

Run ID,Agent Name,Task,Result,Timestamp

9188734b-1bf0-4896-beaa-59387304f822,Python API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Sure! Below is a Python code snippet that demonstrates how to send an API request to OpenAI's API using the `requests` library, with error handling included. Make sure to replace `YOUR\_API\_KEY` with your actual OpenAI API key.

### Python Code Snippet

```
```python
```

```
import requests
```

```
def call_openai_api(prompt):
```

```
    api_key = 'YOUR_API_KEY' # Replace with your OpenAI API key
```

```
    url = 'https://api.openai.com/v1/chat/completions'
```

```
    headers = {
```

```
        'Authorization': f'Bearer {api_key}',
```

```
        'Content-Type': 'application/json'
```

```
    }
```

```
    data = {
```

```
        "model": "gpt-3.5-turbo", # Specify the model you want to use
```

```
        "messages": [
```

```
            {"role": "user", "content": prompt}
```

```
        ]
```

```

}

try:

    response = requests.post(url, headers=headers, json=data)

    response.raise_for_status() # Raise an error for bad responses

    result = response.json()

    return result

except requests.exceptions.HTTPError as http_err:

    print(f'HTTP error occurred: {http_err}') # Handle HTTP errors

except Exception as err:

    print(f'Other error occurred: {err}') # Handle other errors


if __name__ == "__main__":

    prompt = "What are the benefits of using AI in healthcare?"

    response = call_openai_api(prompt)

    print(response)

...

```

### ### Instructions to Run

1. Install the `requests` library if you haven't already:

```

```bash

pip install requests

...

```

2. Save the code snippet in a `.py` file.

3. Update the `YOUR\_API\_KEY` placeholder with your OpenAI API key.

4. Run the script using:

```
```bash  
  
python your_script_name.py  
  
```
```

### ### Explanation

- The script defines a function `call\_openai\_api` which takes a prompt as input.
- It constructs the necessary headers and payload.
- The function makes a POST request to the OpenAI API.
- Error handling is implemented to catch and print any HTTP or other errors that may occur during the API request.
- Finally, it demonstrates how to call the function with an example prompt in the main block.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,JavaScript API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Here's a JavaScript code snippet that sends an API request to OpenAI's API using the Fetch API. This code includes error handling and is ready to run in a Node.js environment. You'll need to install the `node-fetch` package to use the Fetch API in Node.js.

### ### Step 1: Install `node-fetch`

First, make sure to install the `node-fetch` library by running:

```
```bash
```

```
npm install node-fetch
```

```
```
```

### ### Step 2: JavaScript Code Snippet

```
```javascript
```

```
const fetch = require('node-fetch');
```

```
const API_KEY = 'YOUR_OPENAI_API_KEY'; // Replace with your OpenAI API key
```

```
const API_URL = 'https://api.openai.com/v1/chat/completions';
```

```
async function callOpenAI(prompt) {
```

```
  const requestBody = {
```

```
    model: 'gpt-3.5-turbo', // Adjust the model as needed
```

```
    messages: [{ role: 'user', content: prompt }],
```

```
    max_tokens: 100, // You can adjust the token limit
```

```
  };
```

```
  try {
```

```
    const response = await fetch(API_URL, {
```

```
      method: 'POST',
```

```
      headers: {
```

```
        'Content-Type': 'application/json',
```

```
        'Authorization': `Bearer ${API_KEY}`,
```

```

    },
    body: JSON.stringify(requestBody),
  });

  if (!response.ok) {
    const errorResponse = await response.json();
    throw new Error(`Error: ${errorResponse.error.message}`);
  }

  const data = await response.json();
  console.log('Response from OpenAI:', data.choices[0].message.content);
} catch (error) {
  console.error('Error occurred while calling OpenAI:', error.message);
}
}

// Example usage

const userPrompt = 'What is the capital of France?';
callOpenAI(userPrompt);
...

```

### ### Explanation:

1. **Dependencies**: The code uses the `node-fetch` library to make HTTP requests.
2. **API Key**: Replace `'YOUR_OPENAI_API_KEY'` with your actual OpenAI API key.
3. **API URL**: The code uses the endpoint for chat completions.

4. **Request Body**: The body of the request includes the model to use, the user's message, and the maximum number of tokens to return.
5. **Error Handling**: The code checks if the response is not OK and throws an error with the message from the API, which is then caught in the catch block.
6. **Output**: The response from OpenAI is logged to the console.

Make sure to run this code in a secure environment, as it contains sensitive API credentials.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,Go API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Below is a Go code snippet that sends an API request to OpenAI's API. This code includes necessary libraries, error handling, and is ready to run.

