SominAI Test Task

Generated by Doxygen 1.12.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

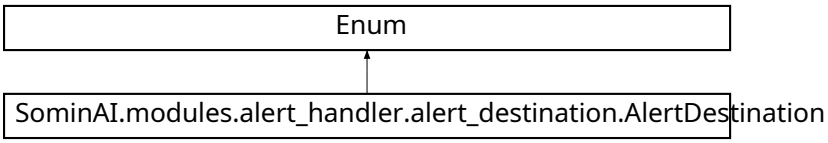
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 SominAI.modules.alert_handler.alert_destination.AlertDestination Class Reference

Inheritance diagram for SominAI.modules.alert_handler.alert_destination.AlertDestination:



Static Public Attributes

- str **MONITORING** = "monitoring"
- str **TELEGRAM** = "telegram"
- str **LOGGING** = "logging"

3.1.1 Detailed Description

Enumeration for alert destinations.

This enumeration defines different destinations where alerts can be sent.
It provides flexibility for handling and routing alerts in various systems.

Attributes:
MONITORING: Destination for monitoring systems (e.g., dashboards or monitoring tools).
TELEGRAM: Destination for sending alerts to Telegram.
LOGGING: Destination for logging alerts into application logs.

The documentation for this class was generated from the following file:

- modules/alert_handler/alert_destination.py

3.2 SominAI.modules.alert_handler.alert_handler.AlertHandler Class Reference

Public Member Functions

- __init__ (self, dict config)
- monitor_services (self, dict services)
- None send_alert (self, AlertDestination destination, str message)

Static Public Member Functions

- get_instance (dict config)

Public Attributes

- **config** = config
- **logger** = Logger

Protected Member Functions

- None _send_to_monitoring (self, str message)
- None **_send_to_telegram** (self, str message)
- None **_log_alert** (self, str message)

Static Protected Attributes

- **_instance** = None

3.2.1 Detailed Description

A universal alert handler that supports multiple notification channels.

This handler initializes once and monitors the state of services. If a signal
is detected from any service, it sends an alert to the specified destination.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 __init__()

SominAI.modules.alert_handler.alert_handler.AlertHandler.__init__ (
self,
dict config)

Initializes the alert handler.

:param config: Configuration dictionary for notification channels.

3.2.3 Member Function Documentation

3.2.3.1 _send_to_monitoring()

None SominAI.modules.alert_handler.alert_handler.AlertHandler._send_to_monitoring (
self,
str message) [protected]

Sending to monitoring services logic.

3.2.3.2 get_instance()

SominAI.modules.alert_handler.alert_handler.AlertHandler.get_instance (
dict config) [static]

Returns the singleton instance of the AlertHandler.

:param config: Configuration dictionary for notification channels.
:return: The AlertHandler singleton instance.

3.2.3.3 monitor_services()

SominAI.modules.alert_handler.alert_handler.AlertHandler.monitor_services (
self,
dict services)

Monitors the state of services and triggers alerts if any service sends a signal.

:param services: Dictionary of services and their states.

3.2.3.4 send_alert()

None SominAI.modules.alert_handler.alert_handler.AlertHandler.send_alert (
self,
AlertDestination destination,
str message)

Sends an alert to the specified destination.

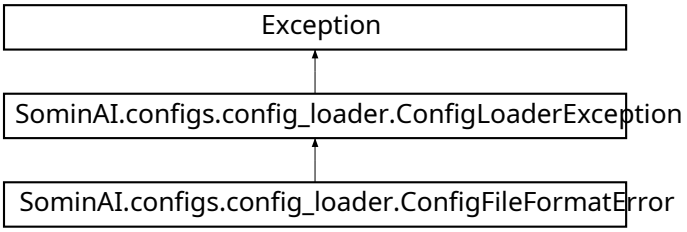
:param destination: Alert destination from AlertDestination enum.
:param message: Message text to be sent.

The documentation for this class was generated from the following file:

- modules/alert_handler/alert_handler.py

3.3 SominAI.configs.config_loader.ConfigFileFormatError Class Reference

Inheritance diagram for SominAI.configs.config_loader.ConfigFileFormatError:



Public Member Functions

- **__init__** (self, str file_path, Exception error)

Public Attributes

- **file_path** = file_path
- **error** = error

3.3.1 Detailed Description

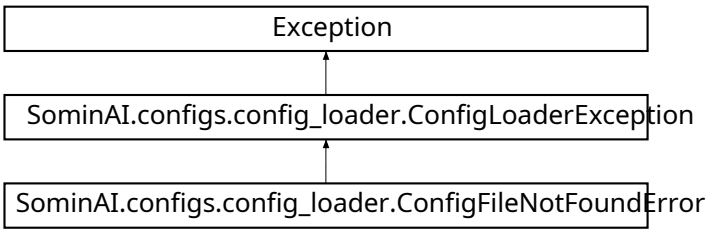
Raised when the configuration file has invalid YAML format.

