



TEAM CLOUD - GITHUB COPILOT – AI PAIR PROGRAMMING

OVERVIEW OF GITHUB COPILOT'S FEATURES AND BENEFITS

Team:

CHANDRASEKHAR KOPPELU

GOPINATH CHIRUVELLA

SARITHA THOTA

PULIKESI SURULIRAJ

PRAKASH VIDUTHALAIRAJU

KEY FEATURES OF GITHUB COPILOT

- Real-Time Code Suggestions
- Context-Aware Completions
- Support for Multiple Languages
- Learning from Code Comments
- Error Detection and Fixes
- Generating Documentation and Tests
- Faster Prototyping

BENEFITS OF AI PAIR PROGRAMMING WITH COPILOT

- Increased Productivity
- Enhances Learning
- Improved Code Quality
- Accelerated Onboarding
- Collaborative Coding Experience

EXAMPLE WORKFLOW WITH GITHUB COPILOT

1. Writing Comments: Describe desired functionality.
2. Copilot Suggests Code: Auto-suggest implementation based on the comment.
3. Accept/Modify Suggestion: Adjust or accept the suggestions.
4. Additional Suggestions: Continue coding with more relevant suggestions.

TEAM CLOUD - DEMO



GITHUB COPILOT PLUGIN

The screenshot shows the IntelliJ IDEA interface with the 'Settings' dialog open to the 'Plugins' tab. The 'Installed' sub-tab is active, showing a list of installed plugins. The 'GitHub Copilot' plugin is highlighted, showing its version (1.5.27.7265) and source (GitHub). Other installed plugins include SonarLint, Gradle, Gradle Extension, Maven, Maven Extension, and Code Coverage for Java. The background shows the project structure of 'TeamCloud' and the code editor with 'TeamCloudApplicationTests.java' open.

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools VCS Window Help

TeamCloudApplication Version control

Project TeamCloud ~/IdeaProjects/TeamCloud

- .gradle
- .idea
- build
- gradle
- src
 - main
 - com.team.cloud
 - TeamCloudApplication
 - TeamCloudController
 - resources
 - static
 - templates
 - application.yaml
 - test
 - .gitattributes
 - .gitignore
 - build.gradle
 - gradlew
 - gradlew.bat
 - HELP.md
 - settings.gradle
 - TeamCloud.iml
- External Libraries
- Scratches and Consoles

TeamCloudApplicationTests.java

```
1 package com.team.cloud;
```

build.gradle (TeamCloud)

```
1 package com.team.cloud;
```

application.yaml

TeamCloudApplication.java

TeamCloudController.java

Settings

Plugins Marketplace Installed

Type / to see options

Downloaded (2 of 2 enabled)

- GitHub Copilot** 1.5.27.7265 GitHub
- SonarLint** 9.1.0.75538 SonarSource

Build Tools Disable all

- Gradle** bundled
- Gradle Extension** bundled
- Maven** bundled
- Maven Extension** bundled

Code Coverage Disable all

- Code Coverage for Java** bundled

Cancel Apply OK

TeamCloud > src > main > java > com > team > cloud > TeamCloudController

8:1 LF UTF-8 4 spaces 2022-10-24

CODE REVIEW

The screenshot displays the IntelliJ IDEA IDE interface during a code review session. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The status bar at the bottom shows the file path: TeamCloud > src > main > java > com > team > cloud > TeamCloudController > getCapital, along with the line number 18:8, LF line ending, UTF-8 encoding, and 4 spaces indentation.

The left sidebar shows the Project view with the following structure:

- TeamCloud ~/Idea
- gradle
- .idea
- build
- gradle
- src
 - main
 - java
 - com.teamcloud
 - AppController.java
 - CountryResponse.java
 - CountryResponse.java
 - TeamCloudController.java
 - TeamCloudController.java
 - resources
 - static
 - template
 - application.yml
 - test
 - .gitattributes
 - .gitignore
 - build.gradle
 - gradlew
 - gradlew.bat
 - HELP.md
 - settings.gradle
 - TeamCloud.iml
- External Libraries
- Scratches and Console

The main editor displays the code for TeamCloudController.java:

```
1 package com.team.cloud;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.http.ResponseEntity;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RestController;
8 import org.springframework.web.client.RestTemplate;
9
10 @RestController
11 public class TeamCloudController {
12
13     @Autowired
14     private RestTemplate restTemplate;
15
16     @GetMapping("/getCapital")
17     public String getCapital(@RequestParam String country) {
18         String url = "https://jsonmock.hackerrank.com/api/countries?name=" + country;
19         ResponseEntity<CountryResponse> response = restTemplate.getForEntity(url, CountryResponse.class);
20         Country countryObj = response.getBody().getData().get(0);
21         return countryObj.getCapital();
22     }
23 }
24
25
```

