1. Adding Test-Scoped Dependencies

- **Purpose:** Test-scoped dependencies are libraries required only for compiling and running tests. They are not included in the final build artifact.
- **How to Add:** In your `pom.xml`, add dependencies with the `<scope>test</scope>` element.

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>3.12.4</version>
    <scope>test</scope>
</dependency>
```

2. Running Tests

- **Command:** Use **`mvn test`** to compile and run tests located in `src/test/java`.
- **Test Frameworks:** Maven supports various testing frameworks like JUnit, TestNG, and others. Ensure the appropriate plugin and dependencies are included in your `pom.xml`.

3. Generating Test Reports

- **Default Reports:** Maven generates test reports by default under `target/surefire-reports` when you run `mvn test`.
- **Plugin Configuration:** Customize reports using the Surefire or Failsafe plugins in your `pom.xml`.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M5</version>
    <configuration>
        <!-- Custom configurations -->
    </configuration>

</plugin>
```

-
- ## 4. Using the Site Lifecycle

- **Purpose:** The Maven Site Lifecycle generates a site with project documentation.
- **Commands:** Use `mvn site` to generate the site and `mvn site:stage` to deploy it to a web server.
- **Customization:** Configure the site generation in the `pom.xml` under the `<reporting>` section.

5. Customized Site Configuration

- **Overview:** Customize your Maven site by configuring plugins and resources in the `pom.xml` or by modifying the `src/site` directory.
- **Example Configuration:**

```
<build>
    <plugins>
        <plugin>

<groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-
plugin</artifactId>
            <version>3.9.1</version>
            <configuration>
                <reportPlugins>
                    <!-- Add custom report plugins
-->
                </reportPlugins>
            </configuration>
        </plugin>
    </plugins>
</build>
```

-

6. Using the Javadoc Plugin

- **Purpose:** Generates API documentation for your project.
- **Command:** Run `mvn javadoc:javadoc` to generate Javadoc.
- **Configuration:** Customize the Javadoc output using the plugin configuration in `pom.xml`.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <version>3.4.1</version>
    <configuration>
        <source>11</source> <!-- Java version -->
    </configuration>
</plugin>
```

7. Integrating Maven with Eclipse

- **Installation:** Use the M2Eclipse plugin for Maven integration.
- **Importing Projects:** Use `File -> Import -> Existing Maven Projects` to import Maven projects into Eclipse.
- **Automatic Configuration:** Eclipse should automatically configure Maven projects based on the `pom.xml`.