Squash TF Documentation

squashtest

Sep 30, 2019

Contents

CHAPTER 1

Squash TF Execution Server

Squash TF *Download Page*

CHAPTER 2

Squash TF Runners

2.1 Environment configuration

Hint: Our runners are build to run on our Execution Server. So it's highly recommended to configure your development environment as describe in the section below.

First we advise to use the same tools versions as those we used in our execution server :

- maven: 3.5.0
- java: 1.8

Our maven library are hosted on our own repository. In consequence, you have to define our repository in your maven settings. To do so, edit (or create) the maven `settings.xml` file in your `.m2` directory (The .m2 directory is generally located in your Home directory) and add a new profile:

```
<settings>

...

  <profiles>
    <profile>
      <id>tf-maven-repos</id>
      <!-- Squash TF maven repository -->
      <repositories>
        <repository>
          <id>org.squashtest.tf.release</id>
          <name>squashtest test factory - releases</name>
          <url>http://repo.squashtest.org/maven2/releases</url>
        </repository>
```

(continues on next page)

(continued from previous page)

```
      </repositories>

      <!-- Squash TF maven plugin repository -->
      <pluginRepositories>
        <pluginRepository>
          <id>org.squashtest.plugins.release</id>
          <name>squashtest.org</name>
          <url>http://repo.squashtest.org/maven2/releases</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>

  <activeProfiles>
    <activeProfile>tf-maven-repos</activeProfile>
  </activeProfiles>

</settings>
```

We also advise to patch your maven by using the procedure below for a better logging with our runners :

Note: In all the procedure `$MVN_HOME` is your maven installation directory, and `$MVN_VERSION` your maven version.

- Add in `$MVN_HOME/lib/ext/` the jars:
  - log4j-slf4j-impl-2.5.jar
  - log4j-core2.5.jar
  - log4j-api-2.5.jar
- Create a logging configuration file called `log4j2.xml` in `$MVN_HOME/conf/logging/` and fill it with :

```
<?xml version="1.0" encoding="UTF-8" ?>
<Configuration>
  <Properties>
    <Property name="maven.logging.root.level">INFO</Property>
  </Properties>
  <Appenders>
    <Console name="console" target="SYSTEM_OUT">
      <PatternLayout pattern="[%p] %msg%n%throwable" />
    </Console>
  </Appenders>
  <Loggers>
    <Root level="${sys:maven.logging.root.level}">
      <Appender-ref ref="console"/>
    </Root>
<!-- <logger name="[USER_MESSAGE]" level="DEBUG"/> -->
```

(continues on next page)

(continued from previous page)

```
  </Loggers>
</Configuration>
```

- Remove if exists :
  - In the directory `$MVN_HOME/lib` the file `maven-sl4j-provider-$MVN_VERSION.jar`
  - In the directory `$MVN_HOME/conf/logging/` the file `deletesimpleLogger.properties`

CHAPTER 3

Our development tool

3.1 Eclipse Tools

3.1.1 Squash-TA Toolbox

Eclipse Toolbox - Components

The Squash-TA toolbox 1.10.0 contains the following tools:

- Maven 3.5.0,
- Sahi v51,
- Jailer 7.3.1,
- Eclipse Oxygen 64-bits as well as a preconfigured workspace
  - m2e
  - Squash TA eclipse plugin
  - “Run configurations”.
- An uninstall assistant

The version 1.9.0 contains:

- Maven 3.0.4 (for compatibility with framework versions before1.9.0),
- A patched version of Maven 3.3.3 (you can find here details about the patch),
- Sahi,
- Selenium-server,
- Jailer,
- Eclipse Mars as well as a preconfigured workspace
  - m2e

  - Squash TA eclipse plugin
  - “Run configurations” and eclipse preferences to import.
- An uninstall assistant

Before the 1.9.0 version the toolbox used to contain:

- Maven 3.0.4,
- Sahi,
- Selenium-server,
- Jailer,
- Eclipse Helios as well as a preconfigured workspace
  - m2e
  - Squash TA eclipse plugin
  - “Run configurations” and eclipse preferences to import.
- An uninstall assistant
- open-jdk 6.

Install Squash-TA Toolbox

Prerequisite : Before starting the installation of SQUASH-TA Toolbox, you must install a JDK between 1.7 and 1.8, both 32 and 64 bits version are compatible. See here to download it from Oracle website.

First of all, download the toolbox installer a the the following link.

- Launch the installer by double-clicking on it:
- On the first screen, click on **Next**:

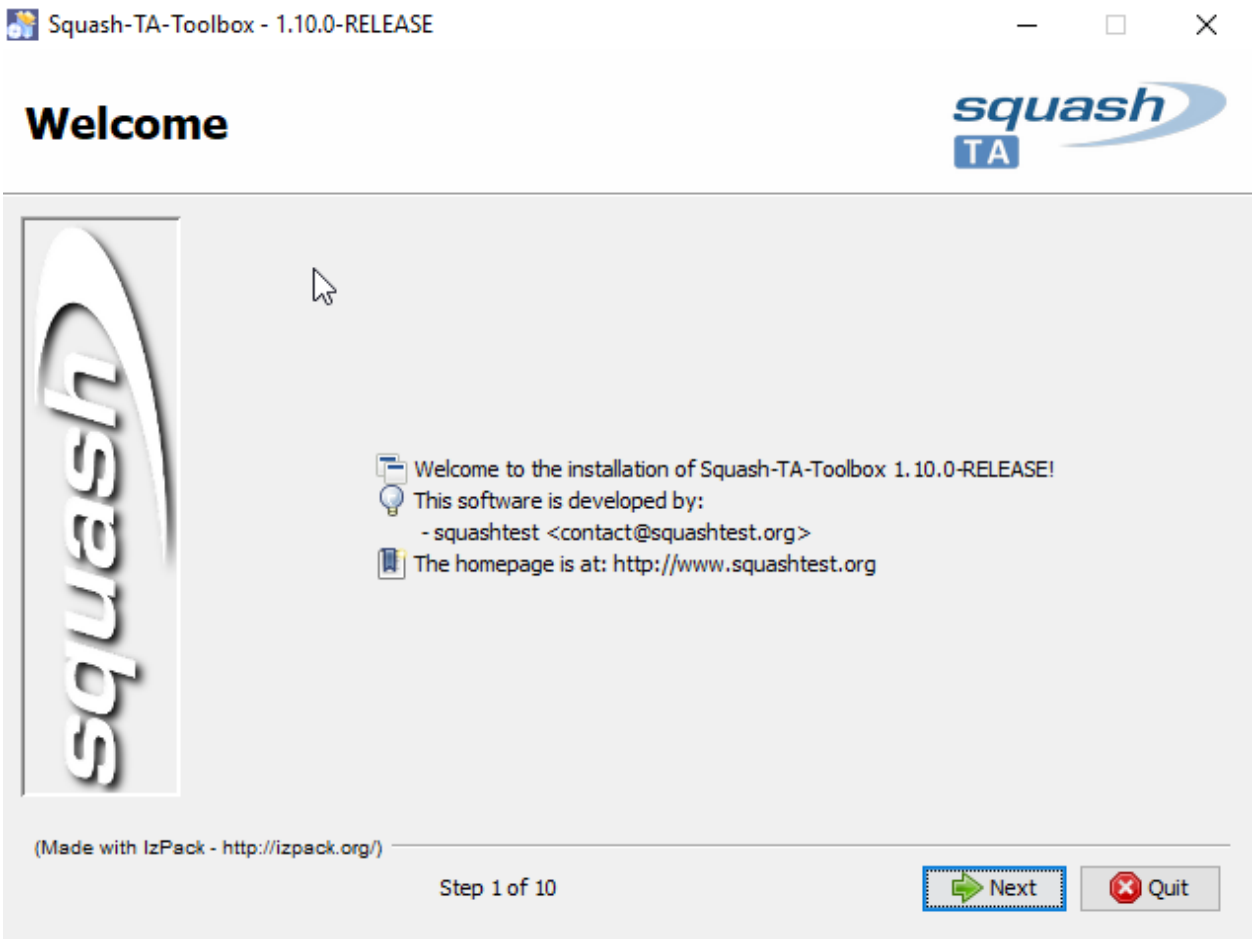
- You will then be prompted to indicate the path of your JDK (to do so, click on the **Browse** button. Click on **Next** when it is done:

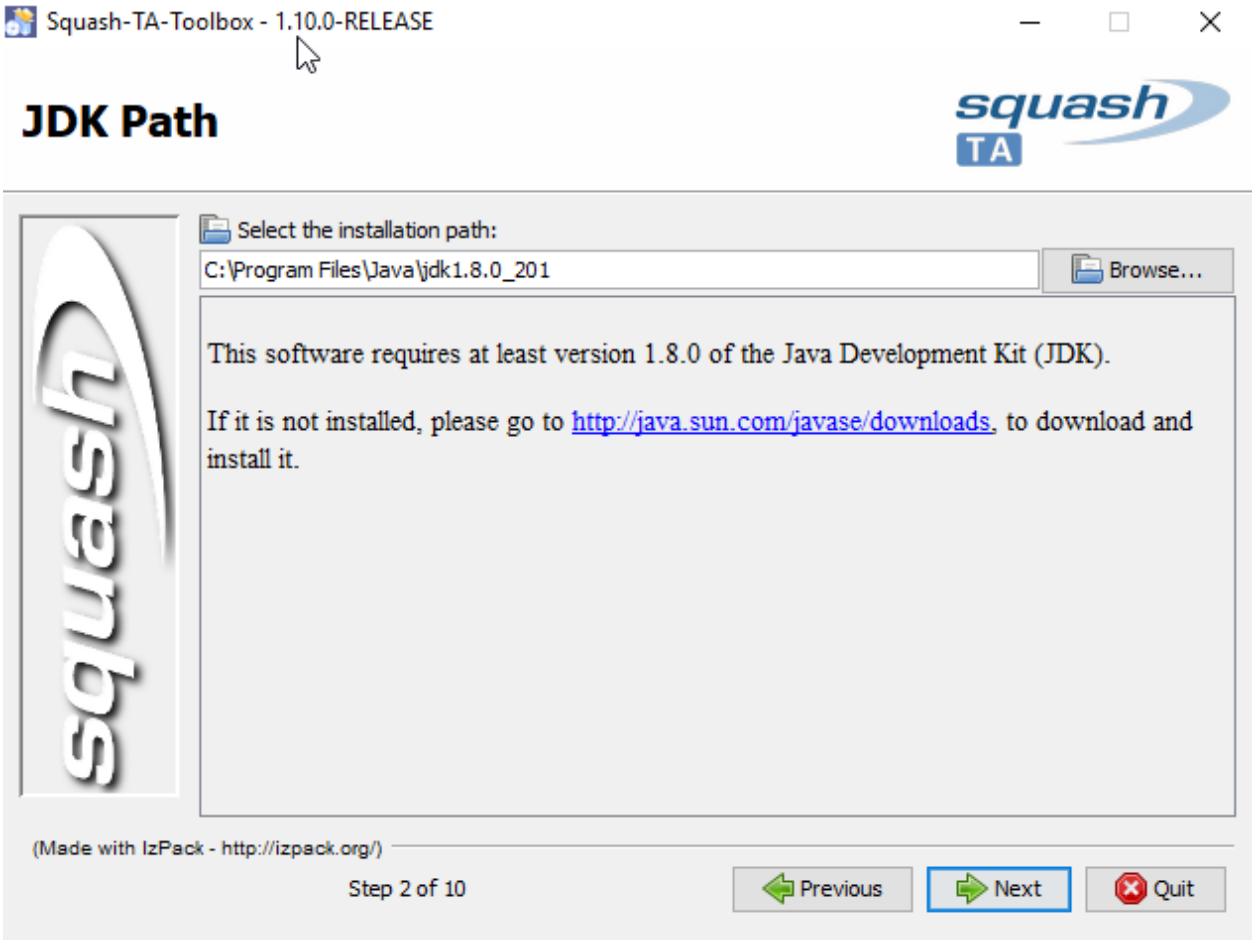
- On the next screen, you will see all the licenses of the different components of Squash TA Toolbox. Click on **Next**:

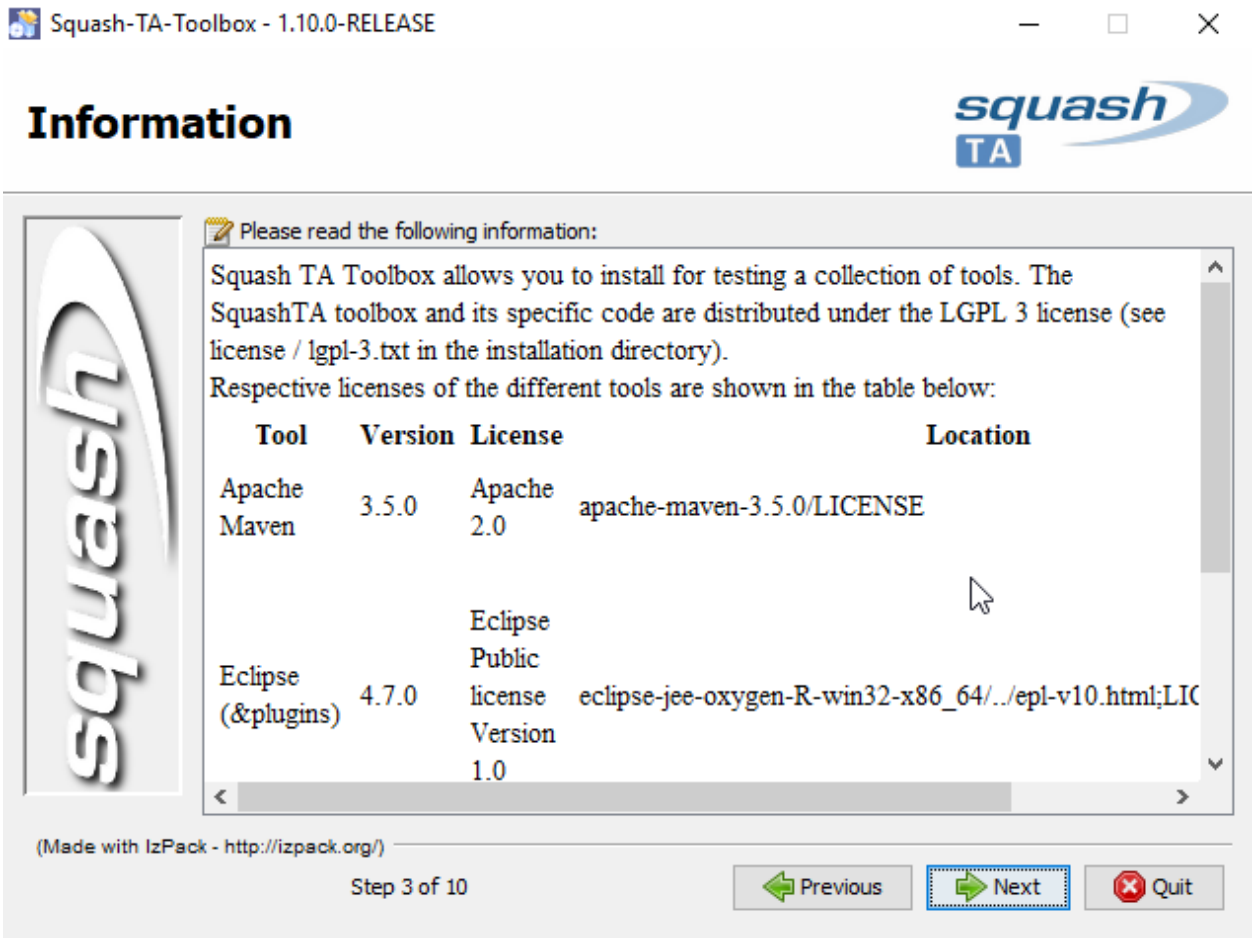
- The next step allows you choose your toolbox install location. You can directly type the path in the input field, or click on **Browse** and select your chosen location (you may create your installation folder from there if needed). After choosing the location, click on **Next**:

