MODULE 3

Documentation and Reporting

Facilitator Notes:

Introduce the module to the participants and tell them that you will be talking about build documentation and reporting, understand site life cycle, advanced site configuration, how to generate unit test reports, how to generate code coverage reports.

You will be discussing about build documentation and reporting, site life cycle, advanced site configuration, generation of unit test and code coverage reports.

Module Objectives

At the end of this module, you will be able to:

- Explain build documentation and reporting
- Understand site life cycle
- Define advanced site configuration
- Learn how to generate unit test reports
- Discuss how to generate code coverage reports

Facilitator Notes:

Explain the module objectives to the participants.

Module Topics

Let us take a quick look at the topics that we will cover in this module:

1. Documentation Overview
2. Reporting
3. Advanced site reports
4. Unit testing
5. Code coverage

Facilitator Notes:

Inform the participants about the topics that they will be learning in this module.

1. Documentation and Reporting

Following are the key details of project documentation:

- Project documentation plays an important role in software engineering for software product development and use.
- All software applications, whether created by a large corporation or a small team, require documentation.
- Different types of documentation are created throughout the software development lifecycle (SDLC).
- Documentation is created to explain the product functionality and project-related information.
- There are two major categories of documentation, namely:
  - Product documentation
  - Process documentation

Facilitator Notes:

Discuss about project documentation and the types of documentation in detail. We will discuss about reporting later.

Documentation is divided into two main categories:

Product documentation

Process documentation

Product documentation documents the details of the product that is developed and contains instructions on how to use the product and perform various tasks. It can be classified into:

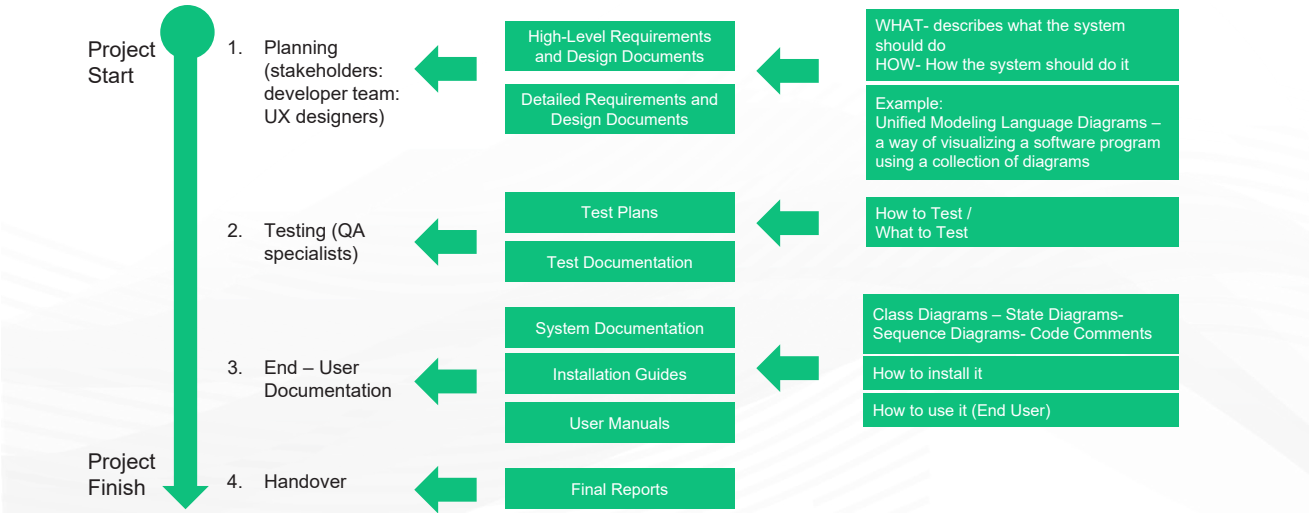
- System documentation and
- User documentation
    - **System documentation** covers the system itself and its parts. It includes requirements documents, architecture descriptions, program source code, design decisions, and help guides.
    - **User documentation** is mainly intended for system administrators and end-users of the product. It includes user guides, troubleshooting manuals, tutorials, installation, and reference manuals.

Process documentation is developed during product development. The common examples are project plans, product API documentation, product standards, meeting notes, test schedules, and reports.

- The main difference between product and process documentation is that the first one describes the product that is developed and the second one record the process of development.

1.1 Documentation Overview

The following figure gives an overview of the documentation and stakeholders involved.



Facilitator Notes:

Discuss about the various types of documentation created from the project starting to finish by looking at the picture.

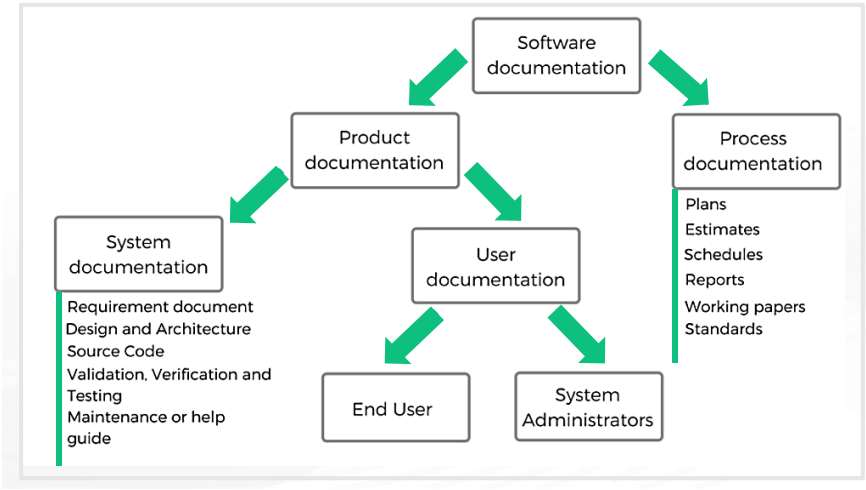
An essential part of any project is the documentation. The project documentation contains ideas and actions that are explained accurately about all the stages involved during each stage of the project.

