

《OpenAI 代码自动评审组件》- 第5节-学习记录

来自：码农会锁

 爱海贼的无处不在

2024年08月04日 23:48



学习系列文章目录：

- 1、《OpenAI 代码自动评审组件》- 第1节-学习记录：https://articles.zsxq.com/id_bpapuj05ur03.html
- 2、《OpenAI 代码自动评审组件》- 第2节-学习记录：https://articles.zsxq.com/id_uokpg7mvt1uc.html
- 3、《OpenAI 代码自动评审组件》- 第3节-学习记录：https://articles.zsxq.com/id_bi51vtdhpp36.html
- 4、《OpenAI 代码自动评审组件》- 第4节-学习记录：https://articles.zsxq.com/id_mw3srnic84z7.html

一、概述

本节作者开始讲解如何将AI代码评审的结果进行数据持久化，作者这里采用的是基于独立Git仓库存储代码评审结果的实现方案，作者先开始进行了代码的编写，然后再Github上创建了个人的Github Token访问令牌，然后作者在Github的action的工作流文件中设置了变量和环境的映射信息，从而串联起来将SDK的评审结果写入到代码仓库。

二、正文

1、首先作者使用jgit第三方工具包，实现了Java代码操作Git仓库的能力，这个是非常重要的，以前我们只知道用IDEA的客户端或者sourceTree客户端来操作Git，在这一节中作者，使用了这个工具包完成了对Git仓库的操作，帮助我们打开了新世界的大门，作者本节实现的核心方法如下：

```
public static String writeLog(String token, String result) throws Exception {
    // 克隆Github仓库到本地目录 "repo"
    Git git = Git.cloneRepository()
        .setURI("xxx.git")
        .setDirectory(new File("repo"))
        .setCredentialsProvider(new UsernamePasswordCredentialsProvider(token, ""))
        .call();

    // 获取当前日期，并创建相应的文件夹
    String dateFolderName = new SimpleDateFormat("yyyy-MM-dd").format(new Date());
    File dateFolder = new File("repo/" + dateFolderName);
    if (!dateFolder.exists()) {
