Automation Testing

Integrating JUnit5

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   cout << "Hello, world!" << endl;
   return 0;
}
```

A Day in the Life of an Automation Test Engineer

Marcelo had understood the concept of Extending Junit, Parameter Extensions, and Meta Annotations.

To complete his project, Marcelo has to understand the Integration in JUnit5, which makes sure that integrated modules work properly as intended.

To achieve the above, he will learn a few concepts in his lesson that can help him to find a solution for the scenario.

Learning Objectives

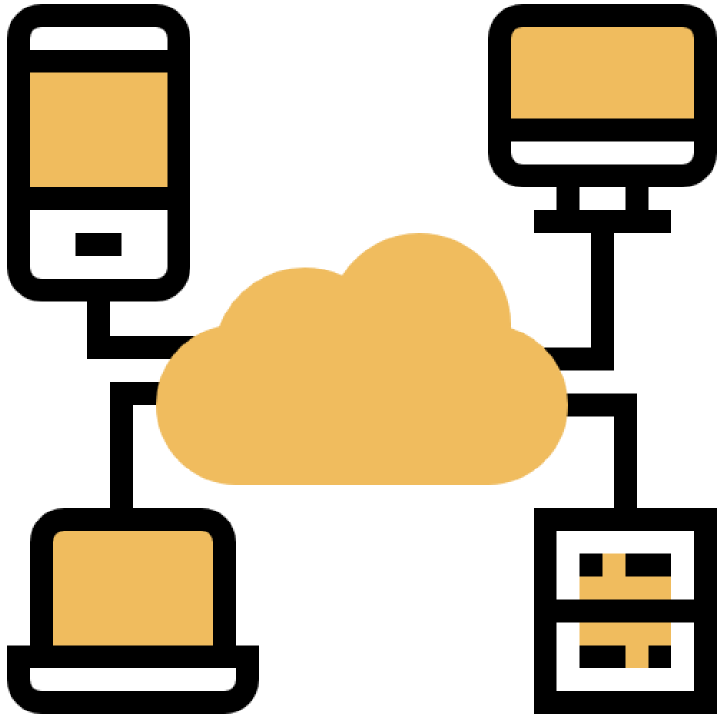
By the end of this lesson, you will be able to:

- Define the JUnit Pro Integration
- Illustrate JUnit Open-Source Integration
- Analyze adding JUnit library in Java project

FULL STACK

JUnit Pro Integration

JUnit Pro Integration



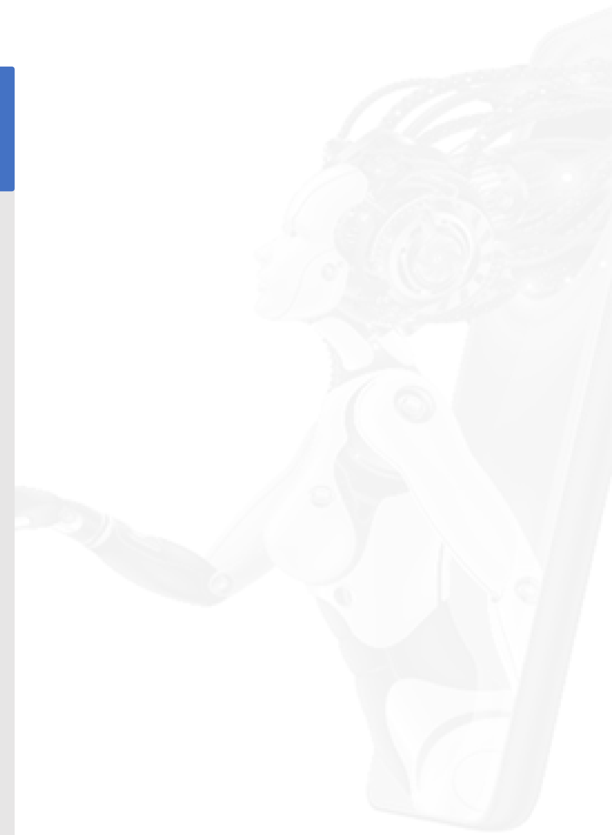
- Pro Integration is the act of bringing together smaller components into a single system that functions as one.

- The JUnit 5 Integration stage can be used in the following topologies: As a source with one or more output links. As a transformer with one or more input links and one or more output or reject links.