```
```go
```

```
package main
```

```
import (
```

```
    "bytes"
```

```
    "encoding/json"
```

```
    "fmt"
```

```
    "io/ioutil"
```

```
    "net/http"
```

```
    "os"
```

```
)
```

```
const apiURL = "https://api.openai.com/v1/chat/completions"
```

```
type RequestBody struct {  
    Model    string `json:"model"`  
    Messages []Message `json:"messages"`  
}
```

```
type Message struct {  
    Role    string `json:"role"`  
    Content string `json:"content"`  
}
```

```
type ResponseBody struct {  
    ID      string `json:"id"`  
    Object  string `json:"object"`  
    Choices []Choice `json:"choices"`  
}
```

```
type Choice struct {  
    Message Message `json:"message"`  
    Index   int    `json:"index"`  
}
```

```
func main() {  
    apiKey := os.Getenv("OPENAI_API_KEY") // Ensure you set your API key in the environment  
    variable  
    if apiKey == "" {
```

```
fmt.Println("Error: OPENAI_API_KEY environment variable is not set.")
```

```
return
```

```
}
```

```
// Create the request body
```

```
requestBody := RequestBody{
```

```
Model: "gpt-3.5-turbo",
```

```
Messages: []Message{
```

```
{Role: "user", Content: "Hello, how can I use OpenAI's API?"},
```

```
},
```

```
}
```

```
// Convert request body to JSON
```

```
jsonData, err := json.Marshal(requestBody)
```

```
if err != nil {
```

```
fmt.Printf("Error marshaling request body: %v\n", err)
```

```
return
```

```
}
```

```
// Create the HTTP request
```

```
req, err := http.NewRequest("POST", apiURL, bytes.NewBuffer(jsonData))
```

```
if err != nil {
```

```
fmt.Printf("Error creating request: %v\n", err)
```

```
return
```

```
}
```



```
// Set the necessary headers

req.Header.Set("Content-Type", "application/json")

req.Header.Set("Authorization", "Bearer "+apiKey)


// Send the request

client := &http.Client{}

resp, err := client.Do(req)

if err != nil {

    fmt.Printf("Error sending request: %v\n", err)

    return

}

defer resp.Body.Close()


// Check the response status

if resp.StatusCode != http.StatusOK {

    body, _ := ioutil.ReadAll(resp.Body)

    fmt.Printf("Error: received status code %d, response: %s\n", resp.StatusCode, body)

    return

}


// Parse the response body

var responseBody ResponseBody

if err := json.NewDecoder(resp.Body).Decode(&responseBody); err != nil {

    fmt.Printf("Error decoding response body: %v\n", err)

    return

}
```

```
// Print the response

for _, choice := range responseBody.Choices {

    fmt.Printf("Response: %s\n", choice.Message.Content)

}

}

...

```

### ### Instructions to Run the Code:

1. **\*\*Set Up Go Environment\*\***: Make sure you have Go installed on your machine. You can download it from [golang.org](https://golang.org/).

2. **\*\*Set API Key\*\***: Before running the code, set your OpenAI API key in the environment variable `OPENAI\_API\_KEY`. You can do this in your terminal:

```
```bash

export OPENAI_API_KEY=""your_api_key_here""

...

```

3. **\*\*Run the Code\*\***: Save the code in a file named `main.go`, navigate to the directory containing the file in your terminal, and run:

```
```bash

go run main.go

...

```

This code snippet will send a request to OpenAI's API and print the response. Make sure to handle your API key securely and avoid exposing it in public repositories.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,Rust API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Here's a Rust code snippet that sends an API request to OpenAI's API. This example uses the `reqwest` library for making HTTP requests and `serde\_json` for handling JSON data. Make sure to include these dependencies in your `Cargo.toml` file.

