Updates in 10.3.3

In this release, we've focused on two key areas:

- Expanding and enhancing the functionality of Unit Test Assistant
- Streamlining desktop workflows

Unit Test Assistant Enhancements

The following features and enhancements were added to Unit Test Assistant.

Support for PowerMock

You can now extend your unit testing frameworks with PowerMock using Unit Test Assistant. You can conveniently mock static methods and constructors by selecting a specific method call or by specifying a mock pattern. See Creating Mocks for details.

Extended Support for Creating Spring Unit Tests

Unit Test Assistant now supports Spring Boot, which makes it easier to create Spring-based applications. Additionally, you can conveniently configure your Spring tests with the ContextConfiguration annotation. See Creating a Spring Unit Test for details.

UI/UX Enhancements

- The UTA context menu now provides quick access to all currently available actions; see Enabling the Unit Test Assistant Interface.
- New action links in the **Recommendations** filter enable you to quickly apply a fix to the test. Additionally, the filter drop-menu enables you to filter recommendations by category; see Executing Unit Tests with Unit Test Assistant.
- You can automatically or manually add missing dependencies to your project; see Adding Required Dependencies.
- We've unified the naming convention replacing the **Quick** action label with **Regular**.
- The Add test case(s) option now allows you to configure the timeout setting when running tests, as well as the depth of test code analysis when objects are initialized.

Streamlined Workflows

The following usability workflow improvements and enhancements were implemented in this release to improve your Jtest experience within the IDE.

Jtest Perspective in Eclipse

The Jtest perspective is now available in Eclipse. This provides you with a set of views that are crucial for working with Jtest on your desktop; see Opening the Jtest Perspective in Eclipse.

Importing Example Projects

We've added new example projects that can help you become familiar with Jtest features and functionalities. The examples are also easier to import into your IDE; see Importing the Jtest Example Projects for details.

Dedicated Run Configurations for Collecting Code Coverage

We've added dedicated run configurations to the Eclipse context menu to facilitate collecting code coverage for JUnit test and Java applications; see Executing and Collecting Coverage for JUnit Tests and Collecting Coverage for Java Applications in Eclipse.

Enhanced Workflow for Test Configurations

We've made it easier for you to interact with test configurations when working with Jtest on your desktop. Now you can easily duplicate and customize an existing test configuration on DTP, as well as run a test configuration right from your IDE Preference page; see Creating Custom Test Configurations and Setting the Active Test Configuration for details.

Extended Support for IDEs and Build Systems

- Support for Eclipse 4.7
- Support for IntelliJ 2017.1
- Support for Gradle 4.1

Updated Code Analysis Rules

- PB.API.CMMT
- GC.OSTM
- OPT.ILUG
- UC.UIMPORT
- JDBC.CDBC - deprecated and replaced with BD.RES.LEAKS

Updated Configurations

- CERT Secure Coding

Resolved Bugs and FRs

Bug/FR ID	**Description**
JT-68803	Code Duplicates performance degradation
JT-68862	AssertionFailedError being reported in "Setup Problems" section
JT-68806	Parsing errors on console during FlowAnalysis
JT-48351	JSON file generated by IntelliJ IDEAcontainsemptyresourceleveltag
JT-68865	Not all parameterized test cases show coverage