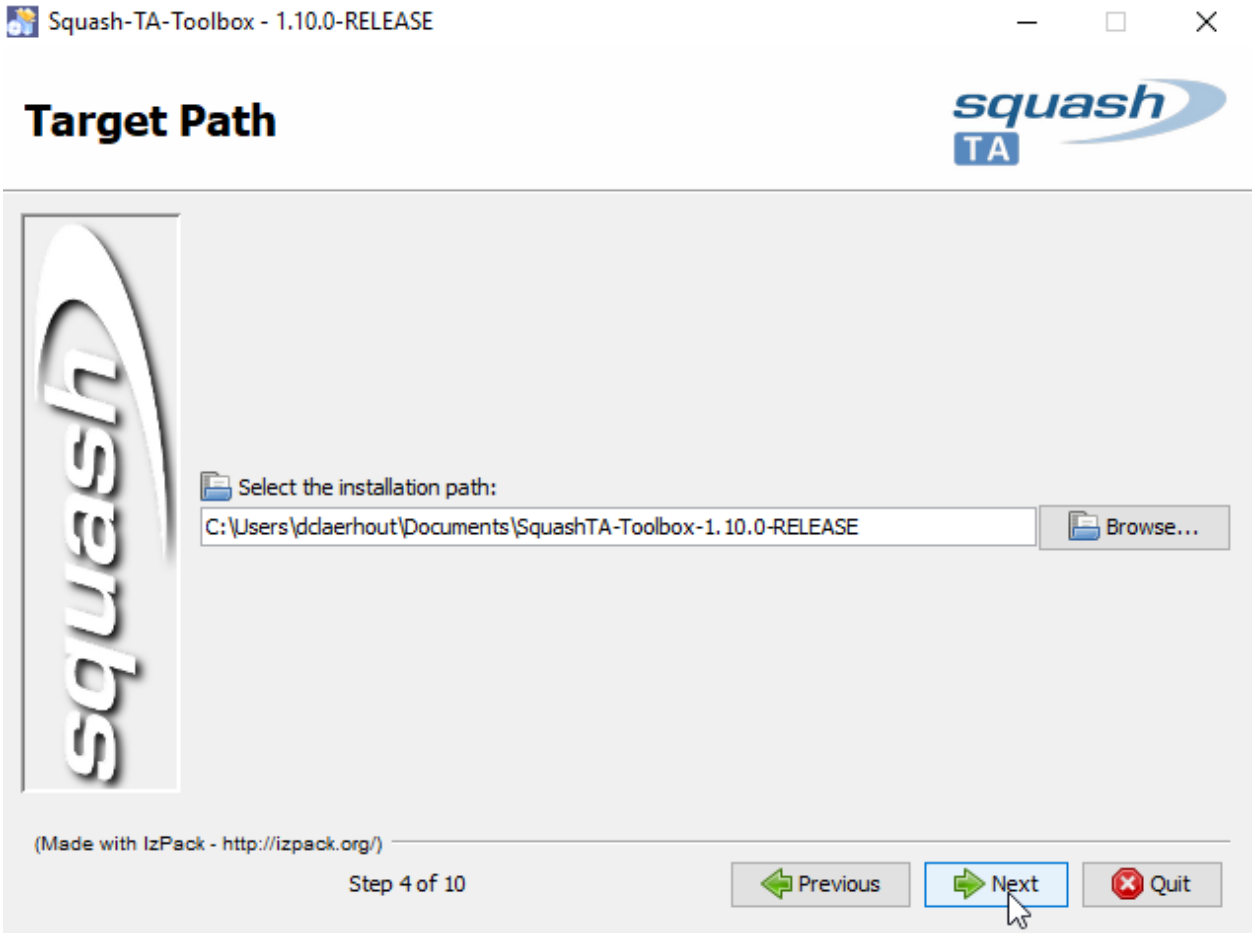
- If the target folder already exists (even if you’ve just created it), you will get the following alertbox. Click on **Yes** to confirm the location:

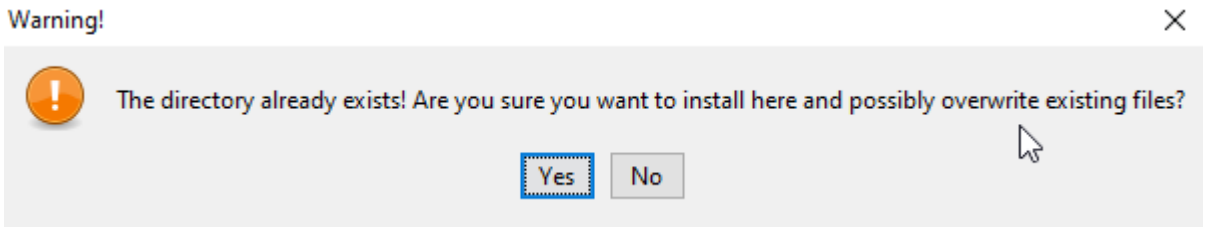
- Next, you will have to choose a location to create your new Eclipse workspace for your Squash-TF projects. As for the install location, you may type the path directly or use **Browse** button. If the location you typed does not exist, it will be automatically created. After choosing your location, click on **Next**:

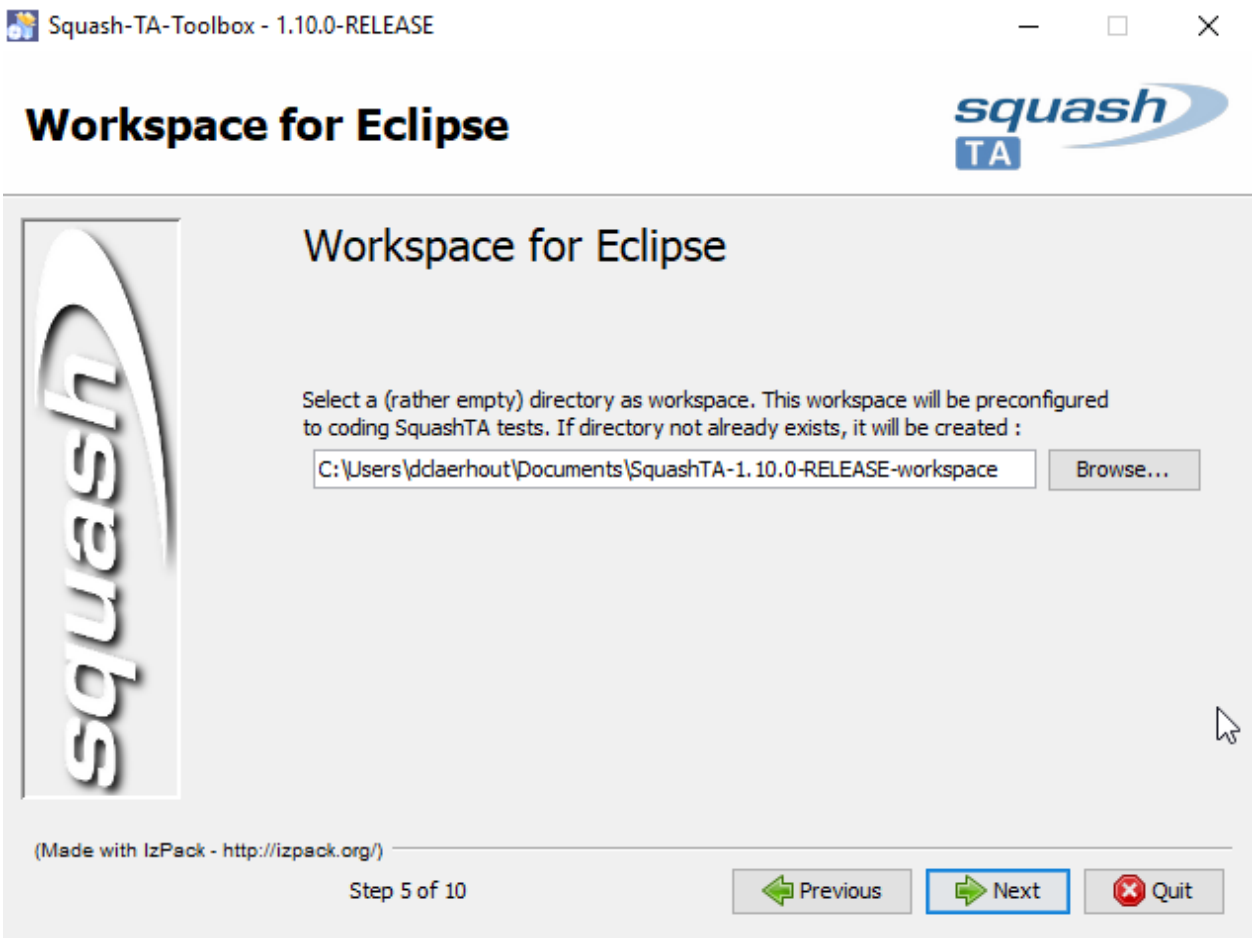

Squash-TA-Toolbox - 1.10.0-RELEASE
Welcome
squash
TA
squash
Welcome to the installation of Squash-TA-Toolbox 1.10.0-RELEASE!
This software is developed by:
- squashtest <contact@squashtest.org>
The homepage is at: http://www.squashtest.org
(Made with IzPack - http://izpack.org/)
Step 1 of 10
Next
Quit