### Cargo.toml

```
``toml
```

```
[dependencies]
```

```
reqwest = { version = "0.11", features = ["json"] }
```

```
tokio = { version = "1", features = ["full"] }
```

```
serde = { version = "1.0", features = ["derive"] }
```

```
serde_json = "1.0"
```

```
``
```

### Rust Code Snippet

```
``rust
```

```
use reqwest::Client;
```

```
use serde::{Deserialize, Serialize};
```

```
use std::error::Error;
```

```
#[derive(Serialize)]
```

```
struct OpenAIRequest {
```

```
    model: String,
```

```
    prompt: String,
```

```
    max_tokens: u32,
```

```
}
```

```
#[derive(Deserialize)]
```

```
struct OpenAIResponse {
```

```
    id: String,
```

```
    choices: Vec<Choice>,
```

```
}
```

```
#[derive(Deserialize)]
```

```
struct Choice {
```

```
    text: String,
```

```
}
```

```
#[tokio::main]
```

```
async fn main() -> Result<(), Box<dyn Error>> {
```

```
    let api_key = ""YOUR_API_KEY""; // Replace with your OpenAI API key
```

```
    let client = Client::new();
```

```
    let request_body = OpenAIRequest {
```

```
        model: ""text-davinci-003"".to_string(),
```

```
        prompt: ""Once upon a time"".to_string(),
```

```
        max_tokens: 50,
```

```
    };
```

```
    let response = client
```

```
        .post("""https://api.openai.com/v1/completions""")
```

```

.header("Authorization", format!("Bearer {}", api_key))

.json(&request_body)

.send()

.await?;

if response.status().is_success() {

    let response_body: OpenAIResponse = response.json().await?;

    println!("Response: {:?}", response_body.choices[0].text);

} else {

    eprintln!("Error: {}", response.status());

    let error_body = response.text().await?;

    eprintln!("Error details: {}", error_body);

}

Ok(())

}

...

```

### ### Explanation

1. **Dependencies**: The code uses `reqwest` for HTTP requests, `tokio` for asynchronous runtime, and `serde`/`serde_json` for serialization and deserialization of JSON data.
2. **Structs**: `OpenAIRequest` is used to structure the request body, while `OpenAIResponse` and `Choice` are used to parse the response from the API.
3. **Main Function**: The `main` function is asynchronous and handles the API request. It constructs the request, sends it, and processes the response.
4. **Error Handling**: The code checks if the response is successful and prints the result or error

details accordingly.

### ### Note

- Replace `""YOUR_API_KEY""` with your actual OpenAI API key.
- Ensure you have the necessary permissions and that your API key is kept secure.",2024-08-24T18:33:54.859277  
9188734b-1bf0-4896-beaa-59387304f822,Java API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.",  
"Here's a Java code snippet that sends an API request to OpenAI's API. This example uses the `URLConnection` class for making the HTTP request, and includes necessary error handling.

Make sure to include the following libraries in your project:

1. `java.net.HttpURLConnection`
2. `java.io.BufferedReader`
3. `java.io.InputStreamReader`
4. `java.io.OutputStream`
5. `java.nio.charset.StandardCharsets`

Here's the code:

```
```java
import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.OutputStream;
```

```

import java.net.HttpURLConnection;

import java.net.URL;


public class OpenAIAPIRequest {

    private static final String API_URL = "https://api.openai.com/v1/your-endpoint"; // Replace with
your actual endpoint

    private static final String API_KEY = "your-api-key"; // Replace with your actual API key


    public static void main(String[] args) {

        try {

            // Create the URL object

            URL url = new URL(API_URL);

            HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();


            // Set the request method and headers

            httpURLConnection.setRequestMethod("POST");

            httpURLConnection.setRequestProperty("Content-Type", "application/json");

            httpURLConnection.setRequestProperty("Authorization", "Bearer " + API_KEY);

            httpURLConnection.setDoOutput(true);


            // Create the JSON payload

            String jsonString = "{ \"model\": \"gpt-3.5-turbo\", \"messages\": [{ \"role\": \"user\", \"content\": \"Hello!\" }]}";


            // Write the payload to the output stream

            try (OutputStream os = httpURLConnection.getOutputStream()) {

```

```

        byte[] input = jsonString.getBytes(StandardCharsets.UTF_8);

        os.write(input, 0, input.length);
    }

    // Get the response code

    int responseCode = httpURLConnection.getResponseCode();

    System.out.println("Response Code: " + responseCode);

    // Read the response

        try (BufferedReader br = new BufferedReader(new
InputStreamReader(httpURLConnection.getInputStream(), StandardCharsets.UTF_8))) {

            StringBuilder response = new StringBuilder();

            String responseLine;

            while ((responseLine = br.readLine()) != null) {

                response.append(responseLine.trim());

            }

            System.out.println("Response: " + response.toString());

        }

    } catch (Exception e) {

        e.printStackTrace();

    }

}

}

...

```



### ### Instructions:

1. Replace `your-endpoint` with the actual endpoint you want to call from OpenAI's API.
2. Replace `your-api-key` with your actual OpenAI API key.
3. Compile and run the Java program.

### ### Notes:

- Ensure that you handle sensitive information like API keys securely in your application.
- You may want to implement more sophisticated error handling depending on your application's needs.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,C# API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Here's a C# code snippet that sends an API request to OpenAI's API. This example uses `HttpClient` to make the request and includes error handling.

Make sure to install the `System.Net.Http` library if it's not already available in your project.

