Jenkins TestLink Plug-in Tutorial

Bruno P. Kinoshita

César Fernandes de Almeida

Jenkins TestLink Plug-in Tutorial

Bruno P. Kinoshita
César Fernandes de Almeida
French Translation.: Flóreal Toumikian
French Translation: Olivier Renault
Review and suggestions on how explain different topics of this tutorial: Oliver Merkel

Abstract

The Jenkins TestLink Plug-in Tutorial is intended to provide a better understanding on how to use the plug-in to integrate Jenkins and TestLink. It is an Open Source project, so contributions and suggestions are welcome.

Table of Contents

1. An Introduction to Jenkins

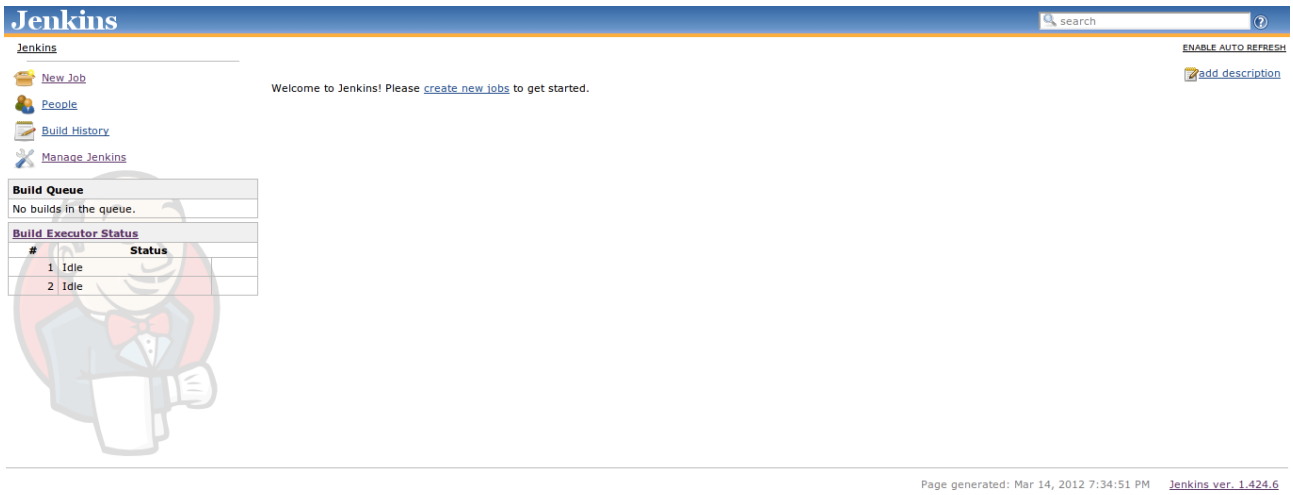
Jenkins (née Hudson) is a continuous integration server written in Java. It can be downloaded from http://www.jenkins-ci.org [http://www.jenkins-ci.org] as an executable war. It means that you can run it with **java -jar jenkins.war**. Jenkins comes with an embedded Servlet Container (Winstone) but you also have the option to deploy the war to an application server like Tomcat, Jetty, JBoss, etc. Jenkins does not use any database to store its configuration. Jenkins uses XStream to save data as XML files.

With Jenkins you can create, monitor and schedule jobs. There are plug-ins for almost anything that you may think about, from different SCMs (git, mercury, SVN, CVS) to plug-ins for integrating your Jenkins with Selenium, Gerrit, TestLink and other tools.

Here is a summarized list of Jenkins features that is available in Jenkins Wiki [http://wiki.jenkins-ci.org/display/JENKINS/TestLink+Plugin]:

1. Easy installation
2. Easy configuration
3. Change set support
4. Permanent links
5. RSS/E-mail/IM Integration
6. After-the-fact-tagging
7. JUnit/TestNG test reporting
8. Distributed builds
9. File fingerprinting
10. Plugin Support

For this guide we suggest you to use the LTS (long term support) version of Jenkins. At the moment that this guide is being written, the current version is *1.424.6*. However it is very likely that what you will learn here will work with newer versions of Jenkins. Jenkins development team does a great job not only developing cool features, but also keeping backward compatibility between versions.



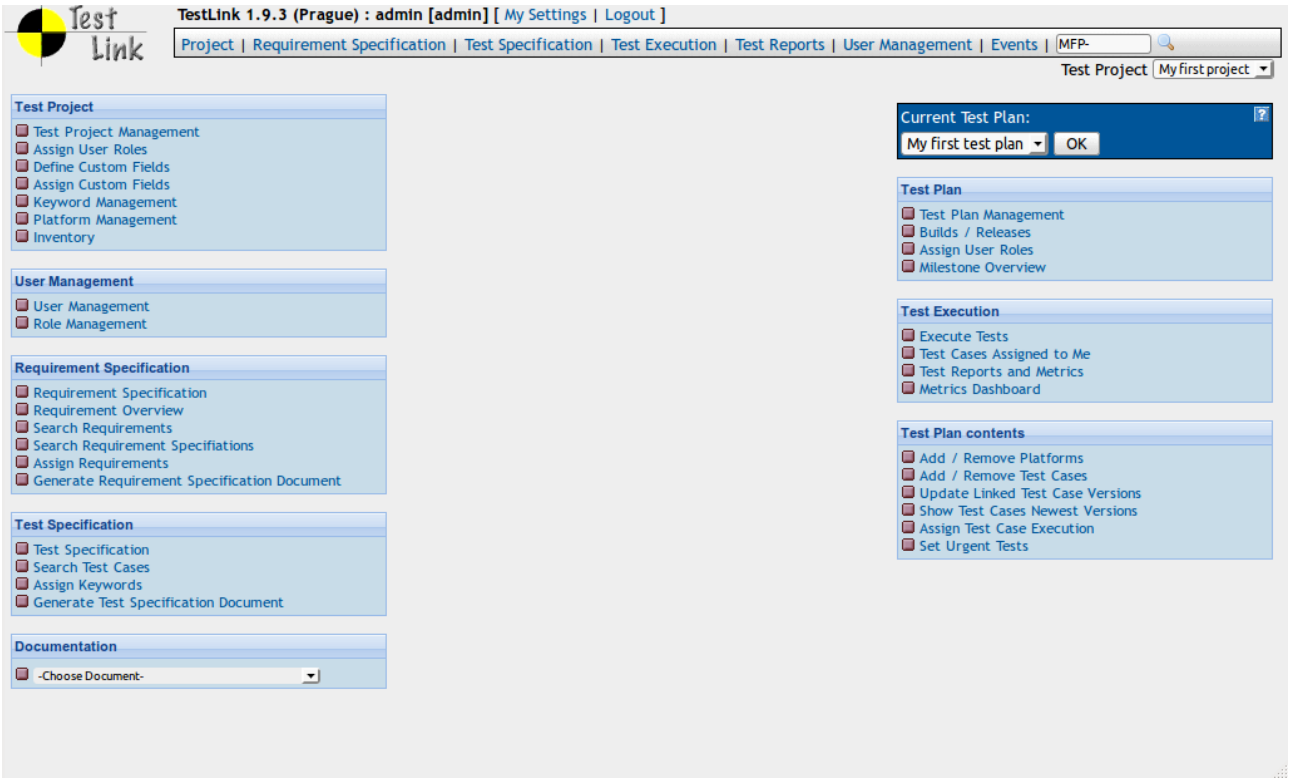
Jenkins is licensed under the MIT License and its code is hosted at GitHub - http://github.com/jenkinsci [http://github.com/jenkinsci].

2. An Introduction to TestLink

TestLink is an open source test management tool written in PHP. In order to install TestLink you will need a HTTP server with PHP 5 and a database. The databases currently supported by TestLink out of the box are MySQL, Postgre SQL and MS SQL, although there are users that managed to use Oracle too. You can download it from http://www.teamst.org [http://www.teamst.org].

In TestLink you have, among other features, Test Plans, Requirement management, Baselines, Custom Fields, Test Suite with Test Cases and Reporting. For external access, there is an XML-RPC API available (it is not enabled by default). Other two nice features in TestLink are the versioning of the some entities, like Test Case and Requirements, and the ability to import and export data in different formats.

There are bindings for the TestLink XML-RPC API available for Java, Perl, PHP, Ruby and other programming languages. The integration between Jenkins and TestLink is done using TestLink Java API - http://testlinkjavaapi.sourceforge.net [http://testlinkjavaapi.sourceforge.net].



At the moment that this guide is being written the latest version of TestLink is *1.9.3*. While the Jenkins version is not a big issue for this integration, the same cannot be said about TestLink. This guide is intended for Testlink 1.9.3 XML-RPC API. In case there are other versions available at the moment that you are reading

this guide, please consult the plug-in Wiki [http://wiki.jenkins-ci.org/display/JENKINS/TestLink+Plugin] for further information on this.

TestLink is licensed under the GPL License and its code is hosted at gitorious - http://www.gitorious.org/testlink-ga/testlink-code [http://www.gitorious.org/testlink-ga/testlink-code].

3. Jenkins and TestLink Integration

The integration between Jenkins and TestLink can be achieved with Jenkins TestLink Plug-in. The plug-in resides in Jenkins, being able to use some of its nice features like executing multiple jobs, scheduling jobs, distributed execution and a plethora of plug-ins. To retrieve the data from TestLink for tests, as well as to report the test case execution status to TestLink, the plug-in uses TestLink XML-RPC API.

This way, while Jenkins handles the execution of jobs and tasks such as downloading source code from an SCM, TestLink maintains the structure of your tests, as well as other assets such as Test Plan, Custom Fields, Reports and Baselines.

We will see how to configure our environment for this integration throughout the next chapters. While it is good to have a written explanation on how this integration works, probably it is a lot easier to understand how it works with a hands-on. The rest of this guide explains how to configure Jenkins and TestLink to run automated tests from a sample test project. This test project is a Java project that uses Maven and TestNG and you can download the source code from https://github.com/tupilabs/jenkins-testlink-plugin-tutorial [https://github.com/tupilabs/jenkins-testlink-plugin-tutorial] (as well as the DocBook source for this book).

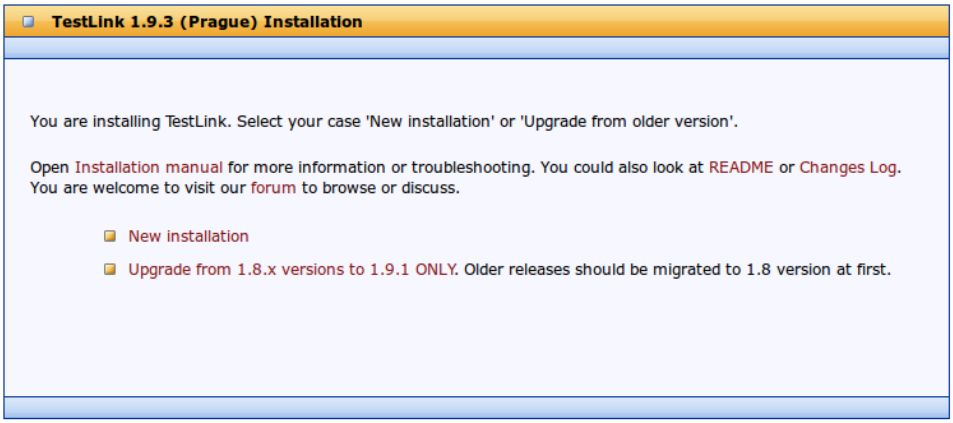
The current version of Jenkins TestLink Plug-in while this book is being written is *3.0.2*.

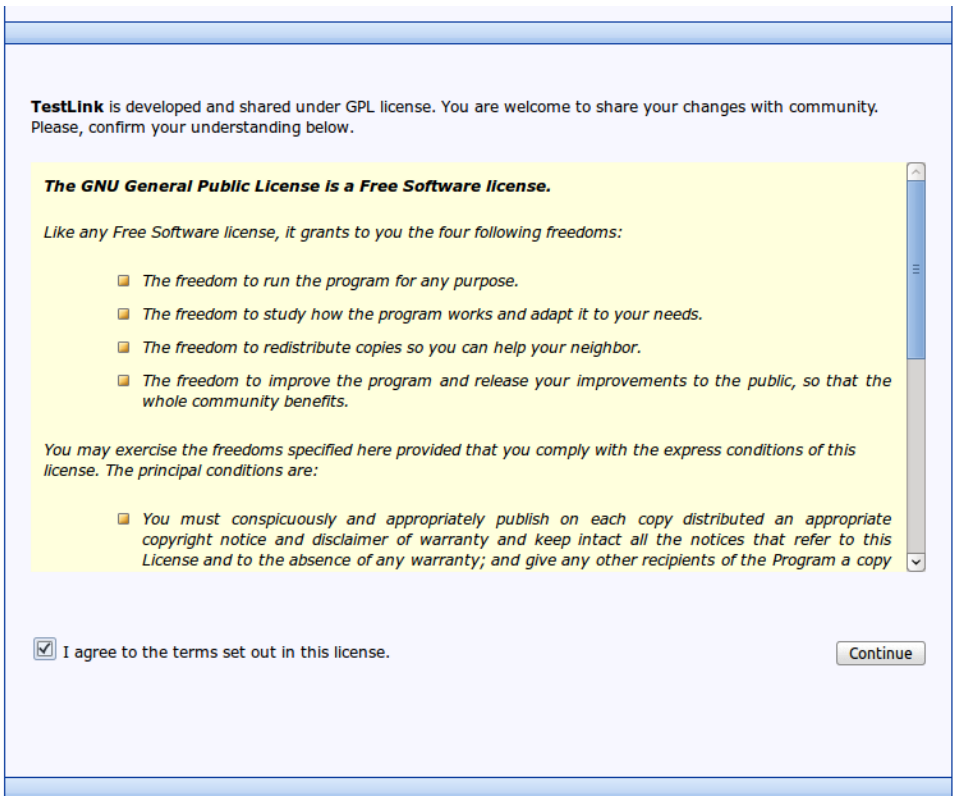
Jenkins TestLink Plug-in is licensed under the MIT License and its code is stored in github - http://github.com/jenkinsci/testlink-plugin [http://github.com/jenkinsci/testlink-plugin].

4. TestLink Configuration

4.1 Installing TestLink

In this part of the tutorial we will show how to install and configure TestLink. Let's start by downloading *testlink-1.9.3.tar.gz* from http://www.teamst.org. Decompress it with **tar -zxvf testlink-1.9.3.tar.gz**. Move the directory created to your HTTP server root directory and open http://localhost/testlink-1.9.3 [http://localhost/testlink-1.9.3] in your browser.





