

Extending Jubula

Jubula Team

BREDEX GmbH

June 29, 2012



BREDEX GmbH
Mauernstr. 33
38100 Braunschweig
Germany
Tel: +49-531 - 243 30 - 0
Fax: +49-531 - 243 30 - 99
www.bredex.de

GUI*dancer* is a registered trademark of BREDEX GmbH

Title: Extending Jubula
Author: Jubula Team
File: EXTEND
State: RELEASE
Version: V6.0.01011
Released by: BREDEX GmbH
Released at: June 29, 2012

Contents

1	Introduction	5
1.1	What does a Tester Class look like?	5
2	General Steps to take	7
2.1	Requirements	7
2.2	Jubula Client Extension	8
2.2.1	Exporting the toolkit plugin to Jubula	8
2.3	Jubula Server Extension	10
2.4	Where to put your Tester Classes	11
2.5	Jubula Example Extension	11
3	Issues	13
3.1	Jubula Updates and Upgrades	13
4	Functions	15

When developing graphical applications, it is often necessary or convenient to alter or combine the functionality of existing toolkit components, or even to write entirely new ones, as the requirements or concept of the software may dictate. These new components generally cannot be tested by Jubula "out of the box", as the behavior of custom components cannot be predicted, or they may deviate from established standards of "look and feel". In order to overcome this limitation, Jubula offers an extension API, which you can use to allow Jubula to test your custom components.

The following sections describe the steps involved in extending Jubula.

Chapter 1

Introduction

Jubula starts, controls, and observes AUT's using its server component. In order for the AUT Agent to know how to control each element of a GUI, we have outfitted the AUT Agent with a pluggable interface for graphic components. Using this interface, each component that Jubula can test is described in a so-called **Implementation Class**. Each GUI toolkit that Jubula supports is described in a toolkit plugin.

Because of the great flexibility that user customization allows, we have opened up this interface to our users. You can add to the existing functionality of our officially-supported Implementation Classes, or provide support for in-house graphic components by defining Implementation Classes of your own, which we will refer to as **Tester Classes**.

1.1 What does a Tester Class look like?

The functionally important aspect of a Tester Class is that it contains a public method for each Jubula test action which will appear in the client. These methods are linked to testable actions within a user-defined Jubula Plugin, which is described later in this handbook. Each plugin provides a XML configuration file, which tells Jubula which method to call, what parameters it needs to send, as well as string externalization information.

Chapter 2

General Steps to take

The following chapter describes the general steps to take when extending Jubula with custom defined components and actions. Detailed information for each step can be found in the corresponding example extension files in "InstallationDirectory/examples/development/extension/src"

In general you always have to extend two parts of Jubula

- The Jubula Client by writing your own `Toolkit-Plugin`:
This will tell the client which new components and actions are available.
- The Jubula Server Component by putting a compiled class file to a specific directory:
This is the part which is actually performing the actions on the new component.

2.1 Requirements

To create your own Jubula extension, you need:

- Jubula 2.0 or later
- Eclipse 3.4 or later
- JDK 5.0

You must also have write access in the directories:

InstallationDir\Jubula\plugins

InstallationDir\server\lib\extImplClasses

2.2 Jubula Client Extension

The following steps have to be done to extend the Jubula client:

1. Create an eclipse plug-in project and a corresponding feature project.
2. Set "InstallationDir\Jubula\plugins" as your target platform
3. Define plugin dependencies to the toolkit support plugin
4. Enter the toolkit support plugin in your toolkit plugin project
5. Create a MyToolkitProvider class
6. Define and configure the toolkit extension at the extension point
7. Create a myComponentExtension.xml
 - Extend existing Jubula components with new actions
 - Derive components from existing Jubula components
 - Define a new component
8. Manage the i18n keys
9. Export the toolkit feature to Jubula

Under "InstallationDirectory/examples/development/extension/src" you will find a "eclipseProjects_ExampleSwingClientExtension.zip" which contains an example Jubula Client Extension for the Swing component "JSlider" as well as a corresponding feature project. These projects are a showcase for steps 1-8.

2.2.1 Exporting the toolkit plugin to Jubula

The only steps you have to do after importing the projects into your Eclipse workspace and setting the target platform (step no. 2) is to export the feature to an update site and then use the update site to install the feature into the Jubula you want to extend.

To export the toolkit feature to an update site:

1. In the package explorer, right click on the feature project (ex. *org.eclipse.jubula.examples.extension.swing.feature*) and select: *Export...* from the context-sensitive menu.

2. In the dialog that appears, select *Deployable features* and click "Next".
3. In the next dialog, in the *Available Features* area, ensure that the checkbox next to the feature you wish to export is selected.
4. In the *Destination* tab, enter the location where the feature's update site should be exported to in the *Directory* field. This can be any writable directory. This directory will serve as an update site, which can later be used to install your feature into Jubula.
5. In the *Options* tab, ensure that the *Package as individual JAR archives* checkbox is selected and click "Finish".