```
```csharp
```

```
using System;
```

```
using System.Net.Http;
```

```
using System.Net.Http.Headers;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
using Newtonsoft.Json;
```

```
namespace OpenAIExample
```

```
{
```

```
class Program
{
    private static readonly HttpClient client = new HttpClient();

    static async Task Main(string[] args)
    {
        string apiKey = "YOUR_API_KEY"; // Replace with your OpenAI API key
        string apiUrl = "https://api.openai.com/v1/chat/completions"; // Example endpoint

        var requestBody = new
        {
            model = "gpt-3.5-turbo", // Specify the model you want to use
            messages = new[]
            {
                new { role = "user", content = "Hello, how can I use OpenAI's API?" }
            }
        };

        try
        {
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
apiKey);

            var json = JsonConvert.SerializeObject(requestBody);
            var content = new StringContent(json, Encoding.UTF8, "application/json");

            HttpResponseMessage response = await client.PostAsync(apiUrl, content);
```

```
response.EnsureSuccessStatusCode()); // Throw if not a success code.
```

```
string responseBody = await response.Content.ReadAsStringAsync();
```

```
Console.WriteLine("Response from OpenAI API:");
```

```
Console.WriteLine(responseBody);
```

```
}
```

```
catch (HttpRequestException e)
```

```
{
```

```
    Console.WriteLine("Request error: " + e.Message);
```

```
}
```

```
catch (Exception e)
```

```
{
```

```
    Console.WriteLine("An error occurred: " + e.Message);
```

```
}
```

```
}
```

```
}
```

```
}
```

```
...
```

### Instructions:

1. Replace `YOUR\_API\_KEY` with your actual OpenAI API key.
2. Ensure you have the `Newtonsoft.Json` package installed for JSON serialization. You can install it via NuGet Package Manager with the command:

```
...
```

```
Install-Package Newtonsoft.Json
```

```
...
```

3. Run the program, and it will send a request to the OpenAI API and print the response.

### Note:

Make sure you have the necessary permissions and that your API key is kept secure.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,Ruby API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Here's a Ruby code snippet that sends an API request to OpenAI's API, including necessary libraries and error handling. Make sure to replace `YOUR\_API\_KEY` with your actual OpenAI API key.

```
```ruby
```

```
require 'net/http'
```

```
require 'uri'
```

```
require 'json'
```

```
def send_openai_request(prompt)
```

```
  api_key = 'YOUR_API_KEY'
```

```
  uri = URI.parse("https://api.openai.com/v1/engines/davinci-codex/completions")
```

```
  request = Net::HTTP::Post.new(uri)
```

```
  request.content_type = "application/json"
```

```
  request["Authorization"] = "Bearer #{api_key}"
```

```
  request.body = JSON.generate({
```

```
    prompt: prompt,
```

```
max_tokens: 100,  
n: 1,  
stop: nil,  
temperature: 0.5  
})
```

```
response = Net::HTTP.start(uri.hostname, uri.port, use_ssl: true) do |http|  
  http.request(request)  
end
```

```
case response  
when Net::HTTPSuccess  
  JSON.parse(response.body)  
else  
  puts "Error: #{response.code} - #{response.message}"  
  puts response.body  
  nil  
end  
end
```

# Example usage

```
response = send_openai_request("Once upon a time")  
if response  
  puts "Response from OpenAI:"  
  puts response  
end
```

### ### Explanation:

1. **Libraries**: The script uses ``net/http`` for making HTTP requests, ``uri`` for URI parsing, and ``json`` for handling JSON data.
2. **Function**: The ``send_openai_request`` function takes a ``prompt`` as an input and sends a request to OpenAI's completions endpoint.
3. **Request Setup**: It sets the necessary headers, including the Content-Type and Authorization header with the Bearer token.
4. **Error Handling**: The code checks the response status. If its successful, it parses and returns the response body; otherwise, it prints an error message.
5. **Example Usage**: At the end, an example call to the function demonstrates how to use it.

Make sure to run this code in an environment that has internet access and where you can install Ruby libraries if needed.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,PHP API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Here's a PHP code snippet that sends an API request to OpenAI's API, including necessary libraries, error handling, and is ready to run.

