CaptainCasa Enterprise Client

Automated UI Testing

Please pay attention:

The “Automated UI Testing” tool is an add-on to the CaptainCasa Enterprise Client toolset – though being part of the standard installation.

This means: you can use it, you can receive the sources for the add-on (and also modify for own purposes if you pass back the results into the community), but: there is no kind of warranty that CaptainCasa takes over.

Table of Contents

Overview & Positioning

The “Automated UI Testing” was derived out of a customer project and does not try to get into competition with “bigger-in-scope” tools that are available on the market.

Its positioning is “simple recording and replaying of test cases”. As a result you can:

- record user activities
- replay user activities, either online or by an ANT script
- validate replayed user activities

And, maybe even more important, there are restrictions:

- You can NOT change any recorded data. For example you may want to iterate through various items and update the data that is entered accordingly – no chance!
- In case the layout changes within your application you have to re-record all your test cases. There's no automated process of trying to find a certain component at its new location.
- You can not compose big test cases out of small ones.
- You can not do any kind of scripting within the test processing.

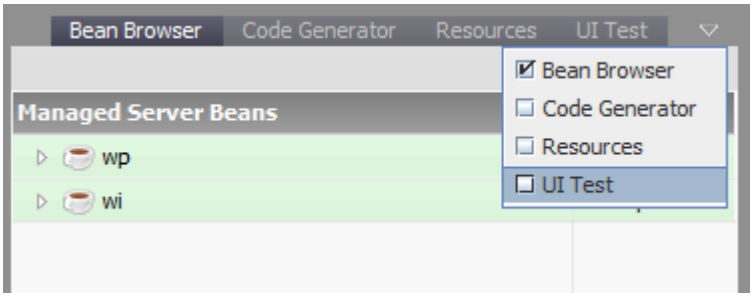
So, you see: the “Automated UI Testing” is simple in what it does: record and replay. - Is this sufficient for automated UI testing? - “Yes and no”, that's what we believe. “Yes”, because it's nicely usable at least for doing all the hot-spot-tests in the critical areas of your application. “No”, because if your test scenarios grow and grow you will have more and more demands towards the test environment, a simple record&replay will not be enough.

CaptainCasa has a nice technical partnership with “Quality First GmbH” (www.qfs.de) – this is a point to go to when looking for a more comprehensive UI testing environment.

