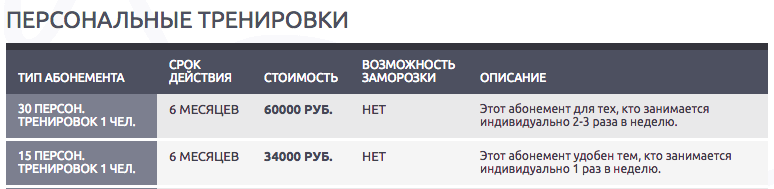
Scrapy: пример работы

# Задача

С сайта [Европейского гимнастического центра](https://www.europegym.ru/) **для всех центров** (Лужники, Локомотив, Киевское Шоссе, Сетунь, Куркино, Динамо, Чертаново, Марьино, Медведково, Строгино)

собрать всю доступную информацию **о персональных тренировках**, записанную в таких таблицах:



Результат сохранить в **json**.

# Решение

**0. Устанавливаем scrapy**

pip3 install scrapy

1. **Создаём проект**
   1. cd <где будем создавать проект>
   2. scrapy startproject europegym
   3. cd europegym
   4. scrapy genspider pricelist europegym.ru
   5. Файл спайдера — europegym/spiders/pricelist.py
2. **Привязываемся к странице со списком**
   1. Находим точку входа: <https://www.europegym.ru/centers/prices/1.html>
   2. Вставляем её в `start\_urls` спайдера
   3. Запускаем через консоль: `scrapy crawl pricelist`.
   4. Замечаем, что скрепи ругается на `robots.txt`. В `settings.py` меняем `ROBOTSTXT\_OBEY = True` на `ROBOTSTXT\_OBEY = False`.
   5. Запускаем и видим, что всё ОК
3. **Находим список ссылок, которые нужно обойти**
   1. Находим глазами, где нужные данные (ссылки)
   2. Смотрим на это место через инспектор (правая кнопка — просмотреть код)
   3. Придумываем селектор и вызываем его в консоли через `$$(‘<selector>’)`, пока не получится что надо (спойлер белым цветом: .side\_info:last-child) a`
4. **Приделываем обход ссылок в спайдера**
   1. В PricelistSpider.parse вместо pass пишем цикл:

for link in response.css('.side\_info:last-child a'):

* 1. В цикле извлекаем из ссылки адрес, на который она ведёт: `link.css('::attr(href)').extract\_first()`
  2. Смотрим, собрались ли нужные адреса:

href = link.css('::attr(href)').extract\_first()

yield { 'href': href }

* 1. Запускаем краулер — ура, ссылки собрались!

1. **Теперь вместо того, чтобы просто выводить адреса страниц, попробуем переходить на них.**
   1. Аккуратно, ссылки относительные (`/centers/prices/21.html`), и скрепи не знает, как по ним ходить — нужно приделать к ним адрес сайта.
   2. Импортируем `import urllib.parse` в начале спайдера — эта библиотека соединяет куски адресов.
   3. Делаем адрес абсолютным:

`abs\_url = urllib.parse.urljoin('https://www.europegym.ru/', href)`

* 1. Пишем в классе затычку-метод `parse\_center`, в котором потом будем разбираться со строками таблицы. Пока пусть он просто делает `yield {}`
  2. Переходим по ссылке!

Похоже на вывод данных, но вызываем со скрепи-запросом:

`yield scrapy.Request(url=abs\_url, callback=self.parse\_center)`

1. **Собираем сами данные.**

*Все* прайс-листы находятся в таблицах внутри `.hold-price`, и перед каждым стоит заголовок `h2`. Мы хотим получить только информацию о персональных тренировках, то есть следующую таблицу после заголовка «Персональные тренировки».

* 1. CSS не умеет фильтровать по тексту, значит, нужен xpath
  2. В parse\_center выбираем все строки нужной таблицы: `rows = response.xpath('//h2[contains(text(), "Персональные тренировки")]/following-sibling::table[1]/tr')`. Вот что это значит:
     + Подходящий заголовок:

`//h2[contains(text(), "Персональные тренировки")]`

* + - *Все* элементы после заголовка: `/following-sibling`
    - Только таблицы: `::table`
    - Только первая таблица: `[1]`
    - Все строки нужной таблицы: `/tr`.
  1. Пробуем достать текст из ячеек и переложить его в массив:

`res = [[td.css('::text').extract\_first().strip() for td in row.xpath('td')] for row in rows]`

`yield { 'table': res }`

* 1. Запускаем посмотреть, лучше уже в файле:
     + scrapy crawl pricelist -o data.json
     + По умолчанию скрепи записывает в файл юникод в виде `\uXXXX`. Если мы хотим настоящий юникод, добавим в `settings.py`

```

FEED\_EXPORT\_INDENT = 2

FEED\_EXPORT\_ENCODING = "utf-8"

```

* 1. Запускаем посмотреть. Внимание! data.json не перезаписывается: новое отправляется в конец. Так что старое удаляем.
  2. Почти успех, но… текст некоторых ячеек обёрнут в `div`, а `::text` возвращает только текст из самого элемента.
  3. Напишем “умную” функцию `cell\_text`, которая решит эту проблему:

def cell\_text(td):

sel = 'div::text' if len(td.css('div')) else '::text'

return td.css(sel).extract\_first().strip()

* 1. И перепишем разбор строки:

`res = [[cell\_text(td) for td in row.xpath('td')] for row in rows]`

* 1. Запускаем посмотреть.
  2. Теперь достанем данные из таблицы вместе с названием клуба:

`yield { 'title': response.css('h1::text').extract\_first(), 'table': res }`

1. Последний раз запускаем и радуемся результату! Чего не хватает?..

(Ответ: заголовков столбцов, вместо них пустое место, ну и ладно.)

Ваша очередь!

# Задача

С сайта <http://rubooks.org> с помощью scrapy сохранить в .json любую многостраничную (5-50 стр.) книгу.