The documentation for this class was generated from the following file:

- configs/config_loader.py

3.4 SominAI.configs.config_loader.ConfigFileNotFoundError Class Reference

Inheritance diagram for SominAI.configs.config_loader.ConfigFileNotFoundError:



Public Member Functions

- **__init__** (self, str file_path)

Public Attributes

- **file_path** = file_path

3.4.1 Detailed Description

Raised when the configuration file is not found.

The documentation for this class was generated from the following file:

- configs/config_loader.py

3.5 SominAI.configs.config_loader.ConfigLoader Class Reference

Static Public Member Functions

- dict load_config (str file_path)

3.5.1 Detailed Description

A utility class to load and parse YAML configuration files.

This class provides a static method to load YAML files and ensures
proper error handling for missing files and invalid formats.

3.5.2 Member Function Documentation

3.5.2.1 load_config()

dict SominAI.configs.config_loader.ConfigLoader.load_config (
str file_path) [static]

Loads a YAML configuration file.

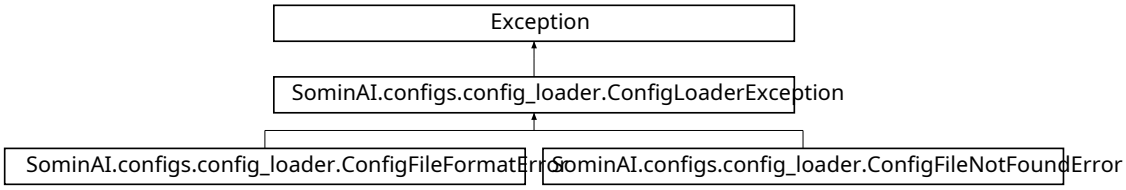
:param file_path: Path to the YAML file.
:return: Configuration as a dictionary.
:raises ConfigFileNotFoundError: If the file does not exist.
:raises ConfigFileFormatError: If the YAML format is invalid.

The documentation for this class was generated from the following file:

- configs/config_loader.py

3.6 SominAI.configs.config_loader.ConfigLoaderException Class Reference

Inheritance diagram for SominAI.configs.config_loader.ConfigLoaderException:



3.6.1 Detailed Description

Base exception for ConfigLoader.

The documentation for this class was generated from the following file:

- configs/config_loader.py

3.7 SominAI.crawler.crawler.Crawler Class Reference

Public Member Functions

- **__init__** (self, str rabbit_config_path, str redis_config_path)
- connect (self)
- process_task (self, dict message)
- consume_tasks (self)
- close (self)

Public Attributes

- **rabbit_publisher** = RabbitMQPublisher(rabbit_config_path)
- **redis_manager** = RedisManager(redis_config_path)
- **parser** = TikTokLibraryParser()
- str **task_queue** = "crawler.task"
- str **result_queue** = "crawler.result"

3.7.1 Detailed Description

Asynchronously consumes tasks from RabbitMQ, processes data, and publishes results.

3.7.2 Member Function Documentation

3.7.2.1 close()

SominAI.crawler.crawler.Crawler.close (
self)

Closes connections to RabbitMQ and Redis.

3.7.2.2 connect()

SominAI.crawler.crawler.Crawler.connect (
self)

Establishes connections to RabbitMQ and Redis.

3.7.2.3 consume_tasks()

SominAI.crawler.crawler.Crawler.consume_tasks (
self)

Asynchronously consumes tasks from the RabbitMQ queue.

3.7.2.4 process_task()

SominAI.crawler.crawler.Crawler.process_task (
self,
dict message)

Processes a single task message asynchronously.

The documentation for this class was generated from the following file:

- crawler/crawler.py

3.8 SominAI.modules.exception_handler.exception_handler.ExceptionHandler Class Reference

Static Public Member Functions

- dict handle_exception (Exception exc, dict context)

3.8.1 Detailed Description

A centralized handler for exceptions in a SaaS environment.

This class handles exceptions, logs them using the Logger, and formats a unified response structure for error handling. In an HTTP context, 'path' and 'method' can be provided; otherwise, default values are used.

