**Техническая документация**

**Функции модуля *parserlib.parsers.gismeteo*:**

* **get\_month\_by\_str(m: *str*) -> int**

Эта функция принимает текстовое сокращение месяца в виде строки **m** и возвращает его в формате YYYY-MM-DD.

* **get\_gismeteio\_diary(city: *str*, start: *tuple*, end: *tuple*, interval: *float* = 1) -> *list***

Эта функция извлекает исторические данные о погоде для указанного города **city** в заданном временном диапазоне. Параметры **start** и **end** представляют кортежи, содержащие год и месяц начала и конца диапазона соответственно. Опциональный параметр **interval** определяет задержку между запросами. Функция возвращает список словарей, где каждый словарь содержит дату и среднюю температуру для заданного диапазона.

* **get\_gismeteo\_month(city\_url: *str*) -> *list***

Эта функция извлекает данные о погоде на месяц для заданного города на сайте Gismeteo. Параметр **city\_url** представляет URL-адрес страницы погоды для города. Функция возвращает список словарей, где каждый словарь содержит дату и среднюю температуру за месяц.

* **main()**

Это функция, которая выполняет основной процесс программы. Она собирает информацию о погоде для разных регионов, заносит данные в базу данных и выполняет необходимые вычисления для каждой записи о погоде. Первая часть функции собирает информацию о погоде для разных регионов и сохраняет её в словаре **weather\_by\_date**, где ключами являются даты, а значениями - температуры для каждого региона. Затем происходит запись данных в базу данных. Сессия базы данных итерирует по словарю **weather\_by\_date**, и для каждой даты обновляются соответствующие записи о погоде в базе данных.

**Функции модуля *parserlib.parsers.*** ***futures*:**

* **get\_date\_by\_coal\_contract\_symbol(contract\_symbol: *str*) -> *int***

Эта функция принимает символ контракта фьючерса угля в виде строки **contract\_symbol** и возвращает дату в формате YYYY-MM-DD. Она анализирует символ контракта и извлекает дату из него, а также преобразует месяц в числовой формат.

* **get\_date\_by\_gas\_month(month: *str*) -> *int***

Эта функция принимает название месяца фьючерса газа в виде строки **month** и возвращает дату в формате YYYY-MM-DD. Она анализирует название месяца и преобразует его в числовой месяц и год.

* **write\_gas\_futures()**

Эта функция использует Selenium для парсинга данных о фьючерсах газа с веб-сайта. Она инициализирует драйвер, загружает страницу, ждет, пока появится таблица с данными, парсит строки таблицы и записывает данные в базу данных.

* **write\_coal\_futures()**

Эта функция использует Selenium для парсинга данных о фьючерсах угля с веб-сайта. Она инициализирует драйвер, загружает страницу, ждет, пока появится сетка с данными, парсит строки сетки и записывает данные в базу данных.

* **main()**

Это функция, которая вызывает функции парсинга для фьючерсов угля и газа.

**Функции модуля *parserlib.parsers.*** ***pdf*:**

* **get\_date\_by\_cci\_filename(filename: *str*) -> *str***

Эта функция принимает имя файла CCI в виде строки **filename** и возвращает дату в формате YYYY-MM-DD, извлеченную из имени файла.

* **get\_date\_by\_freight\_filename(filename: *str*) -> *str***

Эта функция принимает имя файла о фрахтах в виде строки **filename** и возвращает дату в формате YYYY-MM-DD, извлеченную из имени файла.

* **extract\_text\_from\_images\_in\_pdf(pdf\_path: *str*) -> *str***

Функция принимает путь PDF файла и извлекает текст из всех изображений этого PDF-файла. Извлеченный текст объединяется в одну строку и возвращается как результат.

* **write\_stockpiles(reader: *PdfReader*, session: *orm.Session*, date: *str*)**

Эта функция парсит данные о запасах CPR из PDF-файла с использованием объекта *PdfReader*. Она принимает *PdfReader*, сессию для работы с базой данных и дату. Функция извлекает данные о запасах из PDF, создает объект *CPRStockpile*, записывает в него данные и сохраняет его в базе данных.

* **write\_stockpiles\_v2(text: *str*, session: *orm.Session*, date: *str*)**

Эта функция парсит данные о запасах CPR из текста, извлеченного из PDF-файла. Она принимает текст, сессию для работы с базой данных и дату. Функция извлекает данные о запасах из текста, создает объект *CPRStockpile*, записывает в него данные и сохраняет его в базе данных. Этот метод используется для новых файлов CCI.