JUnit Pro Integration

Here is an example of creating an Integration:

```
public void testRunner() throws Exception
{
 ProTestCaseRunner runner = new ProTestCaseRunner();
 runner.setProjectFile("SoapUIWorkspace/Sample-ATE-project.xml");
 runner.setPrintReport(true);
 runner.run();
}
```



JUnit Pro Integration

The runner.run() call will throw an exception if an error occurs. If the user want more control over your integration or error-reporting, a specific TestCase could be run as follows:

```
public void testRunner() throws Exception
 {
      WsdlProjectPro project = new WsdlProjectPro("Workspace/Sample-ATE-project.xml");
      List testSuites = project.getTestSuiteList();
      for( TestSuite suite : testSuites )
      {
          List testCases = suite.getTestCaseList();
          for( TestCase testCase : testCases )
          {
           System.out.println("Running ATE test [" + testCase.getName() + "]");
           TestRunner runner2 = testCase.run(new PropertiesMap(), false);
           assertEquals(Status.FINISHED, runner2.getStatus());
          }
      }
 }
```

FULL STACK

JUnit Open-Source Integration

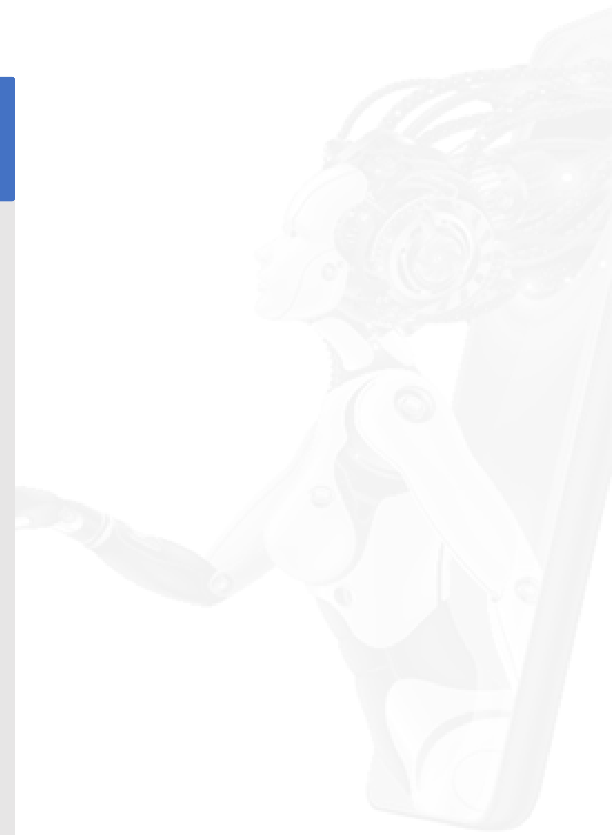
JUnit Open-Source Integration

With JUnit, an Open-Source unit test framework, users can write and run tests for user adapter implementation. Users can use a simple setup() method to prepare each component for testing. Each test method can contain one or more assertions.

JUnit Open-Source Integration

Here is an example for creating a JUnit Open-Source Integration:

```
public void testRunner() throws Exception
{
  TestCaseRunner runner = new TestCaseRunner();
  runner.setProjectFile( "src/dist/sample-Marcelo-project.xml" );
  runner.run();
}
```



JUnit Open-Source Integration

Here is an example for creating a JUnit Open-Source Integration:

```
public void testTestCaseRunner() throws Exception
{
  WsdlProject project = new WsdlProject( "src/dist/sample-ATE-project.xml" );
  TestSuite testSuite = project.getTestSuiteByName( "Test Suite" );
  TestCase testCase = testSuite.getTestCaseByName( "Test Conversions" );

  // create empty properties and run synchronously
  TestRunner runner = testCase.run( new PropertiesMap(), false );
  assertEquals( Status.FINISHED, runner.getStatus() );
}
```

Adding JUnit Library in Java Project

Adding JUnit Library in Java Project

- JUnit is a unit testing framework for the Java programming language.
- JUnit has been important in the development of test-driven developments and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.
- To run the JUnit test users don't have to create a class object or define the main method.
- JUnit provides an assertion library that is used to evaluate the test result.

Adding JUnit Library in Java Project

To add the JUnit library, the user needs to follow the below-mentioned steps:

Go the Java project and right-click.

Go to Build Path->'Configure Build Path'.

Click Libraries.

Click 'Add Library'.

