TestLink Eclipse Test Plan Execution Plugin

Table of Contents

TestLink Eclipse Plugin Manual

Introduction

Place holder for the introduction.

Getting Started

This section assumes the reader has the TestLink web-application installed and has enabled and configured the TestLink API capabilities.

Place holder for plugin installation instructions.

Configuration:

Access the preferences windows in Eclipse and

- Project Name
  The default project that is open by TestLink eclipse plugin at startup. The default is empty and must be configured by the user before the viewer can be used.

- Dev Key
  TestLink API Acces key. See TestLink API documentation. Note: This as of 1.8.2 is the value can be found by accessing the TestLink database user table and viewing values in the source key field. If no value is filled in it may have to be set by database update. The TestLink API may describe other methods.

- TestLink API URL
  The link to the TestLink API URL. The preferences screen provides a default example. See TestLink API documentation for more information.

- User Login Name
  The user login name is not used by the TestLink API for login but it is used for test creation. It assigns this user as the creator of the test case.

- Plan Prepare Class
  There are two principal interfaces that must be implemented by the user in order to use the automated test runner. The first is the test plan preparation interface. A enclipse plugin TestPlan class object is passed to an adjust(TestPlan plan) method. This method call is used by the implementer to add additional test cases to the plan and to register TestCaseExecutors with each test case in the plan. TestCaseExecutor is the second interface the user must implement to use the plugin. As the name implies the executor object that is registered with the test case is executed by the eclipse TestLink plugin test runner. See the javadoc provided with the TestLink

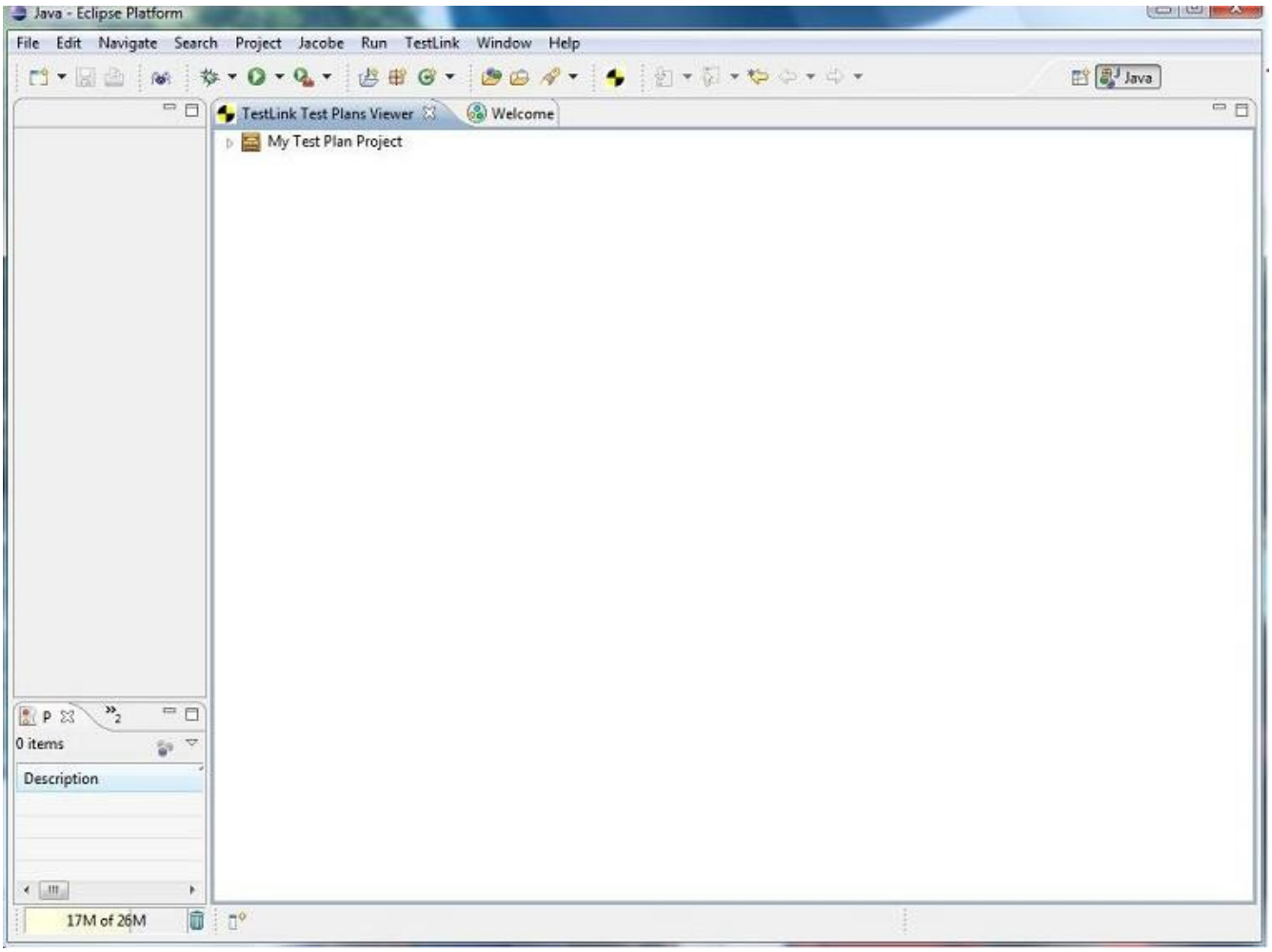
API Java Client for additional information. Also, an example implementation is included in the document.

