文本比较(T)
已产生: 2024/12/24 14:38:33

模式: 差异, 带上下文
左边文件: D:\Source\crewai-updated\task.py 右边文件: D:\Source\crewai\task.py

import threading import uuid from concurrent.futures import Future from copy import copy from hashlib import md5 from typing import Any, Dict, List, Optional, Set, Tuple, Type, Union	=	import threading import uuid from concurrent.futures import Future from copy import copy from hashlib import md5 from typing import Any, Dict, List, Optional, Set, Tuple, Type, Union
from pathlib import Path	+-	
	=	
from jinja2 import Environment, FileSystemLoader	+-	
from opentelemetry.trace import Span from pydantic import (UUID4, BaseModel, Field, PrivateAttr,	=	from opentelemetry.trace import Span from pydantic import (UUID4, BaseModel, Field, PrivateAttr,
tools_errors: int = 0 delegations: int = 0 i18n: I18N = I18N() name: Optional[str] = Field(default=None) prompt_context: Optional[str] = None description: str = Field(description="Description of the actual task.")	=	tools_errors: int = 0 delegations: int = 0 i18n: I18N = I18N() name: Optional[str] = Field(default=None) prompt_context: Optional[str] = None description: str = Field(description="Description of the actual task.")
task_prompt: Optional[str] = Field(description="Prompt to be used for the task.", default="")	+-	
expected_output: str = Field(description="Clear definition of expected output for the task.") config: Optional[Dict[str, Any]] = Field(description="Configuration for the agent", default=None,	=	expected_output: str = Field(description="Clear definition of expected output for the task.") config: Optional[Dict[str, Any]] = Field(description="Configuration for the agent", default=None,
else pydantic_output.model_dump_json() if pydantic_output else result) self._save_file(content) return task_output	=	else pydantic_output.model_dump_json() if pydantic_output else result) self._save_file(content) return task_output
def render_template(self, file_path, inputs): directory_path = os.path.dirname(file_path) file_loader = FileSystemLoader(directory_path) env = Environment(loader=file_loader) template_filename = os.path.basename(file_path) template = env.get_template(template_filename) output = template.render(inputs) return output	+-	
def prompt(self) -> str: """Prompt the task. Returns: Prompt of the task. """	=	def prompt(self) -> str: """Prompt the task. Returns: Prompt of the task. """
# tasks_slices = [self.description]	<>	tasks_slices = [self.description]
output = self.i18n.slice("expected_output").format(expected_output=self.expected_output)	=	output = self.i18n.slice("expected_output").format(expected_output=self.expected_output)
tasks_slices = [self.description, self.task_prompt, output]	<>	tasks_slices = [self.description, output]

return "\n".join(tasks_slices)
 	
def interpolate_inputs(self, inputs: Dict[str, Any]) -> None:
"""Interpolate inputs into the task description and expected output."""
if self._original_description is None:
self._original_description = self.description
if self._original_expected_output is None:
self._original_expected_output = self.expected_output
	=	return "\n".join(tasks_slices)
		
		def interpolate_inputs(self, inputs: Dict[str, Any]) -> None:
		"""Interpolate inputs into the task description and expected output."""
		if self._original_description is None:
		self._original_description = self.description
		if self._original_expected_output is None:
		self._original_expected_output = self.expected_output

`        if inputs:`
`            self.description =`
`self._original_description.format(**inputs)`
`            self.expected_output =`
`self._original_expected_output.format(**inputs)`
		`        if inputs:`
		`            self.description =`
		`self._original_description.format(**inputs)`
		`            self.expected_output =`
		`self._original_expected_output.format(**inputs)`

`            if "prompt_path" in inputs:`
`                prompt_path = inputs["prompt_path"]`
`                if os.path.exists(prompt_path):`
`                    self.task_prompt =`
`self.render_template(prompt_path, inputs)`
	+-	

`    def increment_tools_errors(self) -> None:`
`        """Increment the tools errors counter."""`
`        self.tools_errors += 1`
 	
`    def increment_delegations(self, agent_name: Optional[str])`
`-> None:`
	=	`    def increment_tools_errors(self) -> None:`
		`        """Increment the tools errors counter."""`
		`        self.tools_errors += 1`
		
		`    def increment_delegations(self, agent_name: Optional[str])`
		`-> None:`