3.8.2 Member Function Documentation

3.8.2.1 handle_exception()

dict SominAI.modules.exception_handler.exception_handler.ExceptionHandler.handle_exception (
Exception exc,
dict context) [static]

Handles an exception and formats a unified response.

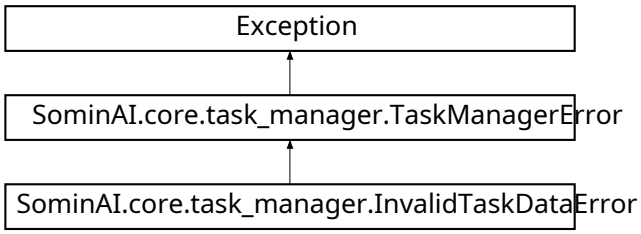
:param exc: Exception to handle.
:param context: Additional context, e.g., request details.
:return: Unified error response.
:raises TypeError: If 'exc' is not an instance of Exception or 'context' is not a dictionary.

The documentation for this class was generated from the following file:

- modules/exception_handler/exception_handler.py

3.9 SominAI.core.task_manager.InvalidTaskDataError Class Reference

Inheritance diagram for SominAI.core.task_manager.InvalidTaskDataError:



Public Member Functions

- **__init__** (self, str message)

3.9.1 Detailed Description

Raised when task data is invalid.

The documentation for this class was generated from the following file:

- core/task_manager.py

3.10 SominAI.modules.logger.logger.Logger Class Reference

Public Member Functions

- None configure_logger(cls, str name, dict|str|None config_path=None, LogLevel level=LogLevel.INFO, str|None log_file=None)
- None log (cls, message, LogLevel level)
- None log_info (cls, message)
- None log_debug (cls, message)
- None log_warning (cls, message)
- None log_error (cls, message)
- None log_critical (cls, message)

Public Attributes

- dict **_logger** = cls._load_yaml_config(config_path)
- str **_logger** = cls._validate_message(message)

Protected Member Functions

- dict _load_yaml_config (cls, str config_path)
- None _ensure_log_directory (cls)
- None _validate_log_level (cls, LogLevel level)
- str _validate_message (cls, message)

Static Protected Attributes

- logging **_logger** = None
- str **_log_file** = None

3.10.1 Detailed Description

A universal logger module that provides configurable logging capabilities.

3.10.2 Member Function Documentation

3.10.2.1 _ensure_log_directory()

None SominAI.modules.logger.logger.Logger._ensure_log_directory (
cls) [protected]

Ensures that the log directory exists. If it doesn't, creates it.

3.10.2.2 _load_yaml_config()

dict SominAI.modules.logger.logger.Logger._load_yaml_config (
cls,
str config_path) [protected]

Loads the YAML configuration file for logging using ConfigLoader.

3.10.2.3 _validate_log_level()

None SominAI.modules.logger.logger.Logger._validate_log_level (
cls,
LogLevel level) [protected]

Ensures the provided log level is valid.

3.10.2.4 _validate_message()

str SominAI.modules.logger.logger.Logger._validate_message (
cls,
message) [protected]

Validates and converts the message to a string.

3.10.2.5 configure_logger()

None SominAI.modules.logger.logger.Logger.configure_logger (
cls,
str name,
dict | str | None config_path = None,
LogLevel level = LogLevel.INFO,
str | None log_file = None)

Configures the logger using a YAML configuration file or defaults.

:param name: Name of the logger.
:param config_path: Path/dict to the YAML configuration. If None, defaults are used.
:param level: Logging level if no config is provided.
:param log_file: File path for log output (optional).
:raises LoggerConfigError: If the config fails to load or apply.

3.10.2.6 log()

None SominAI.modules.logger.logger.Logger.log (
cls,
message,
LogLevel level)

Logs a message at the specified log level.

3.10.2.7 log_critical()

None SominAI.modules.logger.logger.Logger.log_critical (
cls,
message)

Logs a critical message.

3.10.2.8 log_debug()

None SominAI.modules.logger.logger.Logger.log_debug (
cls,
message)

Logs a debug message.

3.10.2.9 log_error()

None SominAI.modules.logger.logger.Logger.log_error (
cls,
message)

Logs an error message.

3.10.2.10 log_info()

None SominAI.modules.logger.logger.Logger.log_info (
cls,
message)

Logs an informational message.

