连接数据库的标准 Scrapy 项目

介绍

上一节中，我们只是基于 scrapy 写了一个爬虫脚本，并没有使用 scrapy 项目标准的形式。这一节我们要将脚本变成标准 scrapy 项目的形式，并将爬取到的数据存储到 MySQL 数据库中。数据库的连接和操作使用 SQLAlchemy。

知识点

连接数据库

创建 Scrapy 项目

创建爬虫

Item 容器

Item Pipeline

Models 创建表

保存 Item 到数据库

Item 过滤

连接数据库准备

本实验会将爬取的数据存入 MySQL，需要做一些准备工作。首先需要将 MySQL 的编码格式设置为 utf8，编辑配置文件：

sudo vim /etc/mysql/my.cnf

添加以下几个配置：

[client]

default-character-set = utf8

[mysqld]

character-set-server = utf8

[mysql]

default-character-set = utf8

保存后，就可以启动 mysql 了：

sudo service mysql start

以 root 身份进入 mysql，实验环境默认是没有密码的：

mysql -uroot

创建 shiyanlou 库给本实验使用：

mysql > create database shiyanlou;

完成后输入 quit 退出。

本实验使用 SQLAlchemy 这个 ORM在爬虫程序中连接和操作 mysql，先安装一下（不要忘记激活虚拟环境）：

pip3 install sqlalchemy

还需要安装 Python3 连接 MySQL 的驱动程序 mysqlclient：

sudo apt-get install libmysqlclient-dev

pip3 install mysqlclient

配置数据库和依赖包操作视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

创建项目

使用 scrapy 提供的 startproject 命令创建一个 scrapy 项目

，需要提供一个项目名称，我们要爬取实验楼的数据，所以将 shiyanlou 作为项目名：

scrapy startproject shiyanlou

进入 shiyanlou，可以看到项目结构是这样的：

shiyanlou/

scrapy.cfg # 部署配置文件

shiyanlou/ # 项目名称

\_\_init\_\_.py

items.py # 项目 items 定义在这里

pipelines.py # 项目 pipelines 定义在这里

settings.py # 项目配置文件

spiders/ # 所有爬虫写在这个目录下面

\_\_init\_\_.py

创建 Scrapy 实验楼项目操作视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

创建爬虫

scrapy 的 genspider 命令可以快速初始化一个爬虫模版，使用方法如下：

scrapy genspider <name> <domain>

name 这个爬虫的名称，domain 指定要爬取的网站。

进入第二个 shiyanlou 目录，运行下面的命令快速初始化一个爬虫模版：

cd /home/shiyanlou/Code/shiyanlou/shiyanlou

scrapy genspider courses shiyanlou.com

scrapy 会在 spiders 目录下新建一个 courses.py 的文件，并且在文件中为我们初始化了代码结构：

# -\*- coding: utf-8 -\*-

import scrapy

class CoursesSpider(scrapy.Spider):

name = 'courses'

allowed\_domains = ['shiyanlou.com']

start\_urls = ['http://shiyanlou.com/']

def parse(self, response):

pass

这里面有一个新的属性 allowed\_domains 是在前一节中没有介绍到的。它是干嘛的呢？allow\_domains 可以是一个列表或字符串，包含这个爬虫可以爬取的域名。假设我们要爬的页面是 https://www.example.com/1.hml, 那么就把example.com 添加到 allowed\_domains。这个属性是可选的，在我们的项目中并不需要使用它，所以可以删除。

除此之外 start\_urls 的代码和上一节相同：

# -\*- coding: utf-8 -\*-

import scrapy

class CoursesSpider(scrapy.Spider):

name = 'courses'

@property

def start\_urls(self):

url\_tmpl = 'https://www.shiyanlou.com/courses/?category=all&course\_type=all&fee=all&tag=all&page={}'

return (url\_tmpl.format(i) for i in range(1, 23))

