# 1．前言

## 1.1 课题描述

爬取北京公交车信息（<https://beijing.8684.cn>）：**公交车名称（lineName）**，**运行时间（time）**，**票价信息（price）**，**所属公司（campony）**，**往返线路（upline和downline）**，并将其保存在**mysql数据库（bus\_information数据库，information表）**中。

## 1.2 课题背景及意义

如今，人类社会已经进入了大数据时代，数据已经成为必不可少的部分，可见数据的获取非常重要。而爬虫作为获取数据的一大利器，可以让我们获取足够的数据并用于实际分析。

练习爬虫能力，获取宜昌公交车线路的详细信息。

## 1.3 相关技术介绍

**Scrapy** ：是用 Python 实现的一个为了爬取网站数据、提取结构性数据而编写的应用框架。Scrapy 常应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。通常我们可以很简单的通过 Scrapy 框架实现一个爬虫，抓取指定网站的内容或图片。

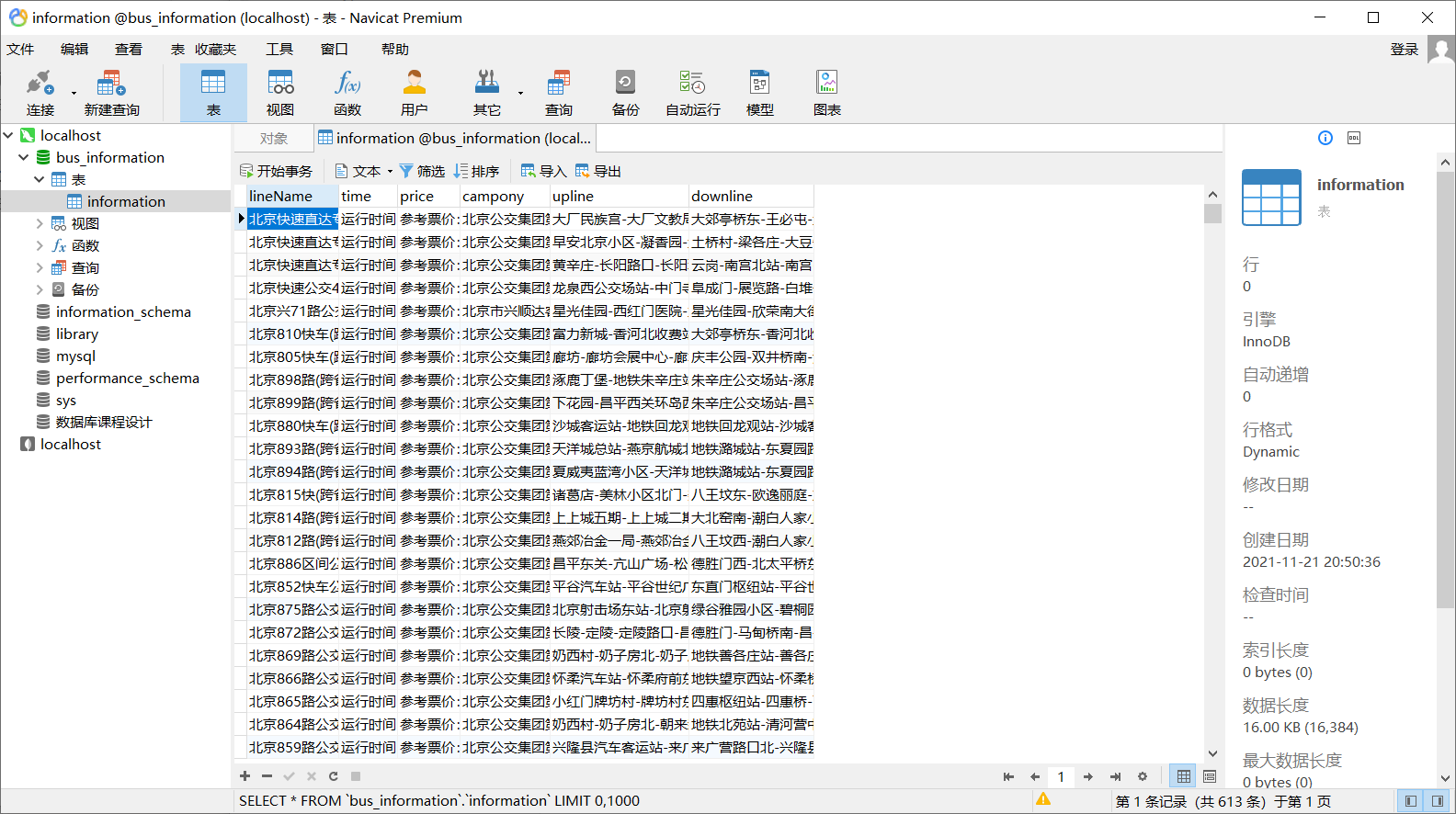
**urllib**：Python 第三方库，用于操作网页 URL，并对网页的内容进行抓取处理。

