SETUP

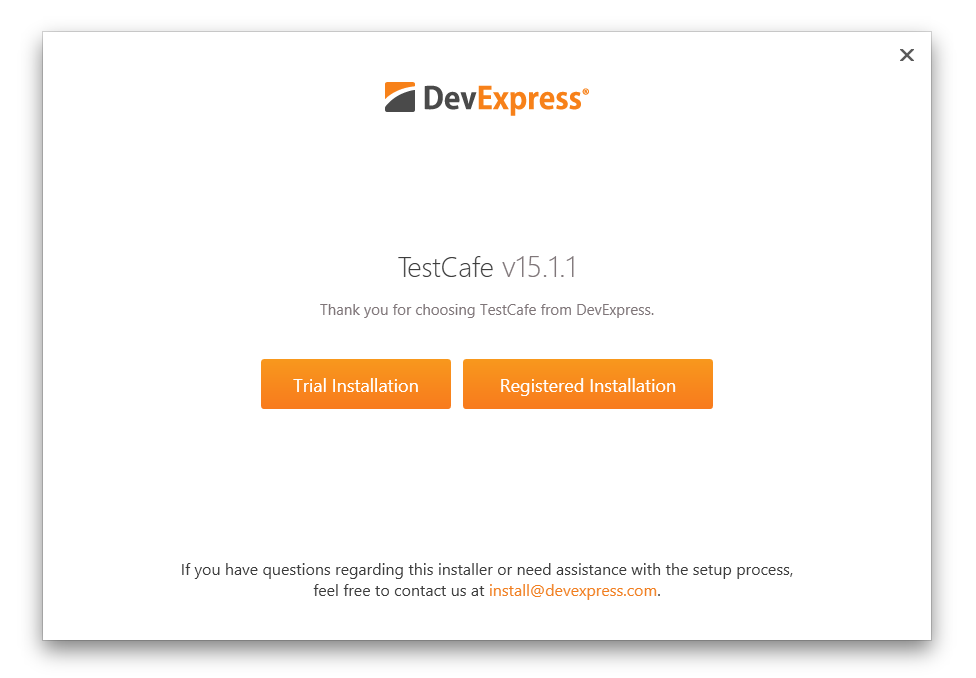
**Distribution Files**

[Download TestCafe](https://testcafe.devexpress.com/Download)

Since TestCafe can be used on different operating systems, it is distributed as an installation file for Windows and MacOS, and a separate ZIP file for Linux.

**Windows**

When using Windows, the installer will ask you to carry out a few simple steps. If you are the registered user, enter your DevExpress.com login credentials or choose the "Trial installation" option. Then, enter the physical path you want to install TestCafe to (the default path: Program Files\DevExpress\TestCafe). After installation is completed, the installer will unzip the required files, create the TestCafe shortcut in the startup menu and open the TestCafe Control Panel in your default browser.



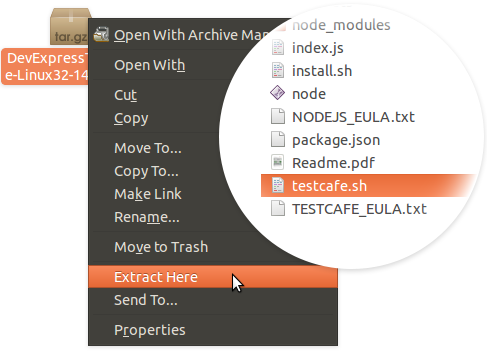
**MacOS**

The installer for MacOS will request the administrator's password and setup TestCafe into Applications/TestCafe.



**Linux**

For a Linux machine, once downloaded, unzip the contents of the TestCafe archive to the directory where you wish to store TestCafe program files. Note that this is not a temporary folder - no further installation or unpacking is required.



If you need to relocate TestCafe program files to a different directory, simply move them to the new location.

**First Run**

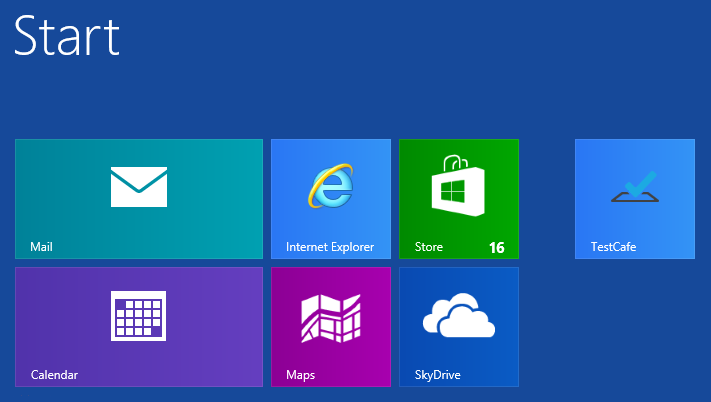
**TestCafe Settings**

During the first run, TestCafe automatically collects the required information, including the installed browser list. To modify TestCafe settings manually, please refer to the [TestCafe Settings](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/" \l "TestCafe_Settings) section in the documentation.

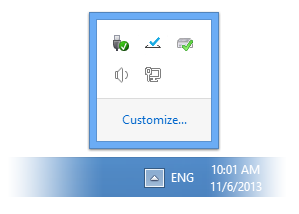
**Windows**

The Control Panel will be opened within your web browser automatically after installation is complete by default.

Alternatively, you can run TestCafe from the Windows startup menu.

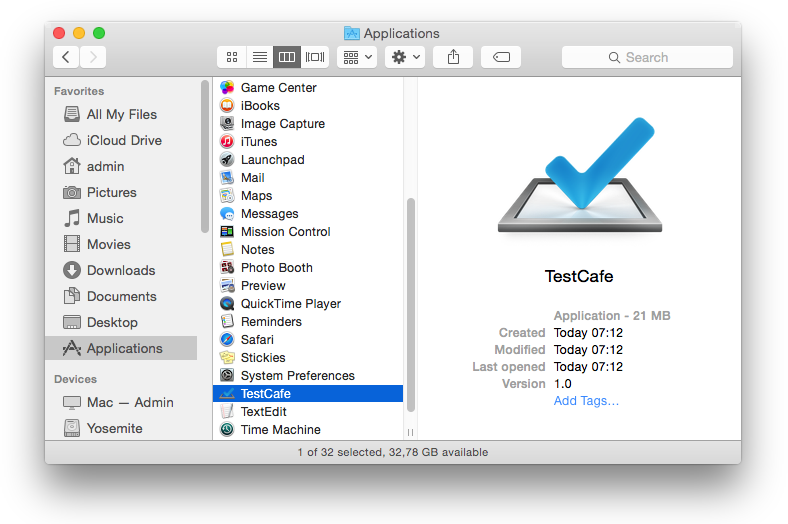


Once you've launched TestCafe, it displays a specific icon within the system tray. Double-click this icon at any time to open the TestCafe Control Panel. You can also right-click the icon to access the context menu allowing for the same action, copying the TestCafe's URL to the clipboard or shutting down the TestCafe process.

