универсальный веб-краулер на Python, используя шаблон проектирования "Стратегия" для обработки HTTP-запросов. Этот шаблон позволяет изменять методы запроса без изменения основной бизнес-логики.

**Шаг 1: Определение интерфейса для стратегии запроса:**

--------------------------------------------------------------------------------------------------  
from abc import ABC, abstractmethod

class HttpRequestStrategy(ABC):

    @abstractmethod  
    def fetch(self, url: str) -> str:  
        pass  
--------------------------------------------------------------------------------------------------

**Шаг 2: Реализация различных стратегий:**

--------------------------------------------------------------------------------------------------

import requests

class DirectRequest(HttpRequestStrategy):

    def fetch(self, url: str) -> str:  
        response = requests.get(url)  
        response.raise\_for\_status()  
        return response.text

class ProxyRequest(HttpRequestStrategy):

    def \_\_init\_\_(self, proxy: str):  
        self.proxy = proxy

    def fetch(self, url: str) -> str:  
        proxies = {  
            "http": self.proxy,  
            "https": self.proxy  
        }  
        response = requests.get(url, proxies=proxies)  
        response.raise\_for\_status()  
        return response.text  
--------------------------------------------------------------------------------------------------

**Шаг 3: Создание краулера:**

--------------------------------------------------------------------------------------------------  
from bs4 import BeautifulSoup

class WebCrawler:

    def \_\_init\_\_(self, strategy: HttpRequestStrategy):  
        self.strategy = strategy

    def fetch\_data(self, url: str) -> str:  
        return self.strategy.fetch(url)

    def fetch\_poems(self, author\_url: str) -> list:  
        poems = []  
        html = self.fetch\_data(author\_url)  
        soup = BeautifulSoup(html, 'html.parser')  
        poem\_blocks = soup.select('.poem')  # Примерный селектор, замените при необходимости  
        for block in poem\_blocks:  
            poem\_text = block.get\_text(separator='\n').strip()  
            poems.append(poem\_text)  
        return poems

    # Аналогичные методы можно добавить для изображений или товаров  
--------------------------------------------------------------------------------------------------

**Пример использования:**

--------------------------------------------------------------------------------------------------  
try:  
    # Использование прямого запроса  
    crawler = WebCrawler(DirectRequest())  
    poems = crawler.fetch\_poems("https://stihi.ru/avtor/authorname")  
    for poem in poems:  
        print(poem)  
        print("-" \* 50)

    # Использование запроса через прокси  
    proxy = "http://your\_proxy\_address:port"  
    crawler\_with\_proxy = WebCrawler(ProxyRequest(proxy))  
    poems\_with\_proxy = crawler\_with\_proxy.fetch\_poems("https://stihi.ru/avtor/authorname")  
    for poem in poems\_with\_proxy:  
        print(poem)  
        print("-" \* 50)  
except Exception as e:  
    print(f"Ошибка: {e}")  
--------------------------------------------------------------------------------------------------

**Вы можете дополнить класс `WebCrawler` другими методами, например, для загрузки изображений или товаров, в зависимости от структуры сайта и ваших потребностей.**