本次项目主要使用urllib.parse.urljoin来连接url。

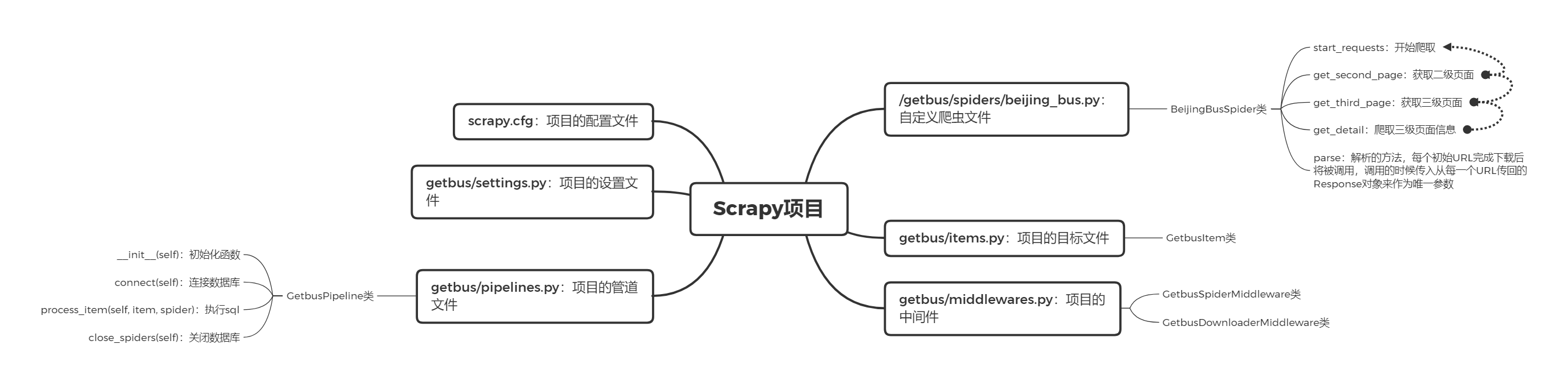
# 2.系统分析

可以做到爬取北京公交车信息（<https://beijing.8684.cn>）：**公交车名称（lineName）**，**运行时间（time）**，**票价信息（price）**，**所属公司（campony）**，**往返线路（upline和downline）**，并将其保存在**mysql数据库（bus\_information数据库，information表）**中。