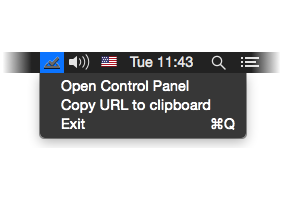


**MacOS**

On a MacOS machine, TestCafe can be run from the Applications directory.



Once you've launched TestCafe, it displays a specific icon within the status bar. Double-click this icon at any time to open the TestCafe Control Panel. You can also click the icon to access the context menu allowing for the same action, copying the TestCafe's URL to the clipboard or shutting down the TestCafe process.



**Linux**

On Linux systems, execute the following two commands for the first run.

$ cd ~/(path to TestCafe)

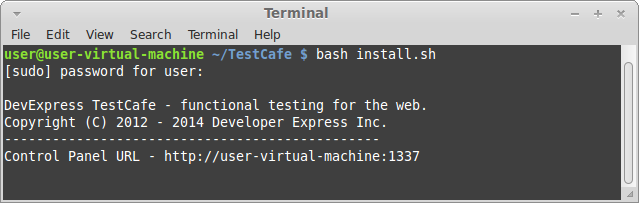
$ bash install.sh

The moment installation is complete, a console application will be launched, which will start the TestCafe process and open your default browser with the TestCafe Control Panel.

To run TestCafe at a later time, use the following command.

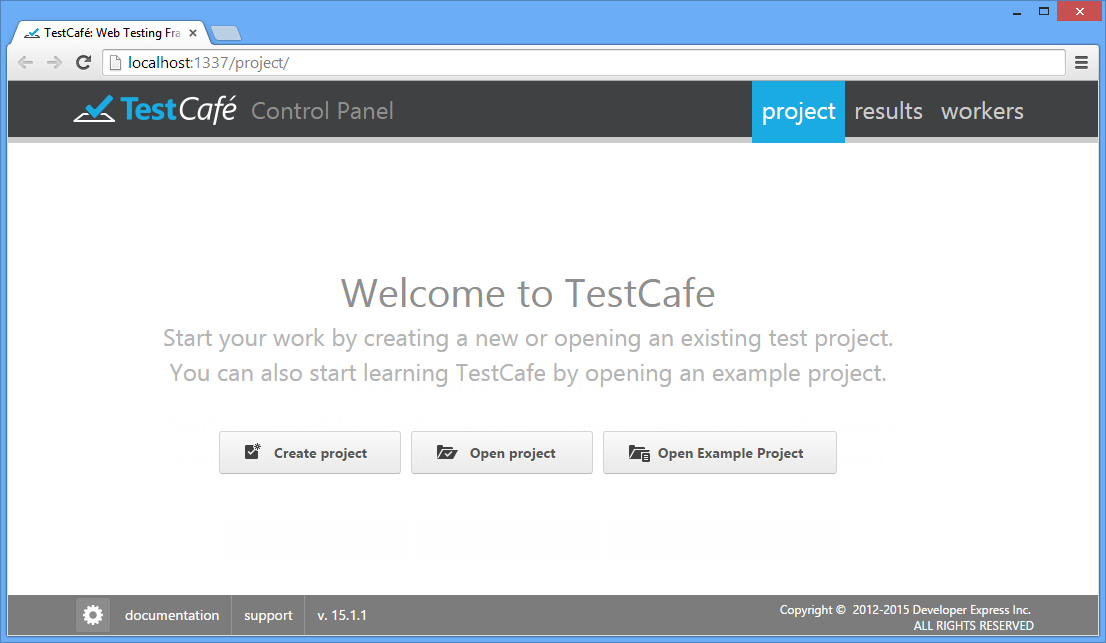
testcafe

To exit TestCafe, close the console application and the Control Panel.



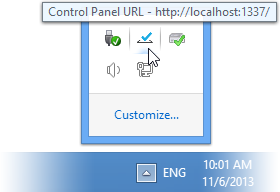
**Access to the Control Panel**

Once the runner completes its initialization, TestCafe automatically starts your default browser and displays the Control Panel.



This UI is available from any browser while TestCafe is running. Simply use the URL displayed in the address line of the browser where you started TestCafe. If you use default port and hostname values, the URL is http://your-computer-name:1337.

If you use the Windows or MacOS operating system, the Control Panel URL is also shown when you point at the TestCafe icon in the system tray (for Windows) or menu bar (for MacOS). You can also copy the URL to the clipboard from the TestCafe icon's context menu.

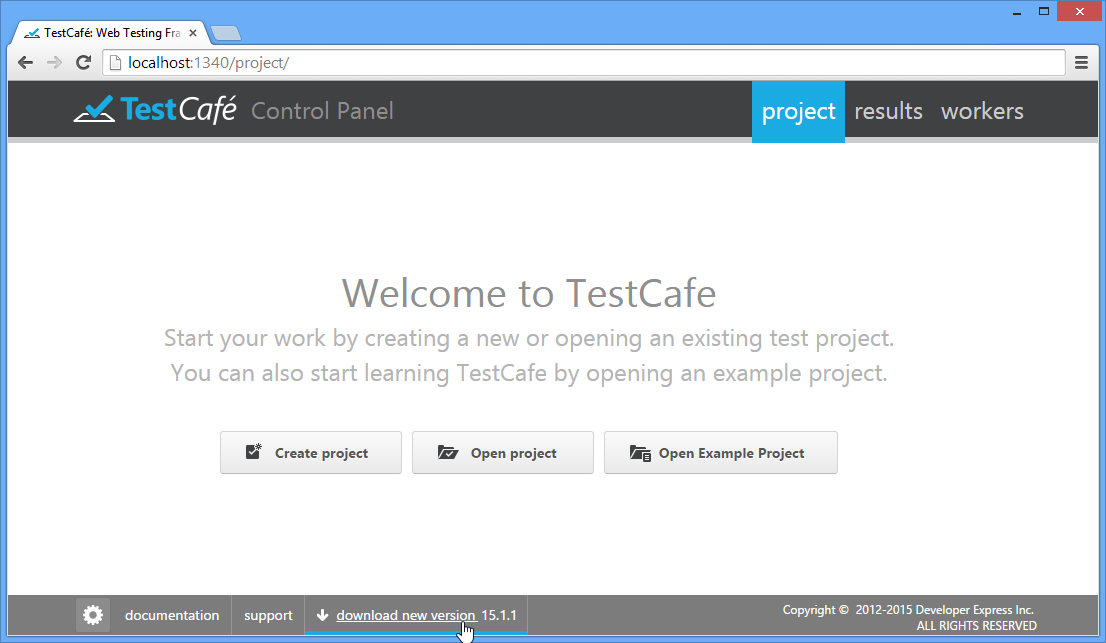


If you use the Linux system, use the URL displayed in the command prompt.