* **write\_cci\_indicies(reader: *PdfReader*, session: *orm.Session*, date: *str*)**

Эта функция парсит данные о CCI индексах из PDF-файла с использованием объекта *PdfReader*. Она принимает *PdfReader*, сессию для работы с базой данных и дату. Функция извлекает данные о CCI индексах из PDF, создает объект *Index*, записывает в него данные и сохраняет его в базе данных.

* **write\_cci\_indicies\_v2(text: *str*, session: *orm.Session*, date: *str*)**

Эта функция парсит данные о CCI индексах из текста, извлеченного из PDF-файла. Она принимает текст, сессию для работы с базой данных и дату. Функция извлекает данные о CCI индексах из текста, создает объект *Index*, записывает в него данные и сохраняет его в базе данных. Этот метод используется для новых файлов CCI.

* **write\_freight(reader: *PdfReader*, session: *orm.Session*, date: *str*)**

Эта функция парсит данные о фрахтах из PDF-файла с использованием объекта *PdfReader*. Она принимает *PdfReader*, сессию для работы с базой данных и дату. Функция извлекает данные о фрахтах из PDF, создает объект *Freight*, записывает в него данные и сохраняет его в базе данных.

* **main()**

Это главная функция, которая выполняет основной процесс. Она сначала ищет и сортирует файлы CCI и Freight по дате, а затем итерируется по каждому файлу, вызывая соответствующие функции парсинга и записи в базу данных. Когда процесс завершается, файлы архивируются.

**Функции модуля *parserlib.parsers.xls*:**

* **write\_coal\_rail\_exports(excel\_file: pd.ExcelFile, session: orm.Session)**

Эта функция выполняет парсинг данных о железнодорожных перевозках угля и последующую запись их в базу данных. Она обрабатывает информацию о датах, направлениях и объемах перевозок, фильтруя данные по заданному списку продуктов и группируя объемы по регионам. После этого функция обновляет соответствующие записи в базе данных.

* **write\_vostochny\_indicies(excel\_file: *pd.ExcelFile*, session: *orm.Session*)**

Эта функция парсит индексы FOB Vostochny из файла Excel и записывает их в базу данных. Она принимает объект *pd.ExcelFile*, представляющий файл Excel, и сессию для работы с базой данных SQLAlchemy. В зависимости от содержимого листа в Excel-файле, функция анализирует данные и записывает их в базу данных.

1. Если лист называется "Price history", функция считывает данные из строк 4 и далее, где дата и индекс указаны в столбцах. Затем она создает или обновляет запись в базе данных *Index* для каждой даты и записывает значения индексов.
2. Если лист называется "Цены", функция считывает данные из соответствующих столбцов, где дата и индекс указаны в разных столбцах. Затем она создает или обновляет запись в базе данных *Index* для каждой даты и записывает значения индексов.

* **write\_ici3\_indicies(excel\_file: *pd.ExcelFile*, session: *orm.Session*)**

Эта функция парсит индексы ICI3 из файла Excel и записывает их в базу данных.

* + 1. Если лист называется "Price history", функция считывает данные из строк 4 и далее, где дата и индекс указаны в столбцах. Затем она создает или обновляет запись в базе данных *Index* для каждой даты и записывает значения индексов.
    2. Если лист называется "Цены", функция считывает данные из соответствующих столбцов, где дата и индекс указаны в разных столбцах. Затем она создает или обновляет запись в базе данных *Index* для каждой даты и записывает значения индексов.
* **write\_manual\_input(excel\_file: *pd.ExcelFile*, session: *orm.Session*)**

Эта функция парсит данные из "ручного" файла Excel и записывает их в базу данных. Она принимает объект *pd.ExcelFile*, представляющий файл Excel, и сессию для работы с базой данных SQLAlchemy. Функция парсит разные листы в файле Excel, такие как "Index," "CPR Stockpile," "Freight," "Futures," и "China Weather."

1. Для каждого листа, функция извлекает данные из соответствующих столбцов и записывает их в соответствующие сущности базы данных, такие как *Index*, *CPRStockpile*, *Freight*, *Future*, и *ChinaWeather*.
2. Если в файле указано, что запись должна быть обновлена (по значению в столбце "Update?"), функция проверяет существование записи и обновляет ее данные.
3. Если произошла ошибка при парсинге данных, функция записывает сообщение об ошибке в лог.

* **parse\_manual\_input\_files()**

Эта функция выполняет парсинг данных из "ручных" файлов XLSX. Она итерирует по файлам в директории **manual\_dir**, открывает каждый файл, вызывает **write\_manual\_input** для парсинга и записи данных в базу данных, а затем архивирует обработанный файл. Если произошла ошибка при парсинге, она также записывает сообщение об ошибке в лог.