3.10.2.11 log_warning()

None SominAI.modules.logger.logger.Logger.log_warning (
cls,
message)

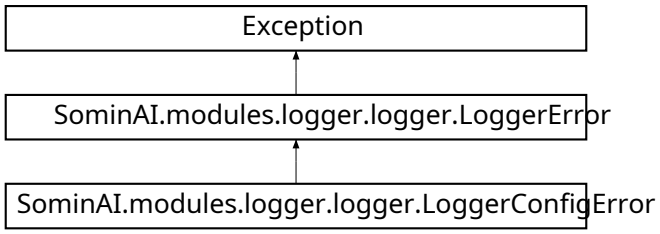
Logs a warning message.

The documentation for this class was generated from the following file:

- modules/logger/logger.py

3.11 SominAI.modules.logger.logger.LoggerConfigError Class Reference

Inheritance diagram for SominAI.modules.logger.logger.LoggerConfigError:



3.11.1 Detailed Description

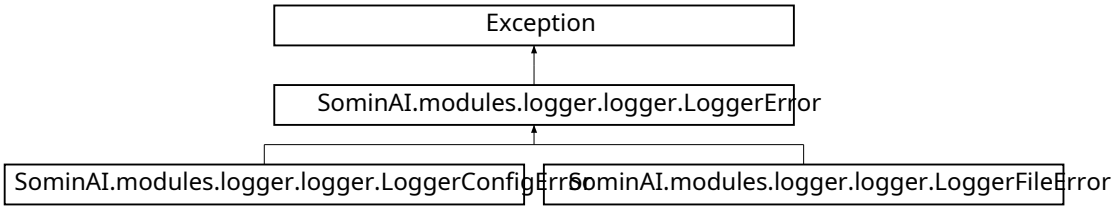
Raised when there is an error in loading or applying the configuration.

The documentation for this class was generated from the following file:

- modules/logger/logger.py

3.12 SominAI.modules.logger.logger.LoggerError Class Reference

Inheritance diagram for SominAI.modules.logger.logger.LoggerError:



3.12.1 Detailed Description

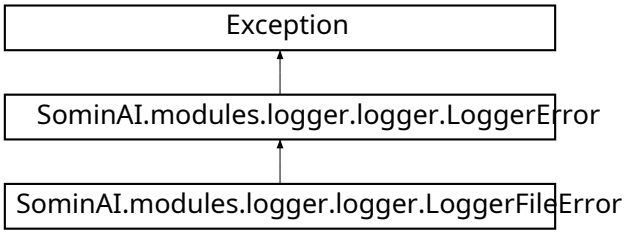
Base exception for Logger errors.

The documentation for this class was generated from the following file:

- modules/logger/logger.py

3.13 SominAI.modules.logger.logger.LoggerFileError Class Reference

Inheritance diagram for SominAI.modules.logger.logger.LoggerFileError:



3.13.1 Detailed Description

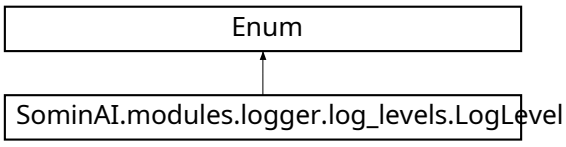
Raised when there is an issue with log file operations.

The documentation for this class was generated from the following file:

- modules/logger/logger.py

3.14 SominAI.modules.logger.log_levels.LogLevel Class Reference

Inheritance diagram for SominAI.modules.logger.log_levels.LogLevel:



Static Public Attributes

- **INFO** = logging.INFO
- **DEBUG** = logging.DEBUG
- **WARNING** = logging.WARNING
- **ERROR** = logging.ERROR
- **CRITICAL** = logging.CRITICAL

3.14.1 Detailed Description

Enumeration for log levels.

This enumeration maps Python's logging levels to a more structured format.
It can be used to specify logging verbosity in applications.

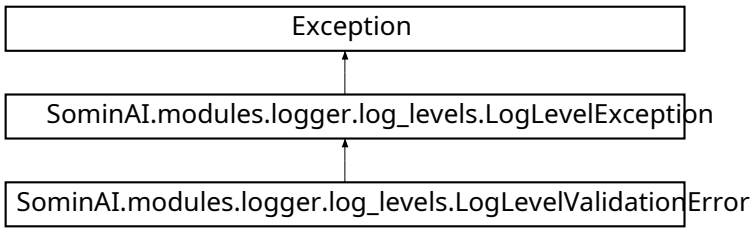
Attributes:
INFO: Informational messages.
DEBUG: Debugging messages.
WARNING: Warning messages indicating potential issues.
ERROR: Error messages for failures that require attention.
CRITICAL: Critical error messages for severe failures.