**Upgrade**

You can upgrade TestCafe to a new version either on the [TestCafe website](http://testcafe.devexpress.com/) or within the TestCafe Control Panel by using a specific software update indicator. The indicator becomes a link automatically when a new TestCafe version is available.



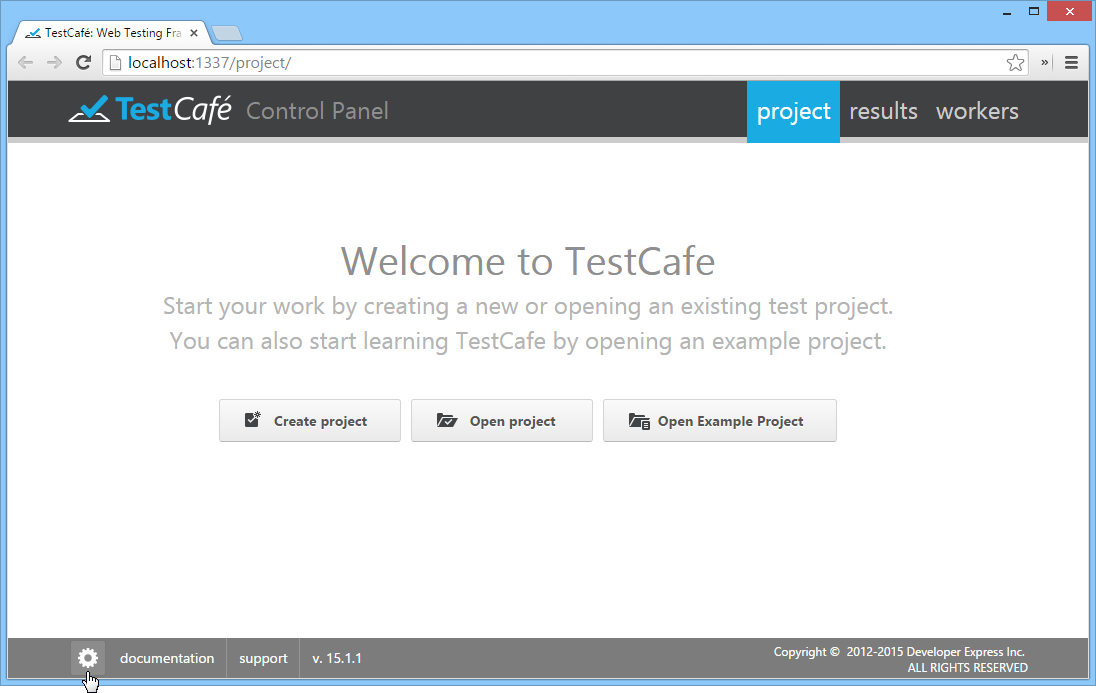
Different versions of TestCafe don't conflict with each other on the same machine. So, one way to upgrade to a new version is to simply launch the installation (for Windows and MacOS) or unpack the newly downloaded archive to a new directory (for Linux), and then use the previously created configuration file and test files.

For Linux, if you wish to replace your existing version with a new one, first make a backup copy of your tests (JavaScript files) and the configuration file (config.json). Then, clear the destination folder, unpack the files, replace the configuration file and tes

# CONTROL PANEL

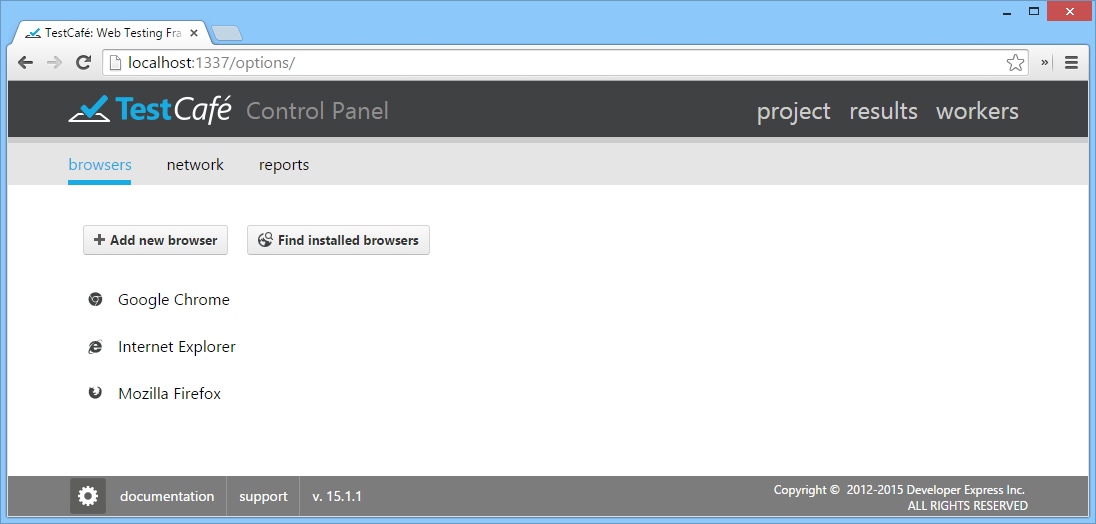
## TestCafe Settings

You can access TestCafe settings from the Control Panel via the TestCafe settings button in the bottom left corner of the window.