        dateFolder.mkdirs(); // 如果文件夹不存在，则创建
    }

    // 生成随机文件名，并创建新文件
    String fileName = generateRandomString(12) + ".md";
    File newFile = new File(dateFolder, fileName);
    try (FileWriter writer = new FileWriter(newFile)) {
        writer.write(result); // 将结果写入文件
    }

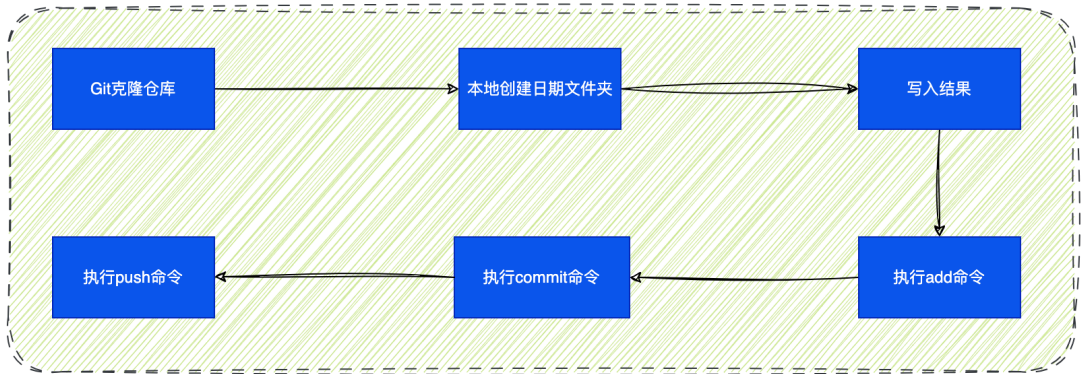
    // 将新文件添加到Git仓库中
    git.add().addFilepattern(dateFolderName + "/" + fileName).call();
    // 提交变更
    git.commit().setMessage("Add new file via GitHub Actions").call();
    // 推送到远程仓库
    git.push().setCredentialsProvider(new UsernamePasswordCredentialsProvider(token, "")).call();

    // 打印推送成功的消息
    System.out.println("Changes have been pushed to the repository.");

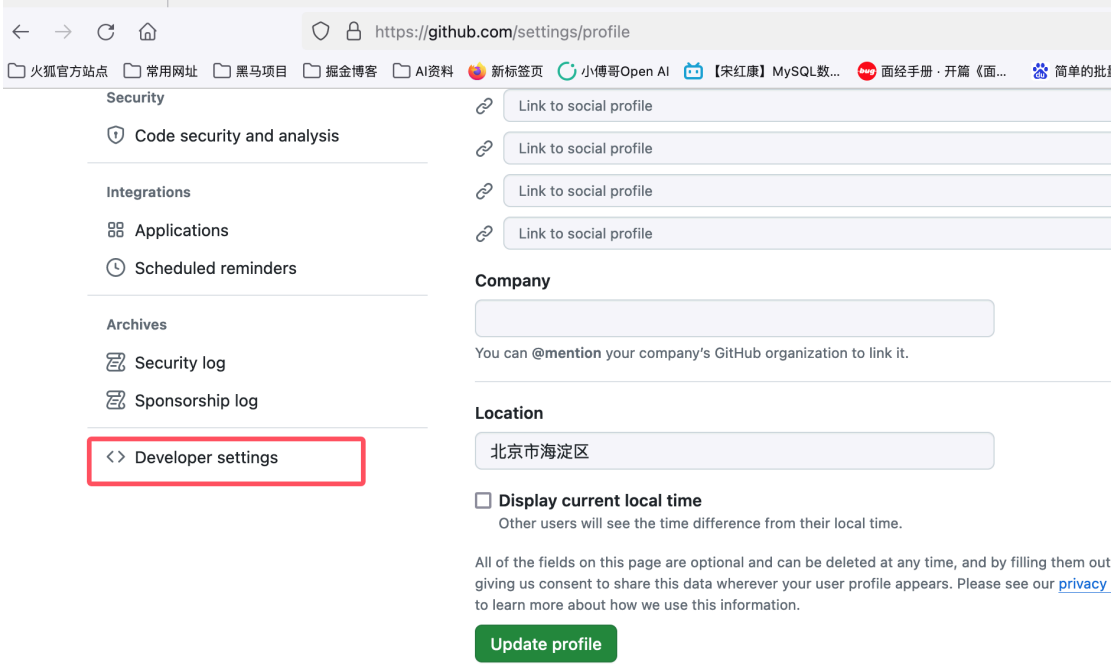
    // 返回新文件在Github仓库中的URL
    return "https://github.com/authorName/ProjectName/blob/master/" + dateFolderName + "/" + fileName;
}

// 生成指定长度的随机字符串
private static String generateRandomString(int length) {
    String characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    Random random = new Random();
    StringBuilder sb = new StringBuilder(length);
    for (int i = 0; i < length; i++) {
        sb.append(characters.charAt(random.nextInt(characters.length())));
    }
    return sb.toString();
}
```

