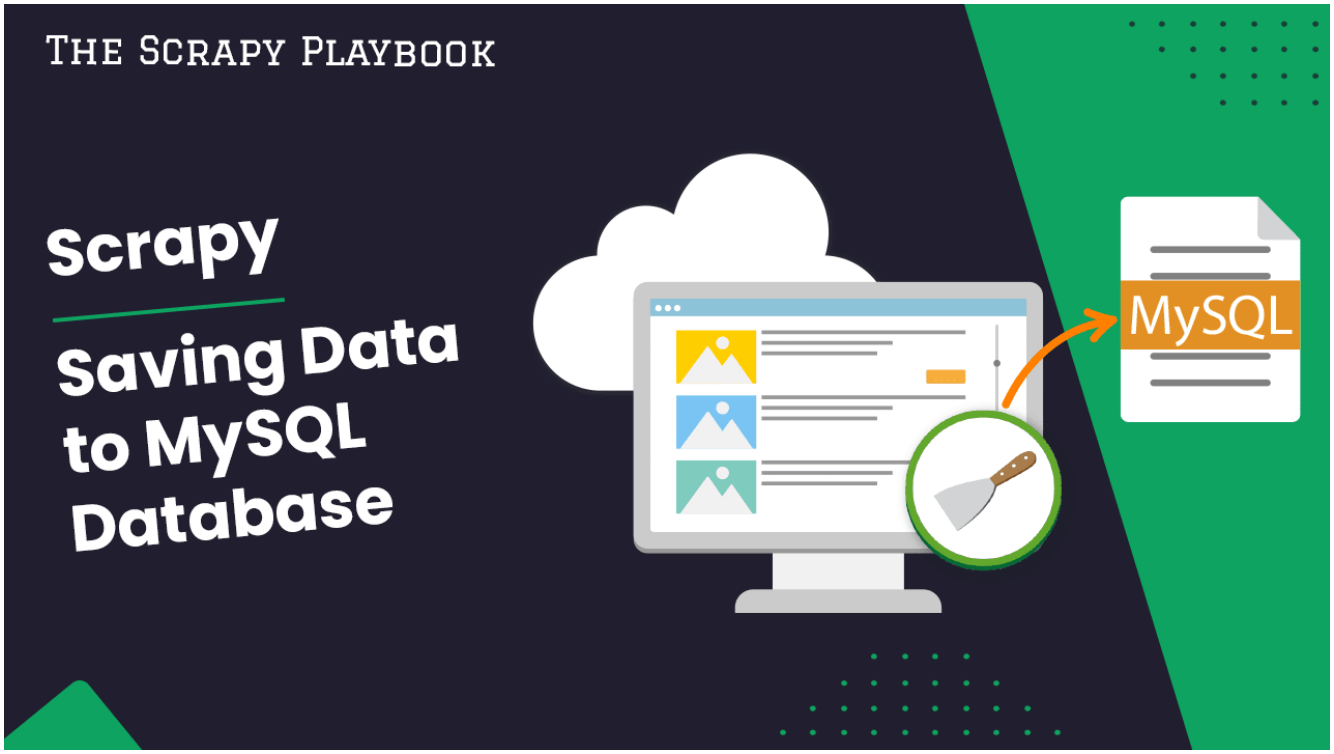

Saving Scraped Data To MySQL Database With Scrapy Pipelines

If your scraping a website, you need to save that data somewhere. A great option is MySQL, one of the most popular and easy to use SQL databases out there.

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

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

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 MySQL database.

Setting Up A MySQL Database

To get started we first need to setup a MySQL database.

Either you can set one up on your local machine by using one of the appropriate installer for your operating system.

Or you could get a hosted version with cloud provider like DigitalOcean.

Once setup you should have access to the database connection details of your database:

```
host="localhost",
database="my_database",
user="root",
password="123456"
```

Saving Data to a MySQL Database

Okay, now let's now integrate saving data into our MySQL database.

1. Install mysql

To interact with our database we will need a library to handle the interaction. For this will install `mysql` and `mysql-connector-python`.

```
pip install mysql mysql-connector-python
```

We will use `mysql` to interact with our MySQL database.