**结果保存到数据库的图片（部分）**：



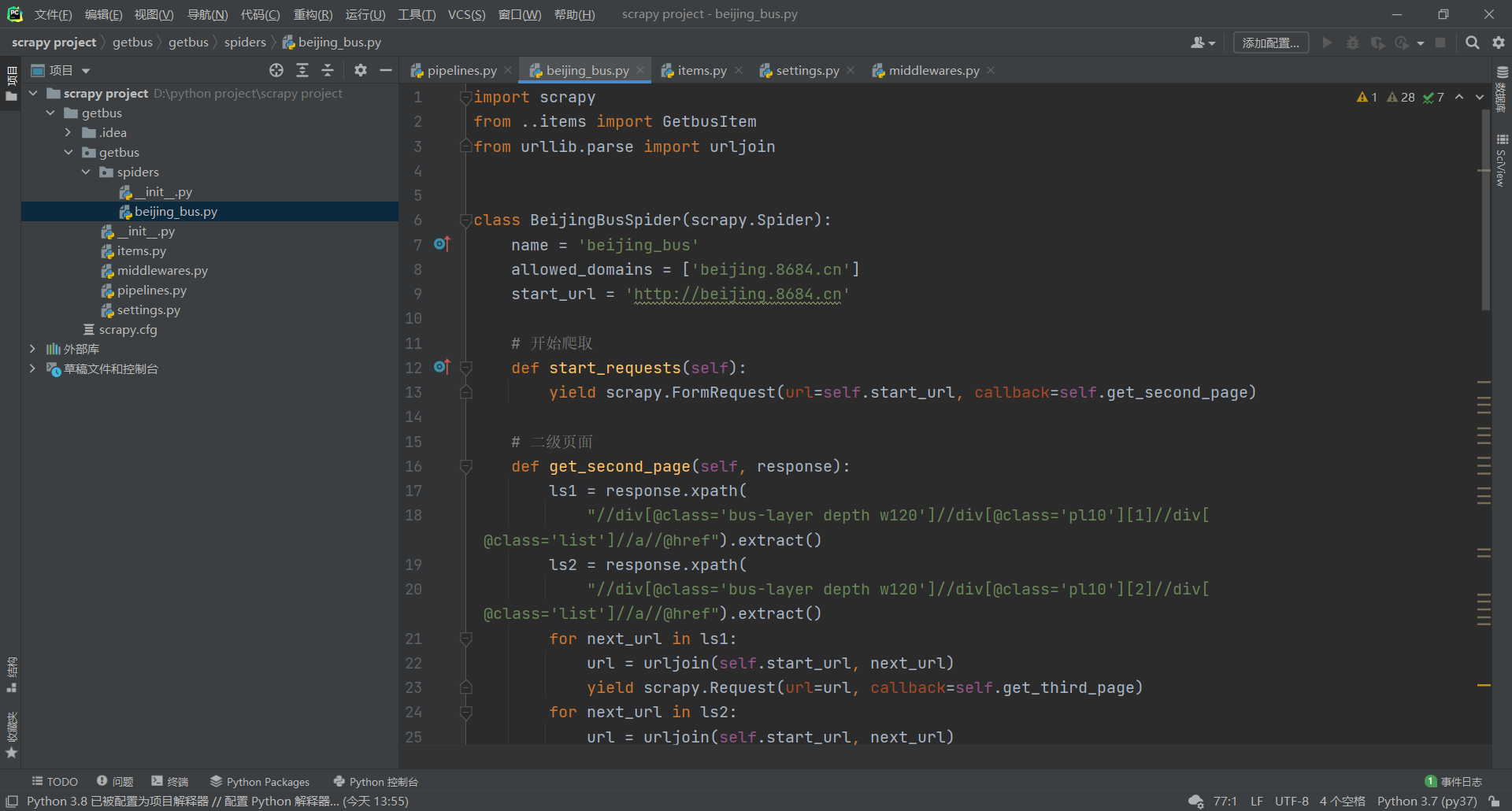
# 3.系统设计

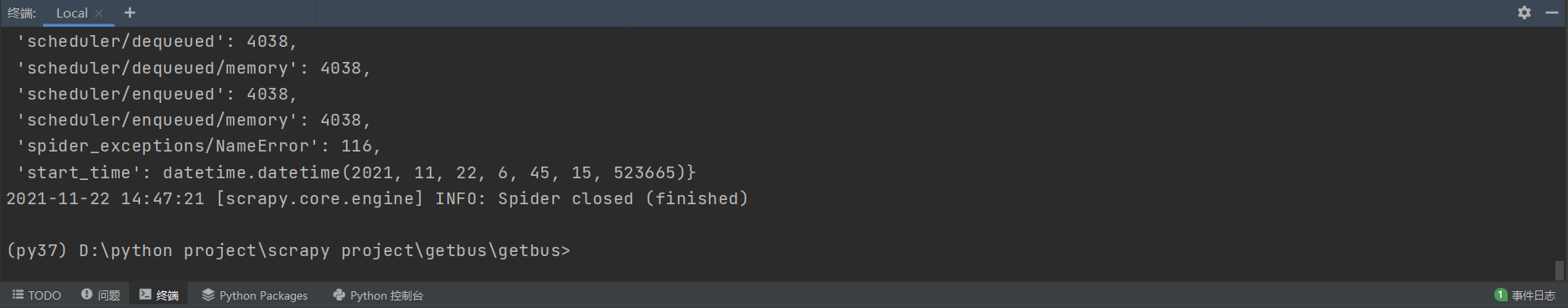


# 4.系统实现

## 4.1 运行界面

（终端进入到项目目录下，输入scrapy crawl beijing\_bus）





## 4.2 代码实现

**beijing\_bus.py（自定义spider文件）**：

import scrapy
  
from ..items import GetbusItem
  
from urllib.parse import urljoin
  
   
   
class BeijingBusSpider(scrapy.Spider):
  
 name = 'beijing\_bus'
  
 allowed\_domains = ['beijing.8684.cn']
  
 start\_url = 'http://beijing.8684.cn'
  
   
 # 开始爬取
  
 def start\_requests(self):
  
 yield scrapy.FormRequest(url=self.start\_url, callback=self.get\_second\_page)
  