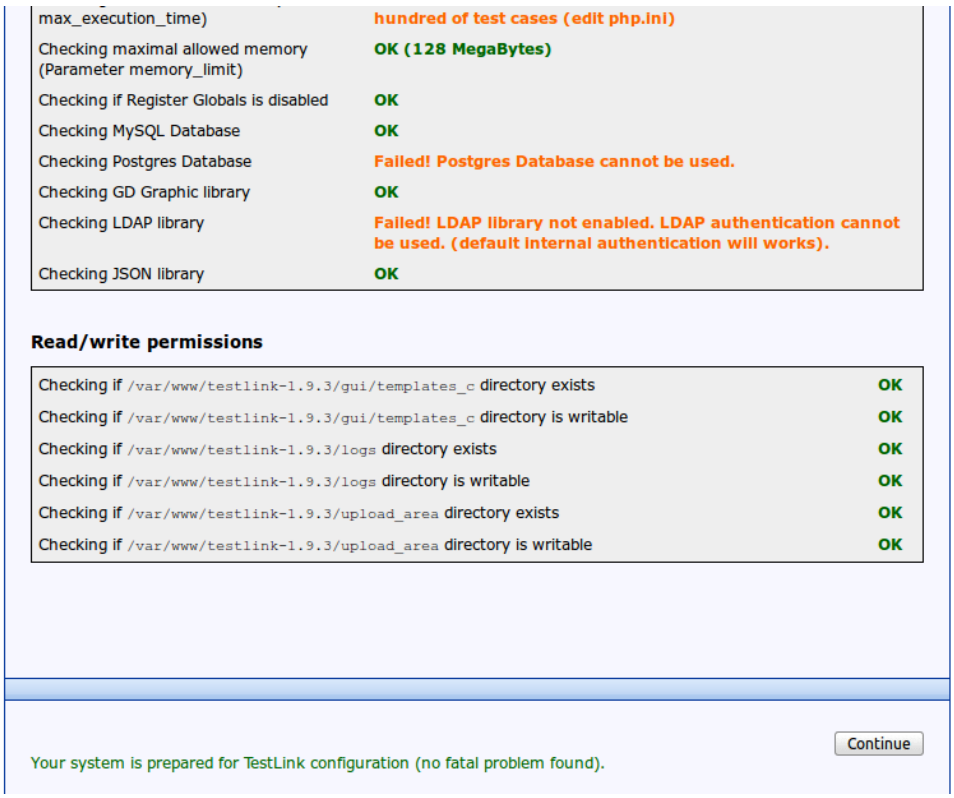
Now the installation wizard will guide you through the rest of the installation. But before going on, we need to create a database in MySQL.

mysql> create database testlink;

The next step is to create a user that TestLink will use to access the database.

mysql> grant all privileges on testlink.* to 'testlink' identified by 'testlink';

mysql> flush privileges;



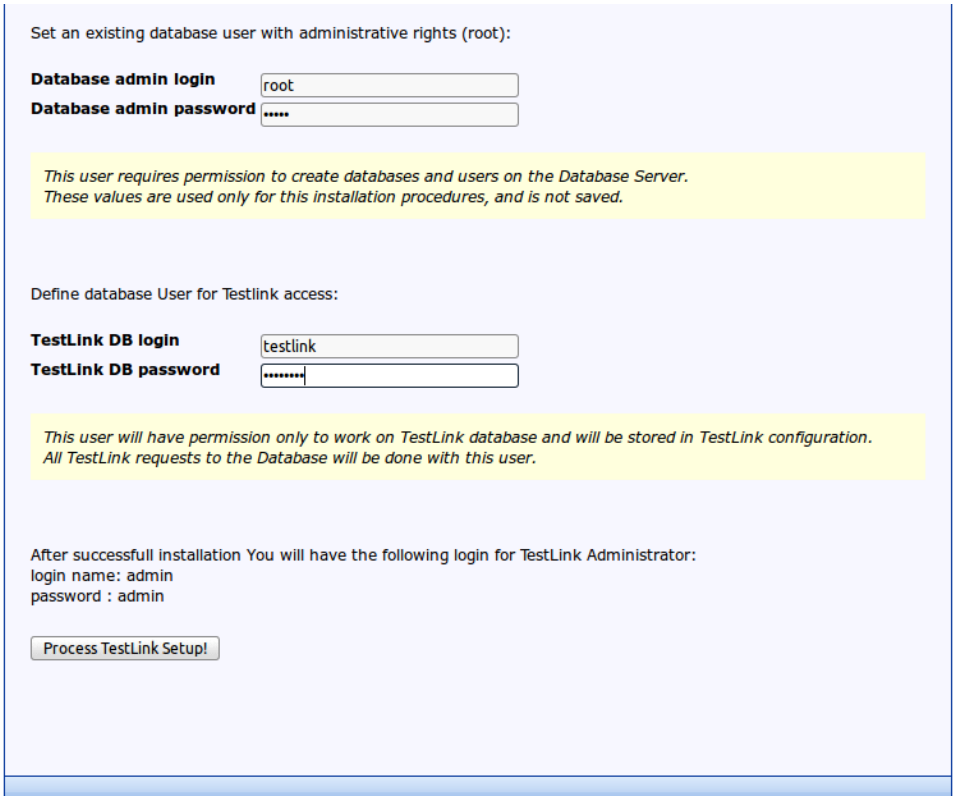
max_execution_time)	**hundred of test cases (edit php.ini)**
Checking maximal allowed memory (Parameter memory_limit)	**OK (128 MegaBytes)**
Checking if Register Globals is disabled	**OK**
Checking MySQL Database	**OK**
Checking Postgres Database	**Failed! Postgres Database cannot be used.**
Checking GD Graphic library	**OK**
Checking LDAP library	**Failed! LDAP library not enabled. LDAP authentication cannot be used. (default internal authentication will works).**
Checking JSON library	**OK**

Read/write permissions

Checking if `/var/www/testlink-1.9.3/gui/templates_c` directory exists	**OK**
Checking if `/var/www/testlink-1.9.3/gui/templates_c` directory is writable	**OK**
Checking if `/var/www/testlink-1.9.3/logs` directory exists	**OK**
Checking if `/var/www/testlink-1.9.3/logs` directory is writable	**OK**
Checking if `/var/www/testlink-1.9.3/upload_area` directory exists	**OK**
Checking if `/var/www/testlink-1.9.3/upload_area` directory is writable	**OK**

Continue

Your system is prepared for TestLink configuration (no fatal problem found).



Set an existing database user with administrative rights (root):

Database admin login root

Database admin password •••••

This user requires permission to create databases and users on the Database Server.
These values are used only for this installation procedures, and is not saved.

Define database User for Testlink access:

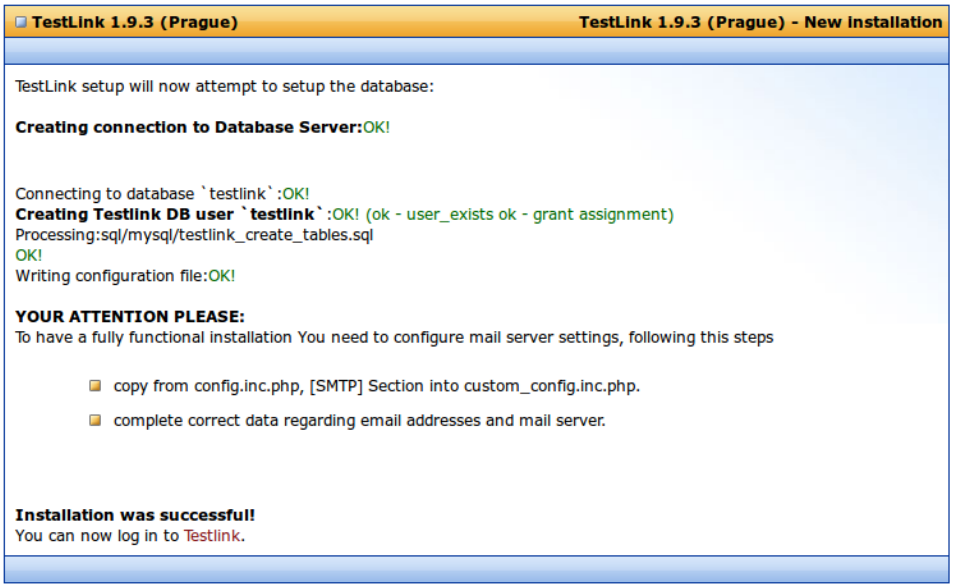
TestLink DB login testlink

TestLink DB password ••••••••

This user will have permission only to work on TestLink database and will be stored in TestLink configuration.
All TestLink requests to the Database will be done with this user.

After successfull installation You will have the following login for TestLink Administrator:
login name: admin
password : admin

Process TestLink Setup!



If everything worked out correctly you should be asked to log in with user admin and password admin. The examples in this tutorial require you to have a user with administrator rights in TestLink.

By default, the XML-RPC comes disabled in TestLink. Let's enable it by editing config.inc.php, located in TestLink root folder.

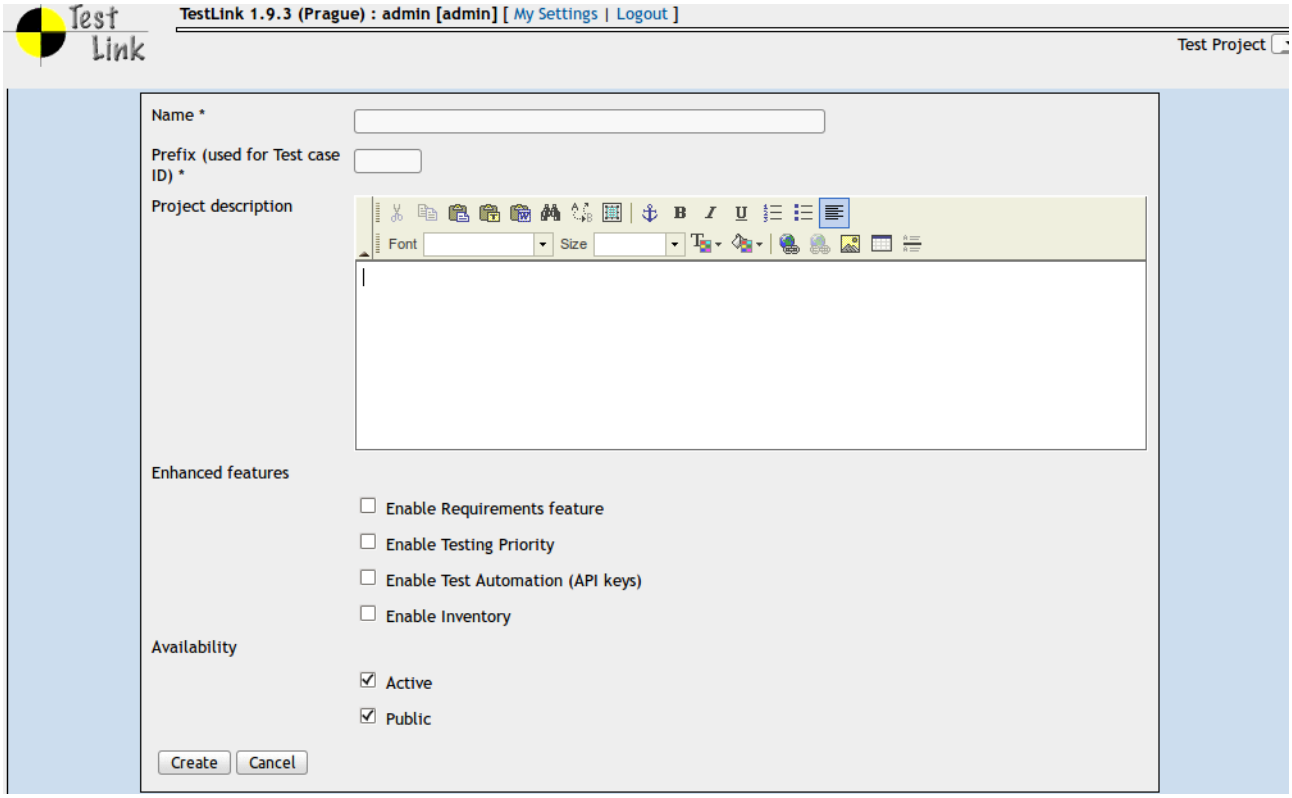
$tlCfg->api->enabled = TRUE;

Finally, let's make sure that the attachments retrieved from the database are ordered by its ID. This way, the order of attachments will be preserved in TestLink. We could use the date that the attachment was inserted in database, however the precision of the date_added column is in seconds, what could lead to inconsistencies in the way that attachments are displayed in TestLink.

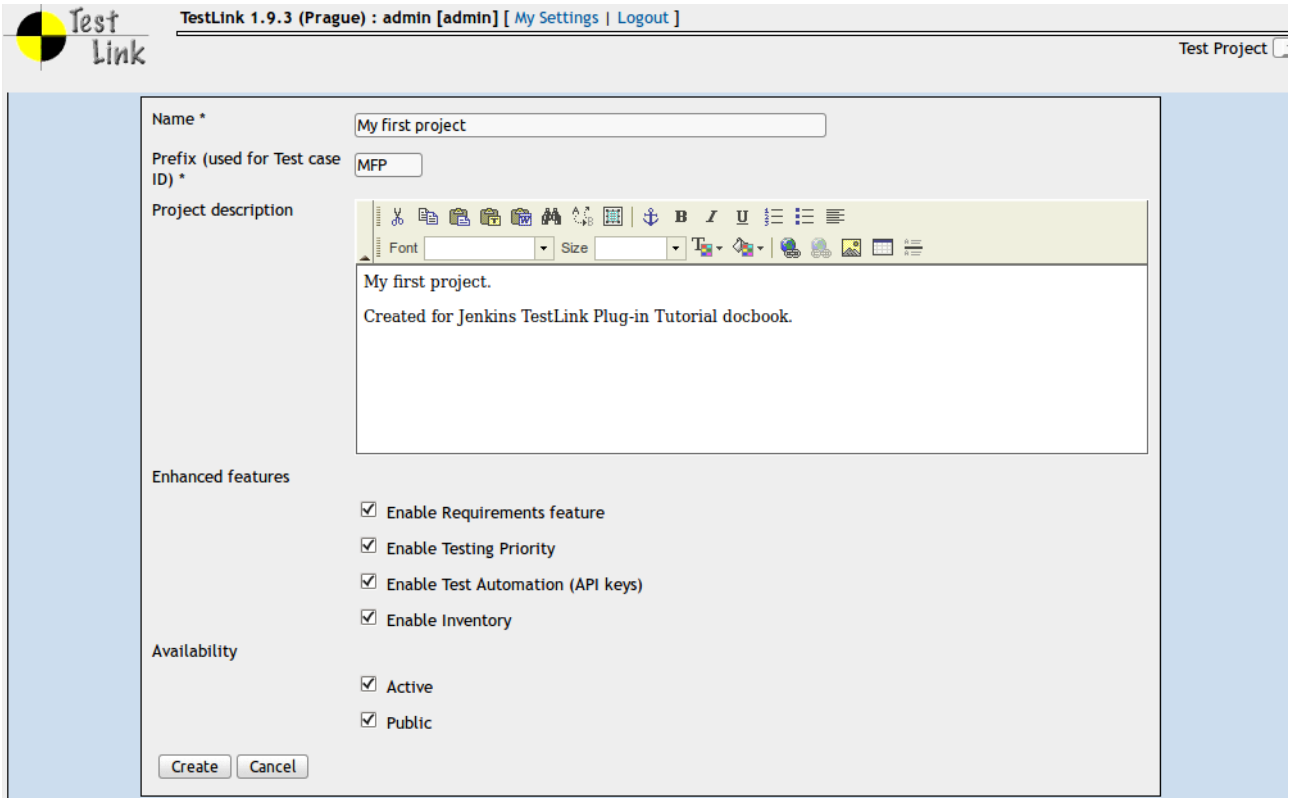
$g_attachments->order_by = " ORDER BY id ASC ";

4.2 Creating a Test Project

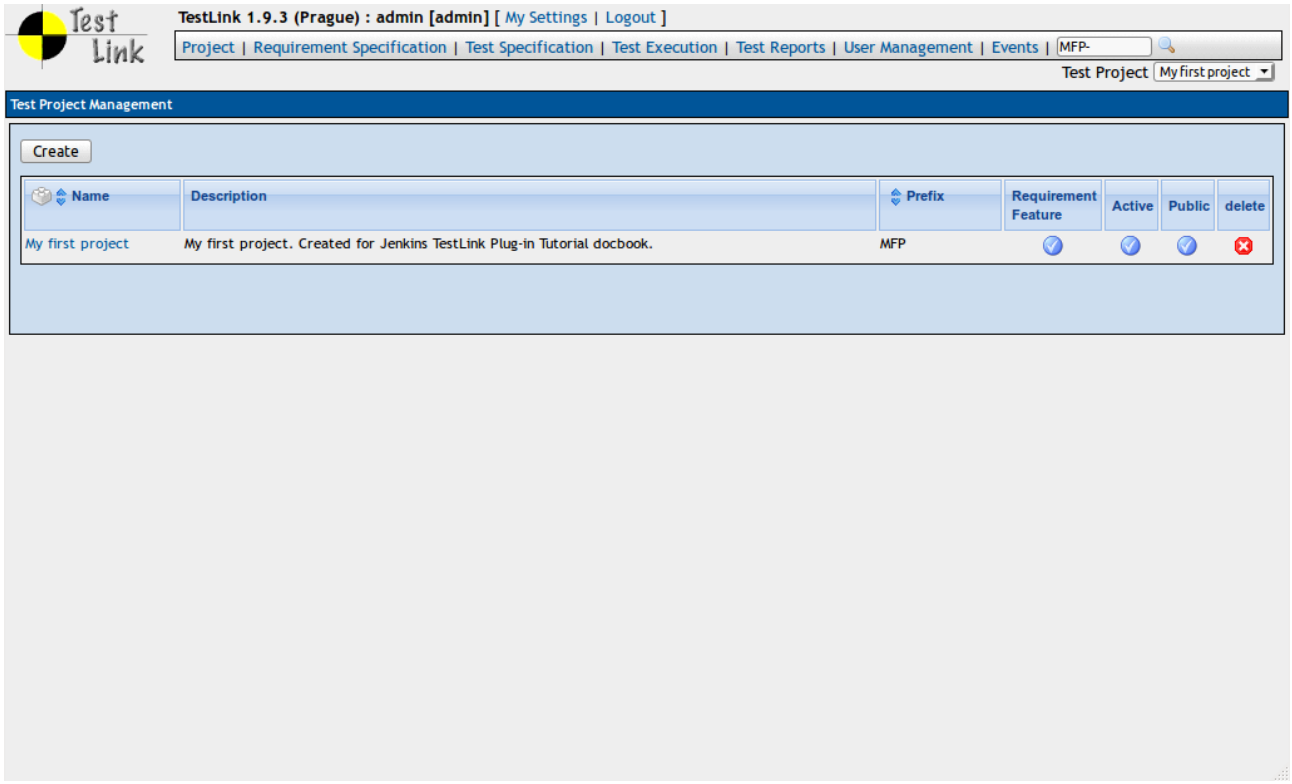
When you log in by the first time in TestLink it is showed to you a form to create a Test Project. After creating a test project you will be able to create test plans, requirements, specify and execute your tests.