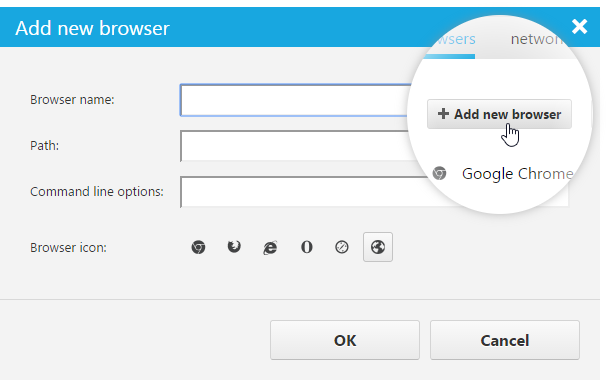


This button opens a settings view that contains the following sections: **Browsers**, **Network** and **Reports**.

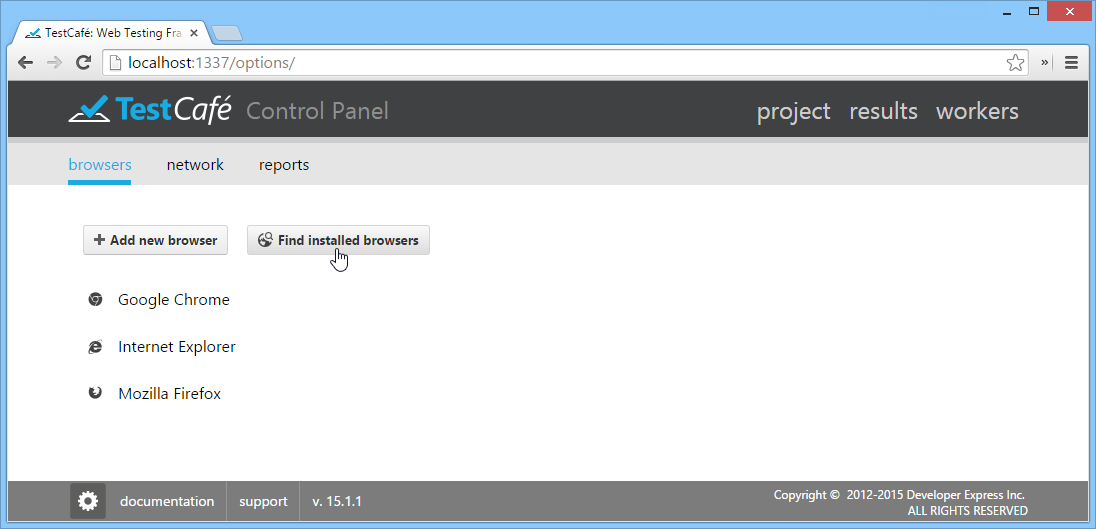
Use the **Browsers** section to edit the list of browsers used by TestCafe.



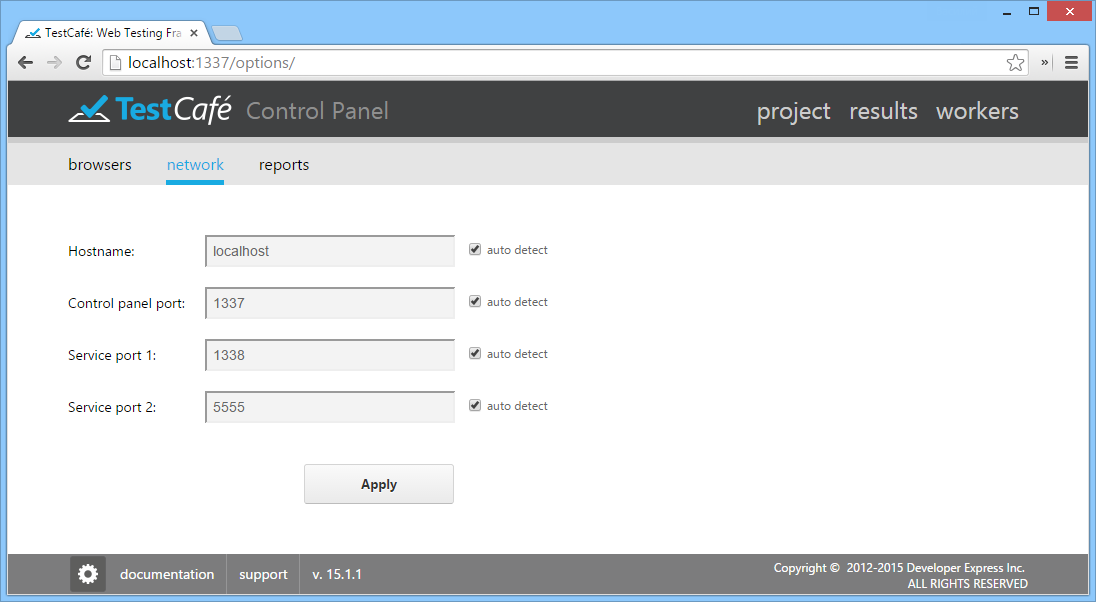
To add a new browser, click the **Add new browser** button. This button invokes a dialog in which you can select the browser's executable file, specify its name, as well as optionally add command line parameters and choose an icon.



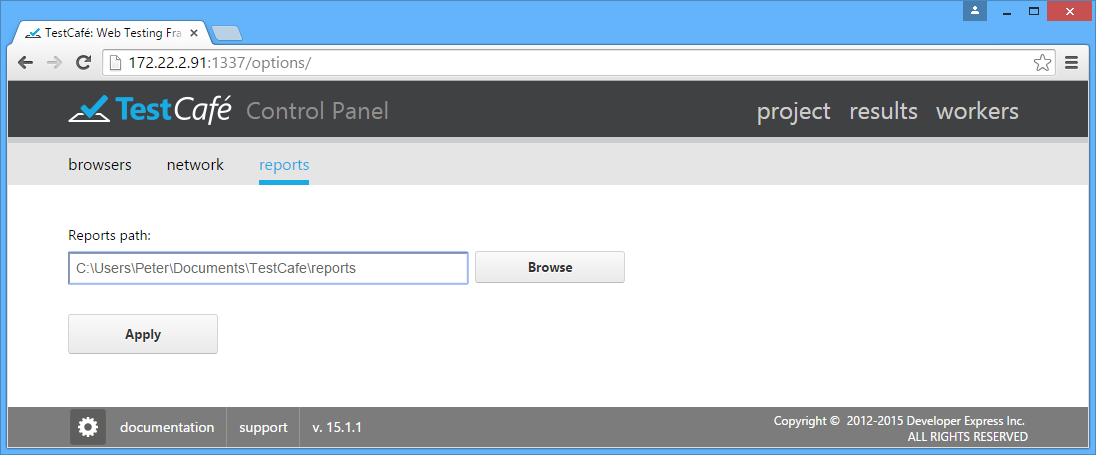
Use the **Find installed browsers** button to automatically populate the list with all browsers available on your system.