Squash-TA-Toolbox - 1.10.0-RELEASE
JDK Path
squash
TA
Select the installation path:
C:\Program Files\Java\jdk1.8.0_201
Browse...
This software requires at least version 1.8.0 of the Java Development Kit (JDK).
If it is not installed, please go to http://java.sun.com/javase/downloads, to download and install it.
(Made with IzPack - http://izpack.org/)
Step 2 of 10
Previous
Next
Quit

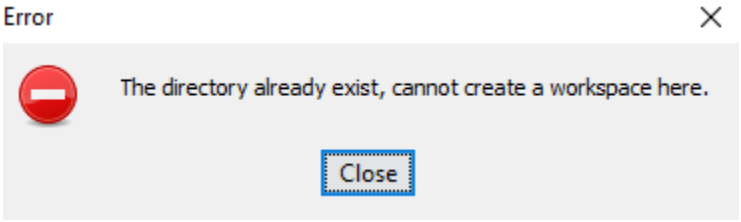

Squash-TA-Toolbox - 1.10.0-RELEASE
Information
squash TA
Please read the following information:
Squash TA Toolbox allows you to install for testing a collection of tools. The SquashTA toolbox and its specific code are distributed under the LGPL 3 license (see license / lgpl-3.txt in the installation directory).
Respective licenses of the different tools are shown in the table below:
Tool
Version
License
Location
Apache Maven
3.5.0
Apache 2.0
apache-maven-3.5.0/LICENSE
Eclipse (&plugins)
4.7.0
Eclipse Public license Version 1.0
eclipse-jee-oxygen-R-win32-x86_64/../epl-v10.html;LIC
(Made with IzPack - http://izpack.org/)
Step 3 of 10
Previous
Next
Quit


Squash-TA-Toolbox - 1.10.0-RELEASE
Target Path
squash
TA
squash
Select the installation path:
C:\Users\dclaerhout\Documents\SquashTA-Toolbox-1.10.0-RELEASE
Browse...
(Made with IzPack - http://izpack.org/)
Step 4 of 10
Previous
Next
Quit

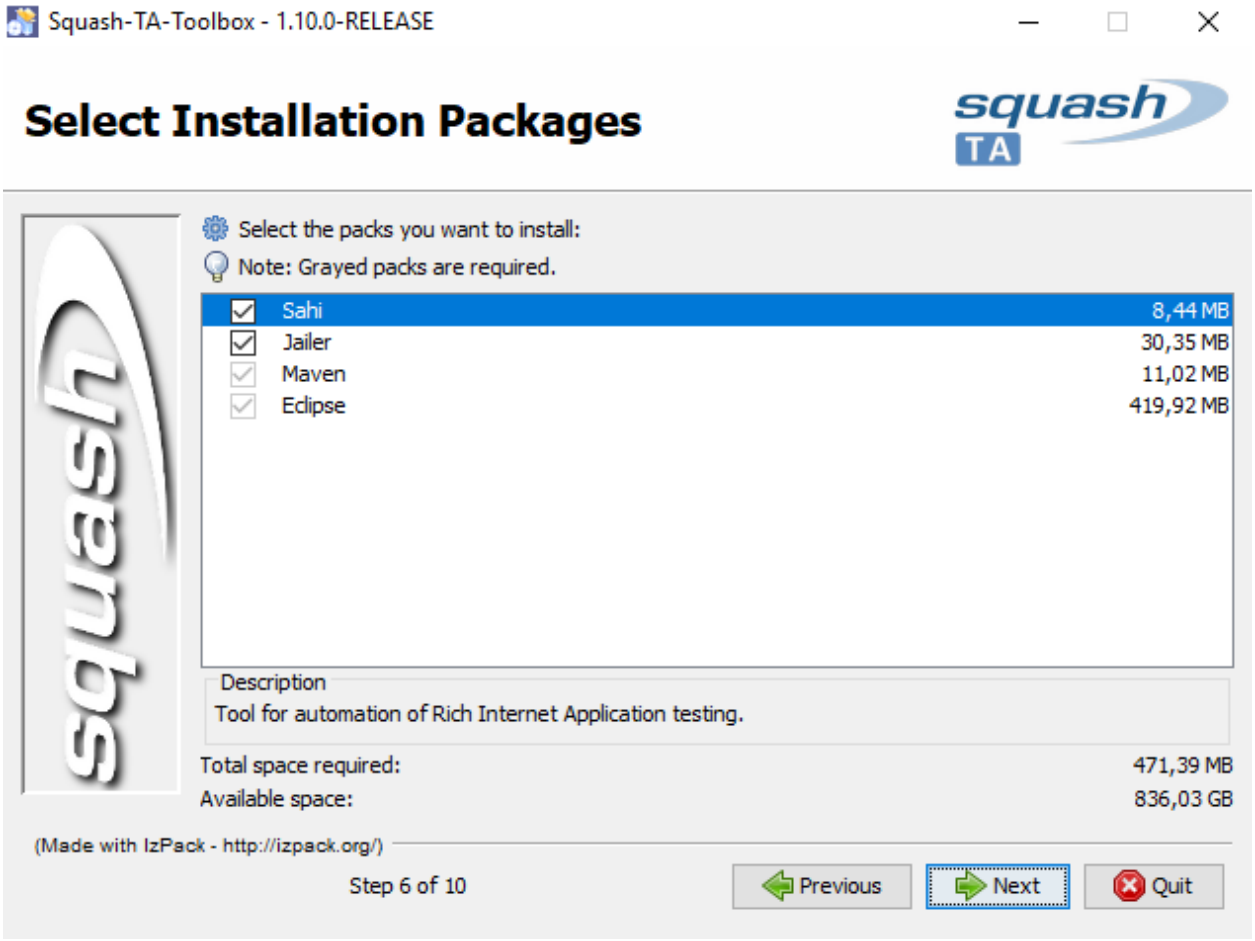

Warning!
The directory already exists! Are you sure you want to install here and possibly overwrite existing files?
Yes
No


Squash-TA-Toolbox - 1.10.0-RELEASE
Workspace for Eclipse
squash TA
Workspace for Eclipse
Select a (rather empty) directory as workspace. This workspace will be preconfigured to coding SquashTA tests. If directory not already exists, it will be created :
C:\Users\dclaerhout\Documents\SquashTA-1.10.0-RELEASE-workspace
Browse...
(Made with IzPack - http://izpack.org/)
Step 5 of 10
Previous
Next
Quit

- If the location already exists, you will get the following error message. You can't create the Eclipse workspace in an existing location:

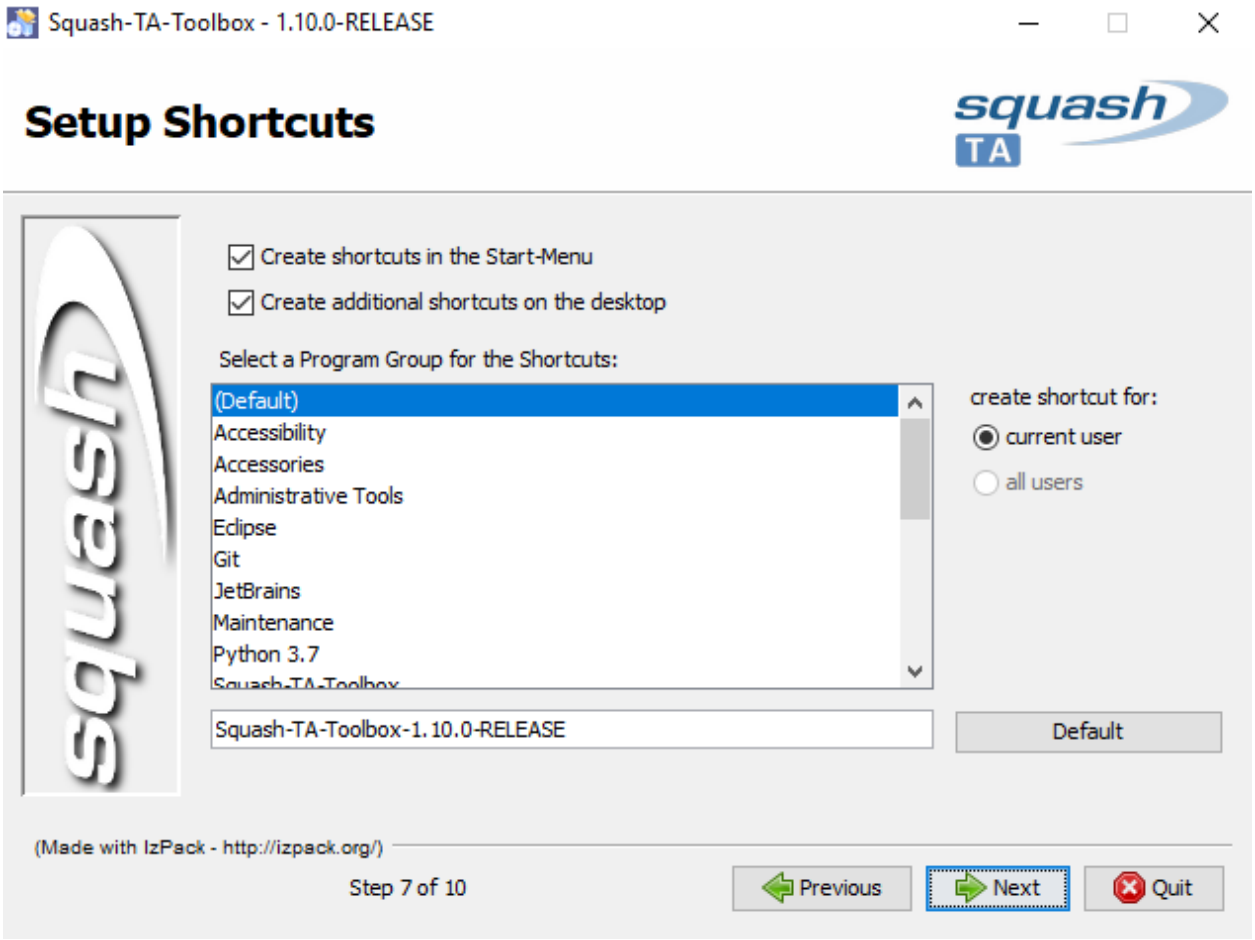


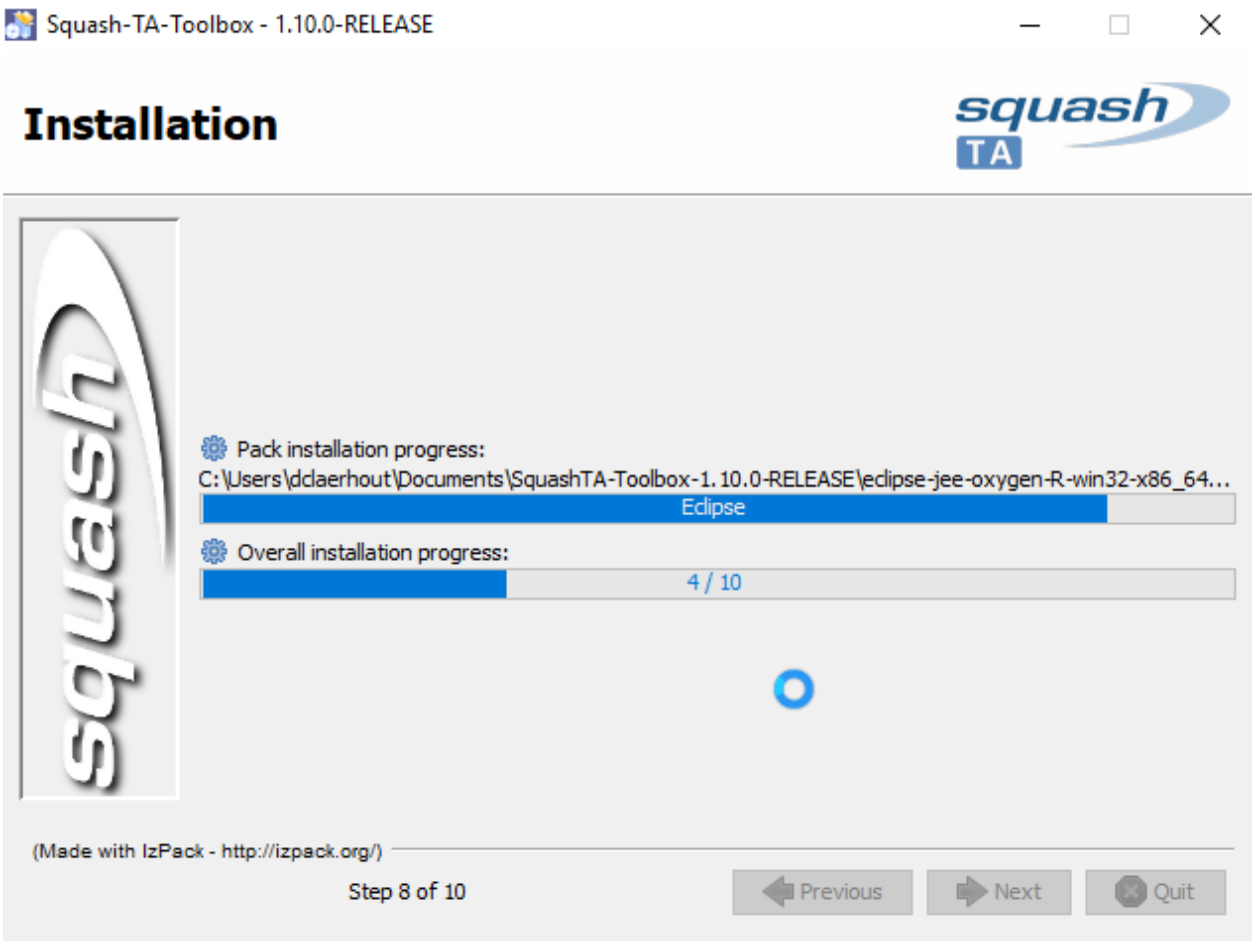
- The next step lets you check the list of installed components. You may reuse some compatible components already installed on your workstation but for now let's assume that you will be getting all tools from the toolbox, and leave all boxes checked. Click on **Next**:



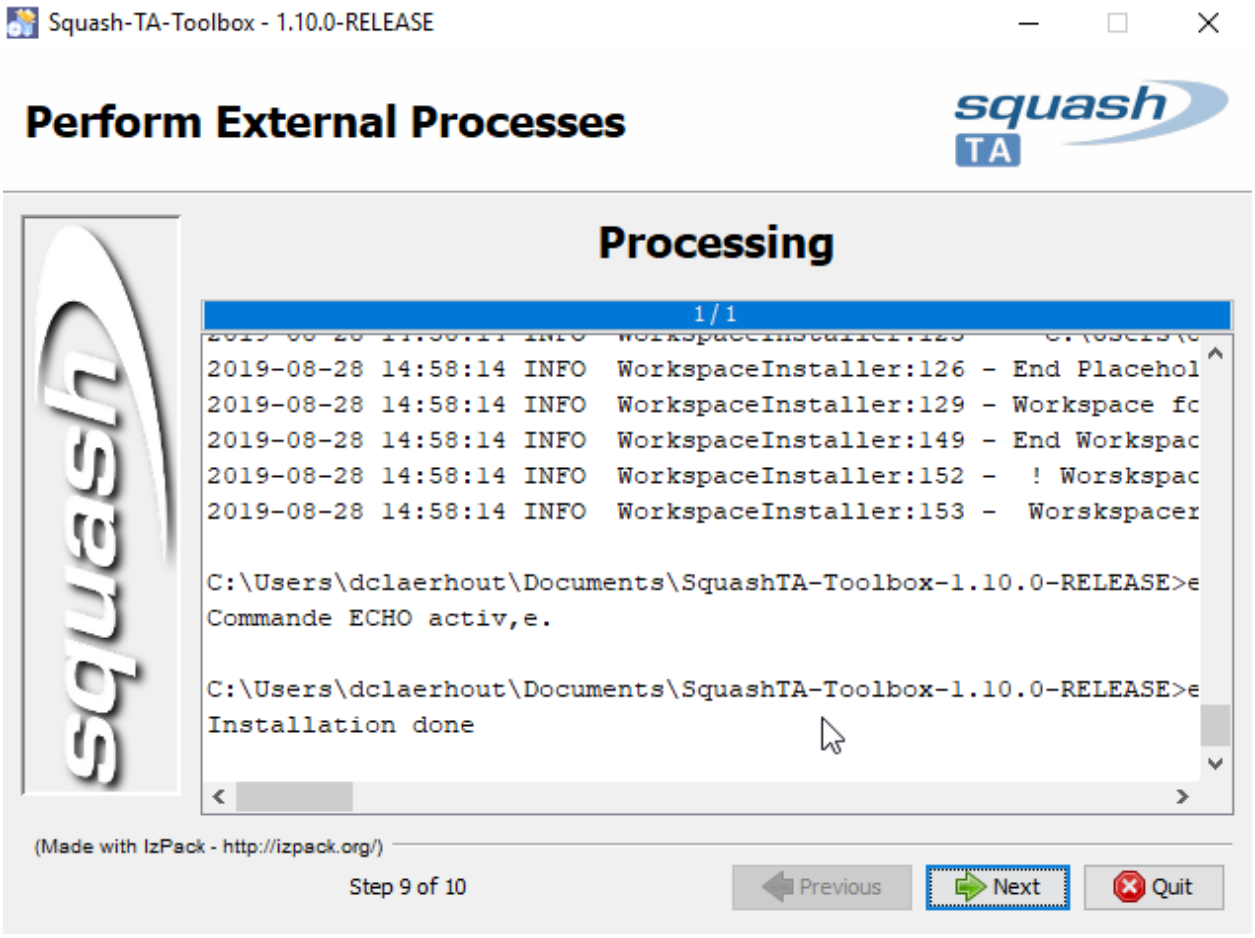
- You can choose the shortcuts you want to create for which users. To install the tools for yourself only, check **current user** on the right radio group. Click on **Next**:

- During this step, installation files are copied to the selected location. When it's done, click on **Next**:


Squash-TA-Toolbox - 1.10.0-RELEASE
Setup Shortcuts
squash
TA
Create shortcuts in the Start-Menu
Create additional shortcuts on the desktop
Select a Program Group for the Shortcuts:
(Default)
Accessibility
Accessories
Administrative Tools
Eclipse
Git
JetBrains
Maintenance
Python 3.7
create shortcut for:
current user
all users
Squash-TA-Toolbox-1.10.0-RELEASE
Default
(Made with IzPack - http://izpack.org/)
Step 7 of 10
Previous
Next
Quit


Squash-TA-Toolbox - 1.10.0-RELEASE
Installation
squash
TA
squash
Pack installation progress:
C:\Users\dclaerhout\Documents\SquashTA-Toolbox-1.10.0-RELEASE\eclipse-jee-oxygen-R-win32-x86_64...
Eclipse
Overall installation progress:
4 / 10
(Made with IzPack - http://izpack.org/)
Step 8 of 10
Previous
Next
Quit

- Some more information about the installation. Click on **Next**:



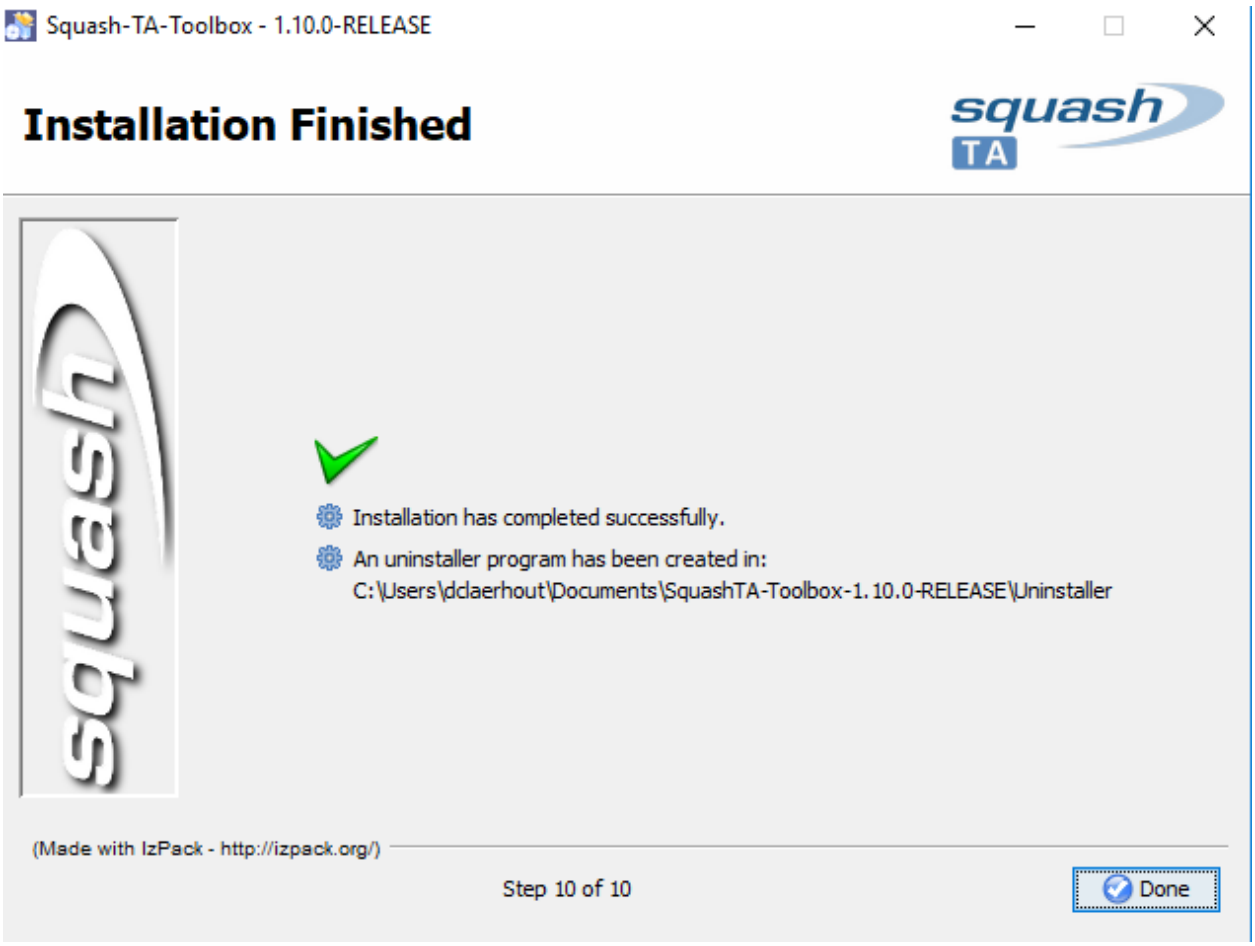
- Done! You have just installed your Squash-TF test automation environment:

Eclipse Toolbox - Upgrade

- Download the new Squash TA toolbox you want to install here.
- Install the new toolbox version in a separate directory as classical new install (For more information, please consult this *page*)
- You are ready to work.

Remark:

- Be careful, if you had special configurations for your project in your previous toolbox, then they will be lost. You had to set it again. You could use the eclipse import existing project feature, but for maven multimodule projects this doesn't work.
- **For 1.9.0** : If you are migrating to version 1.9.0 of Squash-TA-Framework, the logging has been upgraded to log4j 2-5. If you want to use project developped with older version of TA you need to modify your log4j


Squash-TA-Toolbox - 1.10.0-RELEASE
Installation Finished
squash
TA
squash
Installation has completed successfully.
An uninstaller program has been created in:
C:\Users\dclaerhout\Documents\SquashTA-Toolbox-1.10.0-RELEASE\Uninstaller
(Made with IzPack - http://izpack.org/)
Step 10 of 10
Done

