OpenAI API Integration with Spring Boot in Eclipse using Maven

Introduction

This document walks through the process of integrating OpenAI API within a Spring Boot project managed by Maven in the Eclipse IDE. This setup will allow you to interact with OpenAI's machine learning models using RESTful web services in a Spring Boot application.

Prerequisites

- Eclipse IDE with Maven and Spring Boot installed.
- An active OpenAI account with API keys.
- Basic knowledge of Spring Boot framework.

Step by Step Implementation

Step 1: Create a Spring Boot Maven Project in Eclipse

1. Launch **Eclipse IDE**.
2. Navigate to **File** > **New** > **Spring Starter Project**.
3. Fill in the necessary details like **Name**, **Group**, **Artifact**, etc., and click **Next**.
4. Select the required Spring Boot dependencies (Web, Rest Repositories), then click **Finish**.

Step 2: Update application.properties or application.yml

Update your application.properties or application.yml file with OpenAI API credentials and configurations:

propertiesCopy code

openai.api.key=your-api-key-here

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

openai.model=gpt-3.5-turbo-0613

Step 3: Configure RestTemplate Bean

In OpenAIConfig class, a RestTemplate bean is created with an interceptor to add the Authorization header for every request made to OpenAI API.

javaCopy code

@Configuration

```
public class OpenAIConfig {

    @Value("${openai.api.key}")
    private String openaiApiKey;

    @Bean
    public RestTemplate template(){
        RestTemplate restTemplate=new RestTemplate();
        restTemplate.getInterceptors().add((request, body, execution) -> {
            request.getHeaders().add("Authorization", "Bearer " + openaiApiKey);
            return execution.execute(request, body);
        });
        return restTemplate;
    }

}
```

Step 4: Implement OpenAI API Calls

In the CustomBotController class, implement methods to handle various endpoints which interact with the OpenAI API using RestTemplate.

javaCopy code

```
@RestController
@RequestMapping("/bot")
public class CustomBotController {

    // ... other fields ...

    @Autowired
    private RestTemplate template;

    @GetMapping("/chat")
    public String chat(@RequestParam("prompt") String prompt){
        ChatGPTRequest request = new ChatGPTRequest(model, prompt);
```

```
        ChatGptResponse chatGptResponse = template.postForObject(apiURL, request,
ChatGptResponse.class);
        return chatGptResponse.getChoices().get(0).getMessage().getContent();
    }

    // ... other methods ...
}
```

Step 5: Run the Application

5. Right-click on the project.
6. Select **Run As** > **Spring Boot App**.
7. Once the application is running, use a tool like Postman to send requests to your endpoints or access them through the browser.

Step 6: Test the Implementation

Create or use existing front-end pages to interact with the /bot/chat endpoint. Pass in the necessary parameters and observe the responses from OpenAI API being returned and displayed in your application.

httpCopy code

```
GET /bot/chat?prompt=Translate the following English text to French: 'Hello, how are you?'
```

Conclusion

With the outlined steps, you have successfully integrated the OpenAI API within a Spring Boot project in Eclipse managed by Maven. This setup allows seamless interaction with OpenAI's models, providing a basis for extending the application's capabilities with machine learning services.