The documentation for this class was generated from the following file:

- modules/logger/log_levels.py

3.15 SominAI.modules.logger.log_levels.LogLevelException Class Reference

Inheritance diagram for SominAI.modules.logger.log_levels.LogLevelException:



3.15.1 Detailed Description

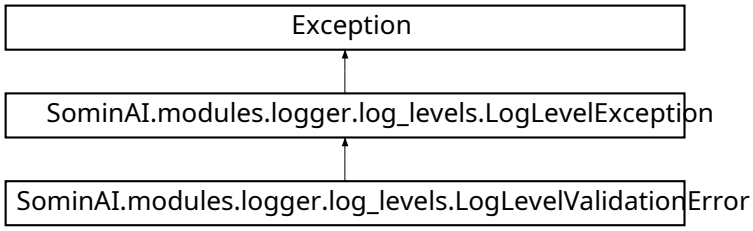
Base exception for LogLevel errors.

The documentation for this class was generated from the following file:

- modules/logger/log_levels.py

3.16 SominAI.modules.logger.log_levels.LogLevelValidationError Class Reference

Inheritance diagram for SominAI.modules.logger.log_levels.LogLevelValidationError:



3.16.1 Detailed Description

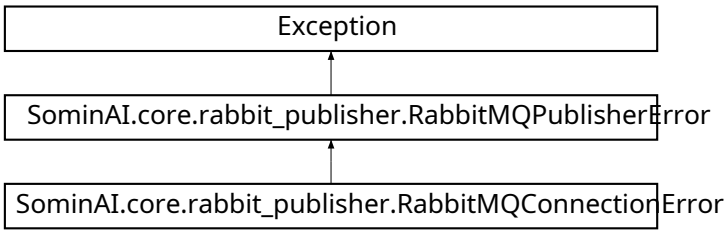
Raised when there is a validation error in LogLevel input.

The documentation for this class was generated from the following file:

- modules/logger/log_levels.py

3.17 SominAI.core.rabbit_publisher.RabbitMQConnectionError Class Reference

Inheritance diagram for SominAI.core.rabbit_publisher.RabbitMQConnectionError:



3.17.1 Detailed Description

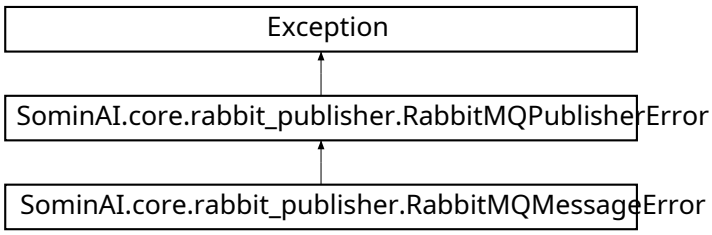
Raised when there is a connection error.

The documentation for this class was generated from the following file:

- core/rabbit_publisher.py

3.18 SominAI.core.rabbit_publisher.RabbitMQMessageError Class Reference

Inheritance diagram for SominAI.core.rabbit_publisher.RabbitMQMessageError:



3.18.1 Detailed Description

Raised when there is an error publishing a message.

The documentation for this class was generated from the following file:

- core/rabbit_publisher.py

3.19 SominAI.core.rabbit_publisher.RabbitMQPublisher Class Reference

Public Member Functions

- __init__ (self, str config_path="configs/rabbitmq.yaml")
- connect (self)
- None send_task (self, str task_id, str keyword)
- None publish (self, str queue, dict message)
- None close_connection (self)

Public Attributes

- **host** = config.get("host", "localhost")
- **queue** = config.get("queue", "task_queue")
- **username** = config.get("username", "guest")
- **password** = config.get("password", "guest")
- **logger** = Logger
- **exception_handler** = ExceptionHandler()
- **channel**
- connection

Protected Member Functions

- None _ensure_channel_initialized (self)

3.19.1 Detailed Description

Handles asynchronous communication with RabbitMQ for task publishing.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 __init__()

SominAI.core.rabbit_publisher.RabbitMQPublisher.__init__ (
self,
str config_path = "configs/rabbitmq.yaml")

Initializes the RabbitMQPublisher with configuration from a file.

:param config_path: Path to the RabbitMQ configuration file.