Create a test project with name My first project, prefix MFP and make sure the following options are checked: Enable Requirements feature, Enable Testing Priority, Enable Test Automation (API keys), Enable Inventory, Active and Public. Click on Create button.



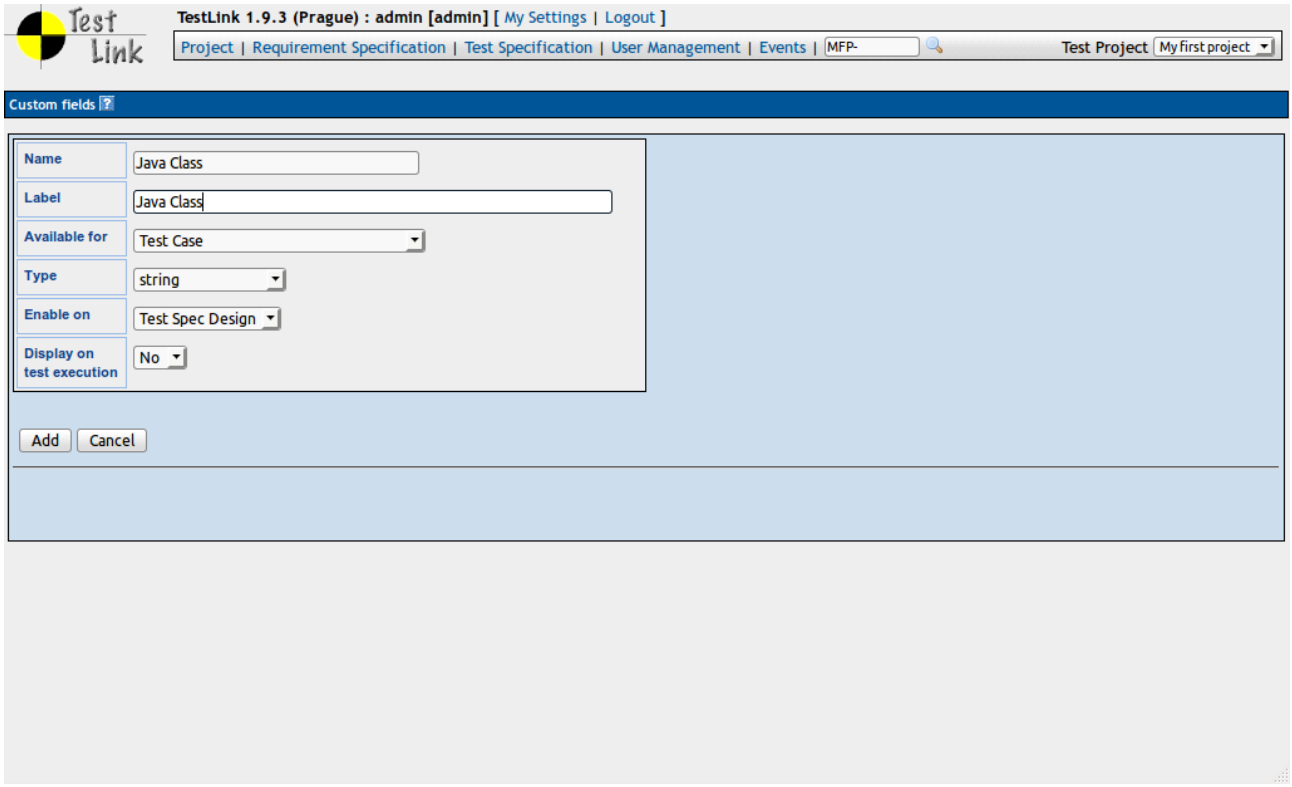
If the following screen is not displayed, review your previous steps or consult TestLink documentation for further assistance.

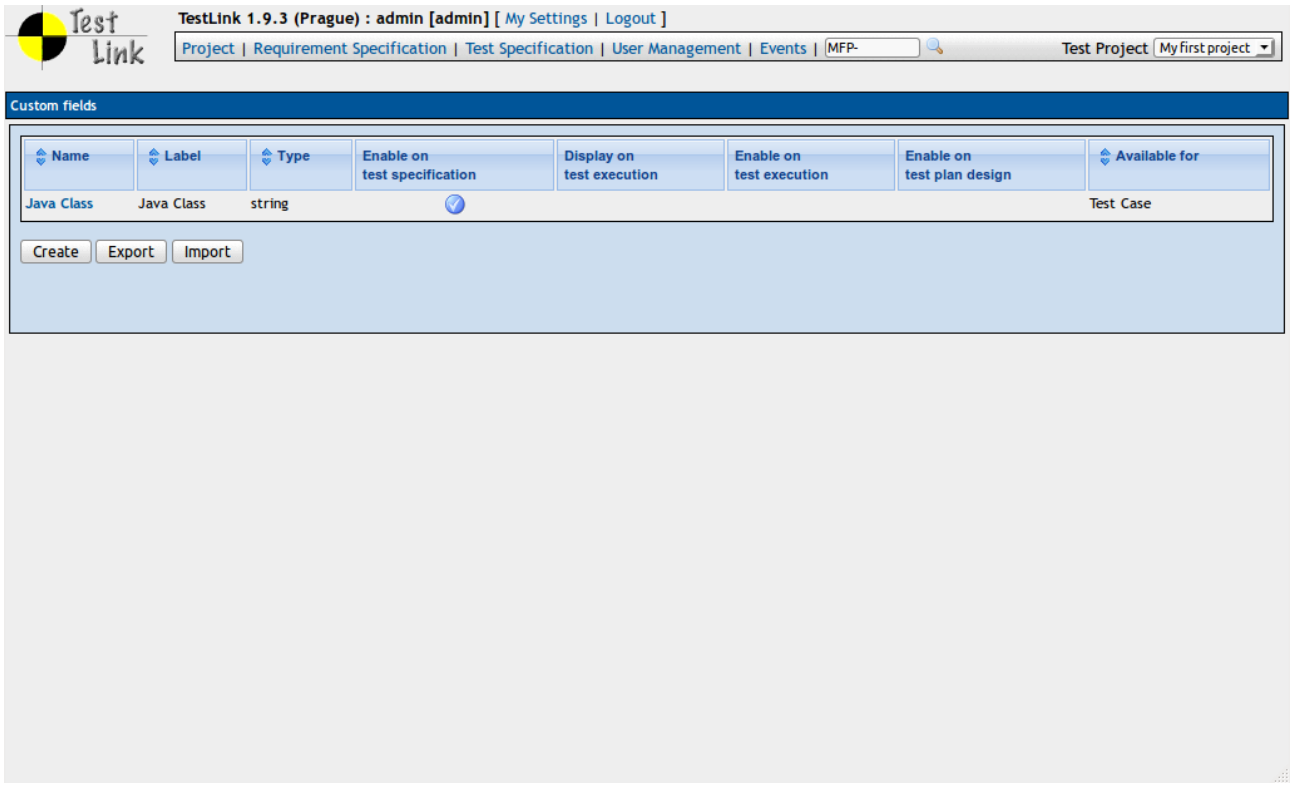


4.3 Creating and assigning a Custom Field

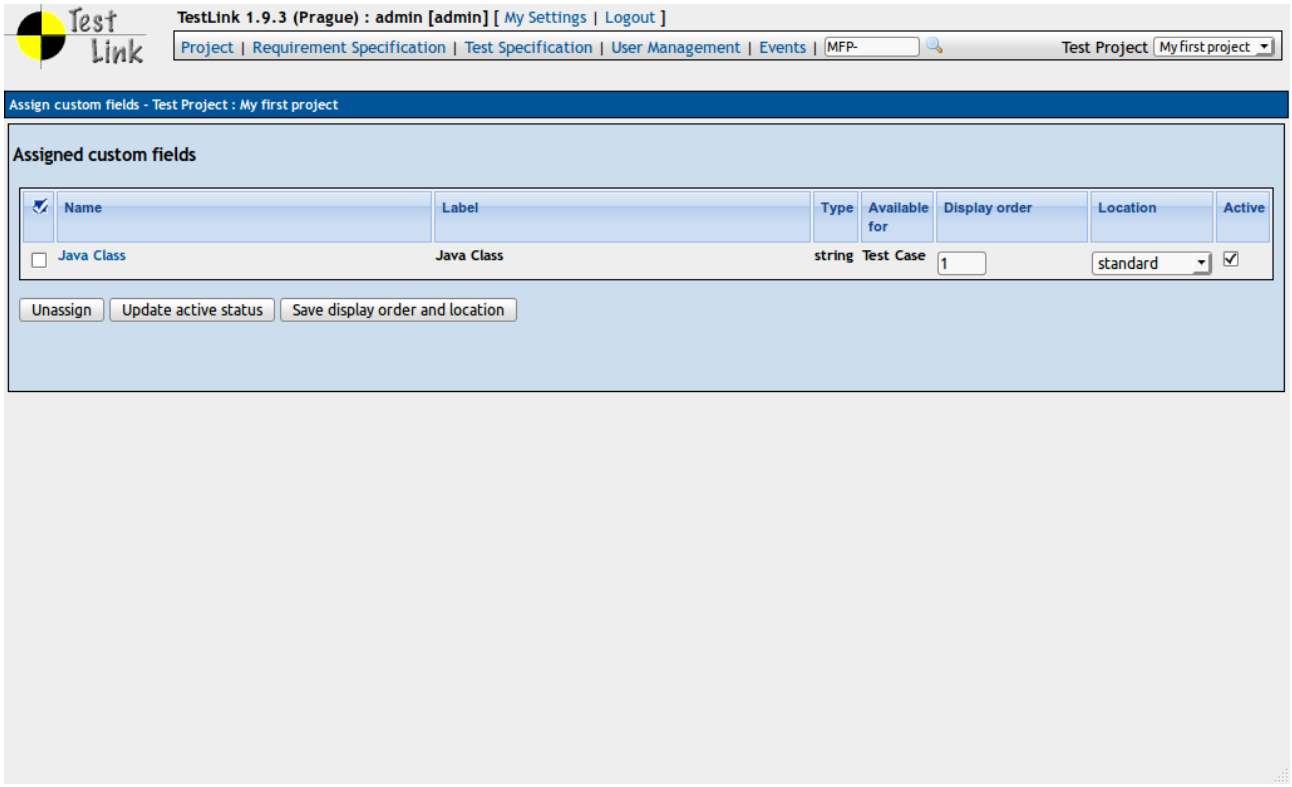
Click on the Project item of the top menu to be redirected to the main screen. Now we will create the custom field used for automation. The plug-in uses this custom field's value to link a test case in TestLink with a test result from your Jenkins build.

Click on Define Custom Fields under the Test Project options box. Now create a custom field using the name Java Class, label Java Class, available for Test Case, type string, enable on Test Spec Design and display on test execution No. The plug-in retrieves the custom fields by its name and not by the value in its label.





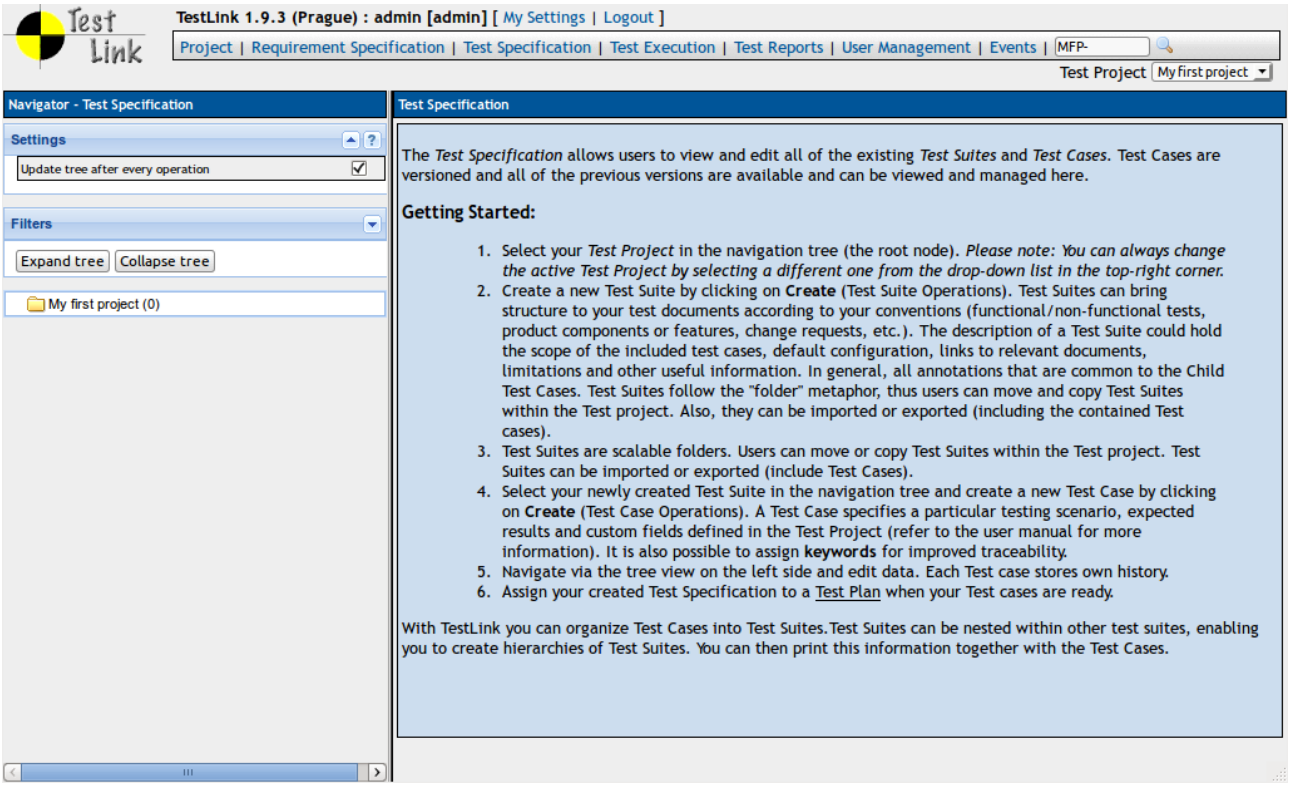
The last step now is to assign this custom field to be used in our test project. Go back to the main screen and click on Assign Custom Fields. It will take you to a screen with the list of the available custom fields. Select the Java Class custom field and click on the Assign button.