- Report Results
  This flag is on or off. If the flag is checked then the plugin automatically records the test results for a test case using the TestLink API.

- External Path (Optional)
  Made available to TestPlanPrepare. Interface as a convinience to the implementers. In most cases the implementer of a prepare will need configuration data from a file. This variable can point to the file of the implementers choice.

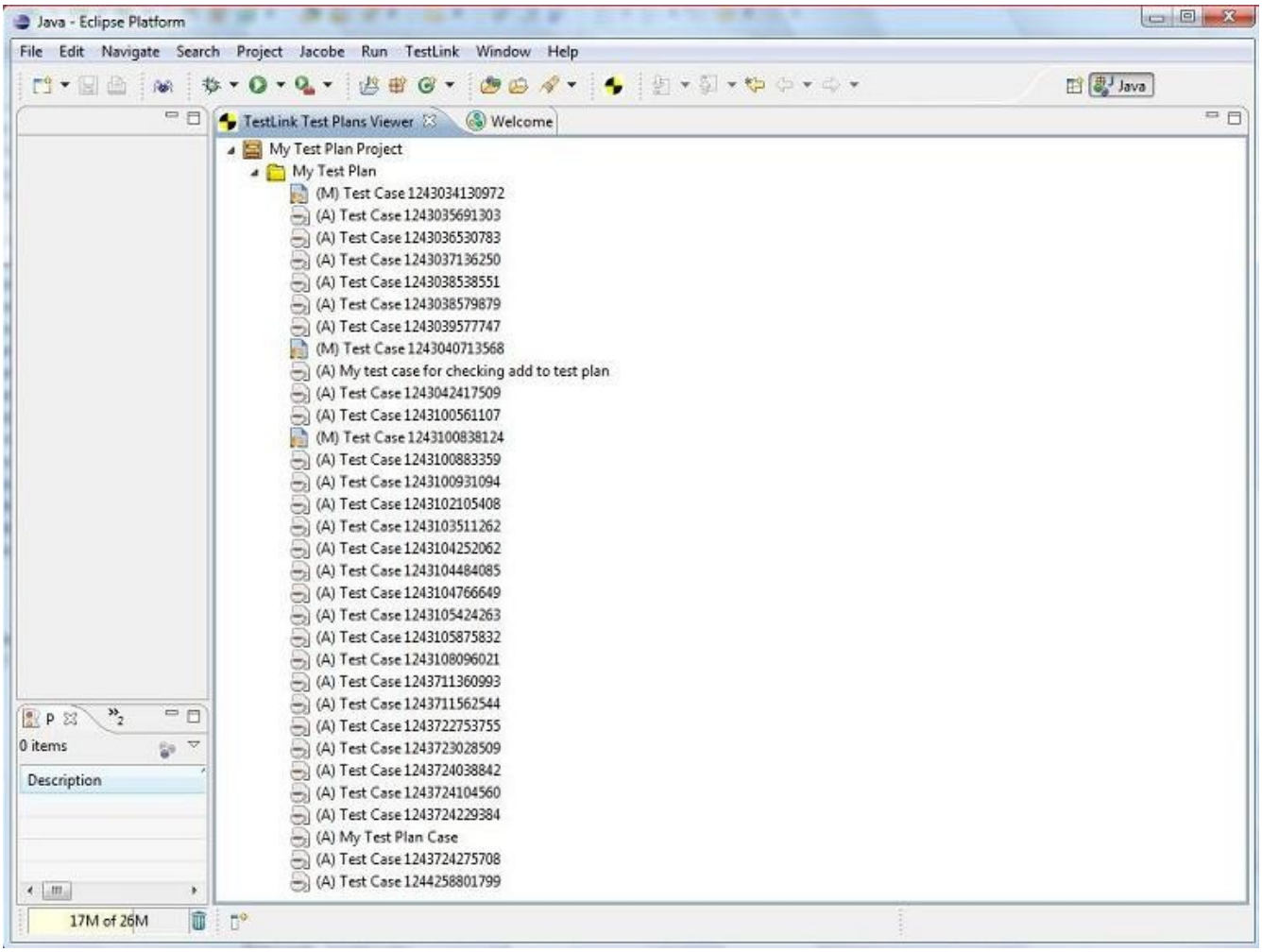
TestLink Test Plan Viewer

Place holder to explain screen



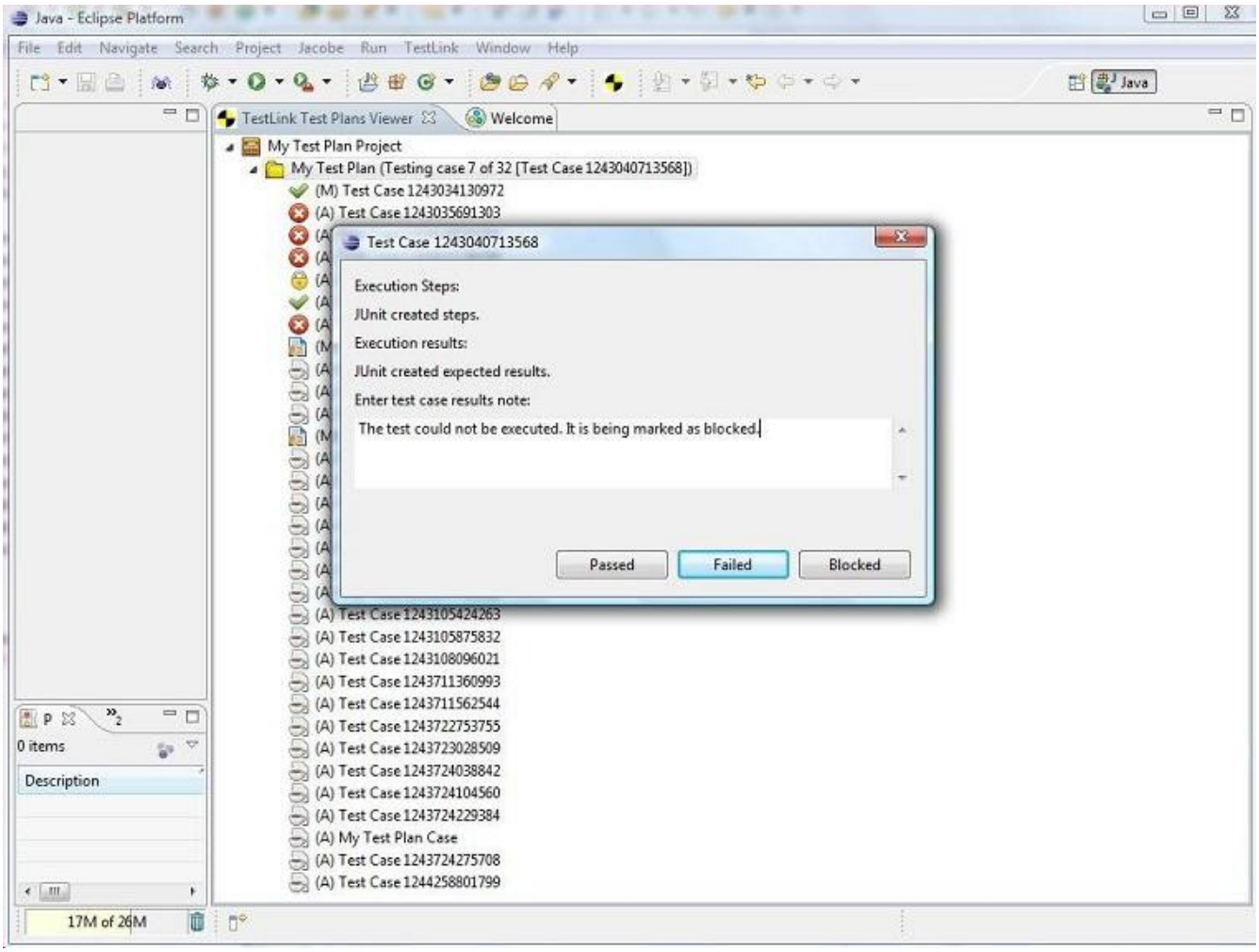
Expanding a Test

Place holder to explain section



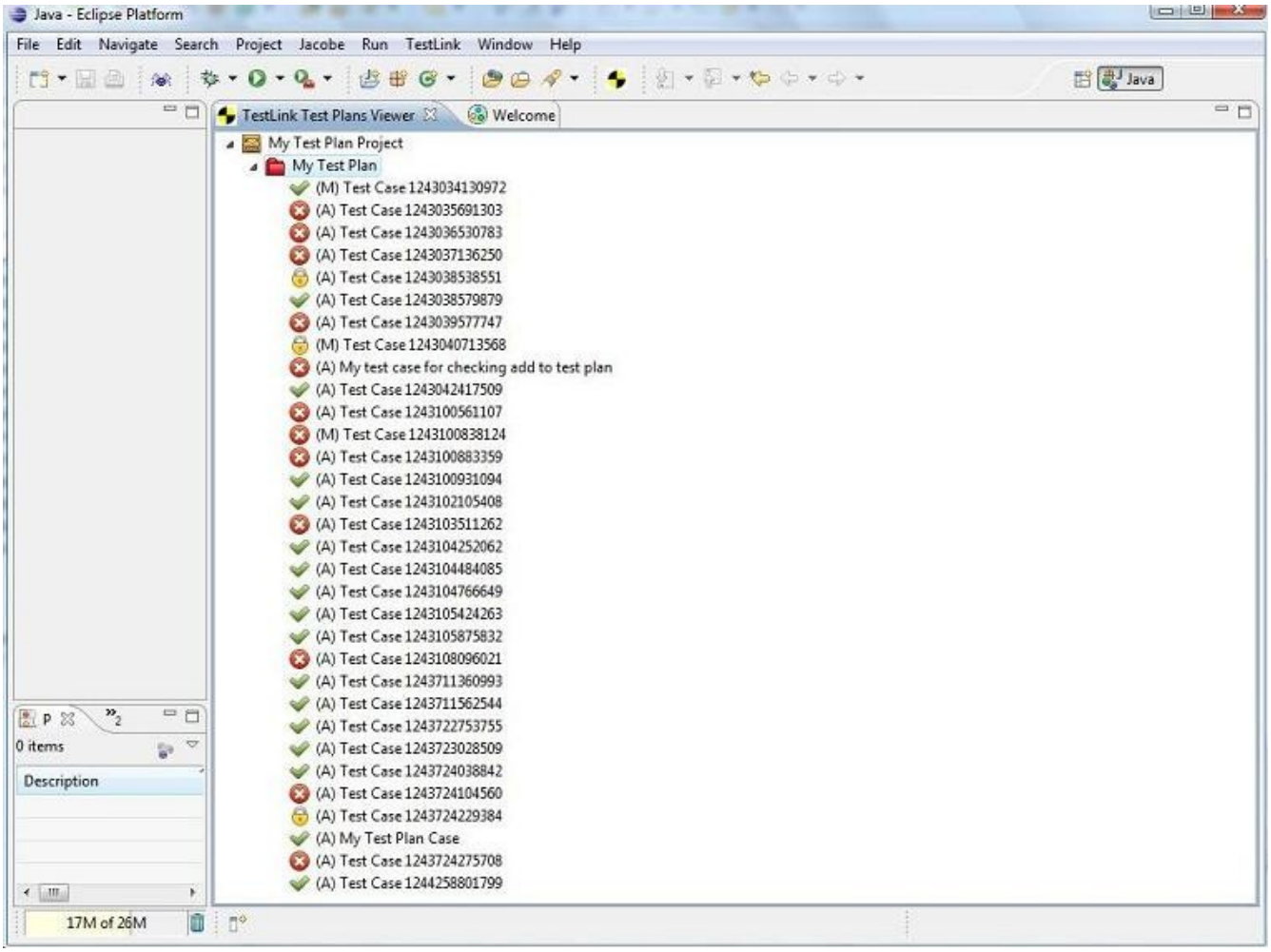
Execute Test

Place holder to explain that manual test can be mixed with automated testing



Test Results

Place holder to explain test results



Interfaces

Place holder for general discussion

TestCasePrepare

```
/*
 * Daniel R Padilla
 *
 * Copyright (c) 2009, Daniel R Padilla
 *
 * This copyrighted material is made available to anyone wishing to use, modify,
 * copy, or redistribute it subject to the terms and conditions of the GNU
 * Lesser General Public License, as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
 * or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
 * for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with this distribution; if not, write to:
 * Free Software Foundation, Inc.
 * 51 Franklin Street, Fifth Floor
 * Boston, MA  02110-1301  USA
 */
package org.dbfacade.testlink.tc.autoexec;


import org.dbfacade.testlink.api.client.TestLinkAPIClient;
import org.dbfacade.testlink.api.client.TestLinkAPIException;


/**
 * An implementer of this interface is expected to do the following:
 *
 * Required:
 *
 * 1)  Assign the executors for all the automated test cases in the test plan.
 *
 * Optional:
 *
 *  1) Add additional test cases that are not listed in the plan. Especially
helpful
 *  when the list of test cases is maintained and added to using programatic means.
 *
 *  2) Replace the test cases with different TestCase interface implementations.
 *  A good example would be when the default ExecutableTestCase could be replaced
 *  with CustomTestCase which extends ExecutableTestCase.
 *
 *  3) etc..
```

```
 *
 * @author Daniel Padilla
 *
 */
public interface TestPlanPrepare
{

      /**
       * Optionally made available by callers to the interface
       *
       * @param directory
       */
      public void setExternalPath(String path);

      /**
       * Optionally made available by callers to the interface
       *
       * @param user
       */
      public void setTCUser(String user);


      /**
       * Make changes to the contents of the test plan and test cases.
       *