Use the **Network** section to modify network settings (hostname, Control Panel Port, Service Port 1 and Service Port 2). If an **auto detect** box is checked, TestCafe automatically determines the respective network setting at every launch. To edit a setting, uncheck its **auto detect** box.



Use the **Reports** section to specify the directory where test run reports are stored.



TestCafe stores its settings within a specific configuration file - config.json. If the configuration file is missing, TestCafe will use settings saved during the first run.

**Important note**

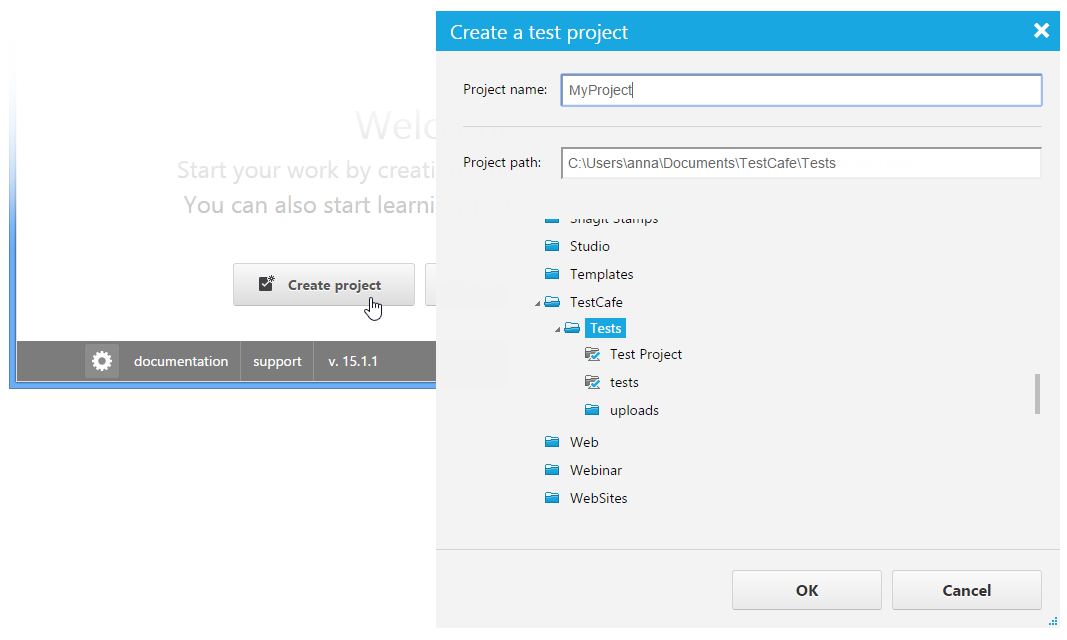
Starting with version 14.1.4, to preserve configuration settings and test file content among TestCafe updates, the default location of the TestCafe configuration file (config.json) and the default location of a directory containing test and fixture files (such as example.test.js) are changed to the following paths.   
  
New locations for Windows:   
\Users\[username]\Documents\TestCafe\config.json   
\Users\[username]\Documents\TestCafe\Tests   
  
New locations for Mac OS:   
/Library/Application Support/TestCafe/[username]/config.json   
/Library/Application Support/TestCafe/[username]/my\_tests   
  
Note that if you have already been using the previous installation of TestCafe, you can use a copy of the existing config.json.

## Projects Tab

### Projects

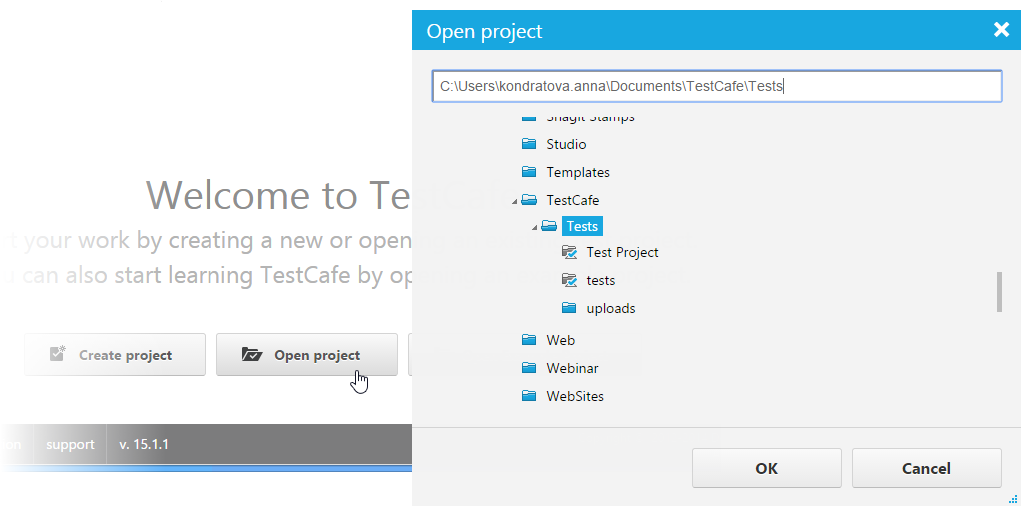
TestCafe's project is a folder that contains a set of individual or organized fixtures and tests located on a specific path, which you can later modify manually by using any JavaScript editor.

Once TestCafe is running, it allows you to create a new project, or open an existing one or example project. To create a new project, click **Create project**to invoke a dialog window in which you can specify the project's name and path.



The newly created project will be opened within the Control Panel without any fixtures and tests. All fixtures and tests you create within that project will be stored at the path specified during project creation. For more information, see [Creating a Fixture](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Fixtures_Creating_a_Fixture) and [Test Recording](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Fixtures_Test_Recording).

To open an existing project, click the **Open Project** button. TestCafe opens a dialog where you specify the project's path and click the **OK** button. All project fixtures and tests will be opened within the Control Panel.



To try sample tests that illustrate TestCafe's API functionality, click the **Open Example Project** button.