* **parse\_downloaded\_files()**

Эта функция выполняет парсинг данных из скачанных файлов XLS. Она итерирует по файлам в директориях **vostochny\_dir**, **ici3\_dir** и **rail\_coal\_export\_dir**, открывает каждый файл, вызывает соответствующую функцию (**write\_vostochny\_indicies**, **write\_ici3\_indicies**, **write\_coal\_rail\_exports**) для парсинга и записи данных в базу данных, а затем архивирует обработанный файл. Если произошла ошибка при парсинге, она также записывает сообщение об ошибке в лог.

* **main()**

Это главная функция, которая вызывает **parse\_downloaded\_files**() и **parse\_manual\_input\_files**() для выполнения всей работы.

**Функции модуля *parserlib.utils*:**

* **archive\_file(src: str)**

Это функция, которая перемещает файл в архив. Она принимает путь к исходному файлу **src**.

1. Функция определяет, к какой категории (директории) принадлежит файл на основе пути к исходному файлу.
2. Затем функция формирует путь назначения **dst** для архивирования, добавляя имя файла к директории архива, соответствующей категории файла.
3. Если файл уже существует в директории архива, он просто удаляется.
4. Если файл не существует в директории архива, он переносится в эту директорию путем переименования.
5. В итоге, функция выполняет перемещение файла в архив и логирует это действие.

**Функции модуля *parserlib.*** ***downloaders.argus*:**

* **signin()**

Эта функция выполняет вход на веб-сайт Argus Media, используя предоставленные учетные данные. Она создает экземпляр веб-драйвера Chrome и ожидает, пока страница для ввода учетных данных не будет загружена. Затем вводит имя пользователя и пароль, а затем производит вход. Функция возвращает кортеж, состоящий из веб-драйвера и объекта ожидания (**wait**).

* **download\_freight\_file(driver: *webdriver.Chrome*, wait: *WebDriverWait*)**

Эта функция выполняет загрузку последнего файла фрахта с веб-сайта Argus Media. Она переключает веб-драйвер на нужные фреймы и выполняет действия для загрузки файла. Затем она переименовывает и перемещает файл в директорию для парсинга.

* **download\_ici3\_file(driver: *webdriver.Chrome*, wait: *WebDriverWait*)**

Эта функция выполняет загрузку последнего файла ICI3 с веб-сайта Argus Media. Она также переключает веб-драйвер на нужные фреймы и выполняет действия для загрузки файла. Затем она переименовывает и перемещает файл в директорию для парсинга.

* **download\_vostochny\_file(driver: *webdriver.Chrome*, wait: *WebDriverWait*)**

Эта функция выполняет загрузку последнего файла FOB Vostochny с веб-сайта Argus Media. Она аналогична предыдущим функциям, но предназначена для загрузки файла FOB Vostochny.

**download\_coal\_rail\_exports(driver: webdriver.Chrome, wait: WebDriverWait)**

Эта функция загружает последний файл Ж/Д перевозок. Она переходит на страницу с загрузками, находит соответствующий элемент, инициирует скачивание файла в формате Excel (XLSX) и перемещает его в директорию для парсинга.

* **main()**

Это главная функция, которая выполняет все вышеуказанные шаги в последовательности. Она входит в аккаунт Argus Media, загружает файлы фрахта, ICI3 и FOB Vostochny, а затем завершает сеанс веб-драйвера.

**Функции модуля *parserlib.*** ***downloaders.email*:**

* **download\_cci()**

Эта функция выполняет скачивание последних 10 файлов CCI из почты. Она подключается к почтовому ящику с использованием учетных данных, фильтрует письма по отправителю и сортирует их по дате получения. Затем она проверяет вложения в каждом письме и, если обнаруживает PDF-файл с названием, содержащим "CCI Daily", скачивает его в директорию **cci\_dir**. После скачивания она перемещает письмо в корзину.

* **download\_manual\_xlsx()**

Эта функция выполняет скачивание последних 10 "ручных" файлов XLSX из почты. Она также подключается к почтовому ящику, фильтрует письма по теме и сортирует их по дате получения. Затем она проверяет вложения в каждом письме и, если обнаруживает XLSX-файл, скачивает его в директорию **manual\_dir**. После скачивания она перемещает письмо в корзину.

* **main()**

Это главная функция, которая вызывает функции **download\_cci()** и **download\_manual\_xlsx()** для скачивания файлов из почты.