This ensures that all those involved know what has happened, what is expected of them, what risk or impact exists and the desired outcome. Project documentation should maintain:

- Clear layout and format
- Keep it simple
- Be positive
- Consistency

1.2 Different types of documentation

The following figure shows the different types of documentation.



Facilitator Notes:

Discuss about the different types of documentation.

Following are the different types of project documentation:

Software Documentation: Software documentation is the broad term that is used to represent all the documents about the software product's development. It is broadly classified into product and process documentation.

- Product documentation: Product documentation is about the product that is being built and the different tasks that can be performed using it. Product documentation, in turn, is classified into system and user documentation.
    - **System documentation:** System documentation refers to the document that gives an overview of the system and is useful for all the stakeholders to understand the technology and concept. System documentation comprises requirements document, source code and other validation docs and user manuals.

  - **User documentation:** User documentation refers to the document that is intended to be used by the product users. Typically, a user documentation is aimed at two major user categories, end users and system administrators.

- Process documentation: This document describes the activities involved in product development. This is intended for use by the developers, so that the development process is structured and streamlined.

In Agile, project management practices enable you to manage schedule risk in a project. Below documentation helps you plan the release in a better way:

- Maintain a backlog of feature requests in prioritized order, and revisit the priority at each iteration boundary
- Enforce acceptance criteria for every user story
- Integrate often
- Verify often
- Deliver and review against a plan

2. Reporting

Following are the key details of reporting:

- Reports present visual view of the projects so that we know where the projects are heading.
- Reports often use charts, images, and graphs to capture and present project status, length of an activity and time spent doing each task.
- Online project reports are the ideal way of communicating the latest project status to clients, project managers, stakeholders, and team members.
- Project reporting is the process of giving updates to the leaders, members, and affiliates of a team or an organization. The reporting process is very important, as this will be the foundation of the team's performance and future activities.

Facilitator Notes:

Discuss about reporting.

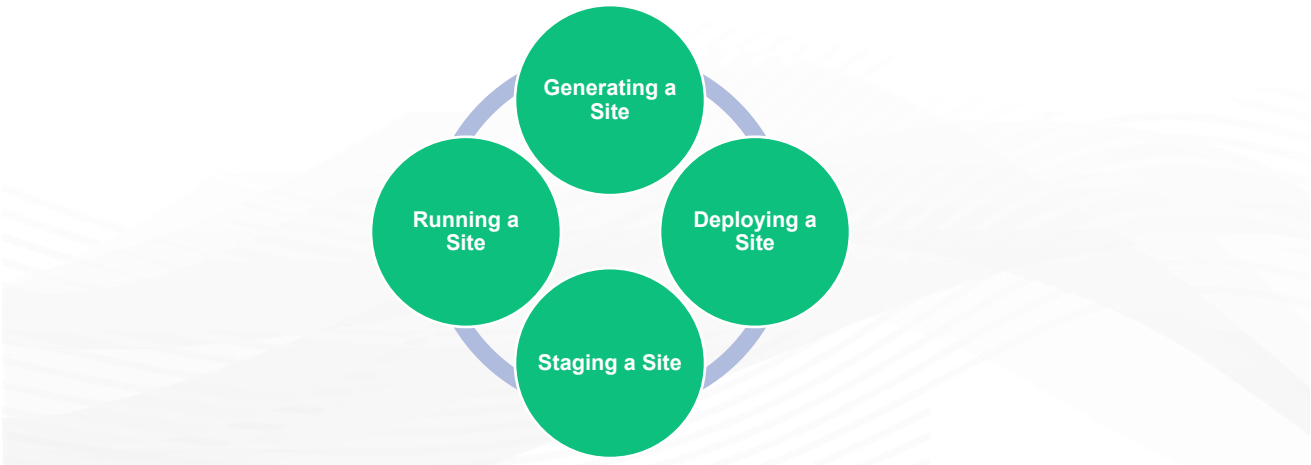
Most of the project reports can be generated using build tools and few reports will be created by the documentation team, project managers and product owners. In maven project reports can be generated by running **mvn site** command.

- Sample configuration in pom.xml to generate site reports:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-project-info-reports-plugin</artifactId>
  <version>3.1.0</version>
</plugin>
```

2.1 Maven Site Plugin

The Site Plugin is used to generate reports and documentation for the project. The generated site reports depend on the contents of the POM. Different stages of site plugin are as follows:



Facilitator Notes:

Give an introduction about the maven site plugin and different stages in site plugin.

Maven plugin has different stages. It includes:

1. Generating a Site

- To generate project site reports **mvn site** command is used.
- By default, the site reports are generated in target/site/directory.
- Maven creates site reports for the for files that are changed since the last build. If the configuration is changed in pom.xml, then you should generate site reports from scratch.

2. Deploying a Site

- To deploy the site, reports provide the location in POM file under <distributionmanagement> tag.
- The <id> element specified in POM is used to link the credentials specified in settings.xml.

- The <url> tag is used to deploy the reports to that location.
- mvn site:deploy command is used to generate and upload site reports.

3. Staging a Site

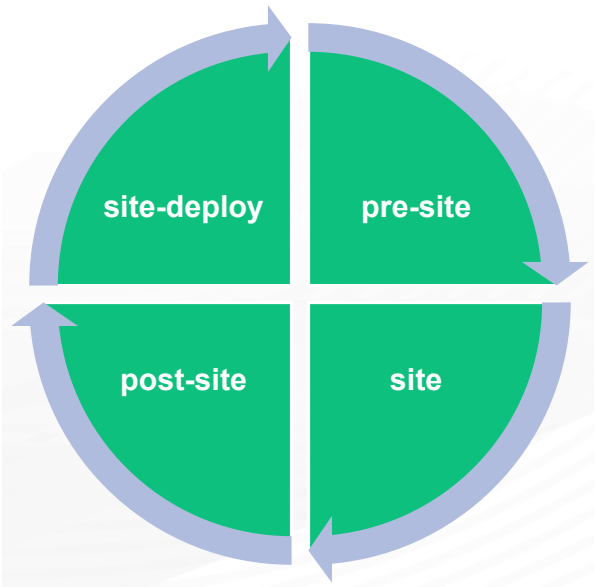
- To test the site reports before deploying it officially is called staging.
- The site reports will be copied to target/staging directory of your project.
- mvn site:stage is used to generate and copy the site reports to the staging directory.

4. Running a Site

- To run a site – **mvn site** command should be executed.
- By default, a jetty server is started and the site reports can be seen at http://localhost:8080
- This works only for single-module sites.

2.2 Maven Site Lifecycle

The Maven Site plugin is generally used to create documentation that includes reports, deploy site, etc. It has the following phases :



Facilitator Notes:

Give an introduction about maven site lifecycle.

Typically maven site lifecycle has 4 phases. If “mvn site” command is executed, it will execute the pre-site and site phases. If “mvn site-deploy“ is executed, it will execute the pre-site, site, post-site, and site-deploy phases.

pre-site: In this phase, maven will run processes that are required before starting the actual site generation.

Site: In this phase, actual project documentation is created.

post-site: In this phase, maven executes the processes that are required to finalize the generation of site reports.

site-deploy: In this phase, maven will deploy the generated site documentation to a web server.

2.3 Maven site Configuration in pom.xml

A sample site configuration file in the maven site configuration is given below:

```
<execution>
          <id>id.site</id>
         <phase>site</phase>
         <goals>
          <goal>run</goal>
          </goals>
          <configuration>
                <tasks>
                <echo>site phase</echo>
             </tasks>
          </configuration>
      </execution>
```

Facilitator Notes:

Discuss about the syntax of mvn site in pom.xml and discuss about the output generated by mvn site.

Sample pom.xml to enable site report configuration:

```
<project xmlns=”http://maven.apache.org/POM/4.0.0” xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
     xsi:schemaLocation=”http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd”>
     <modelVersion>4.0.0</modelVersion>
```



```
<artifactId>api-java</artifactId>
<packaging>jar</packaging>
<name>CDI Java API Module</name>
<description>Java COM API for Common Object Model</description>
<url>http://maven.apache.org</url>
<!-- parent coordinates -->
<parent>
      <groupId>com.hpe.cdi.java.root</groupId>
      <artifactId>com-root</artifactId>
      <version>1.0</version>
</parent>
<build>
<plugins>
      <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-site-plugin</artifactId>
      <version>${maven-site-plugin.version}</version>
      </plugin>

</plugins>
</build>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-pmd-plugin</artifactId>
<version>${pmd.version}</version>
<configuration>
<linkXref>true</linkXref>
<sourceEncoding>utf-8</sourceEncoding>
```



```
<minimumTokens>100</minimumTokens>
<targetJdk>1.7</targetJdk>
<excludes>
<exclude>**/target/*.java</exclude>
</excludes>
</configuration>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-checkstyle-plugin</artifactId>
<version>${checkstyle.version}</version>
<reportSets>
<reportSet>
<reports>
<report>checkstyle</report>
</reports>
</reportSet>
</reportSets>
</plugin>
<plugin>
<groupId>org.codehaus.mojo</groupId>
<artifactId>findbugs-maven-plugin</artifactId>
<version>${findbugs.version}</version>
<configuration>
<effort>Max</effort>
<failOnError>false</failOnError>
<threshold>Low</threshold>
<xmlOutput>true</xmlOutput>
```



```
<findbugsXmlOutputDirectory>${project.build.directory}/findbugs</
findbugsXmlOutputDirectory>
</configuration>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-javadoc-plugin</artifactId>
<version>${maven-javadoc-plugin.version}</version>
<reportSets>
<reportSet>
<id>default</id>
<reports>
<report>javadoc</report>
</reports>
</reportSet>
<reportSet>
<!-- aggregate reportSet, to define in poms having modules -->
<id>aggregate</id>
<!-- don’t run aggregate in child modules -->
<inherited>false</inherited>
<reports>
<report>aggregate</report>
</reports>
</reportSet>
</reportSets>
</plugin>
<plugin>
<groupId>org.jacoco</groupId>
```



```
      <artifactId>jacoco-maven-plugin</artifactId>
      <reportSets>
      <reportSet>
      <reports>
      <report>report</report>
      </reports>
      </reportSet>
      </reportSets>
      </plugin>
      </plugins>
      </reporting>
</project>
```

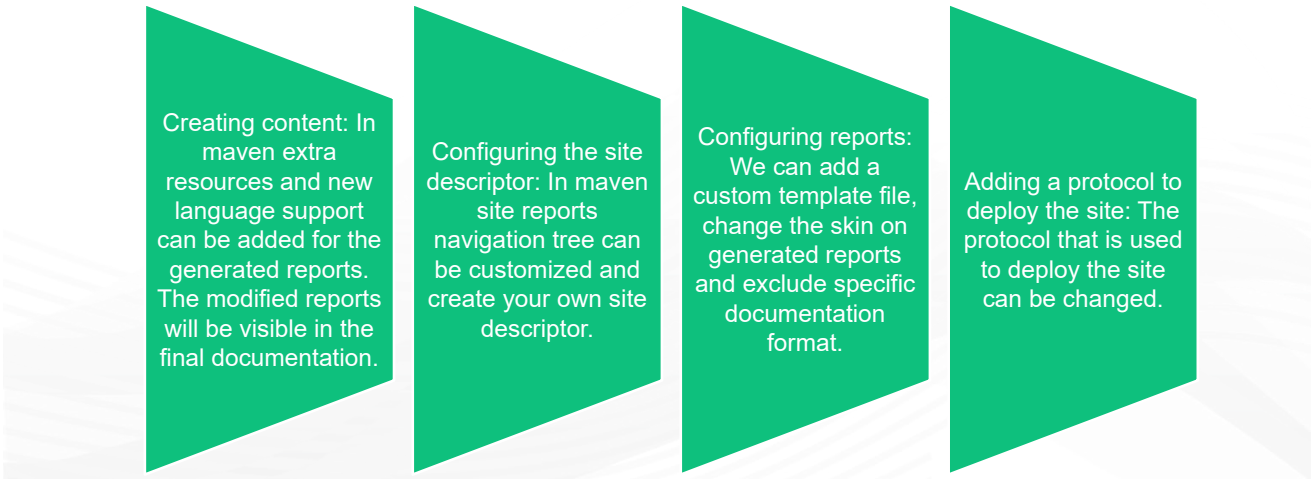
Refer the below output of mvn site command. It includes many project reports.

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO] task-segment: [site]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.pre-site}]
[INFO] Executing tasks
[echo] pre-site phase
[INFO] Executed tasks
[INFO] [site:site {execution: default-site}]
[INFO] Generating “About” report.
[INFO] Generating “Issue Tracking” report.
[INFO] Generating “Project Team” report.
[INFO] Generating “Dependencies” report.
[INFO] Generating “Project Plugins” report.
```



```
[INFO] Generating “Continuous Integration” report.
[INFO] Generating “Source Repository” report.
[INFO] Generating “Project License” report.
[INFO] Generating “Mailing Lists” report.
[INFO] Generating “Plugin Management” report.
[INFO] Generating “Project Summary” report.
[INFO] [antrun:run {execution: id.site}]
[INFO] Executing tasks
[echo] site phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3 seconds
[INFO] Finished at: Sat Nov 04 10:25:10 IST 2018
[INFO] Final Memory: 24M/149M
[INFO] ------------------------------------------------------------------------
```