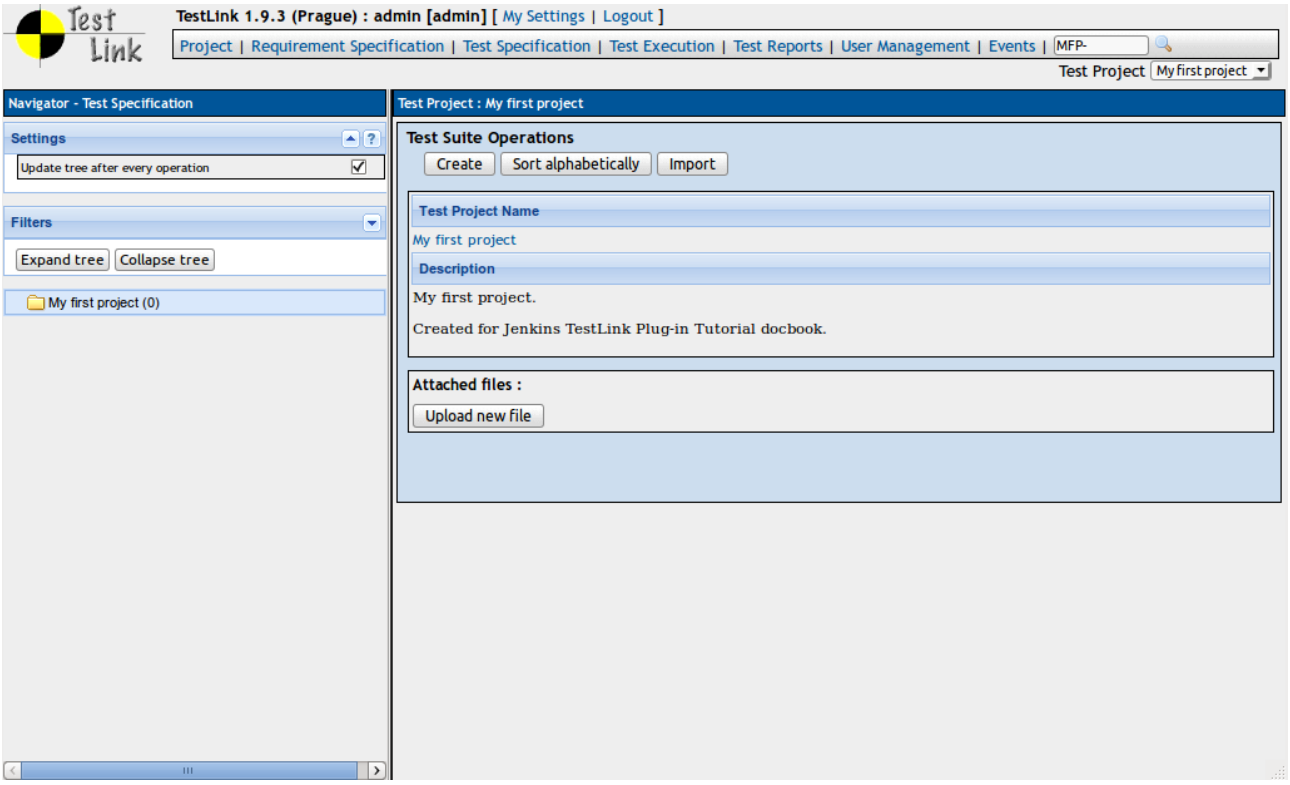
4.4 Specifying Test Suite and Test Cases

For those who work with tests this part may be slightly easier than the previous sections. What we are going to do now is to create a test suite and some test cases. In TestLink your test cases are kept under a test suite which is, by its turn, under a test project.

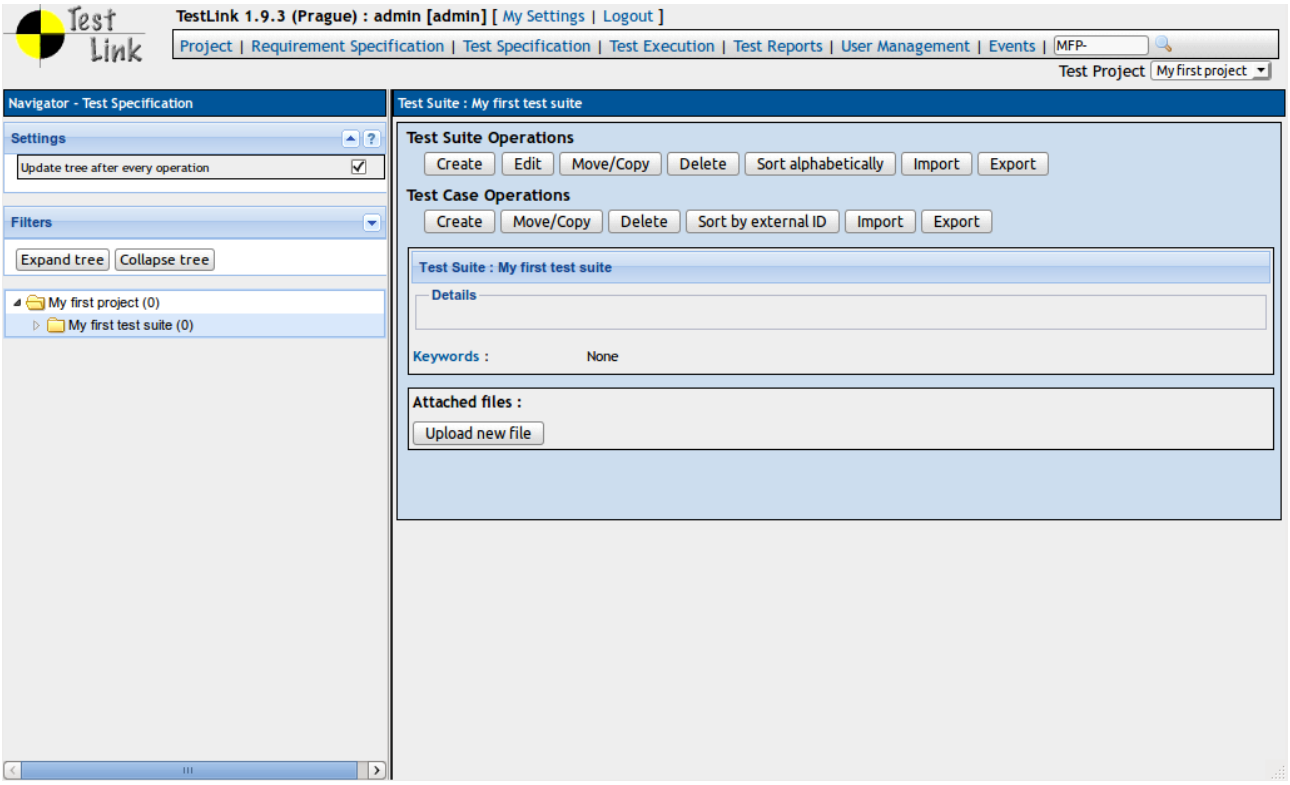
Back to the main screen, in the top menu you will see the option Test Specification. Click on this option. The test specification screen is quite simple. On the left you have the navigator, with the settings, filter and the tree of test suites and test cases. Start clicking on your test project to see the test suite operations available.

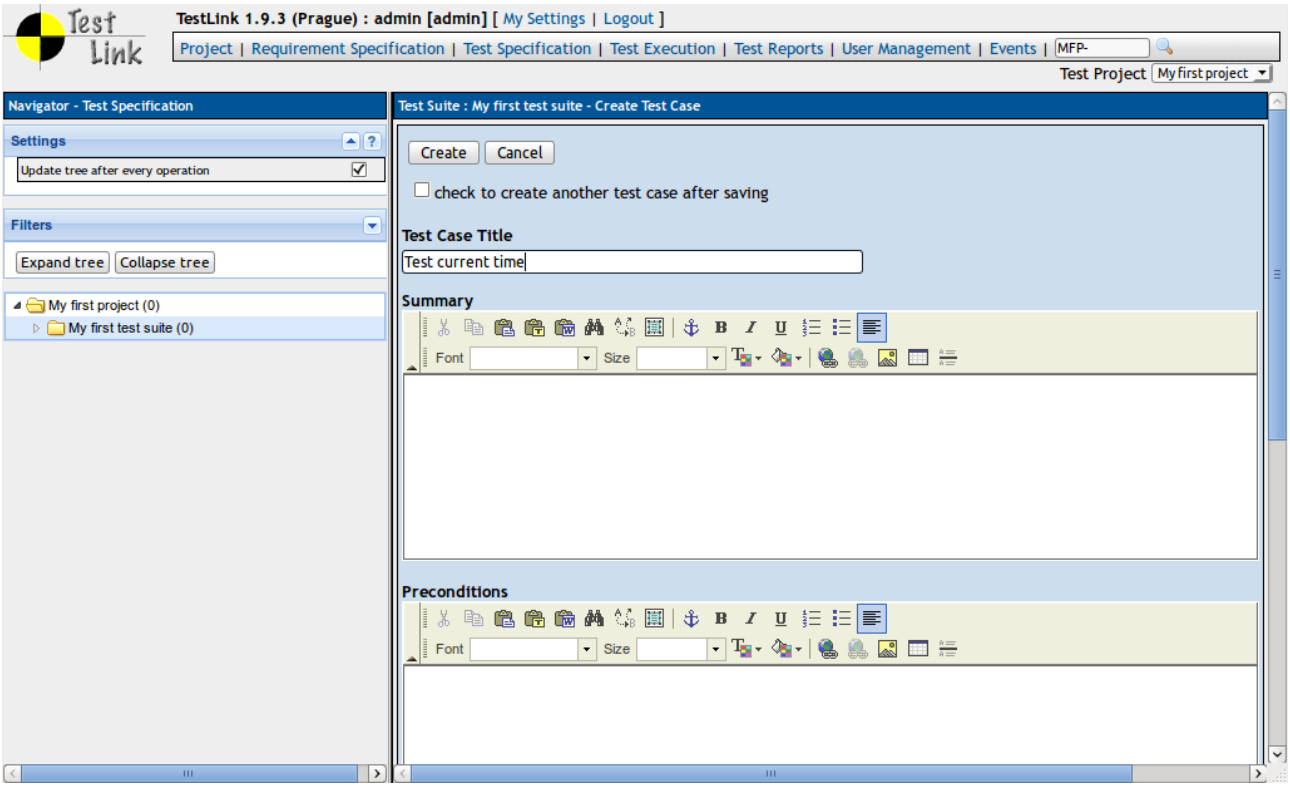


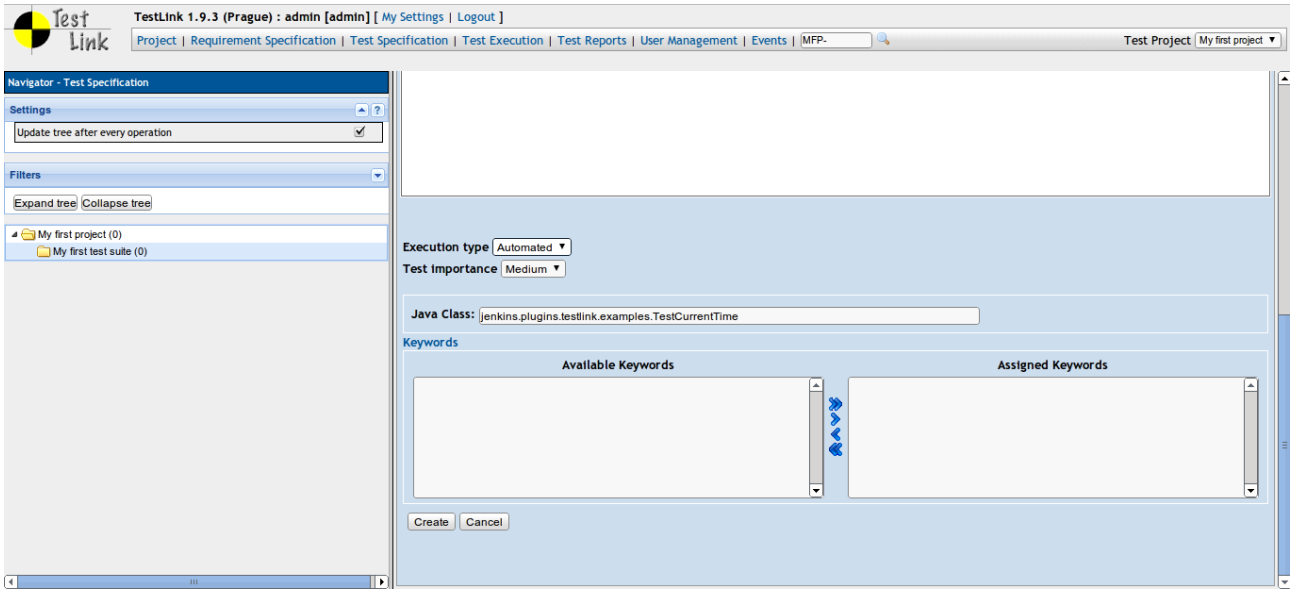
Create a test suite with any name that you want. This field is not important for this tutorial. Once created, your test suite will be displayed in the tree on the left. Now click on the test suite in the tree to see the available test case operations.

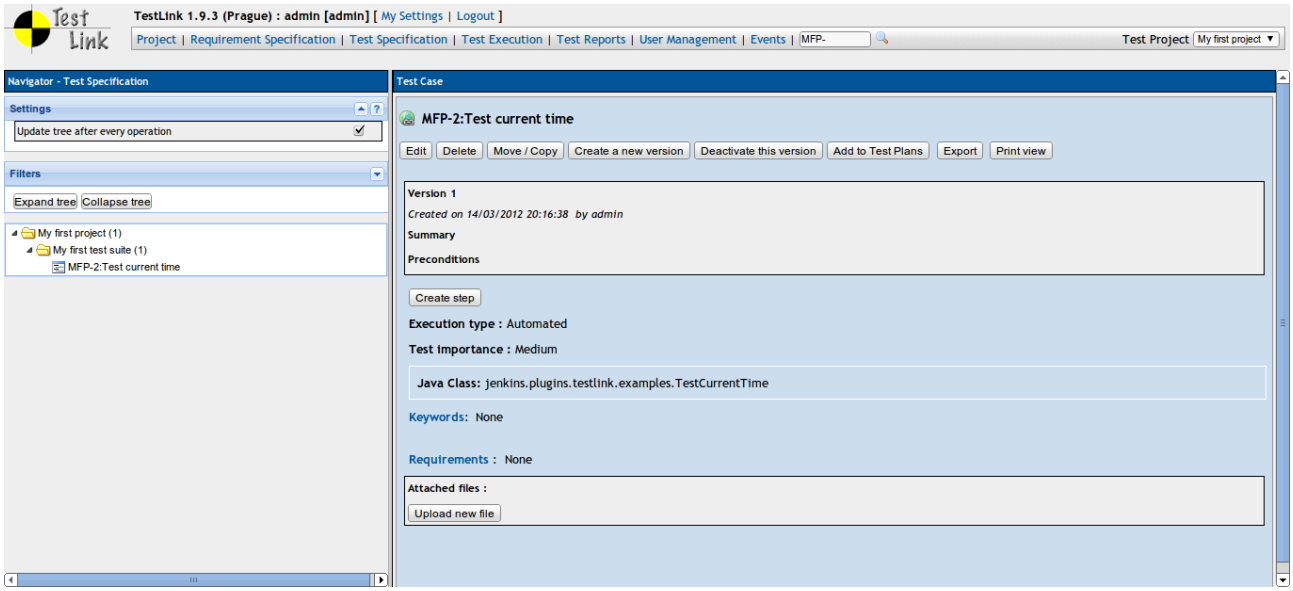


Create a test case with any title or summary. The important information on this screen for the automation are the execution type and the Java class (custom field created in the previous step). In execution type select Automated and for Java class fill with the java class jenkins.plugins.testlink.examples.TestCurrentTime (present in the example project).