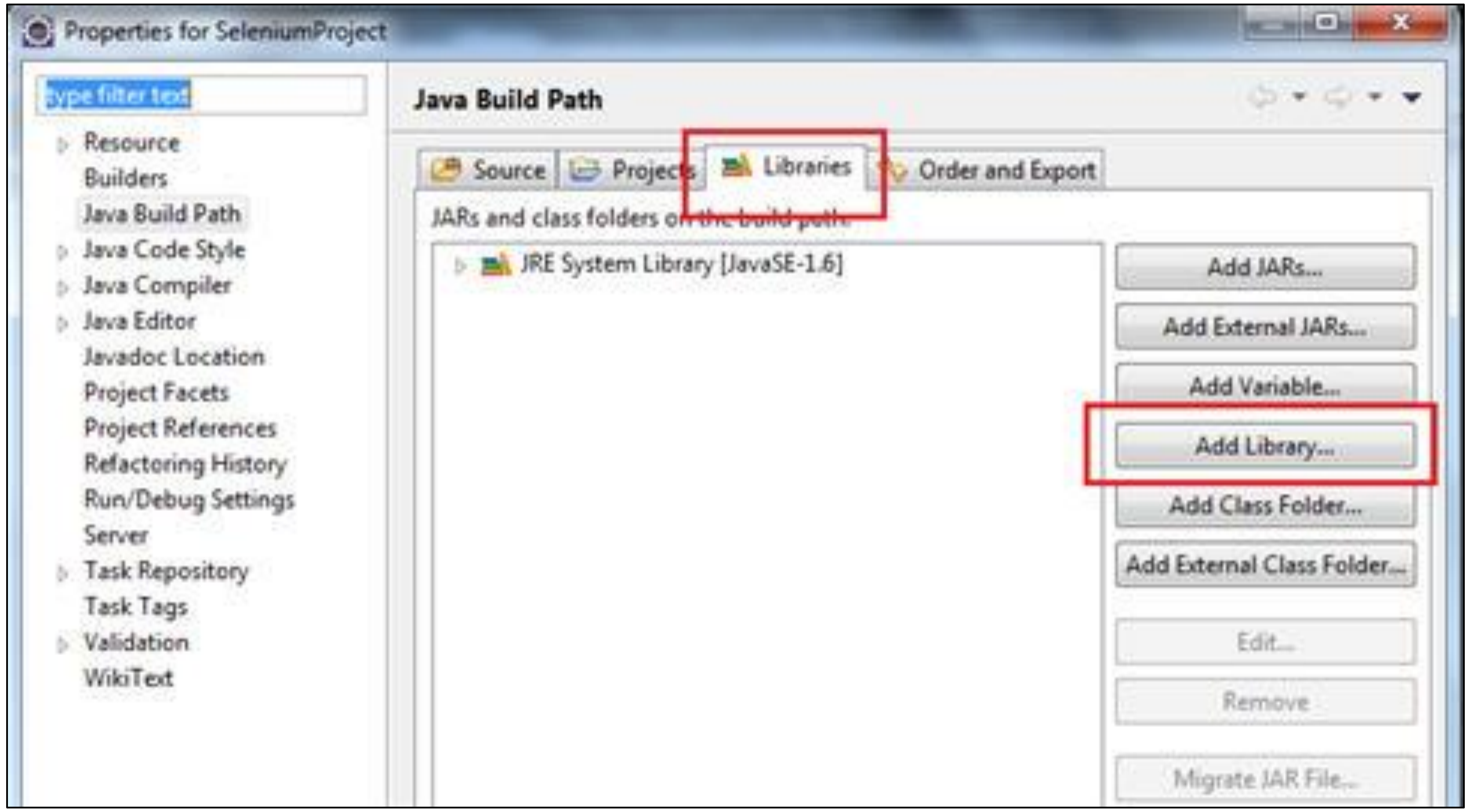
The user will get the 'Add Library' window.

Select 'JUnit' and click 'Next'.

Users will get a screen to select the JUnit version to be used in the project.

Select the JUnit library version as JUnit 5 and click 'Finish'.