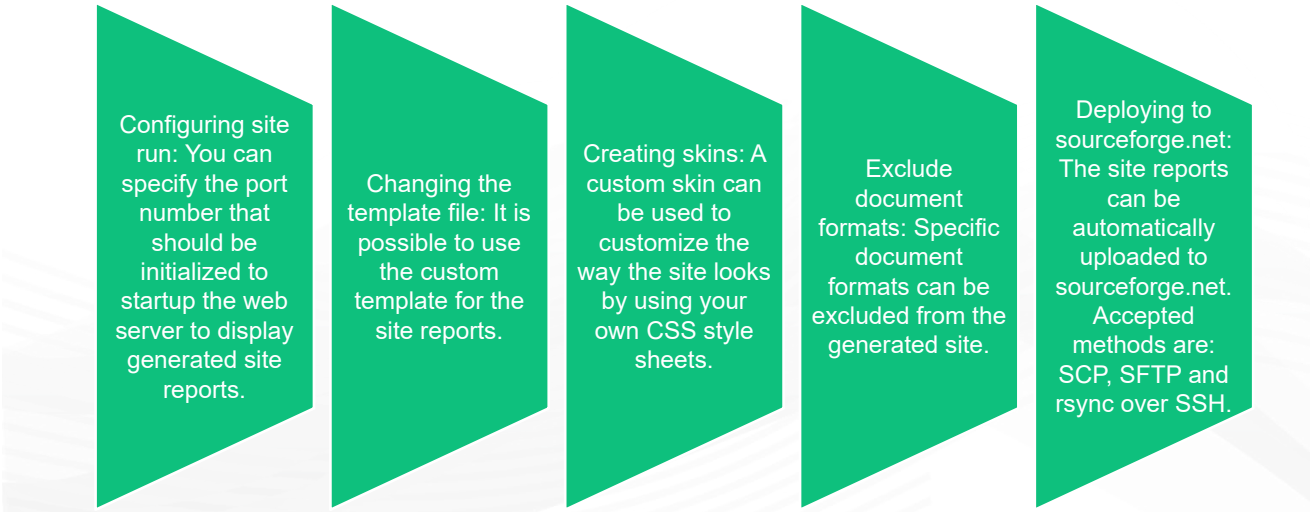
3. Advanced Site Reports



Facilitator Notes:

Discuss about advanced site reporting.

- **Creating content:** We can add extra content to mvn site like adding extra resources (css, png files), internalization (adding new language support), filter the content. This will be visible in the final site documentation.
- **Configuring the site descriptor:** You can use your own site descriptor to customize, **Title** of each generated page, **Banner, Publish Date, Version, Powered by" Logo, Inject xhtml into <head>, Add custom links, Breadcrumbs, Custom footer, Custom content, Skinning**
- **Configuring reports:** Maven has many reports that can be added to the project website to display the current state of the project. Many custom reports can be added to the site:
  - Continuous Integration
  - Dependencies
  - Issue Tracking
  - License
  - Mailing Lists
  - Project Team
  - Source Repository
- **Adding a protocol to deploy the site**
  - Maven3 supports only http, https, and file as transport protocols. Maven2 supports scp to publish the reports.
  - If the site is deployed with unsupported protocol, it will throw an error.



Facilitator Notes:

Discuss about advanced site reporting.

- **Configuring site run**
  - In site:run goal, the port used to start up the site and the directory where the reports are rendered can be changed.
  - For example, the following configuration can be in your POM:
    - When “mvn site:run” is executed, a Jetty server is started at port 9000. It can be accessed at http://localhost:9000 in the browser.
- **Changing the template file**
  - It is possible to change the default template used for creating the site. The location of the custom plugin should be configured in the Site plugin.
- **Creating skins**
  - A custom CSS template can be used to change the view of site reports. Below parameters can be changed in site reports:
  - About
  - Building
  - Constructing the custom CSS
  - Customizing the HTML Output with a Velocity Template
- **Excluding specific document format**
  - A specific document formats can be excluded from the generated site reports. To exclude, you can specify it in the moduleExcludes parameter of the POM.
- **Deploying to sourceforge.net**
  - The old hostname to upload reports to sourceforge is sf.net announcement. It has been changed and the new hostname is web.sourceforge.net and the supported methods are: SFTP, SCP and SSH.

What did You Grasp?

1. Which protocols Maven3 uses to publish the reports?
   - **A) FTP**
   - **B) HTTPS**
   - **C) HTTP**
   - **D) All of the above**

2. In which phase site reports are published?
   - **A) Pre-site**
   - **B) Site**
   - **C) Post-site**
   - **D) Site-deploy**

3. In which folder site reports are generated?
   - **A) target/site/**
   - **B) target/reports/site/**
   - **C) target/modules/site/**
   - **D) None of the above**

4. Which command is used to generate and deploy site reports?
   - **A) mvn site**
   - **B) mvn site deploy**
   - **C) mvn install deploy**
   - **D) mvn install site**

Facilitator Notes:

Tell the participants that they will be going through a knowledge check question.