The right sidebar shows the GitHub Copilot Chat panel with the following content:

Creating a GET method to fetch the capital of a country using Spring REST

```
private List<Country> data;

public List<Country> getData() {
    return data;
}

public void setData(List<Country> data) {
    this.data = data;
}
}
```

package com.team.cloud;

```
public class Country {
    private String capital;

    public String getCapital() {
        return capital;
    }

    public void setCapital(String capital) {
        this.capital = capital;
    }
}
```

This version includes error handling for cases where the country is not found and ensures that the JSON response is correctly parsed.

> Using 1 Reference

What are the best practices for error handling in Spring REST APIs?

please d

+

SECURITY VULNERABILITIES

The screenshot shows the IntelliJ IDEA IDE interface. The main editor displays the `build.gradle` file for a project named `TeamCloud`. The file contains the following code:

```
1 plugins {
2     id 'org.springframework.boot' version '3.1.0'
3     id 'io.spring.dependency-management' version '1.1.0'
4 }
5
6 group = 'com.team'
7 version = '0.0.1-SNAPSHOT'
8
9 java {
10     toolchain { JavaToolchainSpec it -> {
11         languageVersion = JavaLanguageVersion.of(21)
12     }
13 }
14
15 repositories {
16     mavenCentral()
17 }
18
19 dependencies {
20     implementation 'org.springframework.boot:spring-boot-starter-actuator'
21     implementation 'org.springframework.boot:spring-boot-starter-web'
22     implementation 'org.projectlombok:lombok:1.18.30'
23     implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.0.3'
24     developmentOnly 'org.springframework.boot:spring-boot-devtools'
25     testImplementation 'org.springframework.boot:spring-boot-starter-test'
26     testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
27 }
28
29 tasks.named('test') { Task it -> {
30     useJUnitPlatform()
31 }
32 }
```

The right sidebar shows the `GitHub Copilot Chat` window. It displays a code review snippet and a list of updates:

- Input validation for the `country` parameter.
- Enhanced error handling for network issues and other exceptions.
- Logging of errors for better monitoring and debugging.
- More appropriate HTTP status codes and error messages.

Below the list, it says "Using 1 Reference". The chat window also shows a message from `ChanduRoayl` asking to identify security vulnerabilities in the `build.gradle` file. The `GitHub Copilot` status bar indicates "2 steps completed successfully" and shows progress for "Collecting context", "Resolving recent files", and "Generating response".

The bottom status bar shows the current file is `build.gradle` and the test results are "Tests passed: 2 of 2 tests - 837 ms".

SECURITY ISSUES

The screenshot shows the IntelliJ IDEA IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The status bar at the bottom indicates the file path: TeamCloud > src > main > java > com > team > cloud > TeamCloudController.

The Project tool window on the left shows the project structure:

- Project > .idea
- Project > build
- Project > gradle
- Project > src > main > java > com.team.cloud > config > AppConfig
- Project > src > main > java > com.team.cloud > Country
- Project > src > main > java > com.team.cloud > CountryResponse
- Project > src > main > java > com.team.cloud > TeamCloudApplication
- Project > src > main > java > com.team.cloud > TeamCloudController
- Project > src > test > java > com.team.cloud > TeamCloudApplicationTests
- Project > src > test > java > com.team.cloud > TeamCloudControllerTest

The main editor displays the code for TeamCloudController.java:

```
4 import io.swagger.v3.oas.annotations.responses.ApiResponse;
5 import io.swagger.v3.oas.annotations.responses.ApiResponses;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.HttpStatus;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RestController;
12 import org.springframework.web.client.RestTemplate;
13 import org.springframework.web.server.ResponseStatusException;
14
15 @RestController
16 public class TeamCloudController {
17
18     @Autowired
19     private RestTemplate restTemplate;
20
21     @Operation(summary = "Get a greeting message")
22     @ApiResponses(value = {
23         @ApiResponse(responseCode = "200", description = "Successful operation")
24     })
25     @GetMapping("/capital")
26     public String getCapital(@RequestParam String country) {
27         String url = "https://jsonmock.hackerrank.com/api/countries?name=" + country;
28         ResponseEntity<CountryResponse> response = restTemplate.getForEntity(url, CountryResponse.class);
29         if (response.getBody() == null || response.getBody().getData().isEmpty()) {
30             throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Country not found");
31         }
32         Country countryObj = response.getBody().getData().get(0);
33
34         return countryObj.getCapital();
35     }
36 }
37
38 }
```

The Run tool window at the bottom shows the execution results:

- TeamCloudApplication
- TeamCloudControllerTest
- Tests passed: 2 of 2 tests - 837 ms

The GitHub Copilot Chat window on the right shows a conversation:

Creating a GET method to fetch the capital of a country...

