ai-testsuite

May 27, 2023

0.1 AGiXT AI Testsuite

0.1.1 Jupyter Notebook for AI interactions

First set the uri of your AGiXT server, the Agent's name, and a command name to test with.

[1]:
```
import requests
from pprint import pprint

from cfg import cfg

base_uri = "http://localhost:7437"

agent_name = "test_suite_agent"
agent_settings = cfg["agent_settings_openai"]
agent_commands = cfg["agent_commands"]

shots = 3 # for SMART actions

instruction = cfg["message-1"]
chat = cfg["message-3"]
task = cfg["message-3"]
```

0.2 Agent setup

[2]:
```
# Test POST /api/agent
# Add an agent
data = {"agent_name": agent_name, "settings": agent_settings }
response = requests.post(f"{base_uri}/api/agent", json=data)
pprint(response.json())
```

```
{'agent_file': 'test_suite_agent.yaml', 'message': 'Agent added'}
```

[3]:
```
# Test PATCH /api/agent/{agent_name}/command
# Update the agent's commands
for command_name, enabled in agent_commands.items():
 data = {"command_name": command_name, "enable": enabled}
 response = requests.patch(f"{base_uri}/api/agent/{agent_name}/command",␣
 ↪json=data)
```

```
pprint(response.json())
```

```
{'message': "Command 'Get Response' toggled for agent 'test_suite_agent'."}
{'message': "Command 'Google Search' toggled for agent 'test_suite_agent'."}
{'message': "Command 'Is Valid URL' toggled for agent 'test_suite_agent'."}
{'message': "Command 'Sanitize URL' toggled for agent 'test_suite_agent'."}
{'message': "Command 'Scrape Links' toggled for agent 'test_suite_agent'."}
{'message': "Command 'Scrape Links with Playwright' toggled for agent "
            "'test_suite_agent'."}
{'message': "Command 'Scrape Text' toggled for agent 'test_suite_agent'."}
{'message': "Command 'Scrape Text with Playwright' toggled for agent "
            "'test_suite_agent'."}
{'message': "Command 'Use The Search Engine' toggled for agent "
            "'test_suite_agent'."}
```

0.3 [SMART]Instruct agent

[4]:
```
# Test POST /api/agent/{agent_name}/instruct
# Instruct the agent
data = {"prompt": instruction}
response = requests.post(f"{base_uri}/api/agent/{agent_name}/instruct",␣
 ↪json=data)
assert response.status_code == 200, response.json()
data = response.json()
print(data["response"])
```

```
AI is revolutionizing the way we live by automating tasks, analyzing data, and
providing insights that were once impossible to access. The possibilities are
endless!
```

[5]:
```
# Test POST /api/agent/{agent_name}/smartinstruct/{shots}
# SMART Instruct the agent
data = {"prompt": instruction}
response = requests.post(f"{base_uri}/api/agent/{agent_name}/smartinstruct/
 ↪{shots}", json=data)
assert response.status_code == 200, response.json()
data = response.json()
print(data["response"])
```

```
AI is changing the way we interact with the world, improving healthcare,
education, and many other industries. Let's ensure ethical and responsible use
of its potential. #AI #innovation
```

0.4 [SMART]Chat with agent

[6]:
```
# Test POST /api/{agent_name}/chat
# Chat with agent
data = {"prompt": chat}
response = requests.post(f"{base_uri}/api/agent/{agent_name}/chat", json=data)
assert response.status_code == 200, response.json()
pprint(response.json())
```

```
{'response': 'test_suite_agent: The capital of France is Paris.'}
```

[7]:
```
# Test POST /api/agent/{agent_name}/smartchat/{shots}
# SmartChat with agent
data = {"prompt": chat}
response = requests.post(f"{base_uri}/api/agent/{agent_name}/smartchat/
  ↪{shots}", json=data)
assert response.status_code == 200, response.json()
pprint(response.json())
```

```
{'response': 'The capital of France is Paris, known for its iconic landmarks '
             'and cultural significance. With a population of over 2 million '
             'people, Paris is not only the political center of France but '
             'also a major cultural hub, home to world-renowned museums, art '
             'galleries, and fashion houses.'}
```

0.5 Tasks

[ ]:
```
# Test POST /api/agent/{agent_name}/task
# Create a task for the agent
data = {"objective": task}
response = requests.post(f"{base_uri}/api/agent/{agent_name}/task", json=data)
print(response.json())
```

[ ]:
```
# Test GET /api/agent/{agent_name}/task/status
# Get the agent's task status
response = requests.get(f"{base_uri}/api/agent/{agent_name}/task/status")
pprint(response.json())
```

[ ]:
```
# Test GET /api/agent/{agent_name}/task
# Get the agent's task
response = requests.get(f"{base_uri}/api/agent/{agent_name}/task")
pprint(response.json())
```

0.6 Agent teardown

[ ]:
```
# Test DELETE /api/agent/{agent_name}
# Delete the agent
response = requests.delete(f"{base_uri}/api/agent/{agent_name}")
pprint(response.json())
```