   
 # 二级页面
  
 def get\_second\_page(self, response):
  
 ls1 = response.xpath(
  
 "//div[@class='bus-layer depth w120']//div[@class='pl10'][1]//div[@class='list']//a//@href").extract()
  
 ls2 = response.xpath(
  
 "//div[@class='bus-layer depth w120']//div[@class='pl10'][2]//div[@class='list']//a//@href").extract()
  
 for next\_url in ls1:
  
 url = urljoin(self.start\_url, next\_url)
  
 yield scrapy.Request(url=url, callback=self.get\_third\_page)
  
 for next\_url in ls2:
  
 url = urljoin(self.start\_url, next\_url)
  
 yield scrapy.FormRequest(url=url, callback=self.get\_third\_page)
  
   
 # 三级页面
  
 def get\_third\_page(self, response):
  
 ls = response.xpath("//div[@class='list clearfix']/a//@href").extract()
  
 for next\_url in ls:
  
 url = self.start\_url + next\_url
  
 yield scrapy.Request(url=url, callback=self.get\_detail)
  
   
 # 获取三级页面详细信息
  
 def get\_detail(self, response):
  
 try:
  
 lineName = response.xpath("//h1[1]//text()").extract\_first()
  
 except:
  
 lineName = ''
  
 try:
  
 time = response.xpath("//ul[@class='bus-desc']//li[1]//text()").extract\_first()
  
 except:
  
 time = ''
  
 try:
  
 price = response.xpath("//ul[@class='bus-desc']//li[2]//text()").extract\_first()
  
 except:
  
 price = ''
  
 try:
  
 campony = response.xpath("//ul[@class='bus-desc']//li[3]//a//text()").extract\_first()
  
 except:
  
 campony = ''
  
   
 try:
  
 lines = response.xpath("//div[@class='bus-lzlist mb15']")
  
 # 获取上行线路
  
 ls = lines[0].xpath(".//text()").extract()
  
 str = '-'.join(ls)
  
 upline = str
  
   
 # 获取下行线路
  
 if len(lines) > 1:
  
 ls = lines[1].xpath(".//text()").extract()
  
 str = '-'.join(ls)
  
 downline = str
  
 except:
  
 upline = ''
  
 downline = ''
  
   
 # 格式化数据
  
 bus\_item = GetbusItem()
  
 for field in bus\_item.fields:
  
 bus\_item[field] = eval(field)
  
 yield bus\_item
  
   
 def parse(self, response):
  
 pass

**item.py（目标文件）**：

# Define here the models for your scraped items
  
#
  
# See documentation in:
  
# https://docs.scrapy.org/en/latest/topics/items.html
  
   
import scrapy
  
   
   
class GetbusItem(scrapy.Item):
  
 # define the fields for your item here like:
  
 # name = scrapy.Field()
  
 lineName = scrapy.Field()
  
 time = scrapy.Field()
  
 price = scrapy.Field()
  
 campony = scrapy.Field()
  
 upline = scrapy.Field()
  
 downline = scrapy.Field()

**pipeline.py（管道文件）**：

# Define your item pipelines here
  
#
  
# Don't forget to add your pipeline to the ITEM\_PIPELINES setting
  
# See: https://docs.scrapy.org/en/latest/topics/item-pipeline.html
  
   
   
# useful for handling different item types with a single interface
  
from itemadapter import ItemAdapter
  
from . import settings
  
import pymysql
  
   
   
class GetbusPipeline:
  
 def \_\_init\_\_(self):
  
 self.host = settings.DB\_HOST
  
 self.user = settings.DB\_USER
  
 self.pwd = settings.DB\_PWD
  
 self.db = settings.DB
  
 self.charset = settings.DB\_CHARSET
  
 self.connect()
  
   
 def connect(self):
  
 # 连接数据库，创建一个数据库对象
  
 self.conn = pymysql.connect(host=self.host,
  
 user=self.user,
  
 password=self.pwd,
  
 db=self.db,
  
 charset=self.charset
  
 )
  
 # 开启游标功能，创建游标对象
  
 self.cursor = self.conn.cursor()
  
   
 def process\_item(self, item, spider):
  
 sql = 'insert into information(lineName,time,price,campony,upline,downline) values ("%s","%s","%s","%s","%s","%s")' % (
  
 item['lineName'], item['time'], item['price'], item['campony'], item['upline'], item['downline'])
  
 # 执行SQL语句
  
 self.cursor.execute(sql) # 使用execute方法执行SQL语句
  
 self.conn.commit() # 提交到数据库执行
  
 return item
  
   
 # 用于关闭数据库的连接
  
 def close\_spiders(self):
  
 self.conn.close()
  
 self.cursor.close()