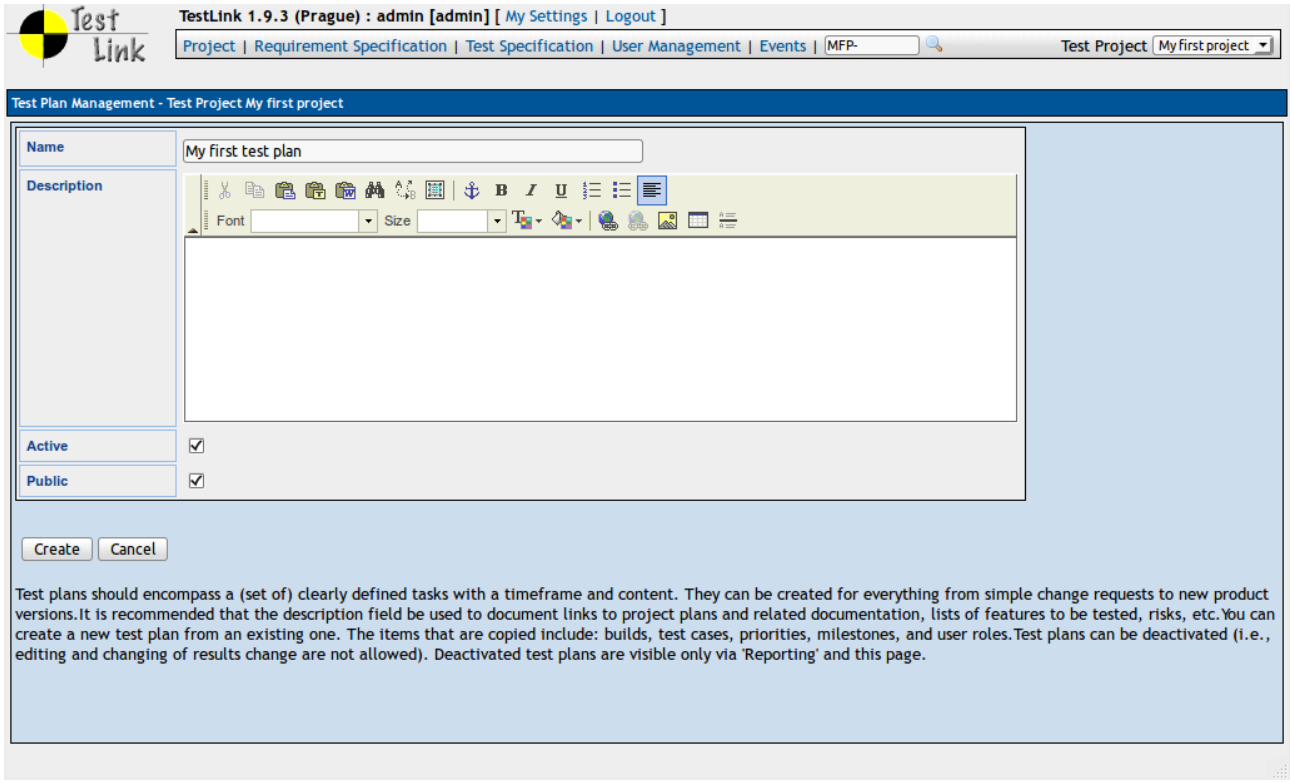
4.5 Create a Test Plan and add the Test Cases

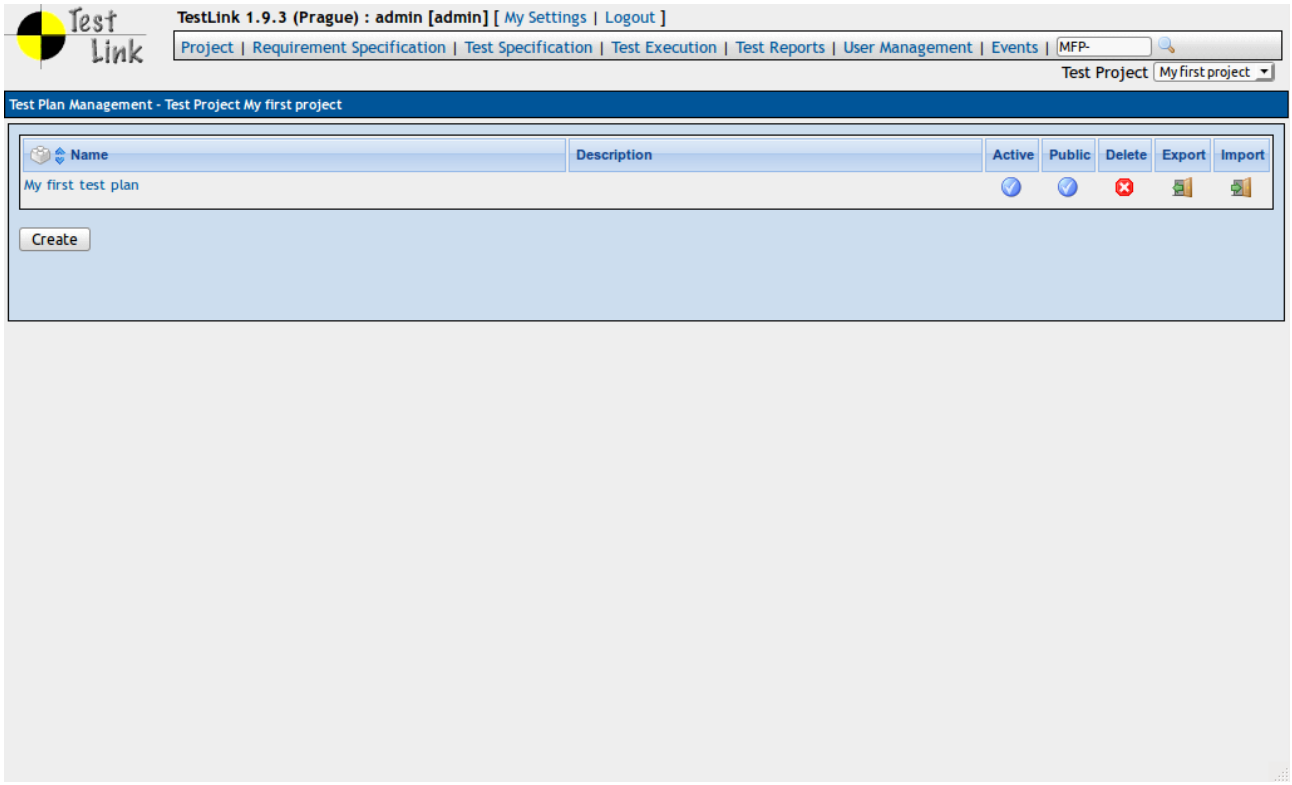
This is the last step for the TestLink configuration. However before we start this step, there is an important concept in TestLink: Builds.

In TestLink, you create a test plan outlining how you will test your application under test. Once you have a test plan you can start to add test cases to your test plan, and then execute the test plan.

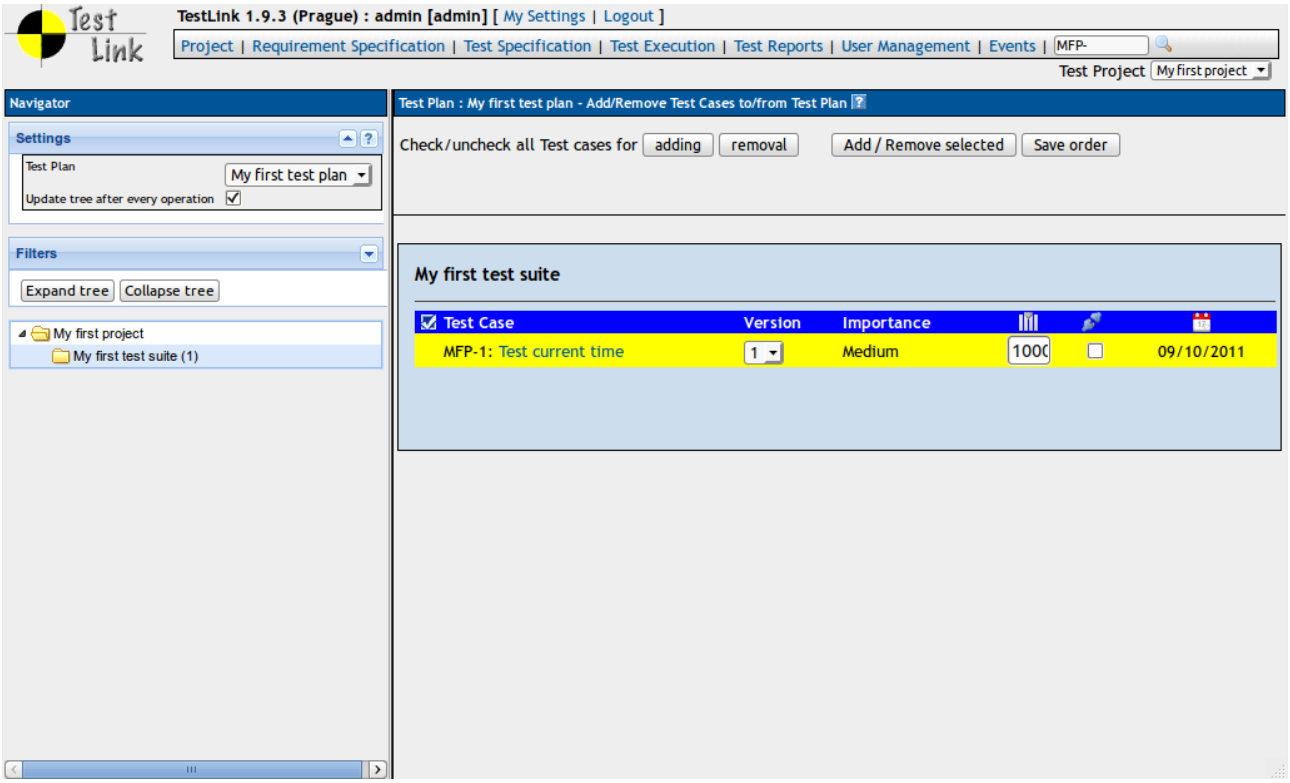
A build in TestLink can be seen as the execution of a test plan. Once the test plan is executed you are not allowed to edit the test cases of this test plan (it wouldn't be right to change the scope or exit criteria of a test case after it had already been executed).

Go back to the main screen and click on the Test Plan Management option available under the Test Plan box on the right of the scren. Create a test plan with the name My first test plan, any description and make sure that Active and Public are checked.





In the last box on the right of the screen, click on Add / Remove Test Cases and add the test case that you created to your test plan.



Create a Build in TestLink is optional, as the plug-in automatically creates a new build if there is none with the name that you provided in the Jenkins job configuration page. When you go back to your test plan, you should see more options available in the Test Plan box and two other boxes: Test Execution and Test Plan contents, as well as other options available in the top menu.

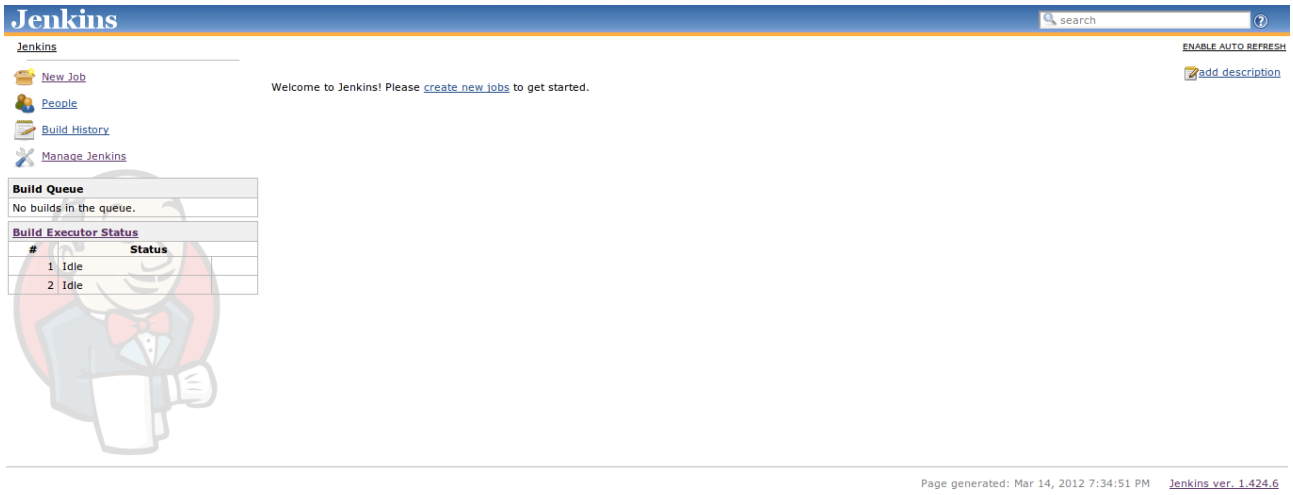
We are done with TestLink for now, the next step is to configure Jenkins.

5. Jenkins Configuration

5.1 Installing Jenkins

Download *jenkins.war* from http://www.jenkins-ci.org. Now open a terminal and execute **java -jar jenkins.war**. This will initiate Jenkins in port 8080 by default (if you need to change that port, use --*httpPort=9999*).

Jenkins creates a default workspace for you at *~/.jenkins* or it uses the folder specified in *JENKINS_HOME* environment variable.

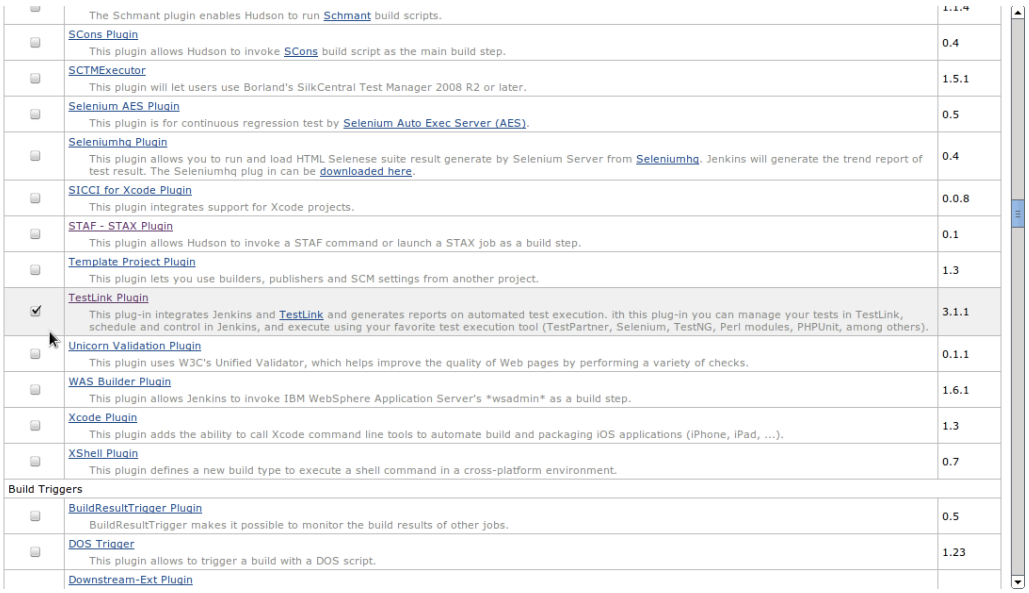


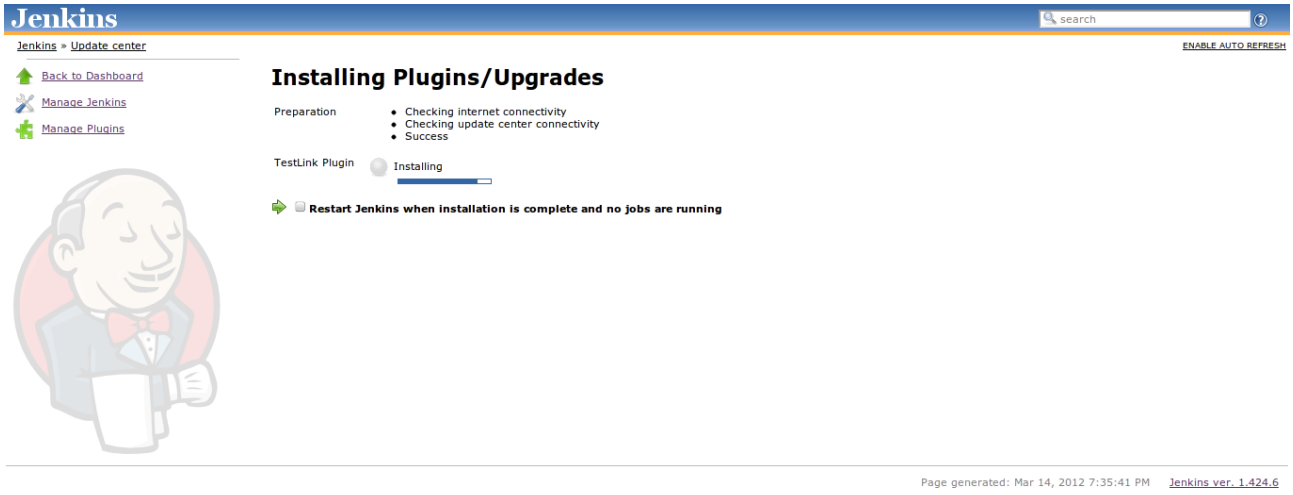
Go to http://localhost:8080 to check if your installation is working. We will call this page as main screen from now on. The examples in this tutorial require you to have administrator rights in Jenkins.