创建 Scrapy 爬虫框架操作视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

Item

爬虫的主要目标是从网页中提取结构化的信息，scrapy 爬虫可以将爬取到的数据作为一个 Python dict 返回，但由于 dict 的无序性，所以它不太适合存放结构性数据。scrapy 推荐使用 Item 容器来存放爬取到的数据。

所有的 items 写在 items.py 中，下面为要爬取的课程定义一个 Item：

import scrapy

class CourseItem(scrapy.Item):

"""定义 Item 非常简单，只需要继承 scrapy.Item 类，将每个要爬取

的数据声明为 scrapy.Field()。下面的代码我们每个课程要爬取的 4

个数据。

"""

name = scrapy.Field()

description = scrapy.Field()

type = scrapy.Field()

students = scrapy.Field()

有了 CourseItem，就可以将 parse 方法的返回包装成它：

# -\*- coding: utf-8 -\*-

import scrapy

from shiyanlou.items import CourseItem

class CoursesSpider(scrapy.Spider):

name = 'courses'

@property

def start\_urls(self):

url\_tmpl = 'https://www.shiyanlou.com/courses/?category=all&course\_type=all&fee=all&tag=all&page={}'

return (url\_tmpl.format(i) for i in range(1, 23))

def parse(self, response):

for course in response.css('div.course-body'):

# 将返回结果包装为 CourseItem 其它地方同上一节

item = CourseItem({

'name': course.css('div.course-name::text').extract\_first(),

'description': course.css('div.course-desc::text').extract\_first(),

'type': course.css('div.course-footer span.pull-right::text').extract\_first(default='免费'),

'students': course.xpath('.//span[contains(@class, "pull-left")]/text()[2]').re\_first('[^\d]\*(\d\*)[^\d]\*')

})

yield item

定义 Item 容器视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

Item Pipeline

如果把 scrapy 想象成一个产品线，spider 负责从网页上爬取数据，Item 相当于一个包装盒，对爬取的数据进行标准化包装，然后把他们扔到Pipeline 流水线中。

主要在 Pipeline 对 Item 进行这几项处理：

验证爬取到的数据 (检查 item 是否有特定的 field)

检查数据是否重复

存储到数据库

当创建项目时，scrapy 已经在 pipelines.py 中为项目生成了一个 pipline模版：

class ShiyanlouPipeline(object):

def process\_item(self, item, spider):

""" parse 出来的 item 会被传入这里，这里编写的处理代码会

作用到每一个 item 上面。这个方法必须要返回一个 item 对象。

"""

return item

除了 process\_item 还有俩个常用的 hooks 方法，open\_spider 和 ｀close\_spider`：

class ShiyanlouPipeline(object):

def process\_item(self, item, spider):

return item

def open\_spider(self, spider):

""" 当爬虫被开启的时候调用

"""

pass

def close\_spider(self, spider):

""" 当爬虫被关闭的时候调用

"""

pass

定义 Item Pipeline 操作视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

定义 Model，创建表

在 items.py 所在目录下创建 models.py，在里面使用 sqlalchemy 语法定义 courses 表结构：

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy import Column, String, Integer

engine = create\_engine('mysql+mysqldb://root@localhost:3306/shiyanlou?charset=utf8')

Base = declarative\_base()

class Course(Base):

\_\_tablename\_\_ = 'courses'

id = Column(Integer, primary\_key=True)

name = Column(String(64), index=True)

description = Column(String(1024))

type = Column(String(64), index=True)

students = Column(Integer)

if \_\_name\_\_ == '\_\_main\_\_':

Base.metadata.create\_all(engine)

运行程序：

python3 models.py

如果运行正确的话，程序什么都不会输出，执行完后，进去 MySQL 中检查是否已经创建了表：

mysql > use shiyanlou;

mysql> show tables;

