src

src package

Subpackages

src.config package

Submodules

src.config.app module

Модуль содержит настройки приложения, который меняются редко и только при внесении изменений в код приложения.

class **src.config.app.ConfigApp**

Базовые классы: `object`

BASE_URL= *'https://iforgot.apple.com'*

FORM_URL= *'https://iforgot.apple.com/password/verify/appleid'*

CAPTCHA_IMAGE_URL= *'https://iforgot.apple.com/captcha'*

CAPTCHA_SOURCE= *'apple-recovery'*

CAPTCHA_TIMEOUT= *5*

TASK_TIMEOUT= *60*

CAPTCHA_SOLUTION_TIMESTAMP_LIFETIME= *180*

src.config.settings module

Module contents

src.fastapi package

Submodules

src.fastapi.main module

src.fastapi.router module

src.fastapi.schemas module

src.fastapi.server module

Module contents

src.interfaces package

Submodules

src.interfaces.abstract_base64_converter module

Модуль содержит интерфейс для работы с конвертером изображений.

class **src.interfaces.abstract_base64_converter.AbstractBase64Converter**

Базовые классы: `ABC`

Интерфейс для работы с конвертером изображений base64 в bytes.

abstract
static **covert_to_bytes**(*base64_string*)

Конвертирует изображение base64 в bytes

Параметры:
base64_string (`str`) – Изображение в формате base64.

Response:
Изображение в формате bytes.

Тип результата:
`bytes`

src.interfaces.abstract_captcha_service module

Модуль содержит интерфейс для работы с сервисом решения капчи.

class **src.interfaces.abstract_captcha_service.AbstractCaptchaService**

Базовые классы: `ABC`

Интерфейс для работы с сервисом решения капчи.

abstract
async **post_captcha**(*image*, *timeout=0*)

Отправляет изображение для решения капчи.

Параметры:

- **image** (`bytes`) – Изображение с капчей.
- **timeout** (`int`) – Максимальное время в течении которого требуется вернуть решение капчи.

Тип результата:

`dict` | `None`

Результат:

Словарь с результатами решения.

abstract
async **result_report**(*task_id*, *correct*)

Отправляет отчет о решении капчи.

Параметры:

- **task_id** (`str`) – ID задачи по решению капчи.
- **correct** (`bool`) – Результат решения.

Тип результата:

`bool`

Результат:

Результат отправки решения.

src.interfaces.abstract_page_parser module

Модуль содержит интерфейс для работы с парсером ответов от сайта.

class **src.interfaces.abstract_page_parser.AbstractPageParser**

Базовые классы: `ABC`

Интерфейс для работы с парсером ответов от сайта.

abstract **parse**(*response*)

Запускает парсинг ответа.

Параметры:

response (`Response`) – Экземпляр класса *requests.Response*.

Response:

Словарь с результатами парсинга.

Тип результата:

`dict`

src.interfaces.abstract_proxy module

Модуль содержит интерфейс для работы с сервисом прокси.

class src.interfaces.abstract_proxy.AbstractProxy

Базовые классы: `ABC`

Интерфейс для работы с сервисом прокси. Используемый сервис прокси должен возвращать словарь с ключом „http“ или „https“, как показано в примере ниже.

Example:

```
get_proxy() -> '{'http': 'http://...', 'https': 'http://...', ...}'
```

abstract
async **get_proxy**()

Возвращает прокси.

Тип результата:
`dict` | `None`

Результат:
Словарь с ключами „http“ или „https“.

Module contents

class src.interfaces.AbstractBase64Converter

Базовые классы: `ABC`

Интерфейс для работы с конвертером изображений base64 в bytes.

abstract
static **covert_to_bytes**(*base64_string*)

Конвертирует изображение base64 в bytes

Параметры:
base64_string (`str`) – Изображение в формате base64.

Response:
Изображение в формате bytes.

Тип результата:
`bytes`

class src.interfaces.AbstractCaptchaService

Базовые классы: `ABC`

Интерфейс для работы с сервисом решения капчи.

abstract
async **post_captcha**(*image*, *timeout=0*)

Отправляет изображение для решения капчи.

Параметры:

- **image** (`bytes`) – Изображение с капчей.
- **timeout** (`int`) – Максимальное время в течении которого требуется вернуть решение капчи.

Тип результата:

`dict` | `None`

Результат:

Словарь с результатами решения.

abstract
async **result_report**(*task_id*, *correct*)

Отправляет отчет о решении капчи.

Параметры:

- **task_id** (`str`) – ID задачи по решению капчи.
- **correct** (`bool`) – Результат решения.

Тип результата:

`bool`

Результат:

Результат отправки решения.

class src.interfaces.AbstractPageParser

Базовые классы: `ABC`

Интерфейс для работы с парсером ответов от сайта.

abstract **parse**(*response*)

Запускает парсинг ответа.

Параметры:

response (`Response`) – Экземпляр класса *requests.Response*.

Response:

Словарь с результатами парсинга.