5.2 Installing and configuring Jenkins TestLink Plug-in

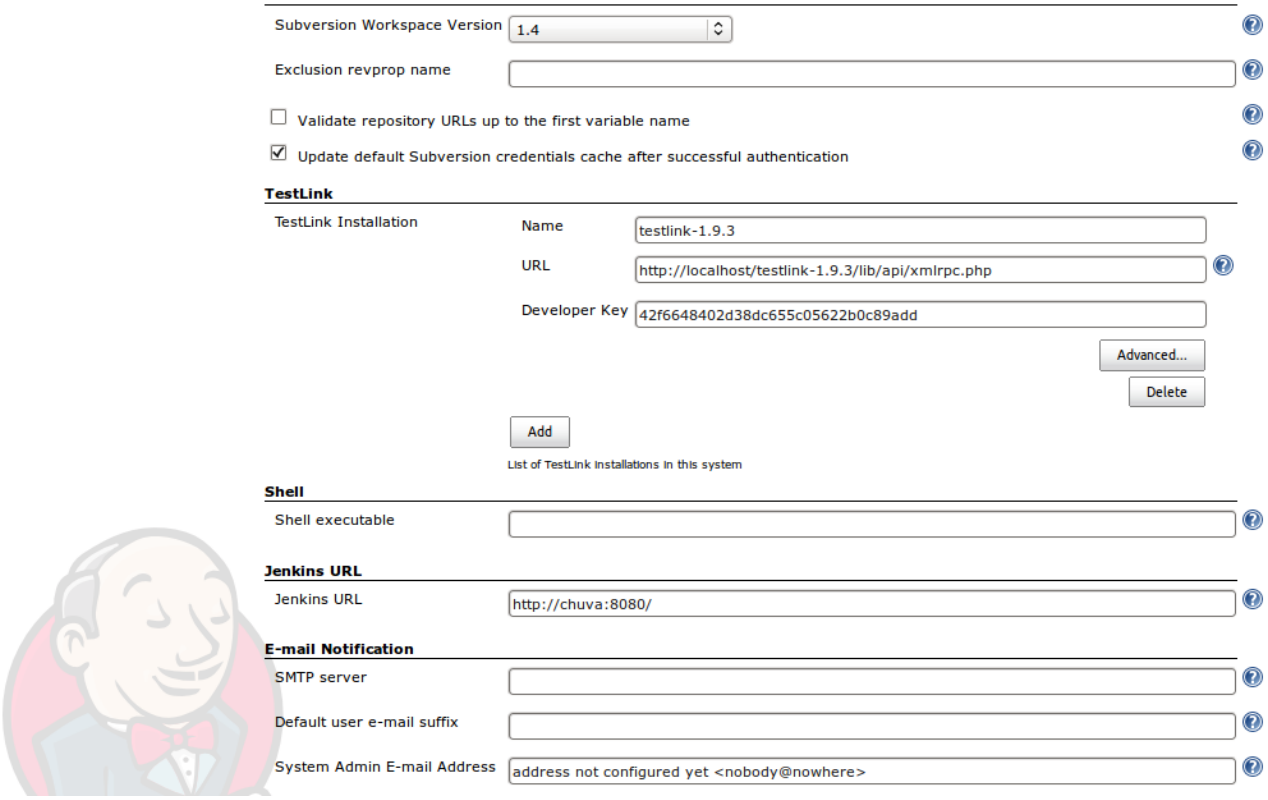
The plug-ins in Jenkins are distributed from a central update site. Select the option *Manage Jenkins* in the left menu and look for the *Manage Plugins* option. Clicking on *Available* will bring you the list of plug-ins ready to be installed in your Jenkins installation.

Just check the check box besides the plug-in name on the list and click on *Install* to install the plug-in. Jenkins will download and install the plug-in automatically for you. Restart Jenkins to enable your plug-in.





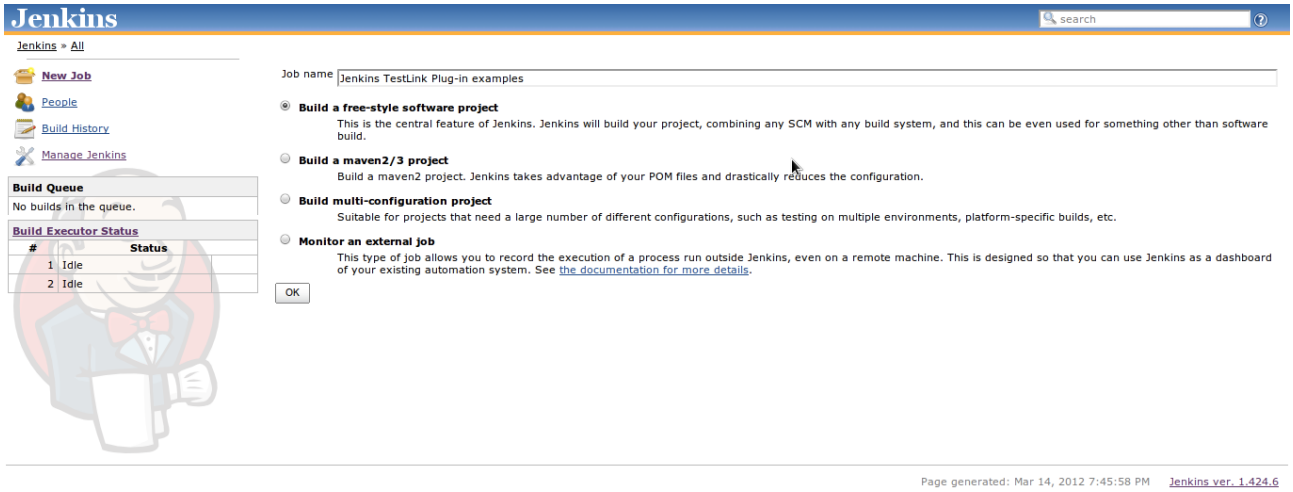
After restarting Jenkins, go to *Manage Jenkins* again, this time click on *Configure System* option and look for the TestLink section. Fill the TestLink configuration form with a name for your TestLink installation, the URL of the XML-RPC API and your *devKey*.



By default the TestLink XML-RPC API URL is *http://<host>:<port>/lib/api/xmlrpc.php*. The *devKey* can be obtained by entering TestLink, clicking on *My Settings* (top menu) and generating a new *devKey*, if there is none yet. If you cannot see the API interface section in *My Settings* page, then it is very likely that you didn't enable it yet. Go back to Chapter 4, *TestLink Configuration* to review your work.

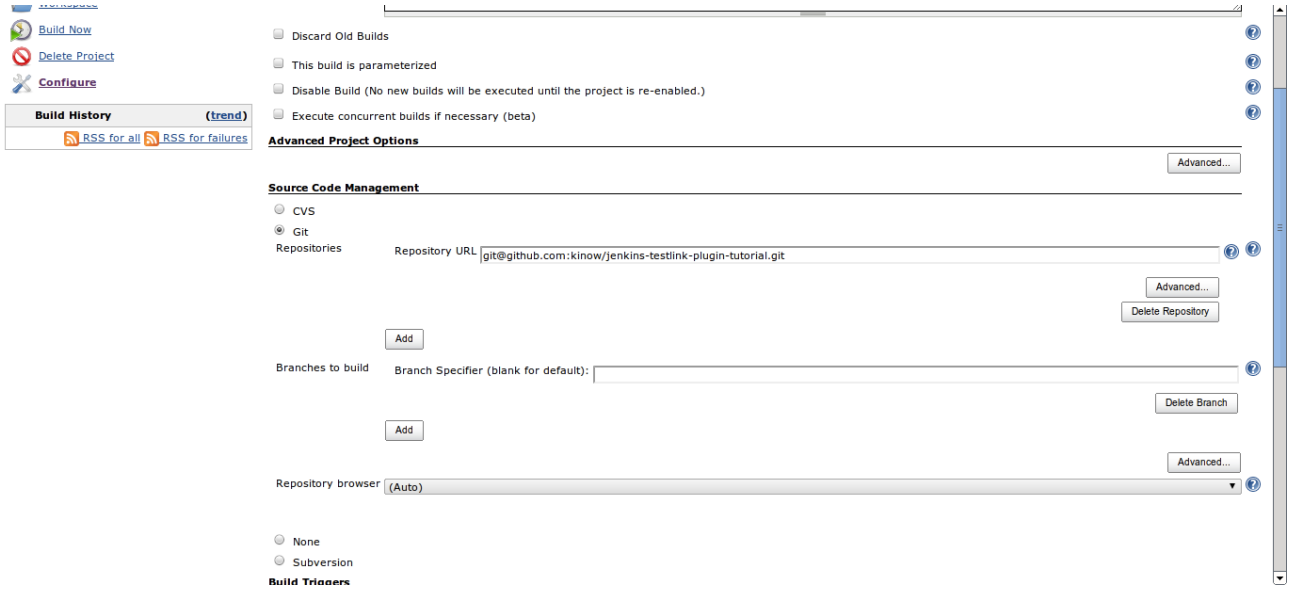
5.3 Creating a job in Jenkins

In order to create a new job all that you need to do is click on *New Job* and give it a name. Choose the option to create a *Free-style project*.



Our job will use git to retrieve the test automation project. This is the project mentioned in Chapter 3, *Jenkins and TestLink Integration*, and can be found at https://github.com/kinow/jenkins-testlink-plugin-

tutorial [https://github.com/kinow/jenkins-testlink-plugin-tutorial]. In a real world project you probably will use a SCM (SVN, Git, CVS, etc) to store your test automation project.



Click on *Add build step* button to expand its options and then click on *Invoke TestLink*. It will show a new form with options to integrate your Jenkins job with TestLink. The plug-in configuration screen contains three sections: *TestLink Configuration*, *Test Execution* and *Result Seeking Strategy*. to fill this form, we will use the configuration created in TestLink throughout Chapter 4, *TestLink Configuration*.

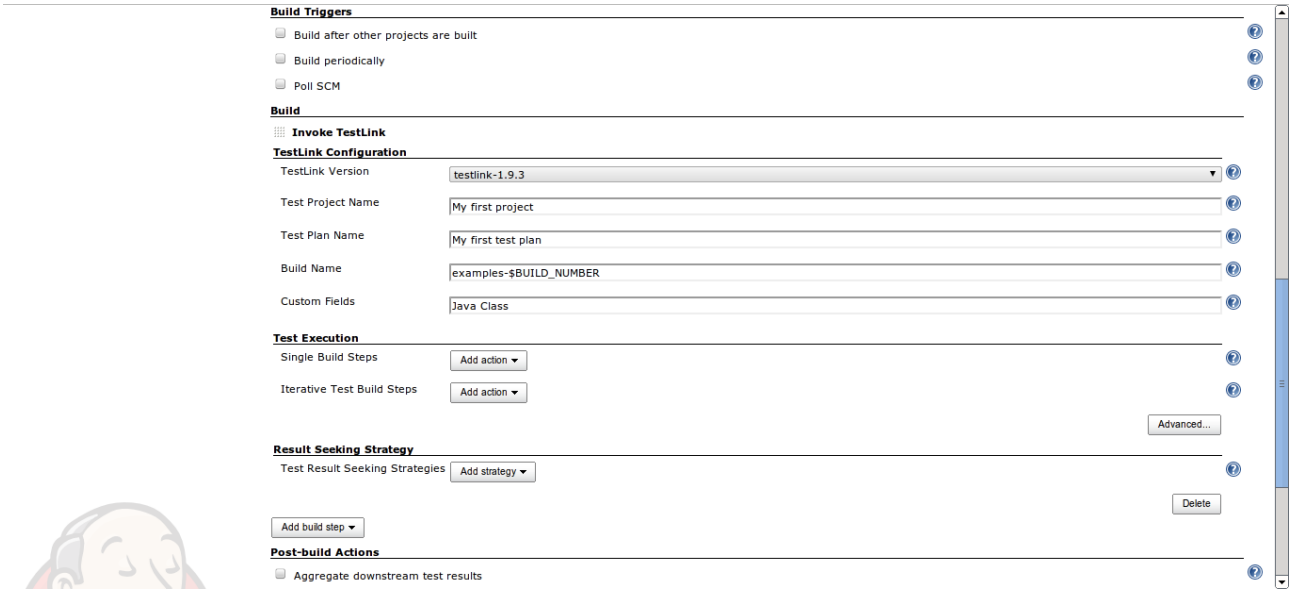
TestLink Configuration

In this section, you are asked to provide the configuration that the plug-in will use to connect to TestLink and retrieve the data for your automated tests.

Use the following settings for this section:

Table 5.1. TestLink Configuration settings explanation

TestLink Version	Select the version that you created in the global configuration
Test Project Name	My first project
Test Plan Name	My first test plan
Build Name	examples-$BUILD_NUMBER
Custom Fields	Java Class



Test Execution

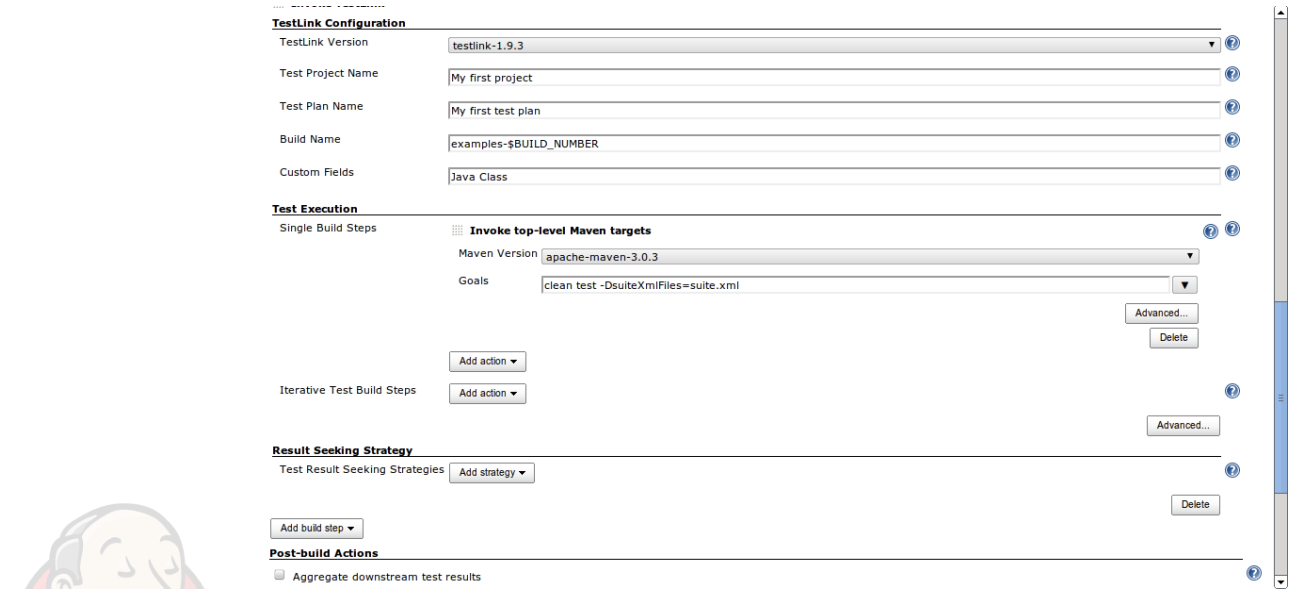
This section contains the configuration to execute your automated tests. You can add *Single Build Steps*, which are executed once per build, or *Iterative Build Steps*, which are executed once per automated test found in TestLink.

There is an advanced section, that is hidden by default, with extra hooks.

