OpenAI Integration

Introduction

FLUX provides integration with the OpenAI API, enabling you to create elements that leverage artificial intelligence capabilities within your workflows.

Overview

The OpenAI integration is implemented through two key components:

- **OpenAI API Profile** – Provides the configuration and credentials needed to connect with OpenAI services
- **OpenAI Element Group** – Contains specialized elements that interact with the OpenAI API to send questions and receive AI-generated responses

You can view the profile configuration here: OpenAI API Profile Example

Available OpenAI Elements

FLUX includes several pre-built elements that demonstrate OpenAI integration capabilities:

Python Helper

- **Purpose**: Generates Python code to extract variables from files
- **Use Case**: Used by other models to automatically create Python code that extracts variables from test data into a dictionary format
- **Link**: View Element

TTP Helper

- **Purpose**: Generates TTP (Template Text Parser) models from test data
- **Use Case**: Used by other models to create TTP templates capable of extracting variables from test data into a dictionary format

- **Link**: View Element

JSON Query

- **Purpose**: Performs complex queries on JSON data structures
- **Use Case**: Used by PublicElementConfig to execute advanced JSON queries
- **Link**: View Element

Using OpenAI Elements in Your Code

To integrate OpenAI elements into your code, follow these steps:

1. **Locate the Element** – Search for the Element that contains your desired OpenAI model
2. **Invoke the Function** – Call `make_question_to_openai_element()` with your question as the second parameter
3. **Receive the Response** – The function returns the OpenAI-generated answer

How It Works

The `make_question_to_openai_element()` function executes an OpenAI API call using:

- The base model context
- Previous conversation history (if any)
- Your current question

The function then returns the AI-generated response for use in your application.

Example Implementation

```
template = None
element = Element.objects.filter(name=f"openai-helper-{model.klass}").first()      You, 3 months a

if not element:
    return 'No openai-helper detected for this model klass.'

last_answer, error = make_question_to_openai_element(element, data)

if last_answer and not error:
    if model.klass == 'pyps':
        last_answer.value = last_answer.value.replace('"\n"', '"\\n"').replace("'\n'", "'\\n'")
        last_answer.save()
    template = last_answer.value
elif error:
    return last_answer
```

Additional Resources

For detailed information on creating custom models with OpenAI integration capabilities, refer to the Models documentation page.

Revision #2
Created 2026-02-13 23:06:43 UTC by ipena@zequenze.com
Updated 2026-03-05 22:30:35 UTC by ipena@zequenze.com