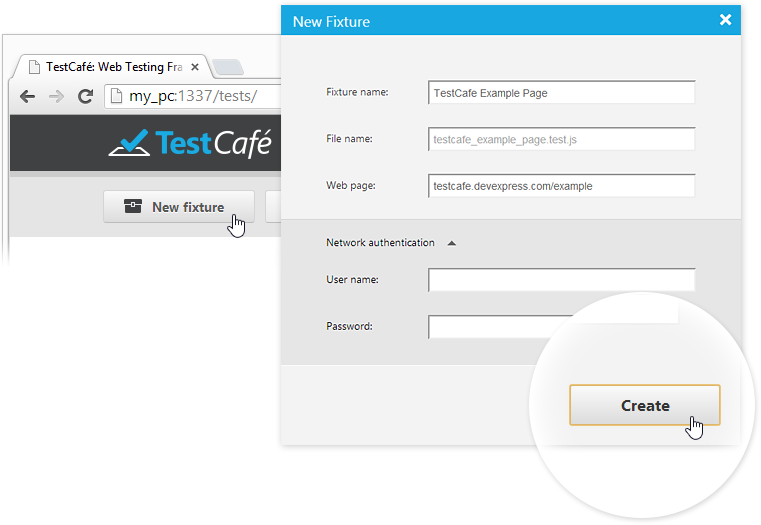
To close the project, use the Close Project button.

### Fixtures

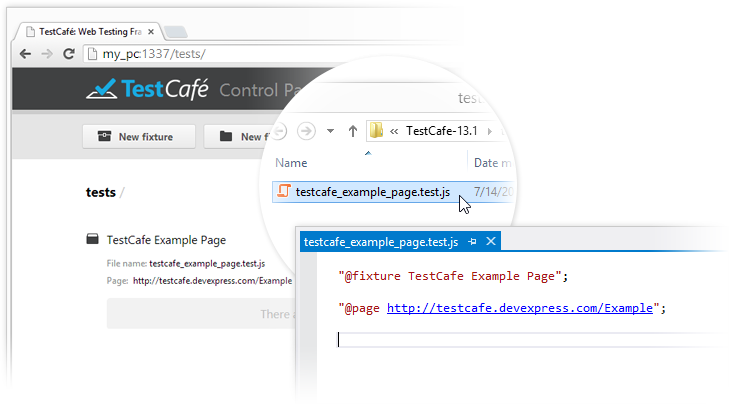
Test fixtures are JavaScript files with \*.test.js extension that contain one or more tests for a single specified target URL. Therefore, you will usually have one test fixture per website page/route.

#### Creating a Fixture

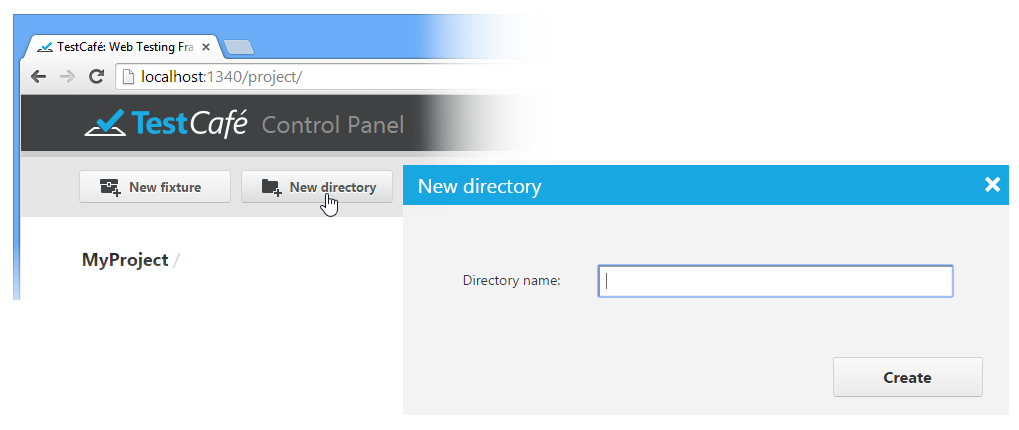
To create a new fixture, click **New Fixture** to invoke a dialog window in which you can specify the fixture name and target web page. If the website you are going to test requires Windows or Basic HTTP Authentication, specify the user credentials within the specific Network Authentication area of the **New Fixture Dialog**. TestCafe will save them to the fixture file within the **[auth](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Common_Concepts/" \l "Http_Authentication)**[directive](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Common_Concepts/" \l "Http_Authentication) (read the [Http Authentication](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/#Http_Authentication) topic to learn how to handle Http Authentication within the Control Panel).



This newly created fixture represents a JavaScript file that stores the settings you have just entered, which is located in the folder that has been specified during the creation of the project (use the [Projects](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Projects) section).



To organize fixtures in the project, you can create one or several directories by using the **New Directory** button and locate fixtures within them.

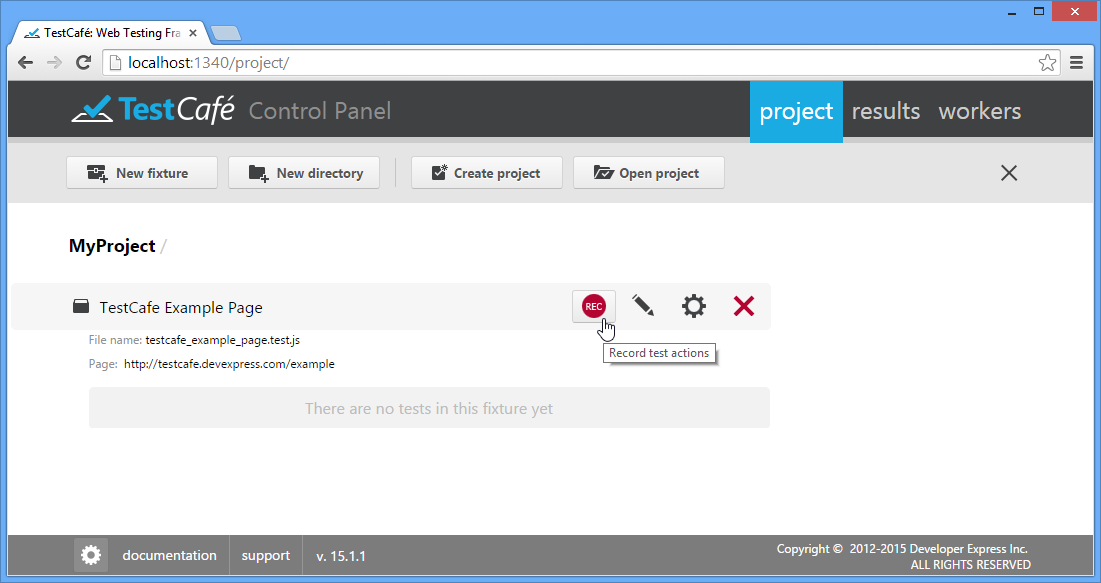


While the TestCafe UI allows you to browse any test hierarchy you created, it doesn't permit you to reorganize your tests. If you need to change the directory structure and move fixtures from one folder to another, use your file explorer of choice. All changes will immediately be reflected in the TestCafe UI.