Answer:

1. d. All of the above

2. d. Site-deploy

3. a. target/site

4. b. mvn site deploy

4. Unit Testing

Unit testing is a testing technique used by a developer in which individual modules are tested to determine if there are any issues in the code.

The main aim of unit testing is to isolate each unit of the system to analyze, identify and fix the bugs.

Facilitator Notes:

Discuss about unit testing.

Unit testing is used to test the smallest testable parts of an application, called units. The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as expected. Each unit is tested separately before integrating them into modules to test the interfaces between modules. By means of effective Unit testing large percentage of defects are identified. Unit testing is performed by developers.

Unit testing is a testing technique used by a developer in which individual modules are tested to determine if there are any issues in the code.

Following are the advantages of unit testing:

- Reduces defects when changing existing functionality or newly developed features.
- Detects bugs in the early phase and reduces cost of testing.
- Allows better code refactoring and improves the design.
- The quality of the build can be increased by integrating Unit Tests with the build.

4.1 Unit Testing Techniques

Unit tests helps in testing individual methods and functions of the classes, components or modules used by your software. Below are a few types of unit testing techniques:

Black Box Testing	Used to test the user interface, input, and output.
White Box Testing	Used to test each one of the function's behavior.
Gray Box Testing	Used to execute tests, risks and assessment methods.

Facilitator Notes:

Discuss about unit testing techniques.

There are multiple types of unit testing like Black box, white box and Gray box testing. They have following benefits:

- Unit tests can be run every time the code is compiled and can identify new bugs if any code is changed.
- With unit test codes are more reusable. With unit testing, the code needs to be modular and it makes code reusable.
- The effort of fixing a defect during unit testing is much lesser than the defects detected in the live production environment.
- Unit testing makes debugging easy. Whenever a unit test fails, only the latest changes should be verified.

4.2 Unit Testing Framework - Java

Unit Testing Framework - Java

- There are many testing frameworks available for Java. The most popular testing frameworks are JUnit and TestNG
- Typically, unit tests and source code are kept in a separate folder to differentiate real code and test cases. The standard convention from the Maven is:
  - src/main/java - for Java classes
  - src/test/java - for test classes

Facilitator Notes:

Discuss about Java testing framework

- JUnit is the most popular unit testing framework in Java and it is recommended for unit testing. JUnit can test the web application without a server, which executes the tests faster.
- JUnit framework allows easy and quick of test cases and test data. The org.Junit package consists of many inbuilt interfaces and classes for JUnit Testing such as Assert, Test, Before, After, etc.

4.3 Junit

To run unit tests a framework is required. JUnit is a unit test framework used in Java programming language.

- ⇥ Junit is an open test framework.
- ⇥ It uses annotations to identify methods that specify a test.
- ⇥ Expected results can be tested using assertions.
- ⇥ JUnit tests allow you to create code faster, which increases quality.
- ⇥ JUnit is less complex, simple and takes less time.
- ⇥ JUnit tests can be run automatically which executing the code and check the results, provide immediate feedback.

Facilitator Notes:

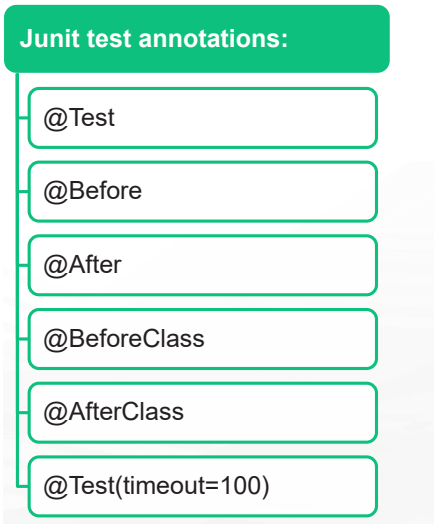
Discuss about JUnit, a unit test framework, in detail

JUnit is a **Unit Testing Framework** used to implement unit testing in Java, increase code quality by developers. JUnit Framework can be easily integrated with multiple build tools like:

- Eclipse
- Ant
- Maven

4.3.1 Junit Annotations

Annotation is a special form of tag that can be added to source code for better code structure and readability.



Facilitator Notes:

Discuss about JUnit annotations in detail.

A JUnit test is a method contained in a class which is only used for testing. This is called a Test class. To define that a certain method is a test method, annotate it with the @Test annotation.

Junit test annotations:

- @Test: Identifies a method as a test method.
- @Before: It is executed before each test and will prepare the test environment. (e.g., initialize the class, read input data).
- @After: It is executed after each test and used to cleanup the test environment (e.g., restore defaults, delete temporary data).
- @BeforeClass: It is executed once before all the tests are started. It is used to perform activities like connecting to a database.
- @AfterClass: It is executed once after all the tests are executed to perform clean up activities like disconnect database connection.
- @Test(timeout=100): Fails if the method execution takes longer than 100 milliseconds.

4.3.2 Example Junit Test Case

The following code shows a JUnit test using the JUnit 5 version.

```
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;

public class MyTests {

    @Test
    public void multiplicationOfZeroIntegersShouldReturnZero() {
        MyClass tester = new MyClass(); // MyClass is tested

        // assert statements
        assertEquals(0, tester.multiply(10, 0), "10 x 0 must be 0");
        assertEquals(0, tester.multiply(0, 10), "0 x 10 must be 0");
        assertEquals(0, tester.multiply(0, 0), "0 x 0 must be 0");
    }
}
```

Facilitator Notes:

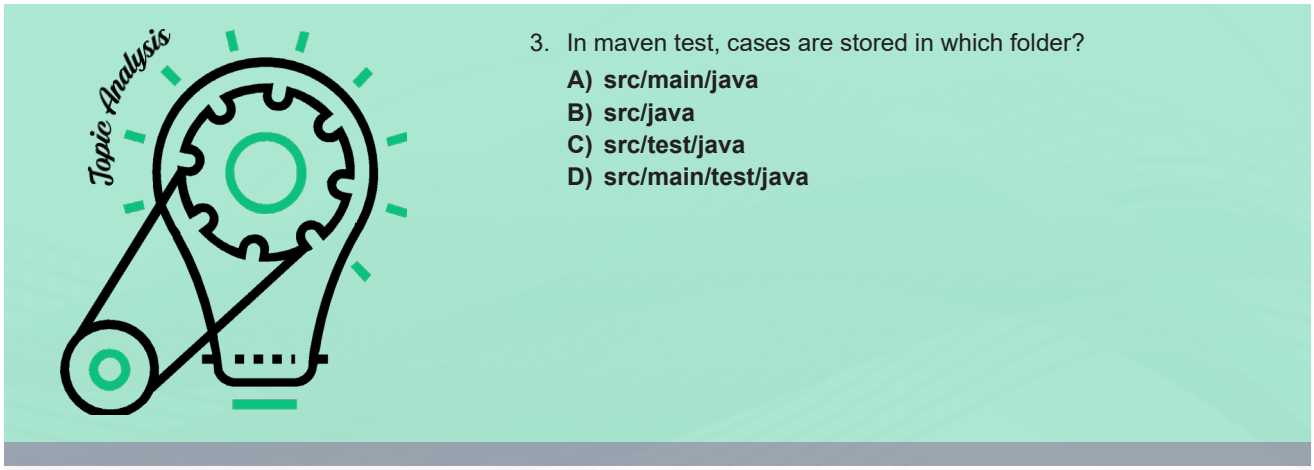
Discuss about the sample Junit test case.

This test assumes that the My Class class exists and has a multiply(int, int) method.

What did You Grasp?

1. In which testing user interface, input and output are tested?
   - **A) White box testing**
   - **B) Black box testing**
   - **C) Gray box testing**
   - **D) None of the above**

2. In which testing, functionality is tested?
   - **A) White box testing**
   - **B) Black box testing**
   - **C) Gray box testing**
   - **D) None of the above**



3. In maven test, cases are stored in which folder?
   - **A) src/main/java**
   - **B) src/java**
   - **C) src/test/java**
   - **D) src/main/test/java**

Facilitator Notes:

Tell the participants that they will be going through a knowledge check question.

Answer:

1. b. Black box testing

2. a. White box testing

3. c. src/test/java

5. Code Coverage

Following are the key details of code coverage:

- Code coverage is a measurement of how many lines/blocks/arcs of your code are executed while the tests are running.
- Code coverage is collected by using specialized tools by adding tracing calls and run a full set of automated tests.
- The goal of code coverage is to find out which part of the code is tested by when running a test.
- Coverage reports can be used to Identify critical features missed in testing.

Facilitator Notes:

Explain in detail about code coverage.

Code coverage is a measurement of how many blocks/lines/arcs of your code are executed while the test cases are executed.

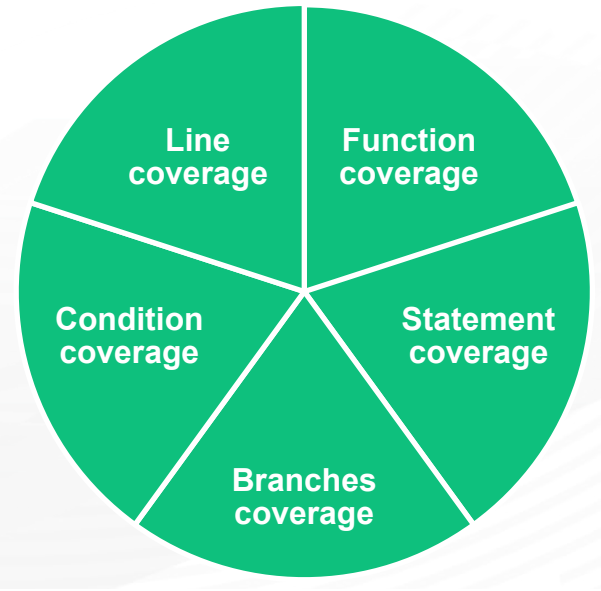
There are various code coverage criteria, like conditions, paths, functions, and statements, etc. Additional criteria for code coverage is:

- **Condition coverage:** All the boolean expressions will be evaluated for true and false.
- **Decision coverage:** All the if-elseif-else conditions will be covered.
- **Loop coverage:** Executes every possible loop at least one time.
- **Entry and exit coverage:** Check the coverage for all calls and its return value.
- **Parameter value coverage:** Check if all the possible values of a parameter are tested. Example, a string can be null, empty, whitespace, valid string, invalid string, single-byte string, double-byte string. If any parameter value is not tested it could be a bug.
- **Inheritance coverage:** While testing coverage for object oriented source, when returning a derived object referred by base class, and check if the sibling object is returned or not, should be tested.

5.1 How is Code Coverage Calculated

Code coverage tools use multiple criteria to test the code during execution of the test suite.

The common metrics that you might see in your coverage reports include:



Facilitator Notes:

Explain details about how code coverage is calculated.

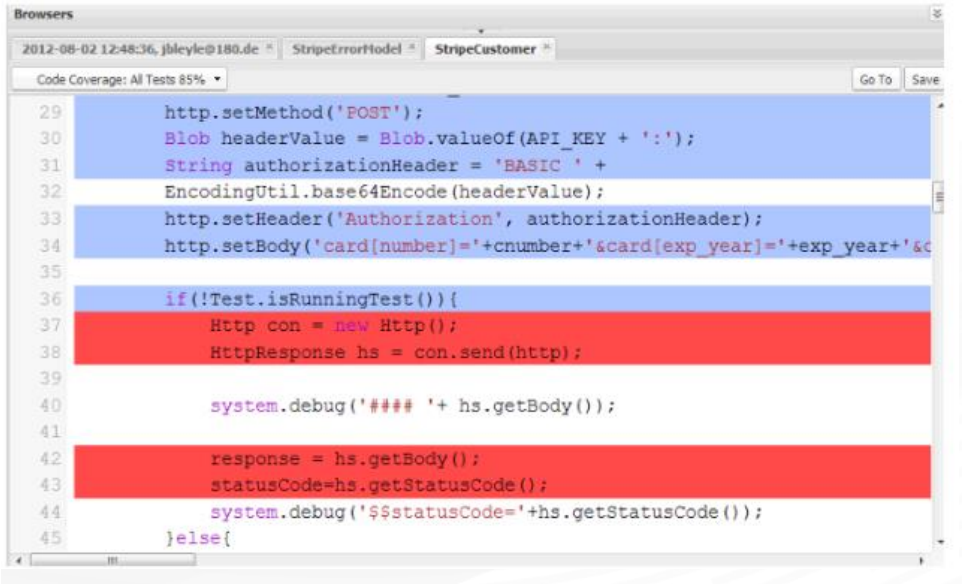
Code coverage tools use multiple criteria to determine how the code was exercised during the execution of the test suite. The common coverage metrics that can be seen in the coverage reports include:

- **Function coverage:** How many defined functions are called.
- **Statement coverage:** How many defined statements in the program are executed.
- **Branches coverage:** how many branches of the control structures (Example - if statements) are executed.
- **Condition coverage:** how many boolean sub-expressions are tested for a value - true and false value.
- **Line coverage:** how many numbers of lines of the source code are tested.

What percentage of code coverage should be better?

There's no fixed % of code coverage for the code. Generally 80% of code coverage is a good goal.

The following screenshot shows the sample code coverage calculation.



Facilitator Notes:

Explain how code coverage is calculated by referring to the above example.

Checking Code Coverage

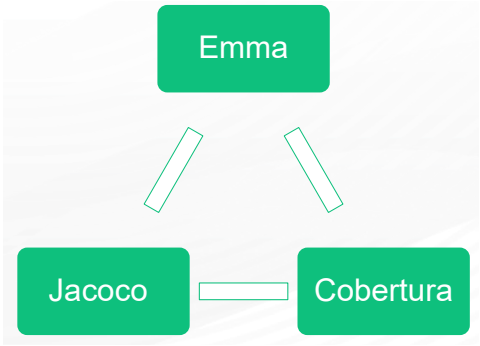
- Code coverage results come from any tests you've run from an API or from a user interface (for example, the Developer Console, the Eclipse IDE, etc.)

- To view line-by-line code coverage for an Apex class, open the class. The code coverage menu will include one or more of the following options depending on the tests you have implemented:
  - **None**
  - **All Tests:** The percentage of code coverage from all test runs.
  - ***className.methodName:*** The percentage of code coverage from a method executed during a test run.
- Lines of code that are covered by tests are blue. Lines of code that aren't covered are red. Lines of code that don't require coverage (for example, curly brackets, comments, and System. debug calls) are left white.

5.2 Code Coverage Tools

There are many open source code coverage tools, but they're not all the same, each tool takes a different approach to check the code coverage.

Few important code coverage tools are as follows:



Facilitator Notes:

Talk about some of the code coverage tools.

There are several options to create code coverage reports depending on the programming language(s) that is used. Popular tools are listed below:

- **Java:** Cobertura, Atlassian Clover, JaCoCo
- **Javascript:** Blanket.js, istanbul
- **PHP:** PHPUnit
- **Python:** Coverage.py
- **Ruby:** SimpleCov

Popular code coverage tools overview:

1. **EMMA:** Emma is one of the most popular and oldest code coverage tools. Emma can run on-the-fly, offline mode to test the coverage of Java applications and it supports multiple types of coverage – class, method, block and line. Emma was developed by Vlad Roubstov and released in 2005.
2. **Cobertura:** It is another open source tool that is developed by Steven Christou in 2015. Cobertura can be integrated with Ant and Maven to generate the code coverage reports.
3. **Jacoco:** Emma and Cobertura is not actively developed. Jacoco is actively developed and can be integrated with Ant, Maven and Jenkins, etc. EclEmma team developed Jacoco plugin.

5.3 Comparison of Code Coverage Tools

The following table summarizes the features of popular code coverage tools.

	Cobertura	Emma	JaCoCo
License	GNU GPL	CPL	EPL
Latest stable release	1.9.4.1 3/3/2010	2.1.5320 22/6/2005	0.5.10 31/8/2012
Sonar Plugin Version	Embedded in Sonar Core	1.2.1	Embedded in Sonar Core
Type of instrumentation	Offline bytecode instrumentation	Offline bytecode instrumentation	On-The-Fly bytecode instrumentation
Process within Sonar	Instrumentation Execution Report generation Report parsing	Instrumentation Execution Data reading	Execution Data reading

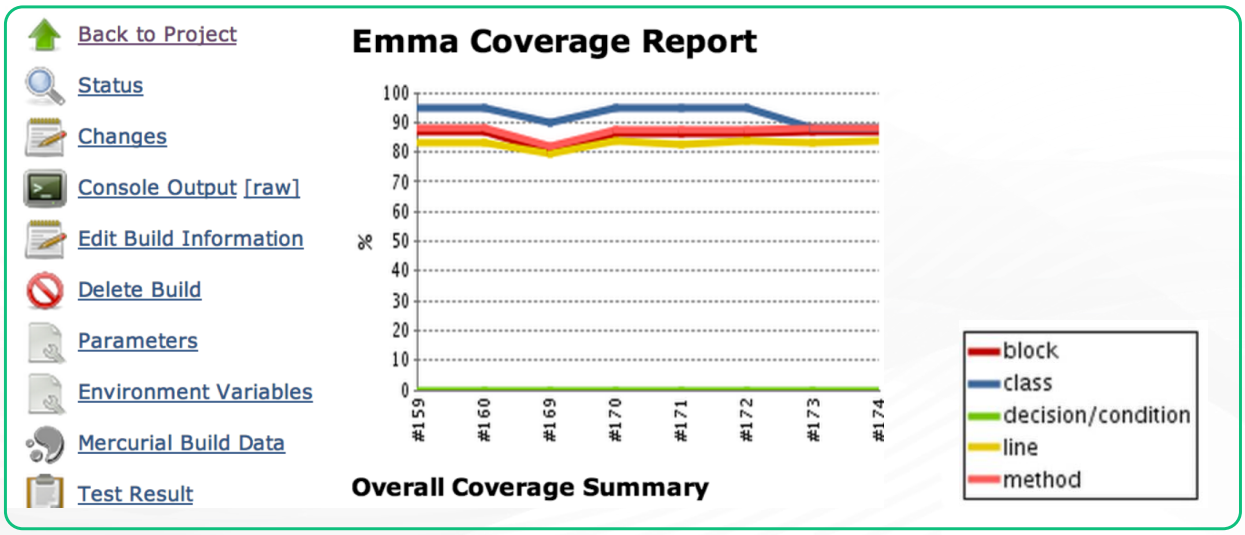
Facilitator Notes:

Compare different types of popular code coverage tools.