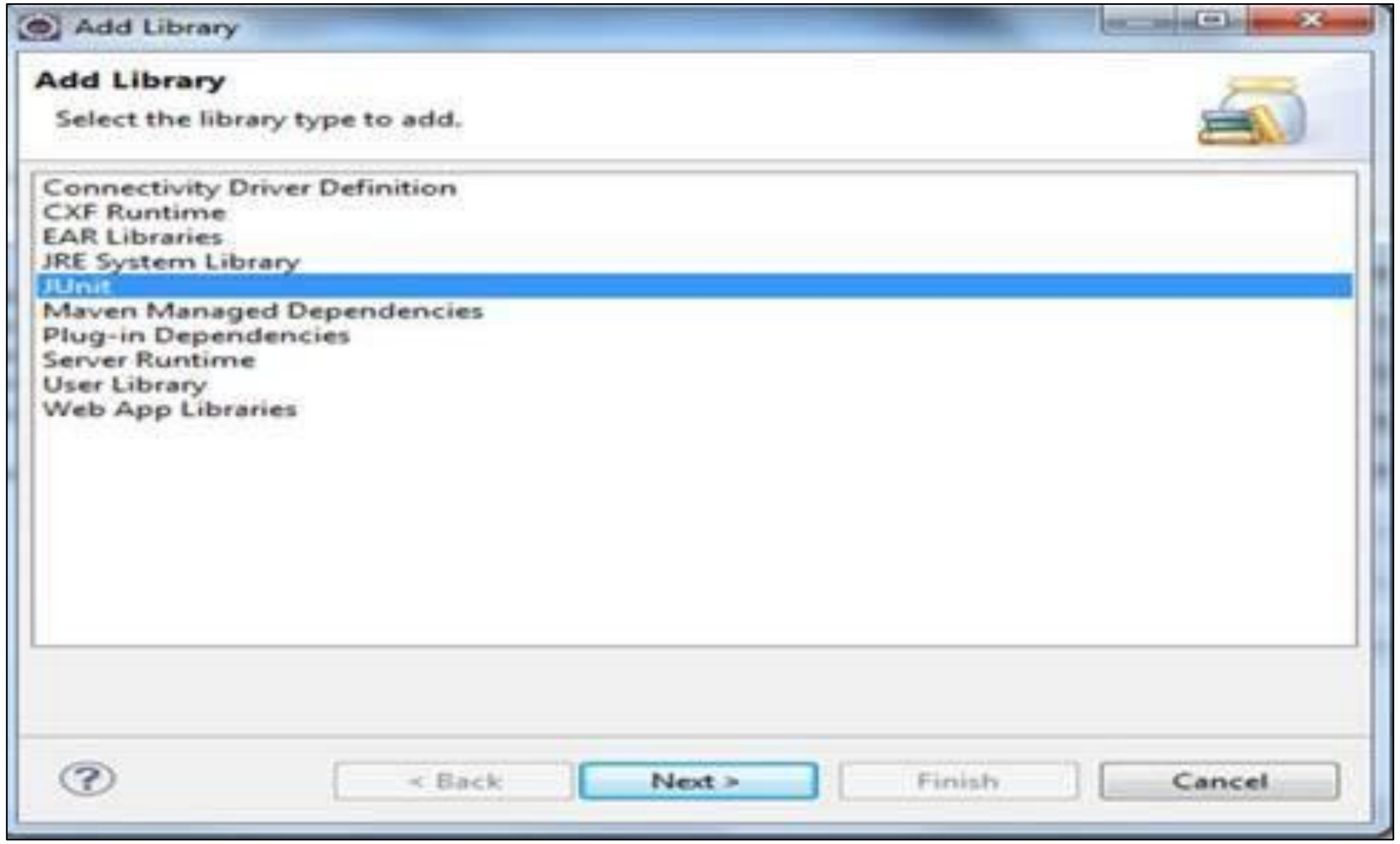
Adding JUnit Library in Java Project



- Go to the Java project and right-click.
- Go to Build Path->'Configure Build Path'.
- Click Libraries.