#### Test Recording

You can create a test either by recording it by using the [visual test recorder](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder) or manually by using TestCafe's [API](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/) and integrated [test code editor](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Fixtures_Test_Code_Editing).

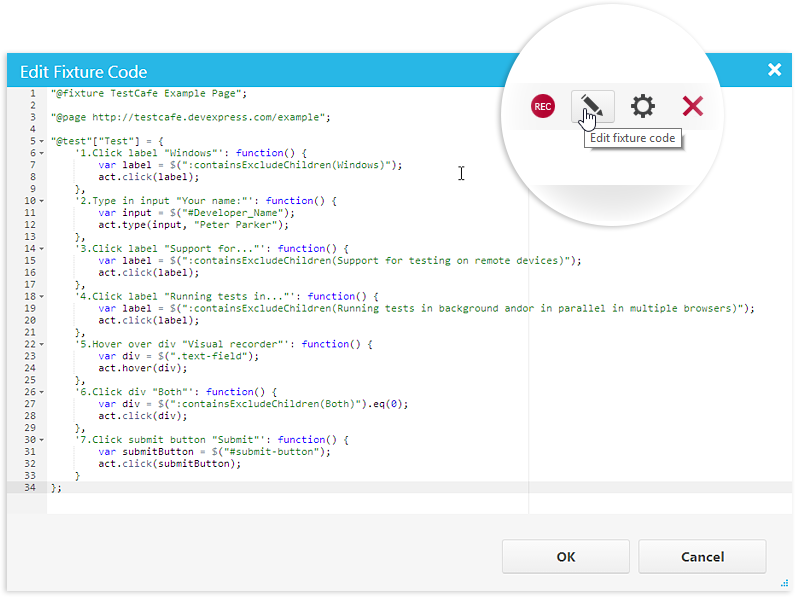
To start the visual test recorder, hover over the fixture in which you wish to create a test, and press **Record Test Actions**.



To learn how to use the test recorder, see [Visual Test Recorder](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/).

#### Test Code Editing

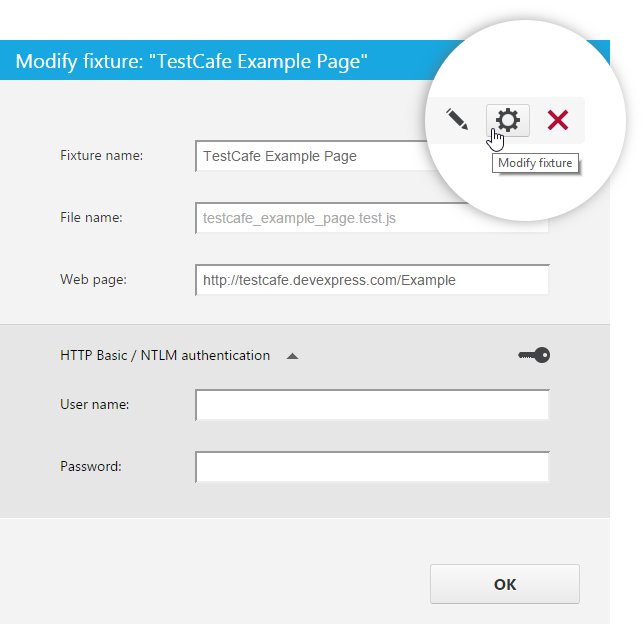
After you have completed the recording, your test will be saved to the fixture. You can review and edit its code by using the built-in code editor that is opened by the Pencil Icon button.



To learn more about creating and editing tests in code, see [Test Fixture API Reference](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API).

#### Modifying Fixtures

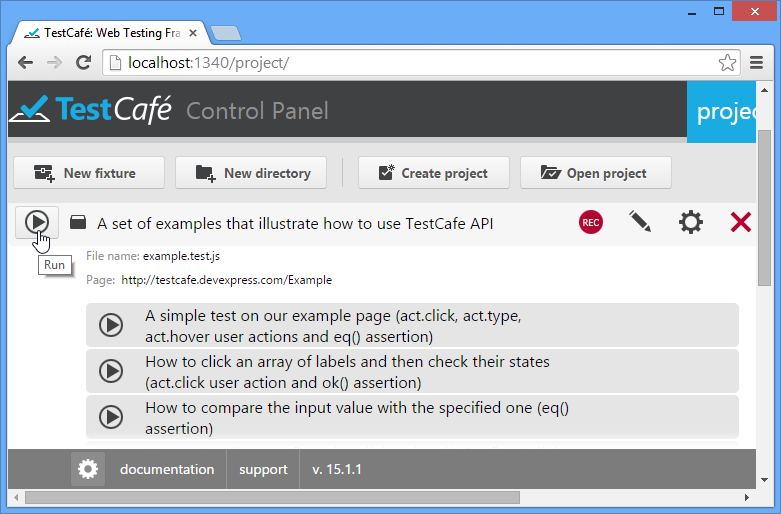
You can change the fixture name, rename a corresponding JS file, specify target webpage URL and authentication credentials by using the Modify Fixture dialog. To invoke this dialog, click the Gear Icon button.



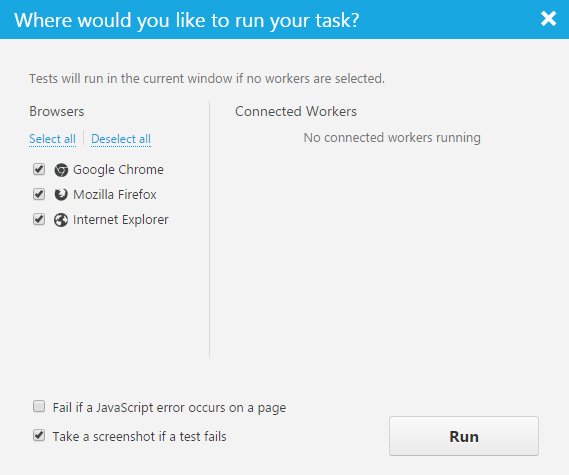
If you do not specify the file name explicitly, it will be automatically generated from the fixture name (and displayed in gray). In case the fixture name contains characters prohibited in file names, they will be replaced or removed from the file name.