+---------------------+

| Tables\_in\_shiyanlou |

+---------------------+

| courses |

+---------------------+

如果出现类似上面的东西说明表已经创建成功了！

注意，如果遇到 MySQLdb 包没有找到的错误，有以下几种可能，依次排查下就可以了：

mysqlclient 没有安装

mysqlclient 被 sudo pip3 install 安装到了系统路径，但在 virtualenv 里执行的scrapy没有找到这个包

mysqlclient 被 pip3 install 安装到了 virtualenv，但没有激活 virtualenv 执行的scrapy没有找到这个包

mysqlclient 被 pip3 install 安装到了 virtualenv，但没有在 virtualenv 安装 scrapy，执行 scrapy 的时候用的是系统的 scrapy（which scrapy 可以查看执行的路径）

mysqlclient 和 scrapy 被 pip3 install 安装到了 virtualenv，但安装完成后没有 deactivate 再次重新激活 virtualenv，执行 scrapy 的时候用的是系统的 scrapy（which scrapy 可以查看执行的路径）

创建数据库表操作视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

保存 item 到数据库

创建好数据表后，就可以在 pipeline 写编写代码将 爬取到的每个 item 存入数据库中。

from sqlalchemy.orm import sessionmaker

from shiyanlou.models import Course, engine

class ShiyanlouPipeline(object):

def process\_item(self, item, spider):

# 提取的学习人数是字符串，把它转换成 int

item['students'] = int(item['students'])

# 根据 item 创建 Course Model 对象并添加到 session

# item 可以当成字典来用，所以也可以使用字典解构, 相当于

# Course(

# name=item['name'],

# type=item['type'],

# ...,

# )

self.session.add(Course(\*\*item))

return item

def open\_spider(self, spider):

""" 在爬虫被开启的时候，创建数据库 session

"""

Session = sessionmaker(bind=engine)

self.session = Session()

def close\_spider(self, spider):

""" 爬虫关闭后，提交 session 然后关闭 session

"""

self.session.commit()

self.session.close()

我们编写的这个 ShiyanlouPipeline 默认是关闭的状态，要开启它，需要在 settings.py 将下面的代码取消注释：

# 默认是被注释的

ITEM\_PIPELINES = {

'shiyanlou.pipelines.ShiyanlouPipeline': 300

}

ITEM\_PIPELINES 里面配置需要开启的 pipeline，它是一个字典，key 表示 pipeline 的位置，值是一个数字，表示的是当开启多个 pipeline 时它的执行顺序，值小的先执行，这个值通常设在 100~1000 之间。

保存数据到数据库操作视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

运行

前面使用的 runspider 命令用于启动一个独立的 scrapy 爬虫脚本，在 scrapy 项目中启动爬虫使用 crawl 命令，需要指定爬虫的 name：

scrapy crawl courses

爬虫运行完后，进入 MySQL，输入下面的命令查看爬取数据的前 3 个：

mysql> use shiyanlou;

mysql> select name, type, description, students from courses limit 3\G;

因为 scrapy 爬虫是异步执行的，所以爬取到的 course 顺序和实验楼网站上的会不一样。

运行爬虫操作视频：

您还没有安装flash播放器，请点击这里安装

安装后重启浏览器即可播放视频

Play Video

item 过滤

有时候，并不是每个爬取到的 item 都是我们想要，我们希望对 item 做一下过滤，丢弃不需要的 item。比如只希望保留学习人数超过 1000 的课程，那么就可以对 pipeline 做如下修改：

from scrapy.exceptions import DropItem

class ShiyanlouPipeline(object):

def process\_item(self, item, spider):

item['students'] = int(item['students'])

if item['students'] < 1000:

# 对于不需要的 item，raise DropItem 异常

raise DropItem('Course students less than 1000.')

else:

self.session.add(Course(\*\*item))

总结

本节内容介绍了如何使用 scrapy 命令行工具快速创建项目，创建爬虫，以及如何基于项目框架编写爬虫，运行爬虫。除此之外，也介绍了如何在 scrapy 中使用 MySQL 数据库，将结果存入数据库。

本节涉及到的知识点如下：

连接数据库

创建 Scrapy 项目

创建爬虫

Item 容器

Item Pipeline

Models 创建表

保存 Item 到数据库

Item 过滤

来源: 实验楼

链接: https://www.shiyanlou.com/courses/996

本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播