       * @param plan
       * @return A test plan which has had the executors set for each test case.
       *
       */
      public TestPlan adjust(
            TestLinkAPIClient apiClient,
            TestPlan plan) throws TestLinkAPIException;
}
```

TestCaseExecutor

```
/*
 * Daniel R Padilla
 *
 * Copyright (c) 2009, Daniel R Padilla
 *
 * This copyrighted material is made available to anyone wishing to use, modify,
 * copy, or redistribute it subject to the terms and conditions of the GNU
 * Lesser General Public License, as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
 * or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser General Public License
 * for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with this distribution; if not, write to:
```

```
 * Free Software Foundation, Inc.
 * 51 Franklin Street, Fifth Floor
 * Boston, MA  02110-1301  USA
 */
package org.dbfacade.testlink.tc.autoexec;


import org.dbfacade.testlink.api.client.TestLinkAPIException;


public interface TestCaseExecutor
{

      // Results
      public static final short RESULT_UNKNOWN = -1;
      public static final short RESULT_PASSED = 0;
      public static final short RESULT_FAILED = 1;
      public static final short RESULT_BLOCKED = 2;

      // States
      public static final short STATE_READY = 100;
      public static final short STATE_RUNNING = 101;
      public static final short STATE_BOMBED = 102;
      public static final short STATE_COMPLETED = 103;
      public static final short STATE_RESET = 104;

      /**
       * Get the state of the execution
       *
       * @return
       */
      public short getExecutionState();

      /**
       * Set the new state of the executor
       *
       * @param newState
       */
      public void setExecutionState(
            short newState);

      /**
       * Return the result state of the test case execution.
       *
       * @return The result of the test case. Implementers should set the initial
status to UNKNOWN.
       */
      public short getExecutionResult();

      /**
       * Set the results of the test from an external source.
       *
       * @param result
       */
```

```
	public void setExecutionResult(
		short result);

	/**
	 * Information about the results of the execution.
	 *
	 * @return Information about the results of the execution.
	 */
	public String getExecutionNotes();

	/**
	 * Set the execution notes
	 *
	 * @param notes
	 */
	public void setExecutionNotes(String notes);

	/**
	 * Execute the test case that has been passed into the execute method.
	 *
	 * @param testCase
	 * @throws TestLinkAPIException
	 */
	public void execute(
		TestCase testCase) throws TestLinkAPIException;
}
```