3.19.3 Member Function Documentation

3.19.3.1 _ensure_channel_initialized()

None SominAI.core.rabbit_publisher.RabbitMQPublisher._ensure_channel_initialized (
self) [protected]

Check whether the RabbitMQ channel is initialized or not.

:raise RabbitMQConnectionError: In case if channel is 'None'.

3.19.3.2 close_connection()

None SominAI.core.rabbit_publisher.RabbitMQPublisher.close_connection (
self)

Asynchronously closes the connection to RabbitMQ.

3.19.3.3 connect()

SominAI.core.rabbit_publisher.RabbitMQPublisher.connect (
self)

Asynchronously connects to RabbitMQ and declares the default queue.

3.19.3.4 publish()

None SominAI.core.rabbit_publisher.RabbitMQPublisher.publish (
self,
str queue,
dict message)

Publishes a message to a specified RabbitMQ queue asynchronously.

:param queue: The name of the queue.
:param message: The message to publish (as a dictionary).

3.19.3.5 send_task()

None SominAI.core.rabbit_publisher.RabbitMQPublisher.send_task (
self,
str task_id,
str keyword)

Publishes a task message to the RabbitMQ queue asynchronously.

:param task_id: Unique ID of the task.
:param keyword: Keyword for the task.

3.19.4 Member Data Documentation

3.19.4.1 connection

SominAI.core.rabbit_publisher.RabbitMQPublisher.connection

Initial value:

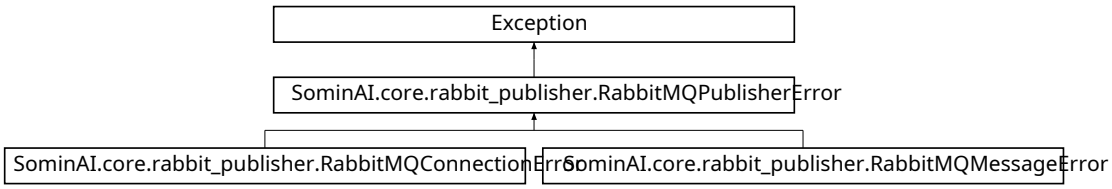
```
= await aio_pika.connect_robust(
                f"amqp://{self.username}:{self.password}@{self.host}/"
            )
```

The documentation for this class was generated from the following file:

- core/rabbit_publisher.py

3.20 SominAI.core.rabbit_publisher.RabbitMQPublisherError Class Reference

Inheritance diagram for SominAI.core.rabbit_publisher.RabbitMQPublisherError:



3.20.1 Detailed Description

Base exception for RabbitMQPublisher.

The documentation for this class was generated from the following file:

- core/rabbit_publisher.py

3.21 SominAI.core.redis_manager.RedisManager Class Reference

Public Member Functions

- **__init__** (self, str config_path="configs/config.yaml")
- None connect (self)
- None close (self)
- None set_status (self, str task_id, TaskData task_data)
- dict|None get_status (self, str task_id)
- None update_task_status (self, str task_id, str status, TaskDa|None result=None)
- None delete_task (self, str task_id)

Public Attributes

- **host** = config.get("host", "localhost")
- **port** = config.get("port", 6379)
- **db** = config.get("db", 0)
- **logger** = Logger
- **exception_handler** = ExceptionHandler()
- redis.Redis|None **redis_client** = None

3.21.1 Detailed Description

Asynchronous manager for interactions with the Redis database for task status storage.

3.21.2 Member Function Documentation

3.21.2.1 close()

None SominAI.core.redis_manager.RedisManager.close (
self)

Closes the Redis connection asynchronously.

3.21.2.2 connect()

None SominAI.core.redis_manager.RedisManager.connect (
self)

Asynchronously connects to the Redis server.

3.21.2.3 delete_task()

None SominAI.core.redis_manager.RedisManager.delete_task (
self,
str task_id)

Deletes a task from Redis asynchronously.

3.21.2.4 get_status()

dict | None SominAI.core.redis_manager.RedisManager.get_status (
self,
str task_id)

Retrieves the status of a task from Redis asynchronously.

3.21.2.5 set_status()

None SominAI.core.redis_manager.RedisManager.set_status (
self,
str task_id,
TaskData task_data)

Stores the status of a task in Redis asynchronously.