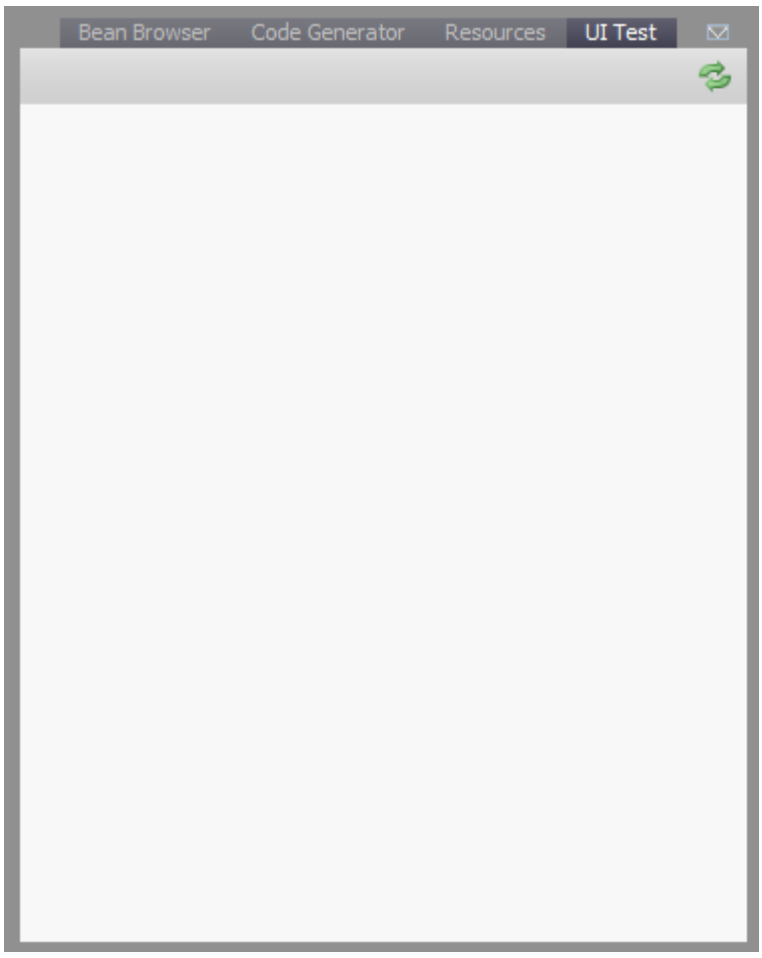
Basics

How to use

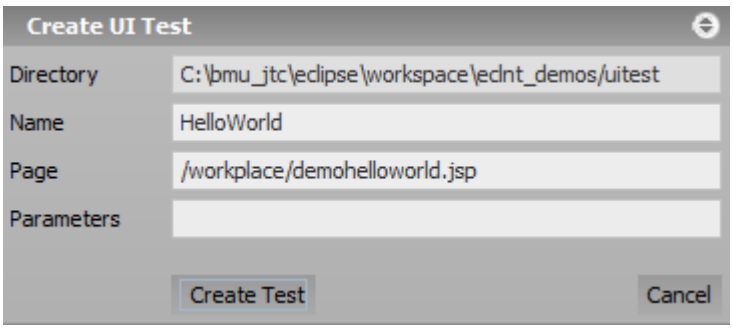
On the right of you CaptainCasa toolset, you see the “UI Test” tool.



After opening it, you will see an empty screen – there is no test case defined so far:

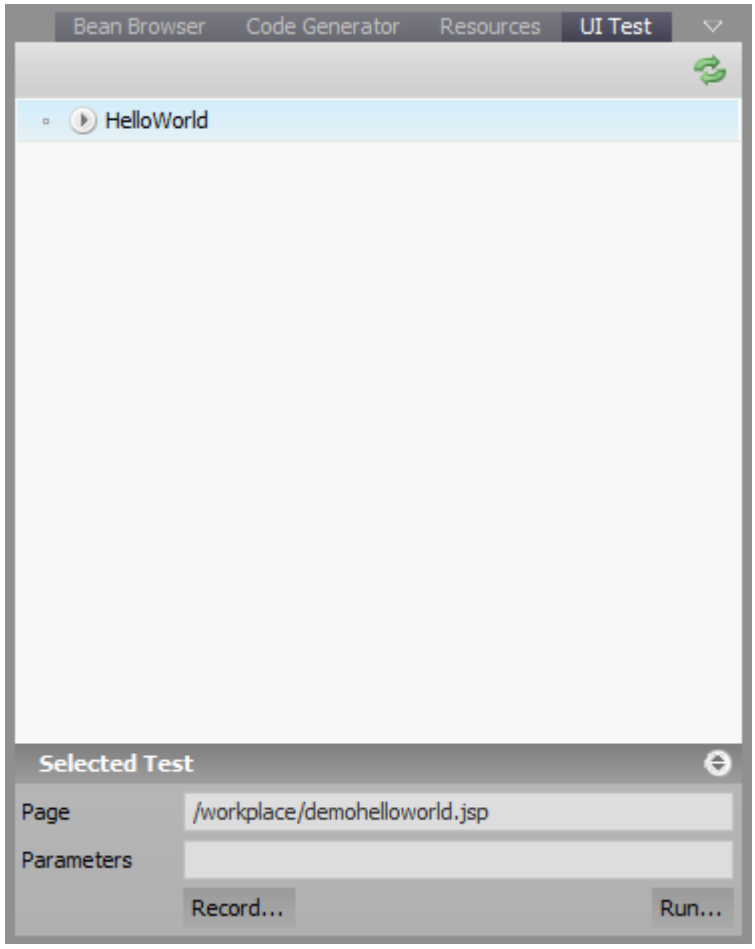


By pressing the right mouse button you can define a new test case:



The data you need to define is:

- The name of the test
- The page to start the test with. The page must be specified from the root-webcontent directory's perspective.