Use the following settings for this section:

Table 5.2. Test Execution settings explanation

Single Build Steps	Add an *Invoke top-level Maven targets* build step, and as goals use **test -DsuiteXmlFiles=suite.xml**
Iterative Test Build Steps	Leave it the way it is



Result Seeking Strategy

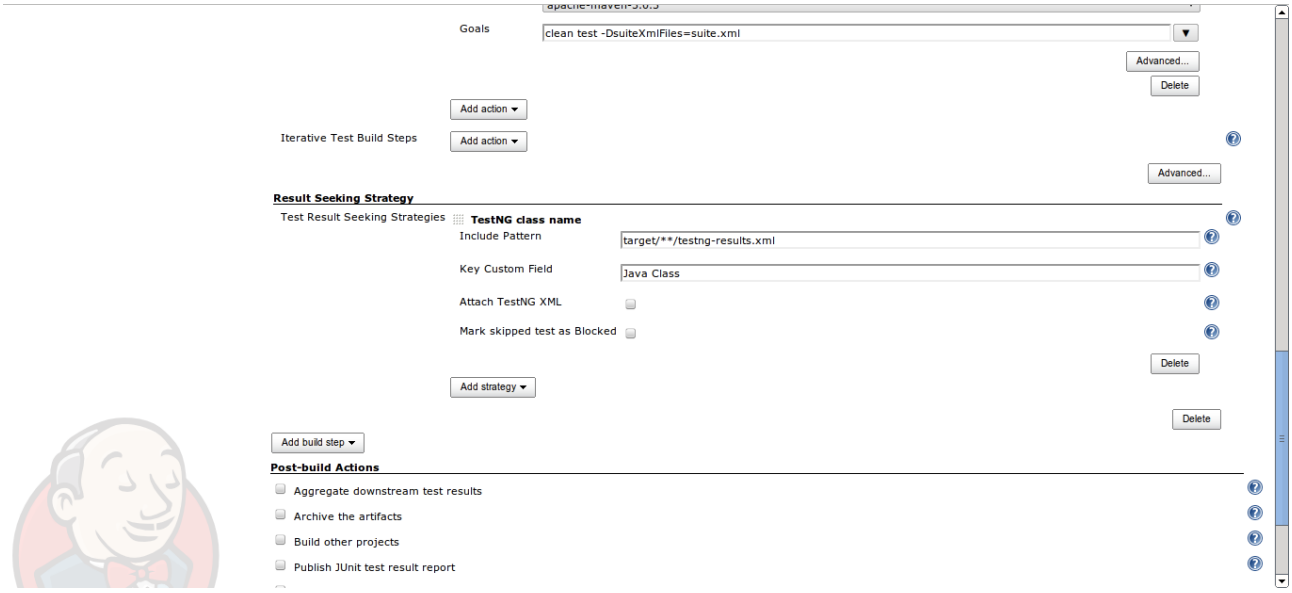
Finally, here you can configure the *Result Seeking Strategy*. The plug-in implements the Strategy pattern for seeking test results. What means that depending on your configuration, the plug-in might use certain algorithm for parsing and updating test results in TestLink.

Each strategy can have different settings. You can use more than one strategy, but you will have to take care to not have the same test in two different strategies.

Use the following settings for this section:

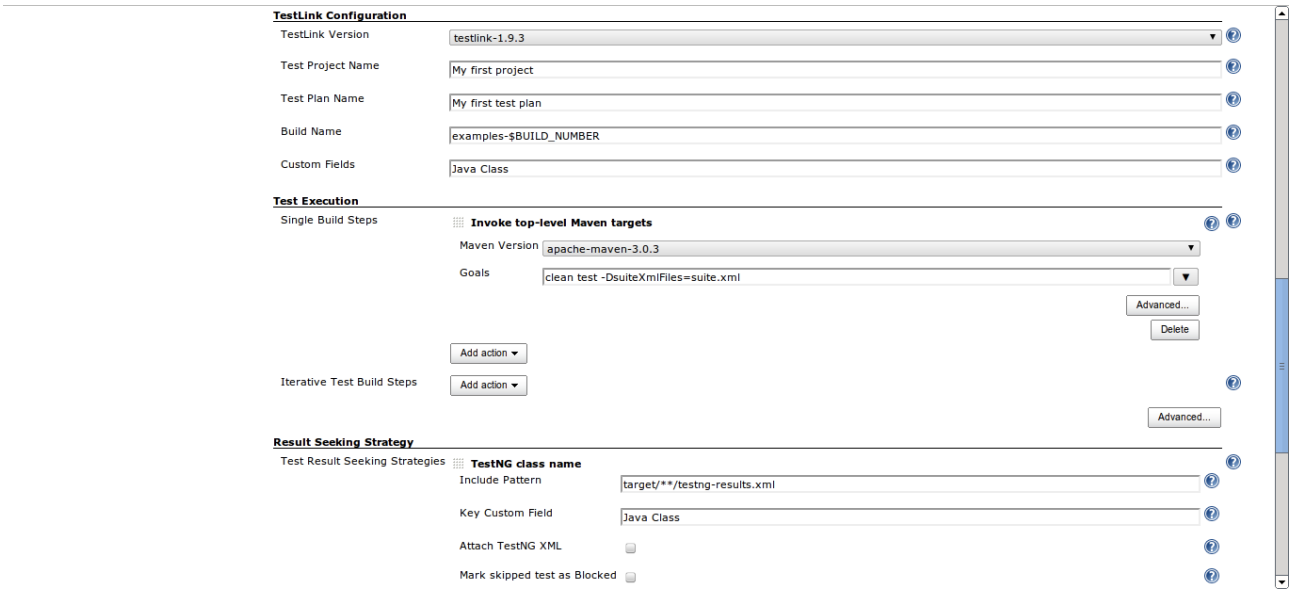
Table 5.3. Result Seeking Strategy settings explanation

Test Result Seeking Strategies	Add the *TestNG class name* strategy.
Include pattern	target/**/testng-results.xml
Key Custom Field	Java Class
Attach TestNG XML	Leave it the way it is
Mark skipped test as Blocked	Leave it the way it is



Save your job.

5.4 Explaining the Job configuration parameters



TestLink Version

This is the name of your TestLink installation in Jenkins global configuration.

Test Project Name

This is the name of your *Test Project* in TestLink. You can use environment variables in this field.

Test Plan name

This is the name of your *Test Plan* in TestLink. You can use environment variables in this field.

Build Name

This is the name of your *Build* in TestLink. You can use environment variables in this field.

Single Build Steps

These Build Steps are executed only once in the job execution.

Iterative Test Build Steps

These Build Steps are executed iteratively for each test case retrieved from TestLink. For these Build Steps, a set of environment variables are available. We will discuss more about them soon.

Test Result Seeking Strategies

This is the Result Seeking Strategy chosen to seek and update test results in TestLink.

Include pattern

The plug-in uses this pattern to find test results of your tests execution (single test command or iterative). It supports Ant-like expressions like TEST*.xml or target/**/testng*.xml.

Key Custom Field

This is the custom field used by the plug-in to link a test case in TestLink with your test results. This custom field must exist in the list of custom fields.

Attach TestNG XML

If this option is checked the plug-in will attach the TestNG XML file to TestLink execution. This way you can open the XML in TestLink.

Mark skipped test as Blocked

By default, in TestNG, skipped tests are not updated in TestLink. It means that they appear as Not Run in TestLink. If this option is checked, the plug-in updates the test case in TestLink as Blocked.

Environment variables

The plug-in retrieves all the information from TestLink for your Test Project, Test Plan, Build and automated Test Cases. You can then use any of this information to execute your tests. Jenkins itself provides the *Environment Variables*, plus *Build Environment Variables* (such as *BUILD_ID*, which holds the date time of your job).

The plug-in injects the information retrieved from TestLink as extra environment variables. This way you can use the value of the *Java class* custom field value that you created in Chapter 4, *TestLink Configuration* in any of your iterative Build Steps. Below you can find an example of how to execute a single test with Maven and one of these environment variables.

mvn test -Dtest=$TESTLINK_TESTCASE_JAVA_CLASS

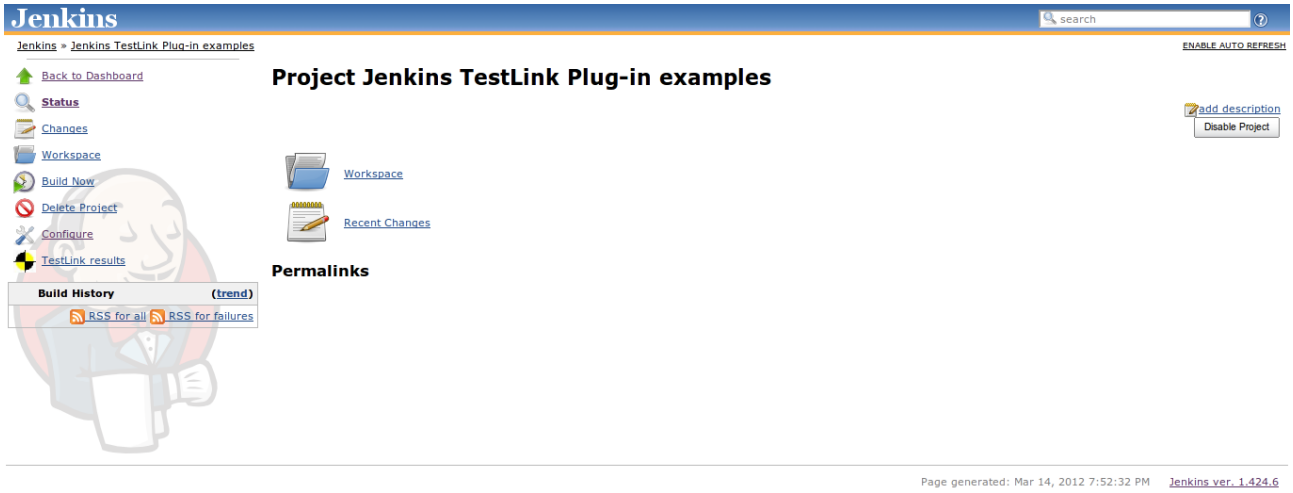
As you can see, our test command uses the *Java class* custom field value to specify the name of the test to Maven (Maven Surefire Plug-in, actually). Below you will find a list with the information that the plug-in makes available for your job configuration. As custom fields names may vary, the strategy used is capitalize the custom field name, replace spaces with _ and append it to *TESTLINK_TESTCASE_*, which represents information of a Test Case in TestLink.

- TESTLINK_TESTCASE_ID
- TESTLINK_TESTCASE_NAME
- TESTLINK_TESTCASE_TESTPROJECTID
- TESTLINK_TESTCASE_AUTHOR
- TESTLINK_TESTCASE_SUMMARY

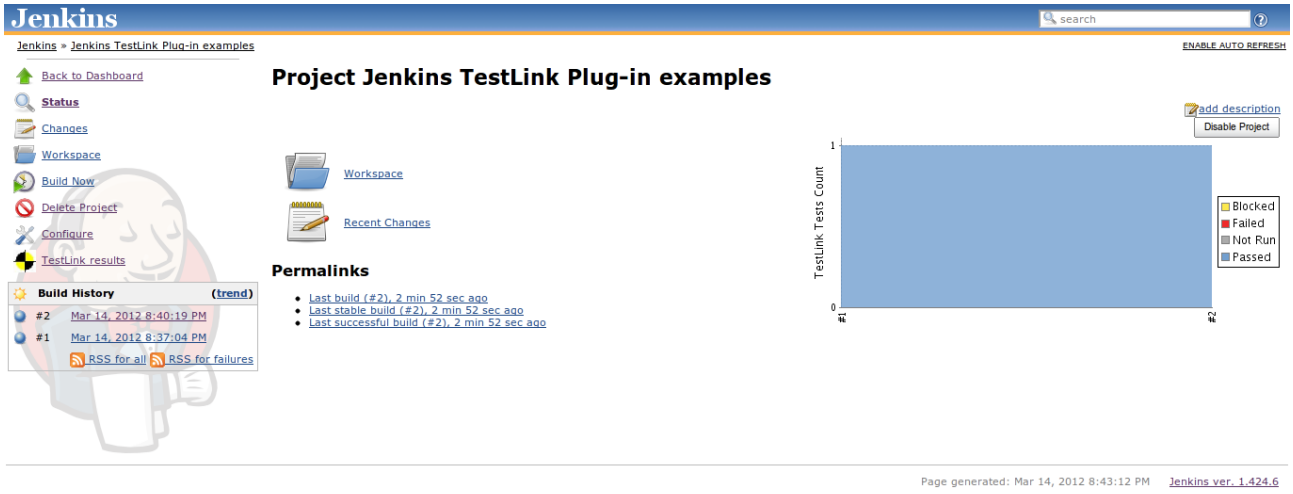
- TESTLINK_BUILD_NAME
- TESTLINK_TESTPLAN_NAME
- TESTLINK_TESTCASE_$CUSTOM_FIELD_NAME

6. Executing Automated Test Cases

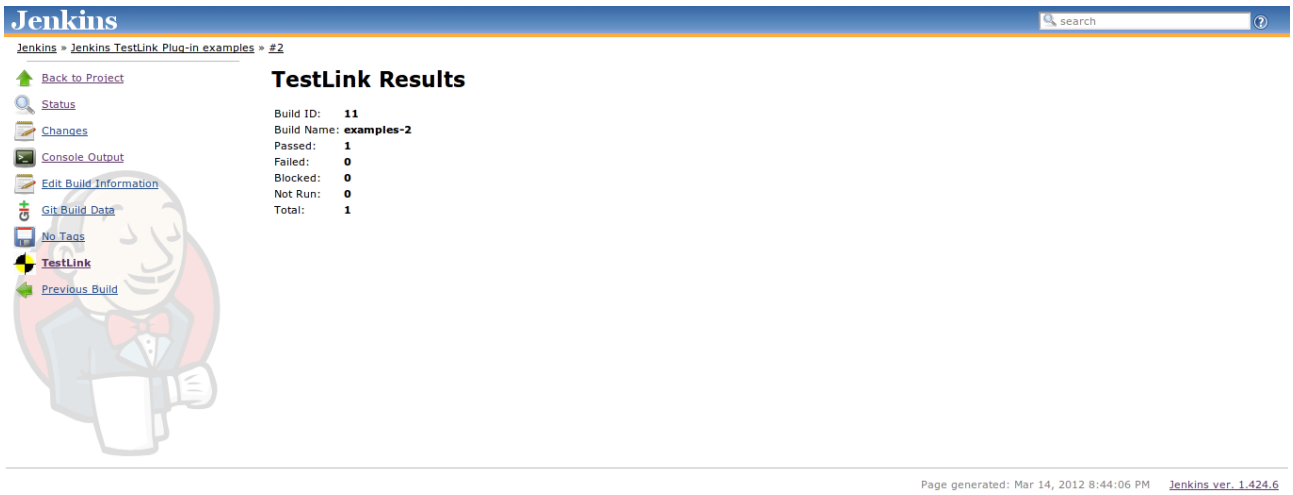
Now that everything is configured it is time to execute our automated test cases. Let's go back to the main screen in Jenkins. Click on your job and trigger your job execution by clicking on *Build Now* .

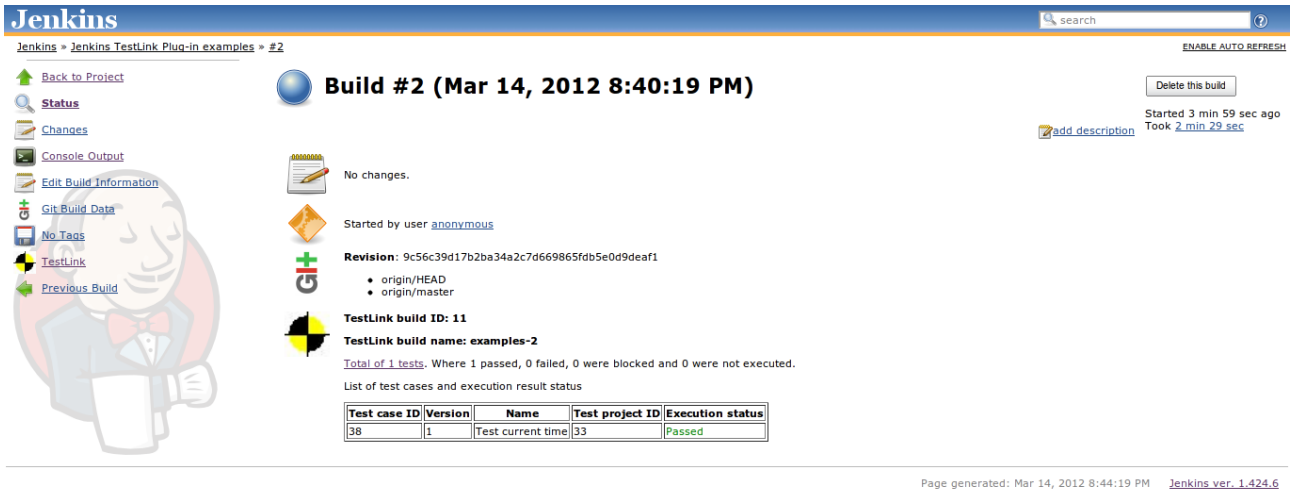


Once your job execution is completed, refresh your current screen either by pressing F5 or by clicking over the build name. The following screen must appear, showing a graph with the test results.