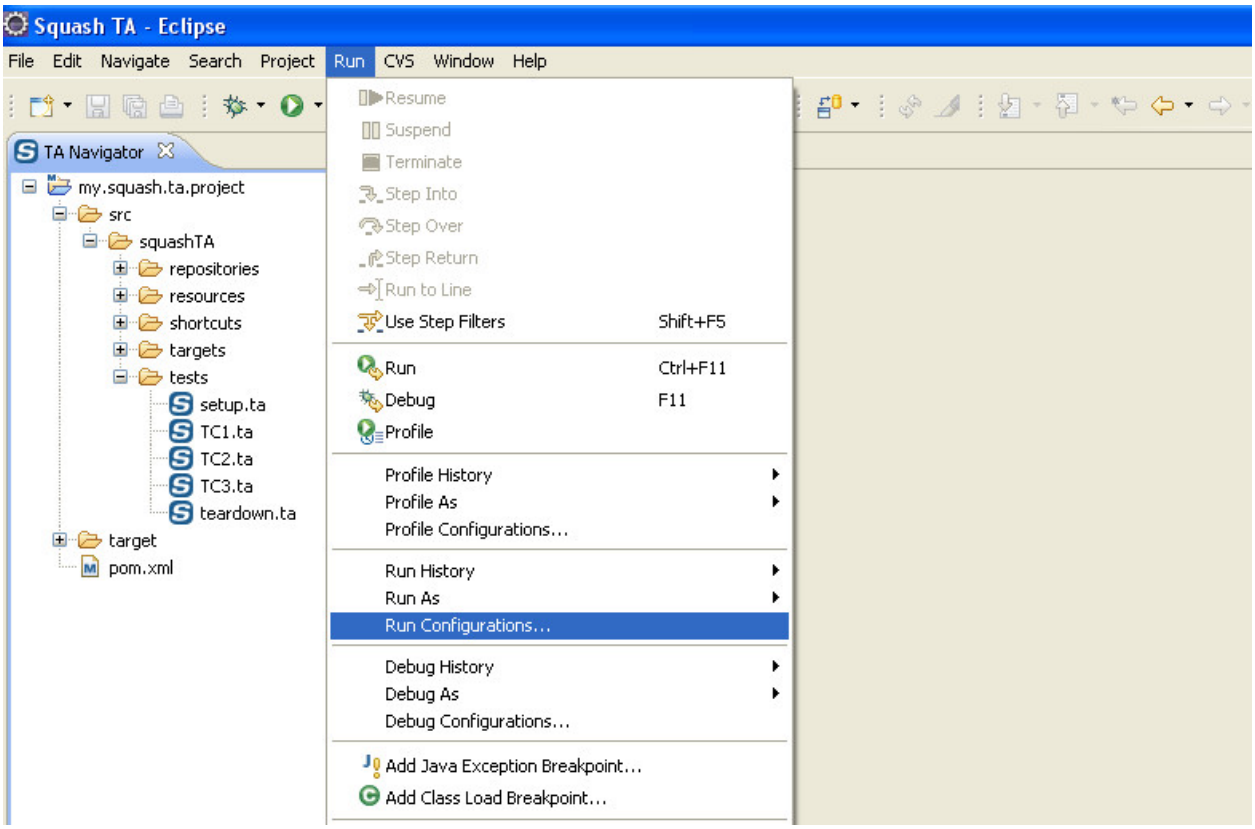
configuration files according to Apache log4j documentation.

Eclipse "Run configuration" for SKF

Squash TA toolbox default workspace embeds some default eclipse "Run configuration" to run Squash Keyword Framework / Squash TA automation project. We will describe them below. Those eclipse **"Run configuration" are part of the workspace and not of the project**. So we will see in second time how to import them.

"Run configuration" in Squash-TA toolbox

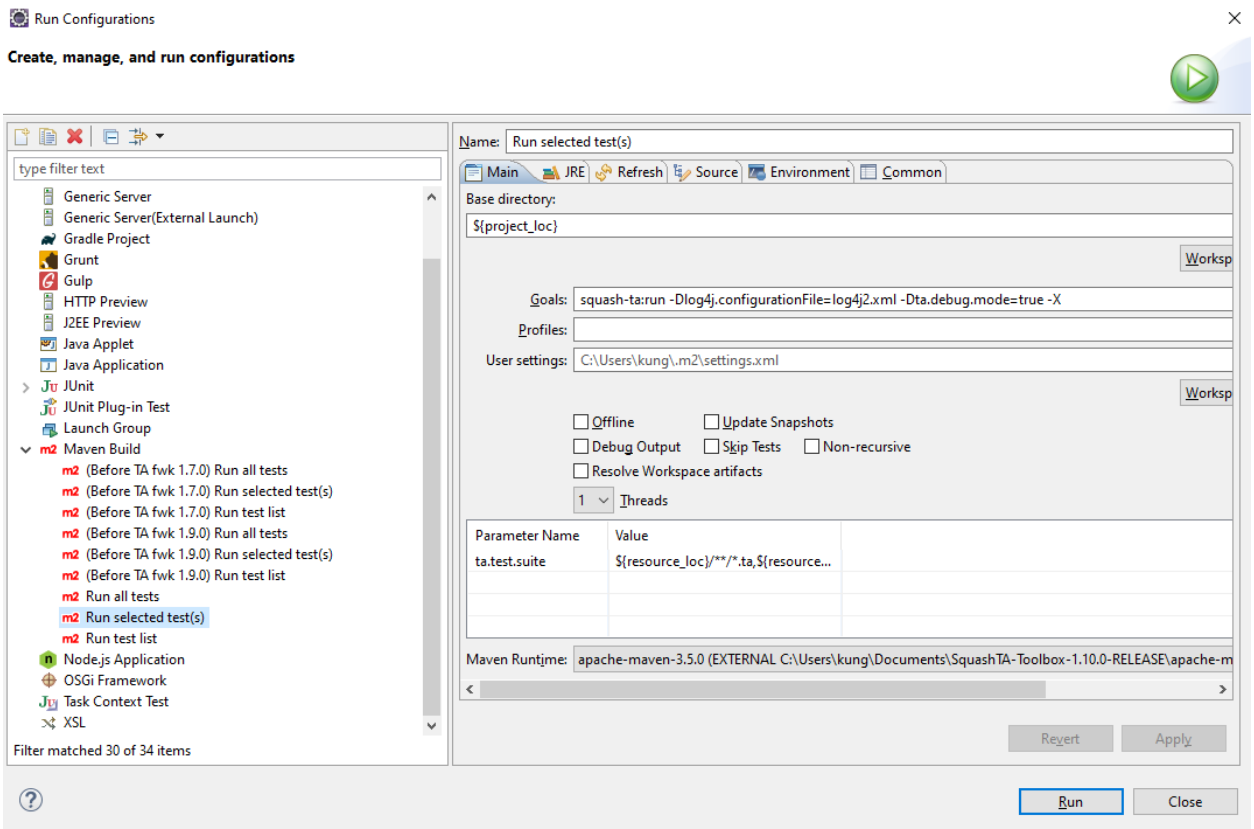
If you have installed the Squash-TA Toolbox and that you are using Squash-TA Eclipse, go to 'Run/Run configurations':



There, you have the pre-installed maven run configurations:

Since Squash-TA toolbox 1.9.0, there is three sets of run configuration:

- One set which use the goal "squash-tf:run" to run the test and specify location of lo4j configuration file "-Dlog4j.configurationFile=log4j2.xml"
  - Run all tests
  - Run selected test(s)
  - Run test list
- One set which use the maven lifecycle "squash-tf:run" to run the test
  - (Before TA fwk 1.9.0) Run all tests



  - (Before TA fwk 1.9.0) Run selected test(s)
  - (Before TA fwk 1.9.0) Run test list

- One set which use the maven lifecycle “integration-test” to run the test
  - (Before TA fwk 1.7.0) Run all tests
  - (Before TA fwk 1.7.0) Run selected test(s)
  - (Before TA fwk 1.7.0) Run test list

Note: To run older projects (< 1.9.0), you must use the prefixed run configuration..

Differences between all Run configuration:

Default Config	**squash-ta:run (+log4j)** • – Before TA Framework 1.6.0 – squash-ta:run
Before TA Framework 1.7.0	integration-test

Import eclipse "Run configuration" for SKF / Squash-TA

To import run configuration in eclipse

Download the "Run configurations" here

In eclipse open : *FileImport*

..figure :: ../../_static/eclipse-toolbox/import.png

In the dialbox, choose *Run/debug* then *Launch Configurations* and click on *:guilabel: Next*

Click on *:guilabel: Browse...* and pick the directory where "Run configurations" were download at the first step.

Select the "Run configurations" you want to import then click on *:guilabel: Finish*

The Squash-TA toolbox is a package which includes all necessary tools to realize automated tests scripts. Apart from the SKF, this package includes Open Source tools as Eclipse, Jailer (For the creation of data sets), Selenium, Sahi. This section describe the components of the Squash TA toolbox and explains how to install and how to upgrade it.

3.2 Other IDE

3.2.1 Notepad++ Squash TF UDL

A Notepad++ UDL (User defined Language) has been developed. It's a generous contribution from **edegenetais**.
This is a Notepad++ extension for syntax highlighting for Squash Keyword Framework

You can download it *here*

Documentation to install an udl extension : How to install UDL files

---

Note: The old wiki is currently still available here : https://sites.google.com/a/henix.fr/wiki-squash-ta/user

---

CHAPTER 4

Squash TF Products Download Page

4.1 Squash TF Execution Server

Latest Version : 2.2.0

`Squash TF Execution Server 2.2.0 Release Note`

4.1.1 > Execution Server : Installer

- Linux : squash-tf-execution-server-2.2.0-RELEASE-linux-installer.jar
- Windows : squash-tf-execution-server-2.2.0-RELEASE-win64-installer.jar

Download previous version: here

4.1.2 > Execution Server : Docker

The latest release tag is 2.2.0-RELEASE

Our images are now available on dockerhub :

- https://hub.docker.com/r/squashtest/squash-tf-execution-server.