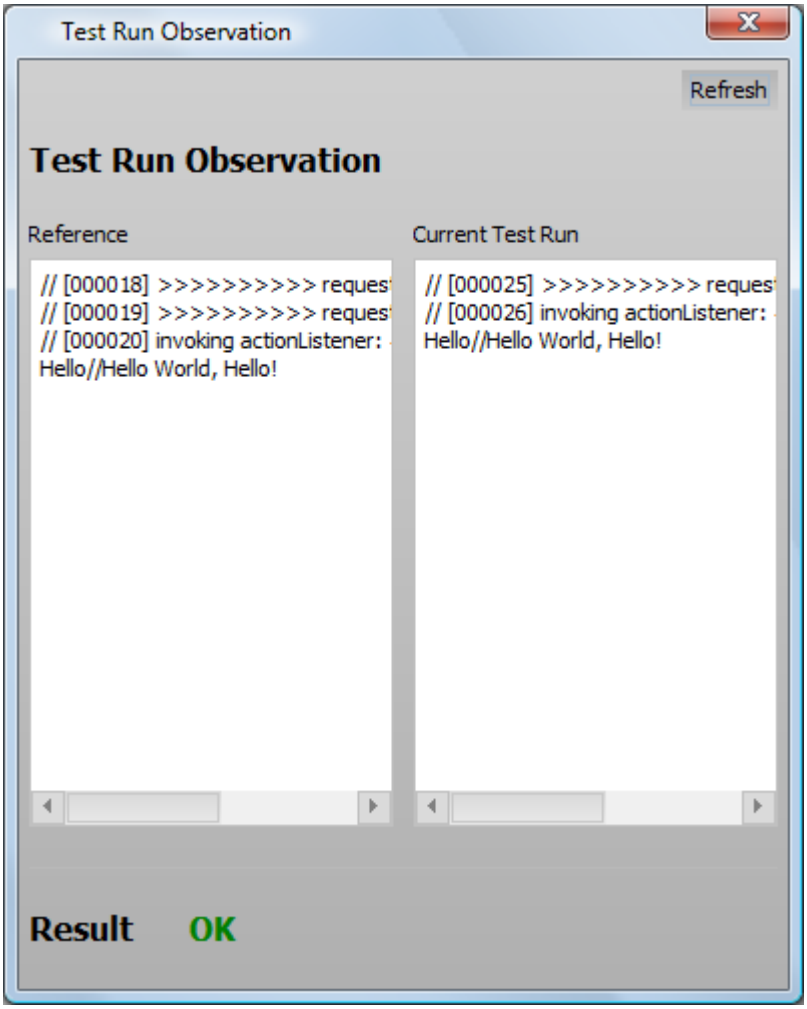
By pressing “Record...” the page that you specified will be shown. From now on any interaction that you do will be recorded in an even-log.

Please do not do one of the following while recording:

- Do not move or resize the screen.
- Do not jump into other applications that run in parallel.

After closing the application client the recording will be automatically stopped. You now can replay the test case. After starting the recording you will see dialog in which certain

recorded data from the record-run will be compared against the data in the replay-run:



The test is OK, when both the left and the right side contain the same recorded data (comments are excluded from the comparison).

Files that are written

Internally the following activities are performed:

- A directory “uitest” will be created within your project.
- For each test case a corresponding directory will be created below “uitest”.