```
```php
```

```
<?php
```

```
// Include the necessary libraries
```

```
require 'vendor/autoload.php'; // Make sure you have installed Guzzle via Composer
```

```
use GuzzleHttp\Client;
```

```
use GuzzleHttp\Exception\RequestException;
```

```
function callOpenAIAPI($apiKey, $prompt) {
```

```
    // Create a new Guzzle client
```

```
    $client = new Client();
```

```
    // Define the API endpoint and headers
```

```
    $url = 'https://api.openai.com/v1/engines/davinci-codex/completions'; // Change engine as needed
```

```
    $headers = [
```

```
        'Authorization' => 'Bearer ' . $apiKey,
```

```
        'Content-Type' => 'application/json',
```

```
    ];
```

```
    // Prepare the request body
```

```
    $body = json_encode([
```

```
        'prompt' => $prompt,
```

```
        'max_tokens' => 100, // Adjust as needed
```

```
        'temperature' => 0.7, // Adjust as needed
```

```
    ]);
```

```
    try {
```

```
        // Send the request to the OpenAI API
```

```
        $response = $client->post($url, [
```

```
            'headers' => $headers,
```

```
            'body' => $body,
```

```

]);

// Get the response body

$responseBody = json_decode($response->getBody(), true);

return $responseBody;

} catch (RequestException $e) {

    // Handle the error

    echo 'API Request failed: ' . $e->getMessage();

    if ($e->hasResponse()) {

        echo ' Response: ' . $e->getResponse()->getBody();

    }

    return null;

}

}

// Example usage

$apiKey = 'YOUR_OPENAI_API_KEY'; // Replace with your OpenAI API key

$prompt = 'Once upon a time in a land far away,';

$response = callOpenAIAPI($apiKey, $prompt);

if ($response) {

    echo 'Response from OpenAI: ' . print_r($response, true);

}

?>

```



```

### ### Instructions to Run the Code:

1. **\*\*Install Guzzle\*\***: Make sure you have Composer installed, then run the following command in your project directory to install Guzzle:

```
```bash

composer require guzzlehttp/guzzle

```
```

2. **\*\*Set Your API Key\*\***: Replace ``YOUR_OPENAI_API_KEY`` in the code with your actual OpenAI API key.

3. **\*\*Run the Script\*\***: Save the code in a PHP file (e.g., ``openai_request.php``) and run it in your PHP environment.

### ### Note:

- Ensure that you have the necessary permissions and the correct API endpoint based on the OpenAI model you intend to use.

- Adjust ``max_tokens`` and ``temperature`` parameters as per your requirements.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,Swift API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Here's a Swift code snippet that sends an API request to OpenAI's API. This example uses ``URLSession`` for networking and includes error handling.

```
```swift
```

```
import Foundation
```

```
// Define the API endpoint and your API key
```

```
let apiKey = ""YOUR_API_KEY""
```

```
let url = URL(string: "https://api.openai.com/v1/chat/completions")!
```

```
// Create the request
```

```
var request = URLRequest(url: url)
```

```
request.httpMethod = "POST"
```

```
request.setValue("application/json", forHTTPHeaderField: "Content-Type")
```

```
request.setValue("Bearer \(apiKey)", forHTTPHeaderField: "Authorization")
```

```
// Define the request body
```

```
let requestBody: [String: Any] = [
```

```
    "model": "gpt-3.5-turbo",
```

```
    "messages": [
```

```
        ["role": "user", "content": "Hello, how can I use OpenAI's API?"]
```

```
    ]
```

```
]
```

```
do {
```

```
    request.httpBody = try JSONSerialization.data(withJSONObject: requestBody, options: [])
```

```
} catch {
```

```
    print("Error serializing JSON: \(error)")
```

```
    exit(1)
```

```
}
```

```
// Send the request
```

```
let task = URLSession.shared.dataTask(with: request) { data, response, error in
```

```
    // Handle error
```

```
    if let error = error {
```

```
        print("""Error sending request: \(error)""")
```

```
        return
```

```
    }
```

```
    // Check for a valid response
```

```
        guard let httpResponse = response as? HTTPURLResponse,
```

```
(200...299).contains(httpResponse.statusCode) else {
```

```
    print("""Server error: \(response!)""")
```

```
    return
```

```
}
```

```
// Parse the response data
```

```
if let data = data {
```

```
    do {
```

```
        if let jsonResponse = try JSONSerialization.jsonObject(with: data, options: []) as? [String:
```

```
Any] {
```

```
            print("""Response JSON: \(jsonResponse)""")
```

```
        }
```

```
    } catch {
```

```
        print("""Error parsing JSON: \(error)""")
```

```
    }
```

```
}  
  
}
```

```
// Start the request
```

```
task.resume()
```

```
// Keep the program running to wait for the async response
```

```
RunLoop.current.run()
```

```
...
```