To install the toolkit feature from the update site:

1. Start Jubula and select **Help** → **Install new software...** from the main menu.
2. In the *Install* dialog that appears, click the *Add...* button.
3. In the *Add Repository* dialog that appears, click the *Local...* button.
4. In the file selection dialog that appears, navigate to the directory that contains your update site and confirm the dialog.
5. Click *OK* to exit the *Add Repository* dialog. The active dialog should now be *Install*.
6. Ensure that the *Group items by category* checkbox is deselected. Your feature should be visible in the central table of the dialog.
7. Ensure that the checkbox next to your feature is selected and click *Next*.
8. Confirm the *Installation Details* by again clicking *Next*.
9. Accept the license agreement terms and click *Finish*.
10. A warning dialog may appear to warn of unsigned content. Click *OK* in this dialog if the feature comes from a trusted source (ex. if you have written the software yourself, as in this example). This will begin installation.

11. After installation, a dialog appears suggesting that Jubula be restarted in order to safely finish the update / installation. Click *Restart Now* to perform the restart. Once the restart completes, your extension feature has been installed in Jubula.

2.3 Jubula Server Extension

AUT's are controlled by the server. There exists a Tester Class for each component that Jubula supports. This class implements the test actions that can be carried out on the component. To add your component to Jubula, you need to write a Tester Class for it.

Please follow the following guidelines for your Tester Classes:

- Your build path must contain the following three JAR files: `org.eclipse.jubula.rc.swing.jar`, `org.eclipse.jubula.rc.swing.implclasses.jar` and `org.eclipse.jubula.tools.jar`, which contain our server classes and some utility classes. They are located in your Jubula installation directory under `server/lib`.
- The class must be compatible with *Java 1.4*.
- Its declared package name must begin with: `"org.eclipse.jubula.rc.swing.swing.implclasses"`
- It must implement the following interface: `"org.eclipse.jubula.rc.swing.swing.implclasses.IImplementationClass"`
- It must provide **public** methods for each action that is implemented for the component.
- Each method that implements an action must throw the following exception upon error: `org.eclipse.jubula.rc.common.exception.ActionFailedException`. This way, Jubula will be able to know that an action has failed.

Under `"InstallationDirectory/examples/development/extension/src"` you will find a `"eclipseProjects_ExampleSwingServerExtension.zip"` which contains an example Jubula Server Extension for the Swing component `"JSlider"`.

Now that you have written your Tester Class, you still need to make Jubula aware of its presence. This is done by first putting the compiled class in a location where Jubula can find

it, then by altering the configuration file, so that Jubula knows which component it refers to, and how it can be used. The following sections explain how this is done.

2.4 Where to put your Tester Classes

If you extend the `SwingToolkitPlugin`, your compiled Tester Classes (*.class files) need to be placed in your Jubula installation directory under the following path:

```
server/lib/extImplClasses,
```

using a directory structure that corresponds to the fully-qualified package name. Therefore, your Tester Classes should be in some sub-directory of the following path within your Jubula installation directory:

```
server/lib/extImplClasses/org/eclipse  
/jubula/swing/swing/implclasses
```

As *Eclipse* already stores compiled classes according to this structure, you need only (recursively) copy the directory structure, starting with `org`, into the `extImplClasses` directory.

Alternatively, you may place a JAR containing the above structure (also starting with `org`) into the `extImplClasses` directory.

Please note that if the `extImplClasses` directory does not already exist, you must create it in the above-mentioned location.



2.5 Jubula Example Extension

Jubula comes with a complete example extension implementation in source and binaries. This example extension extends Jubula for the Swing component "JSlider". After deploying the Jubula Client plug-in and the Jubula Server extension you should be able to test the Swing component "JSlider" at "Graphics Component"-level with Jubula. The example extension code and binaries can be found in the "InstallationDirectory/examples/development/extension":

- AUT

This directory contains a trivial example AUT which uses

the originally unsupported component "JSlider". After installing the extensions, your Jubula will be able to test this new component.

- src

This directory contains several archive files which are all importable Eclipse projects:

- eclipseProjects_ExampleSwingAUT.zip
This is the source code project for the example Swing AUT.
- eclipseProjects_ExampleSwingClientExtension.zip
This is the source code project of the extension plug-in for the Jubula Client.
- eclipseProjects_ExampleSwingServerExtension.zip
This is the source code project of the extension for the Jubula Server.

- bin

This directory contains the compiled sources as directly deployable units.

Chapter 3

Issues

When working with extended or user-defined Tester Classes, certain issues may arise which influence the normal operation of Jubula. The following explains some of these issues and how to avoid problems or work around them, should they arise.

3.1 Jubula Updates and Upgrades

With a new version of Jubula, you need to ensure that your toolkits and tester classes are updated in line with the new Jubula version, as described in the sections earlier in this manual.

Chapter 4

Functions

Functions represent program code that is evaluated during test execution. In order to define a new Function, you will need to create an extension for the extension point *org.eclipse.jubula.client.core.functions*, which can be found in the bundle *org.eclipse.jubula.client.core*. This bundle also contains reference implementations of the extension point.