可以非常清晰的看到这个是一个操作Git仓库的完整过程，作者通过Java代码表达了出来，因此，我们可以梳理出来业务流程图如下所示：

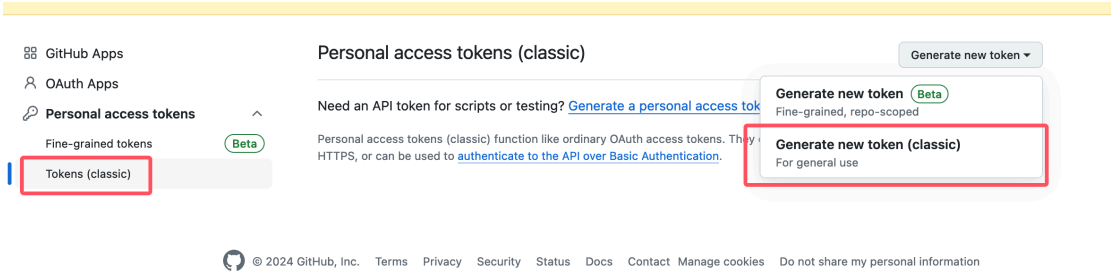


2、作者操作完成代码后，在Git仓库中创建了新的仓库，同时在SDK项目中更新了Action的脚本内容，增加上了用户 token的环境变量的配置，这个内容在如下位置：

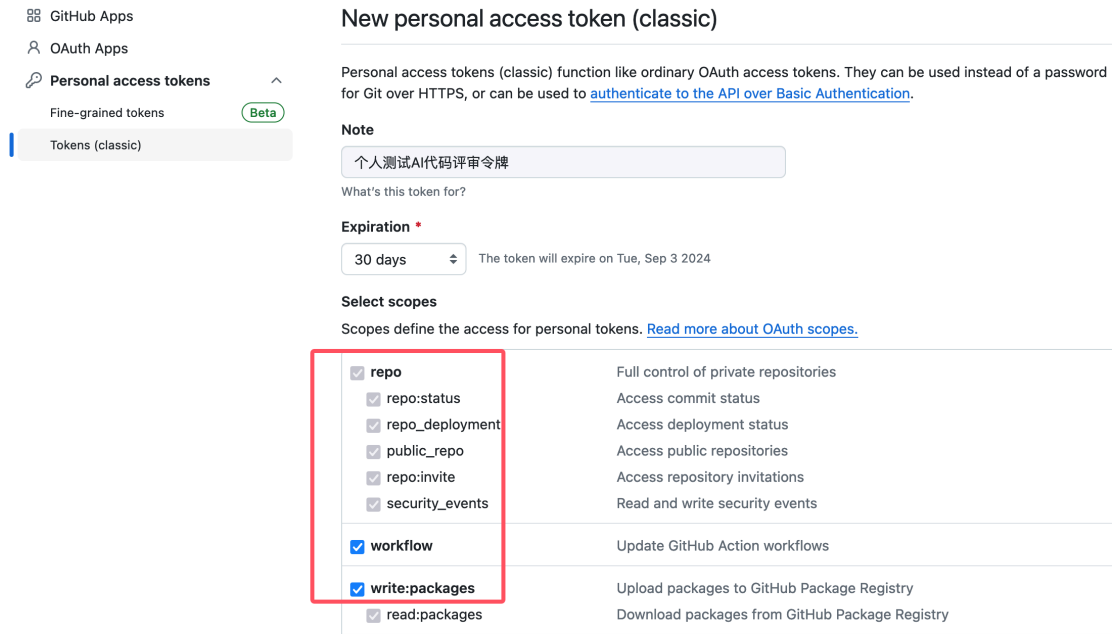


Contributions & activity

然后选择个人的令牌，生成后，将令牌信息拷贝到项目下的环境变量配置中：



在生成的界面，输入令牌的名称和API的权限：



设置完成后，在代码中的工作流文件也配置好变量取值，然后提交本次改动到仓库，就可以看到将评审结果接入到了新的仓库了：



三、本节知识总结

本节作者使用了jgit这个工具包，完成了Java操作代码仓库，基本上和我们日常使用的操作基本一致，

```
org.eclipse.jgit
```

我们来让AI教会我们一下JGit。AI返回的内容如下：

Git是目前最流行的版本控制系统之一，用于管理代码库的历史和协作开发。JGit是一个纯Java实现的Git类库，它提供了对Git功能的全面支持，使开发者能够在Java应用程序中轻松集成和操作Git仓库。本文将详细介绍JGit的功能、优点、常见用例以及如何使用JGit进行基本的Git操作。