3.21.2.6 update_task_status()

```
None SominAI.core.redis_manager.RedisManager.update_task_status (
                self,
               str task_id,
               str status,
               TaskData | None result = None)
```

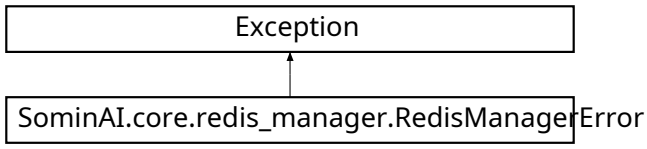
Updates the status and result of a task in Redis asynchronously.

The documentation for this class was generated from the following file:

- core/redis_manager.py

3.22 SominAI.core.redis_manager.RedisManagerError Class Reference

Inheritance diagram for SominAI.core.redis_manager.RedisManagerError:



3.22.1 Detailed Description

Custom exception for RedisManager.

The documentation for this class was generated from the following file:

- core/redis_manager.py

3.23 SominAI.core.result_processor.ResultProcessor Class Reference

Public Member Functions

- **__init__** (self, str rabbit_config_path, str redis_config_path, dict alert_config)
- connect (self)
- process_results (self)
- close (self)

Public Attributes

- **redis_manager** = RedisManager(redis_config_path)
- **rabbit_publisher** = RabbitMQPublisher(rabbit_config_path)
- **alert_handler** = AlertHandler.get_instance(alert_config)
- str **result_queue** = "crawler.result"

3.23.1 Detailed Description

Asynchronously processes completed tasks from the ‘crawler.result‘ queue.

3.23.2 Member Function Documentation

3.23.2.1 close()

SominAI.core.result_processor.ResultProcessor.close (
self)

Closes connections to RabbitMQ and Redis.

3.23.2.2 connect()

SominAI.core.result_processor.ResultProcessor.connect (
self)

Establishes connections to RabbitMQ and Redis.

3.23.2.3 process_results()

SominAI.core.result_processor.ResultProcessor.process_results (
self)

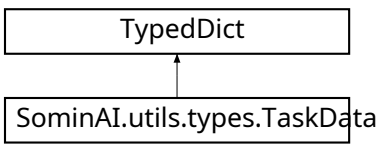
Processes results from the RabbitMQ result queue asynchronously.

The documentation for this class was generated from the following file:

- core/result_processor.py

3.24 SominAI.utils.types.TaskData Class Reference

Inheritance diagram for SominAI.utils.types.TaskData:



Static Public Attributes

- str **region** | None
- dict **result** | None

3.24.1 Detailed Description

TypedDict for task data stored in Redis.

The documentation for this class was generated from the following file:

- utils/types.py

3.25 SominAI.core.task_manager.TaskManager Class Reference

Public Member Functions

- __init__ (self, str redis_config_path, str rabbit_config_path)
- str create_task (self, str keyword)
- TaskData | None get_task_status (self, str task_id)
- update_task_status (self, str task_id, str status, dict | None result=None)

Public Attributes

- **redis_manager** = RedisManager(redis_config_path)
- **rabbit_publisher** = RabbitMQPublisher(rabbit_config_path)
- **logger** = Logger

Protected Member Functions

- bool _is_valid_task_data (self, TaskData | dict data)

3.25.1 Detailed Description

Manages tasks, including creating, updating statuses, and storing results.

This class interacts with Redis for status management and RabbitMQ for
task queueing.

3.25.2 Constructor & Destructor Documentation

3.25.2.1 __init__()

SominAI.core.task_manager.TaskManager.__init__ (
self,
str redis_config_path,
str rabbit_config_path)

Initializes the TaskManager with dependencies.

:param redis_config_path: Path to the Redis configuration file.
:param rabbit_config_path: Path to the RabbitMQ configuration file.

3.25.3 Member Function Documentation

3.25.3.1 _is_valid_task_data()

```
bool SominAI.core.task_manager.TaskManager._is_valid_task_data (
            self,
            TaskData | dict data)        [protected]
```

Validates the structure of task data.

:param data: Data to validate.
:return: True if valid, False otherwise.

3.25.3.2 create_task()

```
str SominAI.core.task_manager.TaskManager.create_task (
            self,
            str keyword)
```

Asynchronously creates a new task and adds it to the queue.

:param keyword: Keyword to be used for crawling.
:return: Unique task ID.

3.25.3.3 get_task_status()

```
TaskData | None SominAI.core.task_manager.TaskManager.get_task_status (
            self,
            str task_id)
```

Retrieves the status of a task.

:param task_id: Unique ID of the task.
:return: Task status and details if found, otherwise None.

3.25.3.4 update_task_status()

```
SominAI.core.task_manager.TaskManager.update_task_status (
            self,
            str task_id,
            str status,
            dict | None result = None)
```

Asynchronously updates the status and result of a task.