**setting.py（设置文件）**：

# Scrapy settings for getbus project
  
#
  
# For simplicity, this file contains only settings considered important or
  
# commonly used. You can find more settings consulting the documentation:
  
#
  
# https://docs.scrapy.org/en/latest/topics/settings.html
  
# https://docs.scrapy.org/en/latest/topics/downloader-middleware.html
  
# https://docs.scrapy.org/en/latest/topics/spider-middleware.html
  
   
BOT\_NAME = 'getbus'
  
   
SPIDER\_MODULES = ['getbus.spiders']
  
NEWSPIDER\_MODULE = 'getbus.spiders'
  
   
ROBOTSTXT\_OBEY = False
  
   
DEFAULT\_REQUEST\_HEADERS = {
  
 'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8',
  
 'Accept-Language': 'en',
  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) '
  
 'AppleWebKit/537.36 (KHTML, like Gecko) '
  
 'Chrome/95.0.4638.69 Safari/537.36',
  
}
  
   
DB\_HOST = 'localhost'
  
DB\_USER = 'root'
  
DB\_PWD = '123456'
  
DB = 'bus\_information' # 数据库　
  
DB\_CHARSET = 'utf8'
  
   
ITEM\_PIPELINES = {
  
 'getbus.pipelines.GetbusPipeline': 300,
  
}

**middlewares.py（中间件文件）**：

# Define here the models for your spider middleware
  
#
  
# See documentation in:
  
# https://docs.scrapy.org/en/latest/topics/spider-middleware.html
  
   
from scrapy import signals
  
   
# useful for handling different item types with a single interface
  
from itemadapter import is\_item, ItemAdapter
  
   
   
class GetbusSpiderMiddleware:
  
 # Not all methods need to be defined. If a method is not defined,
  
 # scrapy acts as if the spider middleware does not modify the
  
 # passed objects.
  
   
 @classmethod
  
 def from\_crawler(cls, crawler):
  
 # This method is used by Scrapy to create your spiders.
  
 s = cls()
  
 crawler.signals.connect(s.spider\_opened, signal=signals.spider\_opened)
  
 return s
  
   
 def process\_spider\_input(self, response, spider):
  
 # Called for each response that goes through the spider
  
 # middleware and into the spider.
  
   
 # Should return None or raise an exception.
  
 return None
  
   
 def process\_spider\_output(self, response, result, spider):
  
 # Called with the results returned from the Spider, after
  
 # it has processed the response.
  
   
 # Must return an iterable of Request, or item objects.
  
 for i in result:
  
 yield i
  
   
 def process\_spider\_exception(self, response, exception, spider):
  
 # Called when a spider or process\_spider\_input() method
  
 # (from other spider middleware) raises an exception.
  
   
 # Should return either None or an iterable of Request or item objects.
  
 pass
  
   
 def process\_start\_requests(self, start\_requests, spider):
  
 # Called with the start requests of the spider, and works
  
 # similarly to the process\_spider\_output() method, except
  
 # that it doesn’t have a response associated.
  
   
 # Must return only requests (not items).
  
 for r in start\_requests:
  
 yield r
  
   
 def spider\_opened(self, spider):
  
 spider.logger.info('Spider opened: %s' % spider.name)
  
   
   
class GetbusDownloaderMiddleware:
  
 # Not all methods need to be defined. If a method is not defined,
  
 # scrapy acts as if the downloader middleware does not modify the
  
 # passed objects.
  
   
 @classmethod
  
 def from\_crawler(cls, crawler):
  
 # This method is used by Scrapy to create your spiders.
  
 s = cls()
  
 crawler.signals.connect(s.spider\_opened, signal=signals.spider\_opened)
  
 return s
  
   
 def process\_request(self, request, spider):
  
 # Called for each request that goes through the downloader
  
 # middleware.
  
   
 # Must either:
  
 # - return None: continue processing this request
  
 # - or return a Response object
  
 # - or return a Request object
  
 # - or raise IgnoreRequest: process\_exception() methods of
  
 # installed downloader middleware will be called
  
 return None
  
   
 def process\_response(self, request, response, spider):
  
 # Called with the response returned from the downloader.
  
   
 # Must either;
  
 # - return a Response object
  
 # - return a Request object
  
 # - or raise IgnoreRequest
  
 return response
  
   
 def process\_exception(self, request, exception, spider):
  
 # Called when a download handler or a process\_request()
  
 # (from other downloader middleware) raises an exception.
  
   
 # Must either:
  
 # - return None: continue processing this exception
  
 # - return a Response object: stops process\_exception() chain
  
 # - return a Request object: stops process\_exception() chain
  
 pass
  
   
 def spider\_opened(self, spider):
  
 spider.logger.info('Spider opened: %s' % spider.name)

# 5.结束语

对爬虫有了新的了解，学会了一些基本爬虫操作