

मaven का अनुवाद

मaven क्या है?

मaven एक build tool है जो आपको अपने project को build करने में मदद करता है। यह आपको अपने project के dependencies को manage करने में मदद करता है, और आपको अपने project को build करने में मदद करता है। यह आपको अपने project को build करने में मदद करता है, और आपको अपने project को build करने में मदद करता है।

मaven का उपयोग कैसे करें?

मaven का उपयोग करने के लिए आपको निम्नलिखित कदमों का पालन करना होगा:

- 1. मaven को install करें: आपको अपने system पर maven को install करना होगा।
- 2. मaven को configure करें: आपको अपने system पर maven को configure करना होगा।
- 3. मaven का उपयोग करें: आपको अपने project में maven का उपयोग करना होगा।
- 4. मaven का उपयोग करें: आपको अपने project में maven का उपयोग करना होगा।

मaven का उपयोग करने के लिए आपको निम्नलिखित कदमों का पालन करना होगा:

मaven का उपयोग करने के लिए आपको निम्नलिखित कदमों का पालन करना होगा:

1. मaven को pom.xml में जोड़ें

मaven को pom.xml में जोड़ने के लिए आपको `<build><plugins>` सेक्शन में `org.apache.maven.plugins` ग्रुप आईडी जोड़नी होगी। `spring-boot-starter-parent` पैरेंट पैरामीटर में `groupId` जोड़ें, `artifactId` जोड़ें; `version` जोड़ें।

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <version>3.1.2</version> <!-- Latest version se replace karein -->
    </plugin>
  </plugins>
</build>
```

2. pom.xml 에 checkstyle.xml 을 추가한다

이 예제에서는 pom.xml 에 checkstyle.xml 을 추가하여 build 시에 checkstyle 을 실행한다. 이 예제에서는 pom.xml 에 checkstyle-plugin 을 추가하여 build 시에 checkstyle 을 실행한다. 이 예제에서는 pom.xml 에 checkstyle-plugin 을 추가하여 build 시에 checkstyle 을 실행한다.

다음과 같이 pom.xml 에 추가한다:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <version>3.1.2</version>
      <configuration>
        <configLocation>checkstyle.xml</configLocation>
      </configuration>
    </plugin>
  </plugins>
</build>
```

3. pom.xml 에 google_checks.xml 을 추가한다

이 예제에서는 pom.xml 에 google_checks.xml 을 추가하여 build 시에 google_checks.xml 을 실행한다. 이 예제에서는 pom.xml 에 google_checks.xml 을 추가하여 build 시에 google_checks.xml 을 실행한다.

```
<configLocation>google_checks.xml</configLocation>
```

이 예제에서는 pom.xml 에 google_checks.xml 을 추가하여 build 시에 google_checks.xml 을 실행한다. 이 예제에서는 pom.xml 에 google_checks.xml 을 추가하여 build 시에 google_checks.xml 을 실행한다.

```
<dependencies>
  <dependency>
    <groupId>com.puppycrawl.tools</groupId>
    <artifactId>checkstyle</artifactId>
    <version>8.44</version>
  </dependency>
</dependencies>
```

이 예제에서는 pom.xml 에 google_checks.xml 을 추가하여 build 시에 google_checks.xml 을 실행한다.

이 예제에서는 pom.xml 에 google_checks.xml 을 추가하여 build 시에 google_checks.xml 을 실행한다. 이 예제에서는 pom.xml 에 google_checks.xml 을 추가하여 build 시에 google_checks.xml 을 실행한다.

□ 0000000000 000 0000000000: 0000000000 00 000000 000000 000 000000000000 000000 00 0000 000000 00 0000:

```
mvn checkstyle:check
```

□ 000000 00 00000000 000 0000000: 00000000 00000, 00000000 verify 000000 00 000000 00000 00000 0000. 0000 0000000:

```
mvn verify
```

00000000 0000000000 0000000 00 00000 0000000 00 00000 00 0000000 00 00000:

```
mvn checkstyle:checkstyle
```

000000000 00000000000 target/site/checkstyle.html 00000 00000000000 00000 00000.

00000000 00 000000000000 0000000 000000000

0000 00000000 00 00000000000 00 <configuration> 000000000 00000 00000000 0000 0000000 00000 0000000 pom.xml 00000:

□ 00000 00 00000000000: 000000000 00000, 0000000 00000 00000 0000 00000 0000000000000 0000000 00000. 0000 000000000 0000000 00 00000:

```
<configuration>
  <failOnViolation>>false</failOnViolation>
</configuration>
```

□ 000000 00 0000000000 00 0000000000 00000000: 000000000 00000000 00000-00 0000000 0000000 00000 00000:

```
<configuration>
  <includes>**/*.java</includes>
  <excludes>**/generated/**/*.java</excludes>
</configuration>
```

□ 000000000000 0000000000 0000 00000000: 00000000 00000000 00000000000 0000000 00 0000000 000000000 000000000 0000000 000000:

```
<configuration>
  <violationSeverity>warning</violationSeverity>
</configuration>
```

checkstyle.xml

checkstyle.xml 是用于配置 Checkstyle 的 XML 文件。它定义了要检查的代码规则。以下是一个基本的配置示例：

```
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC
    "-//Checkstyle//DTD Checkstyle Configuration 1.3//EN"
    "https://checkstyle.org/dtds/configuration_1_3.dtd">

<module name="Checker">
    <module name="TreeWalker">
        <module name="JavadocMethod"/>
        <module name="MethodName"/>
        <module name="ConstantName"/>
    </module>
</module>
```

Checkstyle 配置

Checkstyle 配置文件中，`<module name="Checker">` 是根元素，它包含以下子元素：

- `<module name="TreeWalker">`：这是 Checkstyle 的核心，它负责遍历代码树并应用规则。
- `<module name="JavadocMethod"/>`：检查 Javadoc 注释。
- `<module name="MethodName"/>`：检查方法名称。
- `<module name="ConstantName"/>`：检查常量名称。

Checkstyle 配置选项

- `configLocation`：指定 Checkstyle 配置文件的路径。默认情况下，它会在当前目录下查找 `checkstyle.xml`。
- `configLocation`：指定 Checkstyle 配置文件的路径。默认情况下，它会在当前目录下查找 `checkstyle.xml`。
- `configLocation`：指定 Checkstyle 配置文件的路径。默认情况下，它会在当前目录下查找 `checkstyle.xml`。

Checkstyle 使用

Checkstyle 可以通过以下几种方式使用：

- 通过 IDE 集成。
- 通过命令行运行 `mvn checkstyle:check`。
- 通过 Maven 插件 `mvn checkstyle:checkstyle`。
- 通过 Ant 任务 `checkstyle`。