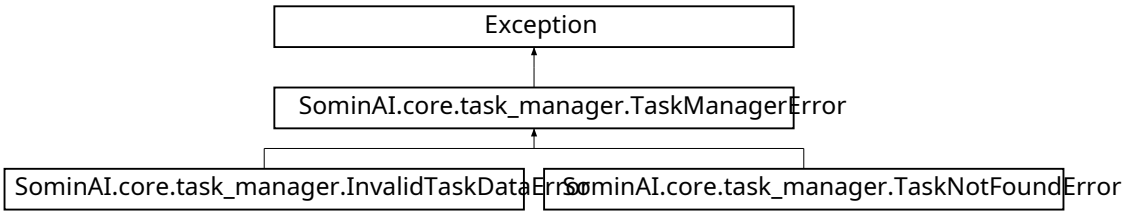
:param task_id: Unique ID of the task.
:param status: New status of the task (e.g., 'completed', 'failed').
:param result: Result of the task (optional).

The documentation for this class was generated from the following file:

- core/task_manager.py

3.26 SominAI.core.task_manager.TaskManagerError Class Reference

Inheritance diagram for SominAI.core.task_manager.TaskManagerError:



3.26.1 Detailed Description

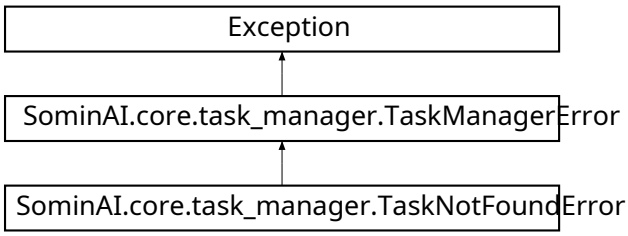
Base exception for TaskManager.

The documentation for this class was generated from the following file:

- core/task_manager.py

3.27 SominAI.core.task_manager.TaskNotFoundError Class Reference

Inheritance diagram for SominAI.core.task_manager.TaskNotFoundError:



Public Member Functions

- **__init__** (self, str task_id)

Public Attributes

- **task_id** = task_id

3.27.1 Detailed Description

Raised when a task is not found.

The documentation for this class was generated from the following file:

- core/task_manager.py

3.28 SominAI.crawler.parser.TikTokLibraryParser Class Reference

Public Member Functions

- **__init__** (self)
- str build_query (self, str region, str adv_name, **kwargs)
- str fetch_data (self, str url)
- list[dict] parse_data (self, str html_content)
- list[dict] search_ads (self, str region, str adv_name, **kwargs)

Public Attributes

- **logger** = Logger

Static Public Attributes

- str **BASE_URL** = "https://library.tiktok.com/ads"

3.28.1 Detailed Description

A universal parser for the TikTok Library search API.
Dynamically constructs URLs based on given parameters.

3.28.2 Member Function Documentation

3.28.2.1 build_query()

str SominAI.crawler.parser.TikTokLibraryParser.build_query (
self,
str region,
str adv_name,
** kwargs)

Constructs a TikTok Library query URL.

:param region: The region code (e.g., "BE", "US", "AT").
:param adv_name: The advertiser's name or keyword for search.
:param kwargs: Optional parameters for the query (e.g., start_time, end_time, sort_type, etc.).
:return: Constructed URL for the TikTok Library search.

3.28.2.2 fetch_data()

str SominAI.crawler.parser.TikTokLibraryParser.fetch_data (
self,
str url)

Fetches the HTML content of the TikTok Library page by navigating to the URL.

:param url: The URL of the TikTok Library search page.
:return: HTML content of the page as a string.

3.28.2.3 parse_data()

list[dict] SominAI.crawler.parser.TikTokLibraryParser.parse_data (
self,
str html_content)

Parses HTML content to extract advertisement details.

:param html_content: HTML content fetched from TikTok Library.
:return: List of parsed advertisements.

3.28.2.4 search_ads()

list[dict] SominAI.crawler.parser.TikTokLibraryParser.search_ads (
self,
str region,
str adv_name,
* * kwargs)

Searches for ads in the TikTok Library with the given parameters.

:param region: The region code (e.g., "BE").
:param adv_name: The advertiser's name or keyword for search.
:param kwargs: Additional parameters for customizing the search.
:return: List of parsed advertisements.

The documentation for this class was generated from the following file:

- crawler/parser.py