite.

Only Saving New Data

Okay, now we have a Item Pipeline that saves all scraped items to our SQLite database. However, what if we only want to save new data that we haven't scraped before.

We can easily reconfigure our pipeline to do this by having it check the database if the item is already in the database before inserting it again.

To do this, I'm going to create a new pipeline in our `pipelines.py` file called **SqliteNoDuplicatesPipeline** and change the `process_item` method so that it only inserts new data into the database.

It will look up the `item['text']` in the database first, and only if it isn't there will insert the new item.

```
# pipelines.py

import sqlite3

class SqliteNoDuplicatesPipeline:

    def __init__(self):

        ## Create/Connect to database
        self.con = sqlite3.connect('demo.db')

        ## Create cursor, used to execute commands
        self.cur = self.con.cursor()

        ## Create quotes table if none exists
        self.cur.execute("""
        CREATE TABLE IF NOT EXISTS quotes(
            text TEXT,
            tags TEXT,
            author TEXT
        )
        """)


    def process_item(self, item, spider):

        ## Check to see if text is already in database
        self.cur.execute("select * from quotes where text = ?", (item['text'],))
        result = self.cur.fetchone()

        ## If it is in DB, create log message
        if result:
            spider.logger.warn("Item already in database: %s" % item['text'])

        ## If text isn't in the DB, insert data
        else:

            ## Define insert statement
            self.cur.execute("""
                INSERT INTO quotes (text, tags, author) VALUES (?, ?, ?)
            """,
            (
                item['text'],
```

```
                str(item['tags']),
                item['author']
            ))

            ## Execute insert of data into database
            self.con.commit()

        return item
```

To activate, this pipeline we also need to update our `settings.py` to use the `SqliteNoDuplicatesPipeline` not the previous `SqliteDemoPipeline` pipeline:

```
# settings.py

ITEM_PIPELINES = {
    # 'sqlite_demo.pipelines.SqliteDemoPipeline': 300,
   'sqlite_demo.pipelines.SqliteNoDuplicatesPipeline': 300,

}
```

Now, when we run our **quotes** spider, the pipeline will only store new data that isn't already in the database.

More Scrapy Tutorials

We've covered pretty much everything you need to know about saving data to a SQLite database with Scrapy Pipelines.

If you would like to learn more about saving data, then be sure to check out these guides:

- Saving Data to CSV
- Saving Data to JSON
- Saving Data to MySQL Database
- Saving Data to Postgres Database
- Saving CSV/JSON Files to Amazon AWS S3 Bucket

If you would like to learn more about Scrapy in general, then be sure to check out The Scrapy Playbook.