Our images are still available as tarball in our repository:

- 2.2.0-release: squash-tf-execution-server.docker.2.2.0-RELEASE.tar

Download previous version: here

4.1.3 > Execution agent : Installer

- Linux : squash-tf-execution-agent-2.2.0-RELEASE-linux-installer.jar
- Windows : squash-tf-execution-agent-2.2.0-RELEASE-win64-installer.jar

Download previous version: here

4.1.4 > Execution agent : Docker

The latest release tag is 2.2.0-RELEASE

Our images are now available on dockerhub :

- execution agent : https://hub.docker.com/r/squashtest/squash-tf-execution-agent
- execution agent + X11 + chrome : https://hub.docker.com/r/squashtest/squash-tf-chrome-ui-execution-agent
- execution agent + X11 + firefox : https://hub.docker.com/r/squashtest/squash-tf-firefox-ui-execution-agent

Our docker images are still available as tarball :

- execution agent (linux):
  - squash-tf-execution-agent.docker.2.2.0-RELEASE.tar
- execution agent + X11 + chrome (linux):
  - squash-tf-chrome-ui-execution-agent.docker.2.2.0-RELEASE.tar
- execution agent + X11 + firefox (linux):
  - squash-tf-firefox-ui-execution-agent.docker.2.2.0-RELEASE.tar

Download previous version: here

4.2 Squash TF Runners

4.2.1 > Java Junit runner

Latest Version : 1.1.0

- `Squash TF Java Junit Runner 1.1.0 Release Note`

4.2.2 > Cucumber Java Runner

Latest Version : 1.2.0

- `Squash TF Cucumber Java Runner 1.2.0 Release Note`

4.3 Our development tools

4.3.1 > Squash Keyword Framework

Latest Version : 1.13.0

- `Squash Keyword Framework 1.13.0 Release Note`

4.3.2 > Squash Eclipse Toolbox

Latest Version : 1.10.0

- `TA toolbox 1.10.0 Release Note`
- squash-ta-tools-bundle-1.10.0-RELEASE-installer.exe.jar

4.3.3 > Squash Eclipse Plugin

Latest Version : 1.2.1

- `eclipse plugin Release Note`
- org.squashtest.ta.eclipse.squash-ta-eclipse-repository-1.2.1.20140326-1001.zip

4.3.4 > Squash IntelliJ IDEA Plugin

Latest Version : 0.0.2

- `IntelliJ IDEA plugin Release Note`
- squash-tf-intellij-plugin-0.0.2-RELEASE.zip

4.3.5 > Noteapad++ Squash UDL

Contribution from **edegenetais**.

Latest Version : 1.1

- `Notepad ++ Squash TF UDL`

4.4 Archives

Archives

CHAPTER 5

Squash TF Roadmap

5.1 Runners

Currently our runners concerns the Java platform (Java Junit Runner, Cucumber Java Runner) We planned to support automated tests developed in other languages.

In our scope we have

- Open the range of programming language supported. Python and C# are actually in our scope (without date for the moment).
- Be able in the future to run tests written with studios like Robot Framework or Katalon

5.1.1 > Cucumber Java Runner

- For this runner, we've planned for the next release an improvement to the reporting.

5.1.2 > Java Junit Runner

- A study for an easier integration of soapui test in maven Javen Junit projects is planned.
- At mid / long term we're thinking to create a library purchasing, to test project written in java pure code approach, some of the Squash Keyword Framework feature. The study on soapui could lead to the first baby step for this library

5.2 Squash TF Execution Server

- An update of the jenkins version and its plugin is planned for the next Squash Execution Server release
- A publication of our docker image in docker hub
- Publication of docker image for agent

- We've also planned to create a jenkins plugin to purchase a more user friendly way to connect Squash TF with Squash TM. It's a mid term goal

5.3 Our development tools

5.3.1 > Squash Keyword Framework

- With our new keyword approach we've planned to provide a large set of macro in order to be able to use all the Squash Keyword Framework features in the automation scripts without using low level instructions
- A dedicated Appium plugin is in the roadmap (instead of using the selenium plugin to execute appium test)
- In the TM TF link context, linkage through metadata in automated test instead of linked done in Squash TM interface is in reflexion

5.3.2 > IntelliJ Plugin

- An IntelliJ plugin is currently in development. The main features would be autocompletion and syntax highlighting
- In the TM TF link context, we consider to create an assistant in our Intellij plugin to :
  - identify test cases (classic or gherkin) written in Squash TM with no implementation in the automation project
  - offer a skeleton for missing implementation in the automation project

CHAPTER 6

Squash TF Community

6.1 Contribution

Note: Under construction

6.2 Developer guideline

Note: Under construction

Note: This part of the site is currently under construction

- Our bugtracker : https://ci.squashtest.org/mantis. In this bugtracker, we have 3 projects wherein you could declare the bug :
  - Squash TF Execution Server
  - Squash TF Runners (for all the runners)
  - Squash TA (For TA dsl & framework and TA toolbox)
- Our forum : https://www.squashtest.org/support-99807/forum-squash

Note: This new documentation site is under construction. Some sections are still missing. We will complete them in the following weeks.

Squash Test **F**actory (aka **Squash TF**) is a set of components whose purpose is to help you implement and run the automated tests of your projects.

A typical cycle to create an automated test has 3 phases:

- Phase 1: Describe the test to achieve
- Phase 2: Implement the automated test based on the description done in Phase 1
- Phase 3: Run the test implemented in phase 2

Squash TF is focus on phase 2 and 3. If you want to address the full cycle, our brother project **Squash TM** (**Squash T**est **M**anagement) is a good solution for phase 1. (Moreover a bridge exist between Squash TF and Squash TM)

Our goal is :

- to be able to run your tests whatever the technology you used to implement them
- to facilitate the run of your tests on multiple environments
- offer a solution which allows the use of multiple test stacks in one test
- bring tools easy to integrate in CI/CD pipelines

CHAPTER 7

The run phase

The **Run** phase is supported by *Squash TF Execution Server*

It is based on jenkins and its distributed build capabilities (for multi environments execution). Base our execution server on Jenkins bring us :

- all the features of the jenkins ecosystem (scm connectors, master/agent mode, pipeline, . . .)
- facilitate the integration in CI/CD pipelines

We *distribute* Squash TF Execution Server in two ways:

- as installer for windows and linux
- as docker image for linux

CHAPTER 8

The implementation Phase

For the implementation phase, there is 3 ways to do it :

- Use a pure code approach (with or without frameworks)
- Use a dedicated studio like Robot Framework or Katalon
- Use our implementation solution : *Squash Keyword Framework*

Whatever way you choose to implement your automated tests, we want to be able to execute them in Squash Execution Server

Pure Code & dedicated studio approach

For people which :

- already have automated tests written
- used specific technology (like their own test robot)

When automated tests are already (will be) written in pure code or with a dedicated studio we need a solution to run them as seamless as possible. To this purpose we decided to create runners.

Runners main goal is to run automated tests in our ecosystem and also ensure the link with our brother project Squash TM for execution of test not written with Squash Keyword Framework. Runners have to :

- manage the test suites to execute (with filter or json)
- launch the executions
- manage the the reports creation
- when linked with Squash TM * report tests execution status to Squash TM * send execution reports to Squash TM

As Squash TF is mainly written in Java, we have started by creating two runners tight to Java :

- Squash TF Java Junit Runner
- Squash TF Cucumber Java Runner

Our implementation : Squash Keyword Framework

We have also created our own implementation stack called *Squash Keyword Framework*, which is a keyword test framework using, most of the time, existing Robots. This framework includes a DSL for writing automated tests scripts and an engine to execute the scripts. It allows the use of multiple robots in one single test.

> *Keyword test framework*

In a keyword tests framework :

- test scripts are a composition of keywords
- keywords are small blocks of reusable features
- the framework provides a library of default keyword.

In Squash Keyword Framework :

- keywords are made of macros
- a large set of built-in macros is provided
- macros can be combined to create a higher level macro
- macros could also be created by using low level instructions, if needed

> *Engine with plugins*

At the heart of our framework, there is an engine which using plugins to pilot test robots. For each robot, a plugin (connector) has been created which provides :

- the built-in macro
- the low level instructions and their implementation (to pilot the robot).

The Squash Keyword Framework plugin architecture makes it an expandable solution : you can contribute by creating a new plugin for a new robot (or by extending an existing one). See community section for contribution.