2. Setup Our Pipeline

The next step is 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 MysqlDemoPipeline:
    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 **mysql_demo** (thus the default pipeline is `MysqlDemoPipeline`), and am use this spider:

```
# spiders/quotes.py

import scrapy
from mysql_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()
```

---

3. Connect to MySQL & Create a Table

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

```
# pipelines.py

import mysql.connector

class MysqlDemoPipeline:

    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 `quotes`, 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 `content`, `tags` and `author`, if one doesn't already exist in the database.

```
# pipelines.py

import mysql.connector

class MysqlDemoPipeline:

    def __init__(self):
        self.conn = mysql.connector.connect(
            host = 'localhost',
            user = 'root',
            password = '******',
            database = 'quotes'
        )

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

        ## Create quotes table if none exists
        self.cur.execute("""
        CREATE TABLE IF NOT EXISTS quotes(
            id int NOT NULL auto_increment,
            content text,
            tags text,
```

```
            author VARCHAR(255),
            PRIMARY KEY (id)
        )
        """)

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

---

4. 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 MySQL 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.

We will also the `close_spider` method, which will be called when the Spider is shutting down, to close our connections to the cursor and database to avoid leaving the connection open.

```
# pipelines.py

import mysql.connector

class MysqlDemoPipeline:

    def __init__(self):
        self.conn = mysql.connector.connect(
            host = 'localhost',
            user = 'root',
            password = '**********',
            database = 'quotes'
        )

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

        ## Create quotes table if none exists
        self.cur.execute("""
        CREATE TABLE IF NOT EXISTS quotes(
```

```
            id int NOT NULL auto_increment,
            content text,
            tags text,
            author VARCHAR(255),
            PRIMARY KEY (id)
        )
        """)


    def process_item(self, item, spider):

        ## Define insert statement
        self.cur.execute(""" insert into quotes (content, tags, author) values (%s,%s,%s)""", (
            item["text"],
            str(item["tags"]),
            item["author"]
        ))

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


    def close_spider(self, spider):

        ## Close cursor & connection to database
        self.cur.close()
        self.conn.close()
```

---

5. Activate Our Item Pipeline

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

```
# settings.py

ITEM_PIPELINES = {
   'mysql_demo.pipelines.MysqlDemoPipeline': 300,
}
```

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

Python Scrapy Playbook - Scrapy Pipelines Saving to MySQL Data

If you don't have a SQL database viewer you can use DBeaver.

Only Saving New Data

Okay, now we have a Item Pipeline that saves all scraped items to our MySQL 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 **MySQLNoDuplicatesPipeline** 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 mysql.connector

class MySQLNoDuplicatesPipeline:

    def __init__(self):
        self.conn = mysql.connector.connect(
            host = 'localhost',
            user = 'root',
            password = 'Eily1990',
            database = 'quotes'
        )

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

        ## Create quotes table if none exists
        self.cur.execute("""
        CREATE TABLE IF NOT EXISTS quotes(
            id int NOT NULL auto_increment,
```

```
            content text,
            tags text,
            author VARCHAR(255),
            PRIMARY KEY (id)
        )
        """)



    def process_item(self, item, spider):

        ## Check to see if text is already in database
        self.cur.execute("select * from quotes where content = %s", (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 (content, tags, author) values (%s,%s,%s)""", (
                item["text"],
                str(item["tags"]),
                item["author"]
            ))

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


    def close_spider(self, spider):

        ## Close cursor & connection to database
        self.cur.close()
        self.conn.close()
```

To activate, this pipeline we also need to update our `settings.py` to use the `MySQLNoDuplicatesPipeline` not the

previous `MysqlDemoPipeline` pipeline:

```
# settings.py

ITEM_PIPELINES = {
#    'mysql_demo.pipelines.MysqlDemoPipeline': 300,
    'mysql_demo.pipelines.MySQLNoDuplicatesPipeline': 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 the basics of saving data to a MySQL 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 SQLite 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.