Тип результата:

`dict`

class src.interfaces.AbstractProxy

Базовые классы: `ABC`

Интерфейс для работы с сервисом прокси. Используемый сервис прокси должен возвращать словарь с ключом „http“ или „https“, как показано в примере ниже.

Example:

```
get_proxy() -> '{'http': 'http://...', 'https': 'http://...', ...}'
```

abstract
async **get_proxy**()

Возвращает прокси.

Тип результата:
`dict` | `None`

Результат:
Словарь с ключами „http“ или „https“.

src.logger package

Submodules

src.logger.context_logger module

class **src.logger.context_logger.ContextLogger**

Базовые классы: `object`

Логер с контекстом

static **get_logger**(*name*)

Возвращает логер по переданному имени.

Параметры:
name (`str`) – Имя логера, который требуется вернуть.

Тип результата:
`LoggingSearchKeyAdapter`

Результат:
Логер.

static **get_root_logger**()

Возвращает корневой логер (с именем root)

Тип результата:
`LoggingSearchKeyAdapter`

src.logger.logger_adapter module

class **src.logger.logger_adapter.LoggingSearchKeyAdapter**(*logger*, *wrapper='||'*)

Базовые классы: `LoggerAdapter`

Адаптер, позволяет задать контекст логера - текст, который будет отображаться в начале каждого сообщения логера. Такой подход применяется в многопоточных приложениях, с целью определить к какому запросу относится сообщение лога.

process(*msg*, *kwargs*)

Обрабатывает сообщение логера, добавляет контекст в сообщение, если он задан.

Параметры:

- **msg** – Сообщение логера.
- **kwargs** – Аргументы сообщения.

Результат:

Обработанное сообщение, аргументы сообщения.

static **get_context_message**()

Возвращает контекст из текущего запроса.

Тип результата:

`str`

Результат:

Контекст.

Module contents

src.logic package

Subpackages

src.logic.adapters package

Submodules

src.logic.adapters.response module

Module contents

src.logic.apple package

Submodules

src.logic.apple.apple module

src.logic.apple.exception_handler module

src.logic.apple.exceptions module

src.logic.apple.search_manager module

Module contents

src.logic.captcha package

Submodules

src.logic.captcha.captcha module

src.logic.captcha.captcha_exceptions module

exception **src.logic.captcha.captcha_exceptions.CaptchaConfiguredError**

Базовые классы: `Exception`

Исключение при работе с сервисом капч.

src.logic.captcha.captcha_service module

Module contents

src.logic.converters package

Submodules

src.logic.converters.base64_converter module

src.logic.converters.base64_exceptions module

Module contents

src.logic.parsers package

Submodules

src.logic.parsers.apple_result_parser module

Module contents

src.logic.proxy package

Submodules

src.logic.proxy.proxy module

Module contents

Module contents

src.request_params package

Subpackages

src.request_params.api package

Subpackages

src.request_params.api.captcha_solver package

Submodules

src.request_params.api.captcha_solver.captcha_solver_create module

src.request_params.api.captcha_solver.captcha_solver_get module

src.request_params.api.captcha_solver.captcha_solver_report module

Module contents

Submodules

src.request_params.api.apple_captcha module

src.request_params.api.apple_form_get module

src.request_params.api.apple_form_post module

src.request_params.api.apple_main module

src.request_params.api.apple_result module

Module contents

src.request_params.interfaces package

Submodules

src.request_params.interfaces.apple_base module

src.request_params.interfaces.apple_base_page module

src.request_params.interfaces.apple_base_requests module

src.request_params.interfaces.base module

src.request_params.interfaces.captcha_service_base module

Module contents

Module contents

src.runners package

Submodules

src.runners.main module

src.runners.mobile_phones module

src.runners.mobile_phones.get_random_mobile_phone()

Тип результата:
`str`

src.runners.mobile_phones.get_random_mobile_phones(*count*)

Тип результата:
`list`[`str`]

src.runners.spammer module

Module contents

Пакет содержит вспомогательные функции для ручного тестирования приложения. В работе приложения функции не участвуют и не требуются.

src.utils package

Submodules

src.utils.utils module

src.utils.utils.now()

Возвращает текущую дату и время в формате unix epoch.

Тип результата:

`int`

Результат:

Unix epoch время и дата.

src.utils.utils.informer(*step_number*, *step_message*)

Декоратор, выводит логи до начала работы функции и после ее завершения.

Параметры:

- **step_number** (`int`) – Шаг в формате int
- **step_message** (`str`) – Сообщение на шаге.

Тип результата:

`Callable`

Результат:

Декорированная функция.

src.utils.utils.strip_str(*input_str*)

Удаляет из строки символы » «, « «.

Тип результата:

`str`

Module contents

src.utils.now()

Возвращает текущую дату и время в формате unix epoch.

Тип результата:

`int`

Результат:

Unix epoch время и дата.

src.utils.informer(*step_number*, *step_message*)

Декоратор, выводит логи до начала работы функции и после ее завершения.

Параметры:

- **step_number** (`int`) – Шаг в формате int
- **step_message** (`str`) – Сообщение на шаге.

Тип результата:

`Callable`

Результат:

Декорированная функция.

Module contents