Image Source: https://www.360logica.com/blog/how-to-use-junit-library-in-java-project/

Adding JUnit Library in Java Project



Add Library

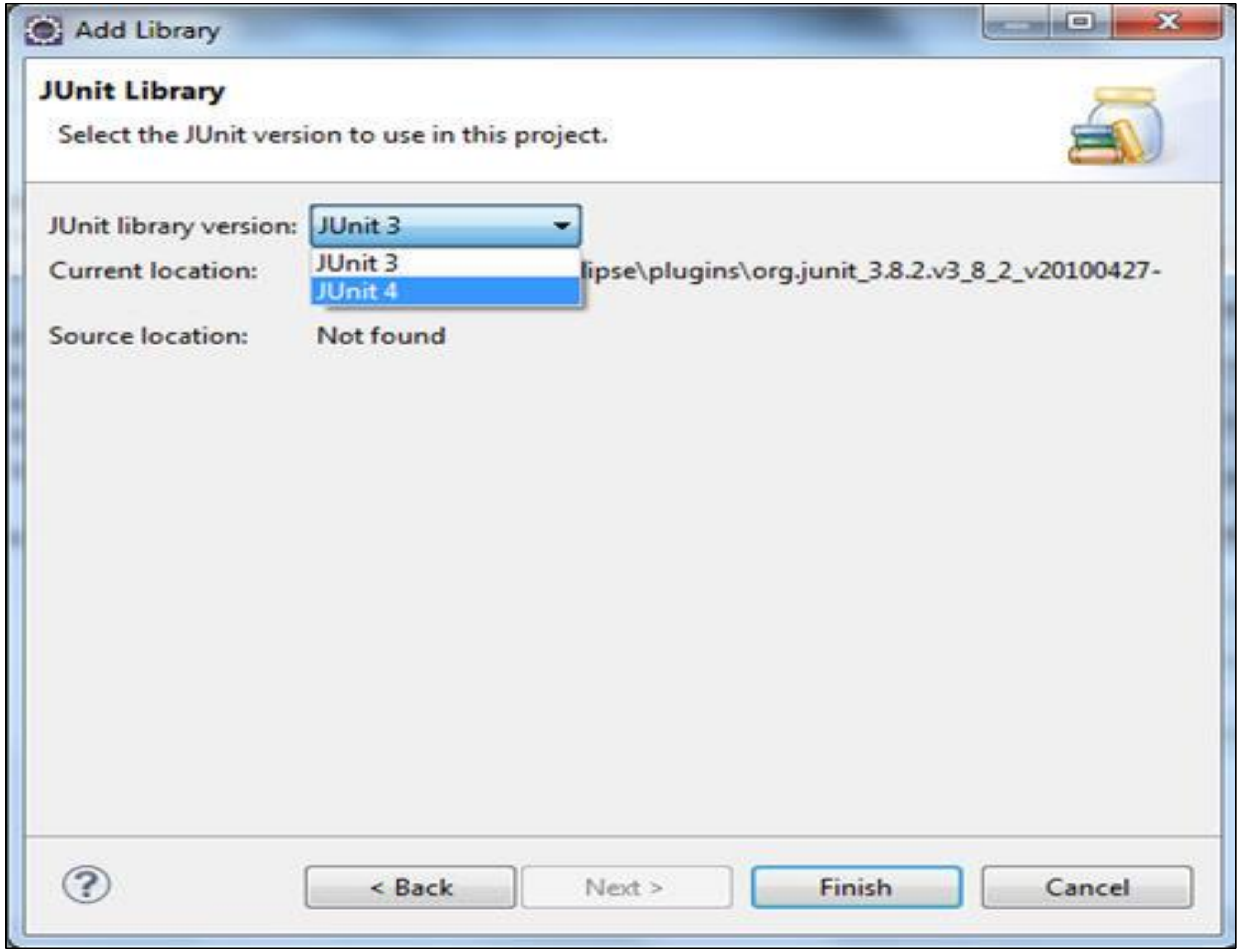
Select the library type to add.

- Connectivity Driver Definition
- CXF Runtime
- EAR Libraries
- JRE System Library
- JUnit
- Maven Managed Dependencies
- Plug-in Dependencies
- Server Runtime
- User Library
- Web App Libraries

- Click ‘Add Library’.
- The user will get the ‘Add Library’ window.
- Select ‘JUnit’ and click ‘Next’.

Image Source: https://www.360logica.com/blog/how-to-use-junit-library-in-java-project/

Adding JUnit Library in Java Project



Add Library

JUnit Library

Select the JUnit version to use in this project.

JUnit library version: JUnit 3

Current location: lipse\plugins\org.junit_3.8.2.v3_8_2_v20100427-

Source location: Not found

Image Source: https://www.360logica.com/blog/how-to-use-junit-library-in-java-project/

- Users will get a screen to select the Junit version to be used in the project.

.

- Select JUnit library version as Junit 5 and click ‘Finish’.

Key Takeaways

- Pro Integration is the act of bringing together smaller components into a single system that functions as one.
- Each test method can contain one or more assertions.
- JUnit provides an assertion library which is used to evaluate the test result.
- JUnit is a unit testing framework for the Java programming language.