### Instructions:

1. Replace ``YOUR\_API\_KEY`` with your actual OpenAI API key.
2. You can modify the `messages` array in the `requestBody` to change the input to the model.
3. Make sure to run this code in an environment that supports Swift, such as Xcode or a Swift playground.

### Note:

- This code uses the `gpt-3.5-turbo` model. You can change the model name based on your requirements.

- Ensure you handle your API key securely and do not expose it in public repositories.",2024-08-24T18:33:54.859277

9188734b-1bf0-4896-beaa-59387304f822,Kotlin API Agent,"Generate code snippets for sending an API request to OpenAI's API in Python, JavaScript, Go, Rust, Java, C#, Ruby, PHP, Swift, and Kotlin.", "Here's a Kotlin code snippet that sends an API request to OpenAI's API. This example uses the `OkHttp` library for making HTTP requests and includes error handling.

Make sure to include the necessary dependencies in your `build.gradle` file:

```
```groovy
dependencies {
    implementation("com.squareup.okhttp3:okhttp:4.9.3")
    implementation("com.squareup.moshi:moshi:1.12.0")
    implementation("com.squareup.moshi:moshi-kotlin:1.12.0")
}
```
```

Now, heres the Kotlin code snippet:

```
```kotlin
import okhttp3.*
import com.squareup.moshi.Moshi
import com.squareup.moshi.kotlin.reflect.KotlinJsonAdapterFactory
import java.io.IOException

// Define your API key and endpoint
const val API_KEY = "YOUR_API_KEY"
const val API_URL = "https://api.openai.com/v1/chat/completions"

data class ChatCompletionRequest(
    val model: String,
    val messages: List<Message>
)
```
```

```
data class Message(  
    val role: String,  
    val content: String  
)
```

```
data class ChatCompletionResponse(  
    val id: String,  
    val choices: List<Choice>  
)
```

```
data class Choice(  
    val message: Message  
)
```

```
fun main() {  
    val client = OkHttpClient()  
    val moshi = Moshi.Builder().add(KotlinJsonAdapterFactory()).build()  
    val jsonAdapter = moshi.adapter(ChatCompletionRequest::class.java)  
  
    // Create the request body  
    val messages = listOf(  
        Message(role = "user", content = "Hello, how can I help you?")  
    )  
  
    val requestBody = jsonAdapter.toJson(ChatCompletionRequest(model = "gpt-3.5-turbo",  
messages = messages))
```

```
val request = Request.Builder()

    .url(API_URL)

    .post(RequestBody.create(MediaType.parse("application/json"), requestBody))

    .addHeader("Authorization", "Bearer $API_KEY")

    .addHeader("Content-Type", "application/json")

    .build()
```

```
client.newCall(request).enqueue(object : Callback {

    override fun onFailure(call: Call, e: IOException) {

        println("Request failed: ${e.message}")

    }

})
```

```
override fun onResponse(call: Call, response: Response) {

    if (!response.isSuccessful) {

        println("Unexpected code: ${response.code()}")

        println("Response body: ${response.body()?.string()}")

        return

    }

}
```

```
response.body()?.string()?.let { responseBody ->

    val responseAdapter = moshi.adapter(ChatCompletionResponse::class.java)

    val chatResponse = responseAdapter.fromJson(responseBody)

    println("Response: $chatResponse")

}

}
```

```
    })  
}  
...
```

### ### Explanation:

1. **Dependencies**: The code uses `OkHttp` for HTTP requests and `Moshi` for JSON serialization/deserialization.
2. **Data Classes**: `ChatCompletionRequest`, `Message`, `ChatCompletionResponse`, and `Choice` are defined to structure the request and response.
3. **API Key and URL**: Replace `YOUR_API_KEY` with your actual OpenAI API key.
4. **Making the Request**: The request is built and sent asynchronously. The response is handled in the `onResponse` and `onFailure` methods.
5. **Error Handling**: The code checks for successful responses and prints error messages if the request fails.

### ### Note:

Make sure to handle your API key securely and not expose it in public repositories.",2024-08-24T18:33:54.859277