This test class includes two test cases:

- testGetCapital_Success: Tests the successful retrieval of the capital.
- testGetCapital_CountryNotFound: Tests the scenario where the country is not found.

Using 1 Reference

ChanduRoayl

is there any security issues with the code?

GitHub Copilot

- No steps completed successfully
- Collecting context

Ask Copilot a question or type '/' for commands

UNIT TEST CASE

The screenshot displays the IntelliJ IDEA IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The toolbar shows various icons for project management, code editing, and running. The left sidebar shows the Project view with a tree structure of the project files. The main editor area shows the source code of the TeamCloudController.java file. The right sidebar shows the GitHub Copilot Chat panel with a message about adding Swagger to the Spring Boot application.

Project Structure:

- build
- gradle
- src
 - main
 - java
 - com.team.cloud
 - config
 - AppConfig
 - Country
 - CountryResponse
 - TeamCloudApplication
 - TeamCloudController
 - resources
 - static
 - templates
 - application.yaml
 - test
 - java
 - com.team.cloud
 - TeamCloudApplicationTests
 - External Libraries

TeamCloudController.java:

```
12 import org.springframework.web.client.RestTemplate;
13 import org.springframework.web.server.ResponseStatusException;
14
15 @RestController
16 public class TeamCloudController {
17
18     @Autowired
19     private RestTemplate restTemplate;
20
21     @Operation(summary = "Get a greeting message")
22     @ApiResponse(value = {
23         @ApiResponse(responseCode = "200", description = "Successful operation")
24     })
25     @GetMapping("/capital")
26     public String getCapital(@RequestParam String country) {
27         String url = "https://jsonmock.hackerrank.com/api/countries?name=" + country;
28         ResponseEntity<CountryResponse> response = restTemplate.getForEntity(url, CountryResponse.class);
29         if (response.getBody() == null || response.getBody().getData().isEmpty()) {
30             throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Country not found");
31         }
32         Country countryObj = response.getBody().getData().get(0);
33
34         return countryObj.getCapital();
35     }
36
37 }
38
```

GitHub Copilot Chat:

Adding Swagger to Spring Boot applicati

```
.paths(PathSelectors.any())
    .build();
}
```

3. Access Swagger UI: Once you have added the dependencies and configuration, you can access the Swagger UI at <http://localhost:8080/swagger-ui/>.

This setup will automatically generate API documentation for all the endpoints in your `com.team.cloud` package.

Using 4 References

ChanduRoayl

lets write unit test for this code

GitHub Copilot

No steps completed successfully

Collecting context

Ask Copilot a question or type '/' for commands

+

Run: TeamCloudApplication

Console:

```
2024-10-25T09:25:44.659-05:00 INFO 7127 --- [TeamCloud] [on(3)-127.0.0.1] o.a.c.c.f.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-10-25T09:25:44.659-05:00 INFO 7127 --- [TeamCloud] [on(3)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2024-10-25T09:25:44.660-05:00 INFO 7127 --- [TeamCloud] [on(3)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
2024-10-25T09:27:19.791-05:00 INFO 7127 --- [TeamCloud] [nio-8080-exec-9] o.springdoc.api.AbstractOpenApiResource : Init duration for springdoc-openapi is: 49 ms
```

TeamCloudController:

38:1 LF UTF-8 4 spaces

TESTING

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Swagger UI x +

localhost:8080/swagger-ui/index.html

Swagger. Supported by SMARTBEAR

/v3/api-docs Explore

OpenAPI definition v0 OAS3

[/v3/api-docs](#)

Servers

http://localhost:8080 - Generated server url

team-cloud-controller

GET /capital Get a greeting message

localhost:8080/swagger-ui/index.html#team-cloud-controller

CONCLUSION

GitHub Copilot enhances coding efficiency, reduces errors, and fosters a collaborative coding environment by acting as an AI pair programmer.