The test case directory contains the following files:

```
<yourproject>
    /src
    /uitest
        /uitest_HelloWorld
            eventlog.log => the events as recorded
            clientmessages_reocrd.log => “left” side of comparison
            clientmessages_replay.log => “right” side of comparison
            record.bat => batch file for recording
            replay.bat => batch file for replaying
            replay.xml => ANT script for replaying
            uitest.xml => some data that is kept for the test case
    /webcontent
```

Starting from Command Line

The best way to start the automated testing for a specific page is to use the ANT script that was created within the test case directory (replay.xml).

The ANT file is created when starting the recording one time from the tool's user interface.

You may use ANT in general to organize your testing environment. You may for example first call some ANT scripts to initialize your application, then call some test cases, then clean up some data, etc. etc.

Recording Client Test Log

When recording or running a test case then a so called client test log is written. This client test log contains application information about what's going on in the application. Though the test log is recorded on client side, the test log is written by the application on server side.

Imagine the test log being a “virtual status bar”. Messages that are written to the test log are not optically rendered to the user (as with the status bar) but are recorded within a file on client side.

The client test log it the one that is recorded both when recording a test case or when replaying a test case. Recorded test log and replayed test log are compared after replaying a test case. If they contain equal information then the whole test case is set to “OK”.

Adding Information to the Client Test Log

There is a simple API to add test log messages within your application:

```
ClientTestLog.addClientTestLogMessage(“a message”);
```

And there are some additional APIs as well:

```
ClientTestLog.checkIfClientTestLogIsActive();
ClientTestLog.addClientTestLogComment(“a comment”);
```

A comment that is added will not be considered when comparing the recorded test log against the replayed test log.

When to add Information

In order to simplify and unify the adding of information to the client test log there is a quite nice function: after each action listener being invoked on server side, the corresponding object is called with the method “onCreateClientTestLogMessage()” - if the application is run either in record- or replay-mode.

The DemoHelloWorld application for example looks as follows:

```
package workplace;

import javax.faces.event.ActionEvent;
import org.eclnt.jsfserver.util.ClientTestLog;

public class DemoHelloWorld extends DemoBase
{
    String m_name;
    String m_output;

    public void setName(String value) { m_name = value; }
    public String getName() { return m_name; }

    public String getOutput() { return m_output; }

    public void onHello(ActionEvent ae)
    {
        if (m_name == null)
            m_output = "No name set.";
        else
            m_output = "Hello World, "+m_name+"!";
    }

    public boolean onCreateClientTestLogMessage()
```

```
    {
        ClientTestLog.addClientTestLogMessage(m_name + "//" + m_output);
        return false;
    }

}
```

So regardless what action is taken within a bean on server side, the onCreateClientTestLogMessage is called. Typically you add information to the log that summarizes the data contained within your page.

In order to simplify the writing of test messages in more complex scenarios there is a certain “drill up” of method calls: if the action listener that is invoked is “#{d.d_1.DemoHelloWorld.onHello}”, then the following sequence of calls is executed just afterwards:

```
{d.d_1.DemoHelloWorld.onCreateClientTestLogMessage}
{d.d_1.onCreateClientTestLogMessage}
{d.onCreateClientTestLogMessage}
```

You see: the call of the “onCreateClientTestLogMessage” is in general done on each level of the object hierarchy on server side. The boolean result of the method indicates when to stop this “walking up the hierarchy”. If returning “false” then the walking up is stopped.

What to add

The messages that you add to the client test log will be the ones to be compared afterwards in order to check if a replayed test case executed “OK” or “not OK”.

As a result you should pay attention to what to write to the log:

- Do not write any data that changes each time you start the application. If you output a timestamp into the log then you will never received equal logs on client side.

CaptainCasa GmbH

Hindemithweg 13
69245 Bammental

Tel +49 6223 484147

http://www.CaptainCasa.com
info@CaptainCasa.com