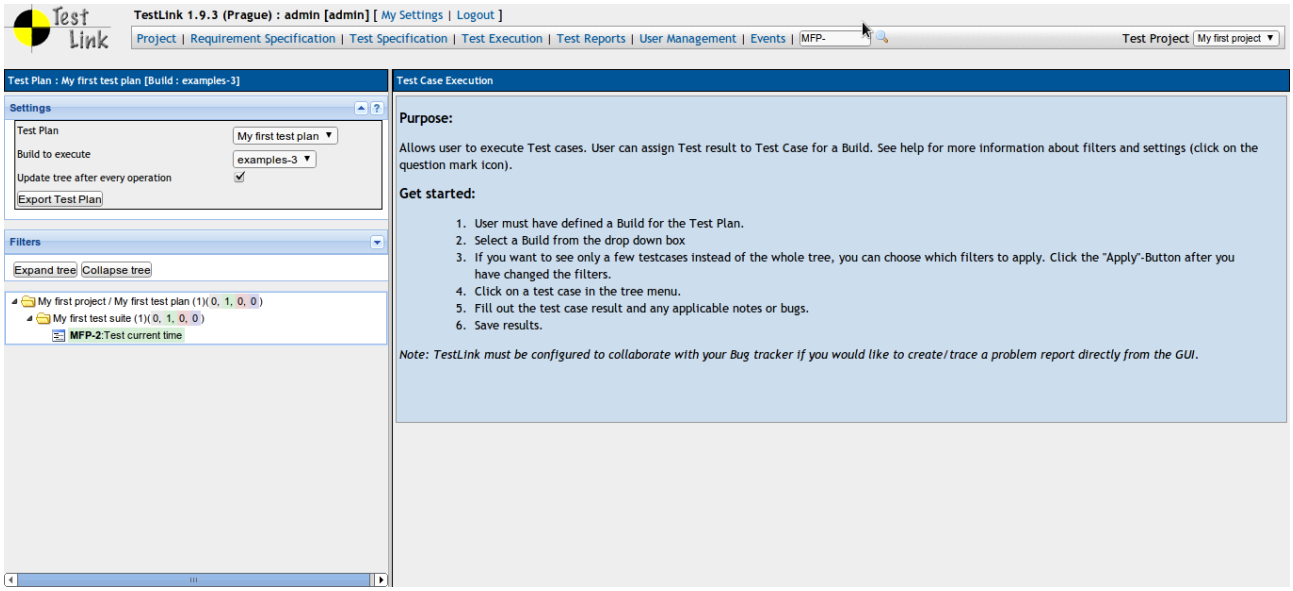


You can find more information on your tests by clicking on *Test Results* or on your build under *Build History*.

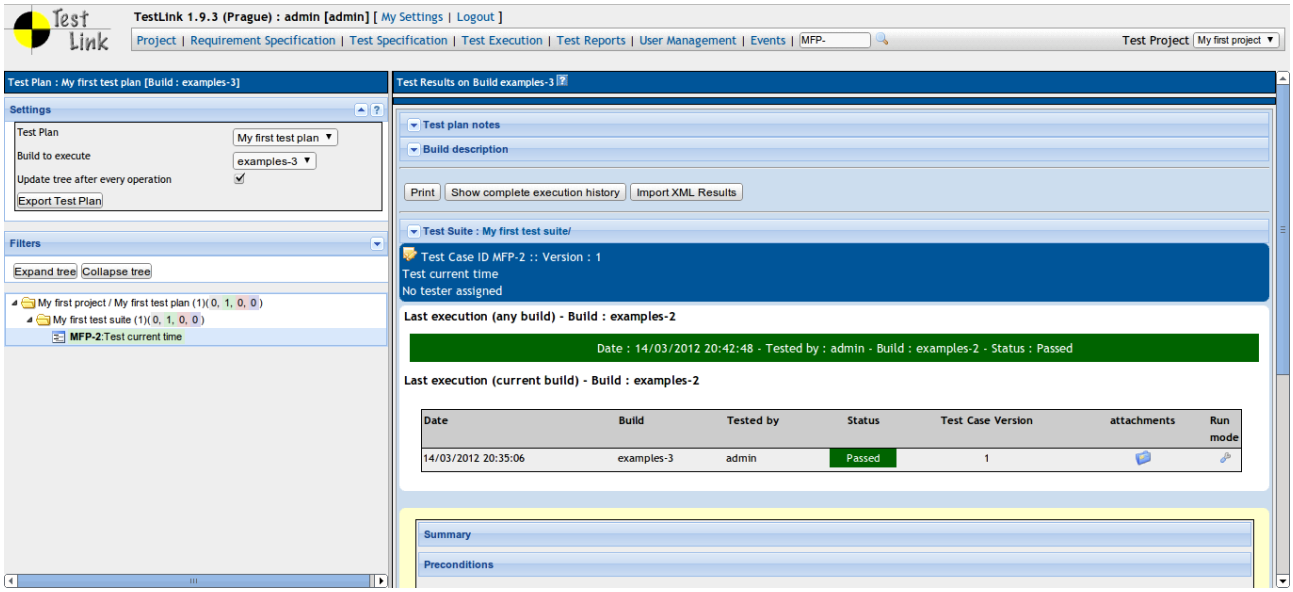




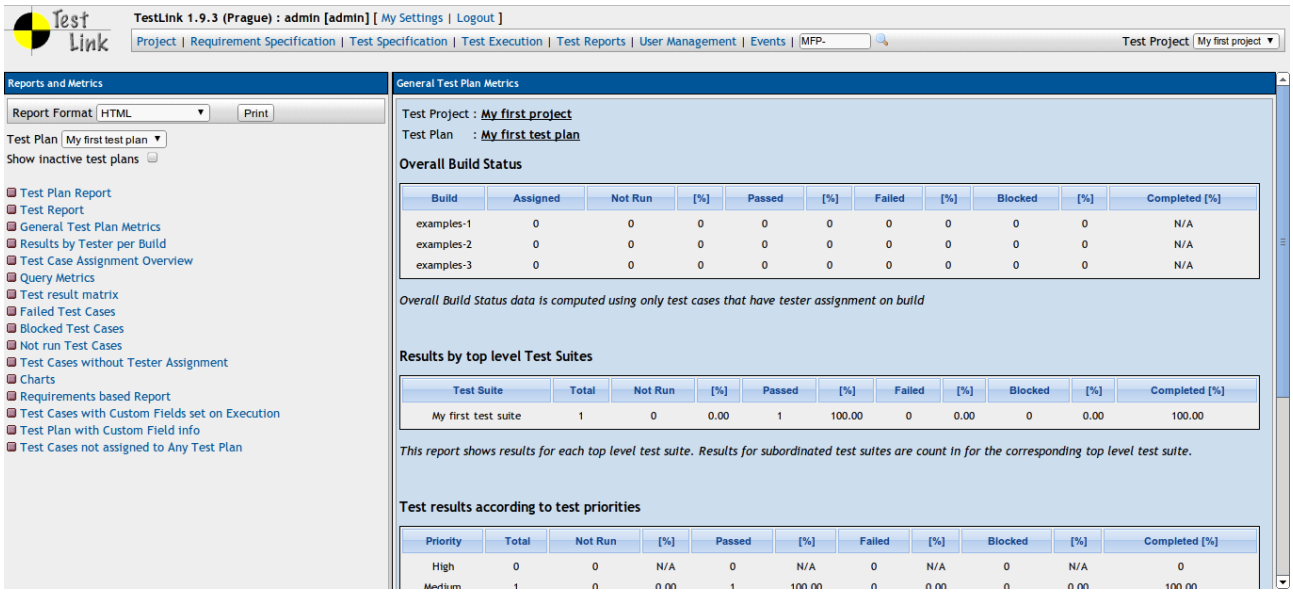
Everything seems to be right in Jenkins, let's go back to TestLink and click on *Test execution* option in the top menu. It will bring a screen showing the result of your test execution.



You can see more information about the execution of your tests by clicking over it on the left tree. Notice that by default the plug-in uploads the test result file. In this case, as we are using TestNG, it uploaded *testng-results.xml*. It is useful as you can see what happened in your tests, e.g.: an exception stack trace.



Explore the reports available in TestLink to see the results of your tests by different perspectives.



7. Result Seeking Strategies

After your automated tests are executed the plug-in has to look for *Test Results*. This way it will know whether an automated test was executed correctly or not and then it can update the test execution status in TestLink.

Jenkins TestLink Plug-in supports three different formats of results: *JUnit*, *TestNG* and *TAP*.

7.1 JUnit

```
<?xml version="1.0" encoding="UTF-8" ?>
<testsuite failures="1" time="0.078" errors="0" skipped="0" tests="1" name="A Suite">
  <testcase time="0" classname="tcA" name="testVoid">
    <failure type="junit.framework.AssertionFailedError" message="null">junit....</failure>
  </testcase>
  <testcase time="0" classname="tcB" name="testVoid" />
  <testcase time="0" classname="tcC" name="testVoid" />
  <testcase time="0" name="nameA" />
  <testcase time="0" name="nameB" />
</testsuite>
```

JUnit case name

This strategy matches the case name with the key custom field value.

JUnit class name

This strategy matches the case class name with the key custom field value.

JUnit method name

This strategy matches <package>.<classname>#<methodname> with the key custom field value (e.g.: com.example.MyClass#testSomething)

JUnit suite name

This strategy matches the suite name with the key custom field value.

7.2 TestNG

```
<testng-results>
  <reporter-output>
  </reporter-output>
  <suite name="Command line suite"
    duration-ms="0"
    started-at="2010-11-17T13:31:41Z"
    finished-at="2010-11-17T13:31:41Z">
    <groups>
    </groups>
    <test name="Command line test"
```

```
        duration-ms="0"
        started-at="2010-11-17T13:31:41Z"
        finished-at="2010-11-17T13:31:41Z">
      <class name="br.eti.kinoshita.Test">
        <test-method status="FAIL"
            signature="testVoid()"
            name="testVoid"
            duration-ms="0"
            started-at="2010-11-17T13:31:41Z"
            finished-at="2010-11-17T13:31:41Z">
        </test-method>
      </class>
    </test>
  </suite>
</testng-results>
```

TestNG class name

This strategy matches the class name with the key custom field value.

TestNG method name

This strategy matches <package>.<classname>#<methodname> with the key custom field value (e.g.: com.example.MyClass#testSomething)

TestNG suite name

This strategy matches the suite name with the key custom field value.

7.3 TAP - Test Anything Protocol

Using TAP, you can define the Platform (defined in TestLink) that you used for your tests or extra attachments to be uploaded to TestLink. These topics are covered in Chapter 9, *Appendix*.

```
1..3
ok 1 testOk
ok 2
not ok 3
```

TAP file name

This strategy matches the TAP Stream file name with the key custom field value.

8. Advantages of using Jenkins TestLink Plug-in

The plug-in is not intended to be a test automation solution, though it helps you to automate your tests. It is not intended to manage your automated tests either, as this can be done in TestLink. The purpose of the plug-in is integrate Jenkins, a continuous integration server, and TestLink, a test management tool.

There are three clear advantages in using this approach. First, if you are already using Jenkins as continuous integration server and TestLink as test management tool, then you won't have another tool to worry about. Simply install the plug-in and then DevOps will keep working in Jenkins, testers will keep working in TestLink and your boss will be more than happy to know that he won't need to buy another tool and training for an automated tests management tool.

Secondly, it is language independent. You can run tests in PHP, Perl, Python, Java, Lua and even in C or C++. The only limitation here is that you have to output your test results either in JUnit, TestNG or TAP.

And lastly, it is free. The plug-in team members are contributors of TestLink and maintainers of the TestLink Java API. We try to fix the issues in Jenkins' JIRA [http://issues.jenkins-ci.org] as fast as they can. So if you use the plug-in, send us your feedback, write in the plug-in Wiki about your case or buy us a beer.

9. Appendix

9.1 Adding attachments to your test results

Each test result file is automagically attached to its automated test case in TestLink. That is the default behavior in the plug-in. You can extend it, only if you are using TAP. You will have to create a *YAMLish* following the example below (it is important that you encode the file content in Base64).

```
1..2
ok 1
not ok 2 - br.eti.kinoshita.selenium.TestListVeterinarians#testGoogle
  ---
  extensions:
   Files:
       /tmp/screenshot3562328890173159732.png:
         File-Location: /tmp/screenshot3562328890173159732.png
         File-Title: screenshot3562328890173159732.png
         File-Description: Main page
         File-Size: 114542
         File-Name: screenshot3562328890173159732.png
         File-Content: "iVBORw0KGgoAAAANSUhEUgAAA+IAAAJqCAYAAACvjvpKAAAgAElEQVR4nOy9d1RU5/r3ff+ff+8dZ\r\
                \n613v8zy/95RfjjkmFmwgVXrvvffee1PBjtKbCoJ0qaIodqMYe8EWwRo1atQkRmOixkTFAkYFu8f\r\
                ...
                ...
            File-Type: image/png
```

An issue that wasn't solved yet is that if you are running your job in a distributed environment, then the plug-in won't be able to find the attachments for your test result.

9.2 Using Platforms

TestLink has a very nice feature, Platforms. Useful specially if you have tests that run is some specific environment and you use some logic (like pairwise) to choose what is going to be tested.

As well as attachments, the only way to use Platforms while integrating Jenkins and TestLink is using TAP. You have to create a *YAMLish* entry like the following.

```
1..3
ok 1
ok 2
  ---
  extensions:
    TestLink:
      Platform: EC1
  ...
ok 3
```

As you can see above, the TAP Stream has the platform EC1. The plug-in scans a TAP Stream looking for this entry structure in each test result and in the test plan.

9.3 Plug-in behavior in distributed environments

The calls to TestLink are executed in the master, as well as the graph generating and some Jelly back end code. The rest, what includes executing Build Steps and looking for and parsing test results is executed in the slave, if present.

9.4 How to use the plug-in with SSL or basic authentication?

In the global configuration of the plug-in, there is a *Advanced* button. Click on it and an extra option will be shown, *TestLink Java API Properties*.

Click on the help icon on the right to see which options you can enable. They are used by testlink-java-api [http://testlinkjavaapi.sf.net] to set properties in the connection.

Among the properties, you may have to set xmlrpc.basicUsername and xmlrpc.basicPassword with the credentials used for basic authentication.

For SSL, there are guides available in the Internet showing how to add a certificate to the Java key store and proceed with a SSL connection.

Bibliography

[JenkinsTestLink] *Jenkins TestLink Plug-in [http://wiki.jenkins-ci.org/display/JENKINS/TestLink+Plugin]*. Copyright © 2011 The Jenkins TestLink Plug-in team.

[JenkinsCI] *Jenkins-CI [http://www.jenkins-ci.org/]*. Copyright © 2011 Jenkins-CI.

[TestLink] *TestLink [http://www.teamst.org/]*. Copyright © 2011 TestLink Community.

[TestNG] *TestNG [http://www.testng.org/]*. Copyright © 2011 TestNG.

[JUnit] *JUnit [http://www.junit.org/]*. Copyright © 2011 JUnit.

[TAP] *Test Anything Protocol Website [http://testanything.org/]*. Copyright © 2011 Test Anything Protocol Community.

[TAPWikipedia] *Test Anything Protocol article in Wikipedia [http://en.wikipedia.org/wiki/Test_Anything_Protocol]*. Copyright © 2011 Wikipedia.

[YAMLorg] *YAML.org [http://www.yaml.org]*. Copyright © 2011 YAML.org.