什么是JGit？

JGit是Eclipse团队开发的一个开源项目，旨在提供对Git的完全实现。由于其纯Java实现，JGit不依赖于任何本地Git安装，这使得它在Java应用程序中非常容易集成。JGit支持Git的所有核心功能，包括克隆仓库、提交变更、管理分支和标签、处理合并和冲突等。

JGit的优点

1. **纯Java实现**：JGit不依赖于本地Git安装，因此可以跨平台运行，特别适合需要在不同环境中部署的Java应用程序。
2. **轻量级**：JGit非常轻量级，不会对项目增加太多的依赖和开销。
3. **易于集成**：JGit提供了丰富的API，可以轻松集成到现有的Java项目中，支持多种操作和定制。
4. **活跃的社区**：作为Eclipse的一部分，JGit拥有一个活跃的开发者社区，定期发布更新和新功能。

常见用例

- **自动化构建和部署**：使用JGit在CI/CD管道中进行代码检出、构建和部署。
- **代码分析工具**：在代码质量检查、代码审查工具中集成JGit，以获取代码变更历史和差异。

- **版本管理**：在任何需要版本控制的Java应用程序中使用JGit，如文档管理系统或配置管理工具。

如何使用JGit

以下是使用JGit进行基本Git操作的示例。

1. 克隆仓库

克隆一个远程Git仓库到本地：

```
import org.eclipse.jgit.api.Git;

public class CloneRepository {

    public static void main(String[] args) throws Exception {

        Git.cloneRepository()

            .setURI("https://github.com/user/repository.git")

            .setDirectory(new File("path/to/repo"))

            .call();

        System.out.println("Repository cloned.");

    }

}
```

2. 添加文件并提交

将文件添加到仓库并提交变更：

```
import org.eclipse.jgit.api.Git;
import org.eclipse.jgit.api.errors.GitAPIException;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class AddAndCommit {

    public static void main(String[] args) throws IOException, GitAPIException {

        Git git = Git.open(new File("path/to/repo"));

        // 创建新文件并写入内容
        File newFile = new File(git.getRepository().getDirectory().getParent(), "newfile.txt");
        try (FileWriter writer = new FileWriter(newFile)) {
            writer.write("Hello, JGit!");
        }

        // 添加文件到Git索引
        git.add().addFilepattern("newfile.txt").call();

        // 提交变更
        git.commit().setMessage("Added newfile.txt").call();

        System.out.println("Changes committed.");

    }

}
```

3. 推送到远程仓库

将本地变更推送到远程仓库：

```
import org.eclipse.jgit.api.Git;
import org.eclipse.jgit.api.errors.GitAPIException;
import org.eclipse.jgit.transport.UsernamePasswordCredentialsProvider;

public class PushToRemote {

    public static void main(String[] args) throws GitAPIException {

        Git git = Git.open(new File("path/to/repo"));

        git.push()
```

```
.setCredentialsProvider(new UsernamePasswordCredentialsProvider("username", "password"))
.call();

System.out.println("Changes pushed to remote repository.");
}
}
```

结论

JGit是一个功能强大且易于使用的Java类库，为开发者提供了全面的Git支持。通过JGit，开发者可以在Java应用程序中实现各种Git操作，简化了Git的集成和使用。无论是自动化构建、代码分析还是版本管理，JGit都能提供高效可靠的解决方案。如果你的Java项目需要与Git交互，不妨试试JGit

四、总结

本文在作者的讲解下，带领大家学会了使用JGit这个类库操作Github的仓库，在学习的过程中也看到了一个不错的项目可以作为学习资料：

<https://github.com/centic9/jgit-cookbook>

希望本篇文章对大家有用。经过这一节的内容，我们的AI代码评审也完成了第4个业务流程玩法：