Refer to the above table to understand about license types, stable release version of popular code coverage tools.

5.4 Sample Code Coverage Report

The following screenshot is a sample code coverage report produced by Emma.

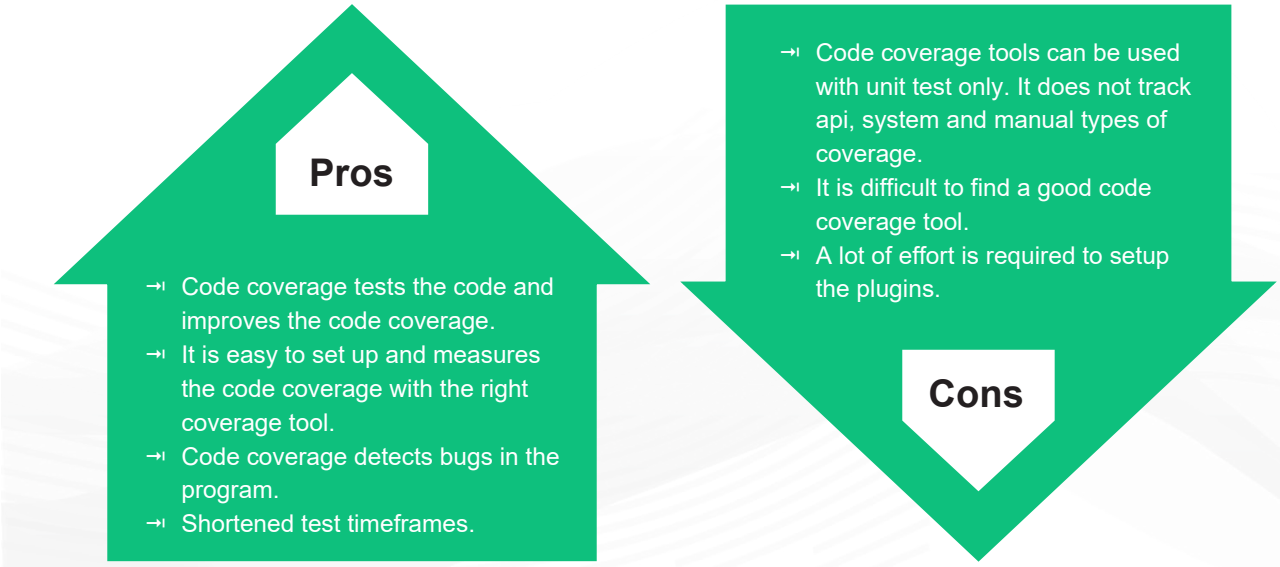


Facilitator Notes:

Discuss about the sample code coverage report

In the above picture, Emma plugin is used to generate code coverage report. It calculated the code coverage for the number of blocks, classes, conditions, lines and methods.

5.5 Code Coverage – Pros and Cons



Facilitator Notes:

Discuss about pros and cons of code coverage.

Every code coverage tool has its pros and cons. But there are multiple advantages using code coverage tools. A few are listed below:

- Identifying the dead code
- Identifying the missing test cases
- Function and line coverage

What did You Grasp?

1. Which of the following units of code is measured by code coverage report?
   - **A) Lines**
   - **B) Arcs**
   - **C) Blocks**
   - **D) All of the above**

2. Code coverage is used to perform which testing?
   - **A) System testing**
   - **B) API testing**
   - **C) Unit testing**
   - **D) Junit testing**

3. Which of the following is/are the types of code coverage?
   - **A) Function coverage**
   - **B) Statement coverage**
   - **C) Condition coverage**
   - **D) Line coverage**
   - **E) All of the above**

4. Code coverage helps you to identify?
   - **A) Function and line coverage**
   - **B) Identifying the missing test cases**
   - **C) Identifying the dead code**
   - **D) All of the above**

5. Which coverage is covered by white box testing?
   - **A) Function coverage**
   - **B) Decision coverage**
   - **C) Line coverage**
   - **D) None of the above**

Facilitator Notes:

Tell the participants that they will be going through a knowledge check question.

Answer:

1. d. All of the above
2. c. Unit testing
3. e. All of the above
4. d. All of the above
5. b. Decision Coverage

Group Discussion

Facilitator Notes:

Divide the students into 4 groups and discuss about project documentation, reporting, role of maven site plugin to generate reports, configuring advanced site reports, unit testing, unit test frameworks, and code coverage tools.

We've so far seen about the important concepts associated documentation/reporting, maven site plugin, unit testing, unit test frameworks and code coverage tools. Form different groups and each group should talk about one concept in detail along with analogies or examples to show your understanding.

In a nutshell, we learnt:

1. Project documentation will play an important role in software engineering for software product development and use.
2. The reports present visual view of the projects so that we know where the projects are heading.
3. The Site Plugin is used to generate reports and documentation for the project.
4. Unit testing is a testing technique used by a developer in which individual modules are tested to determine if there are any issues in the code.
5. JUnit is the most popular unit testing framework in Java and it is recommended for unit testing.
6. Code coverage is a measurement of how many lines/blocks/arcs of your code are executed while the tests are running.

Facilitator Notes:

Share the module summary with the audience.

Ask the participants if they have any questions. They can ask their queries by raising their hands.