#### Running Tests

You can run an individual test or all tests in a fixture from the Projects tab. To do so, click the **Run** button next to the required fixture or test. Note that for fixtures, this button only appears when you hover over the row with the mouse pointer.



This will invoke a dialog that lists all available target browsers and test-running options.



The **Browsers** section on the left hand side will list all browsers specified within the TestCafe Control Panel, as described in the [TestCafe Settings](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/" \l "TestCafe_Settings) topic.

The **Connected workers** section will list browsers on remote computers or mobile devices registered as described in the [Workers Tab](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Workers_Tab).

The **Fail if a JavaScript error occurs on a page** option specifies whether or not a test will fail if a JavaScript error occurs on a tested web page. By default, the option is enabled. Thus, in case of such errors, TestCafe stops the test execution and posts an error message to a report. If you want TestCafe to ignore JavaScript errors, disable the option.

The **Take a screenshot if a test fails** option specifies whether TestCafe will automatically take a screenshot of a tested page if an assertion fails or an error occurs. You can later view the screenshots in a test report on the [Results tab](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Results_Tab).

**Important note**

* The automated screenshot feature is not available on the Linux operating system.
* Screenshots cannot be taken when a test is running in a [remote worker](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel#Using_TestCafe_Control_Panel_Workers_Tab_Connecting_a_Remote_Device).

If you click **Run** without selecting any browser, TestCafe will run tests within the same browser window. If you check any of the listed browsers, TestCafe will automatically run them, execute tests in them, collect the results and close browser windows after completion. Note that tests will be executed in parallel if more than one target browser is selected.

TestCafe emulates cursor movements during tests so that you can view the progress.

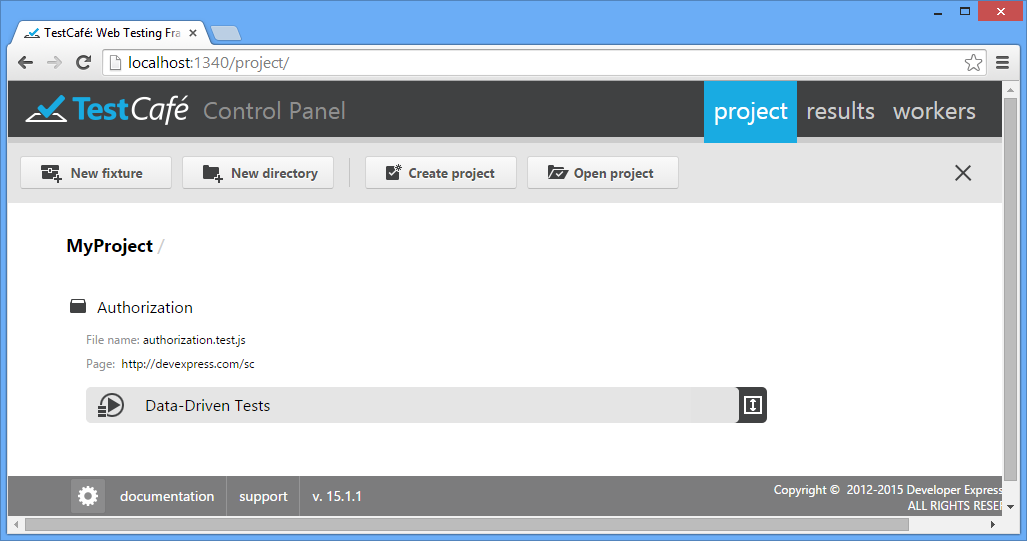
**Important note**

TestCafe always executes tests in an active browser tab. Make sure to keep the active tab open. Do not minimize browser windows. Inactive tabs and minimized browser windows switch to a lower resource consumption mode where tests are not guaranteed to execute correctly.

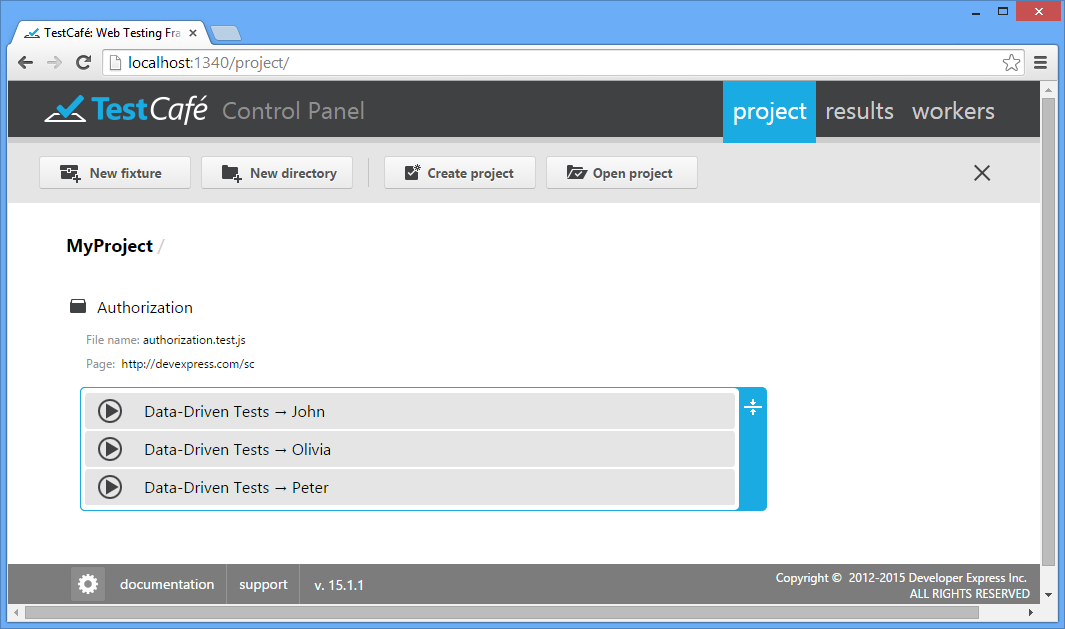
#### Parameterized Tests

[Parameterized testing](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Common_Concepts/#Test_Parameterization) is a way of testing when data sets (input values and expected outputs) are stored separately from the test code. Data sets represent an array of parameters and their values. Each time the test is executed it uses data from each row sequentially. So, a number of test runs corresponds to the array size.

In the Control Panel, data-driven tests represent a group of tests marked by the run group button.

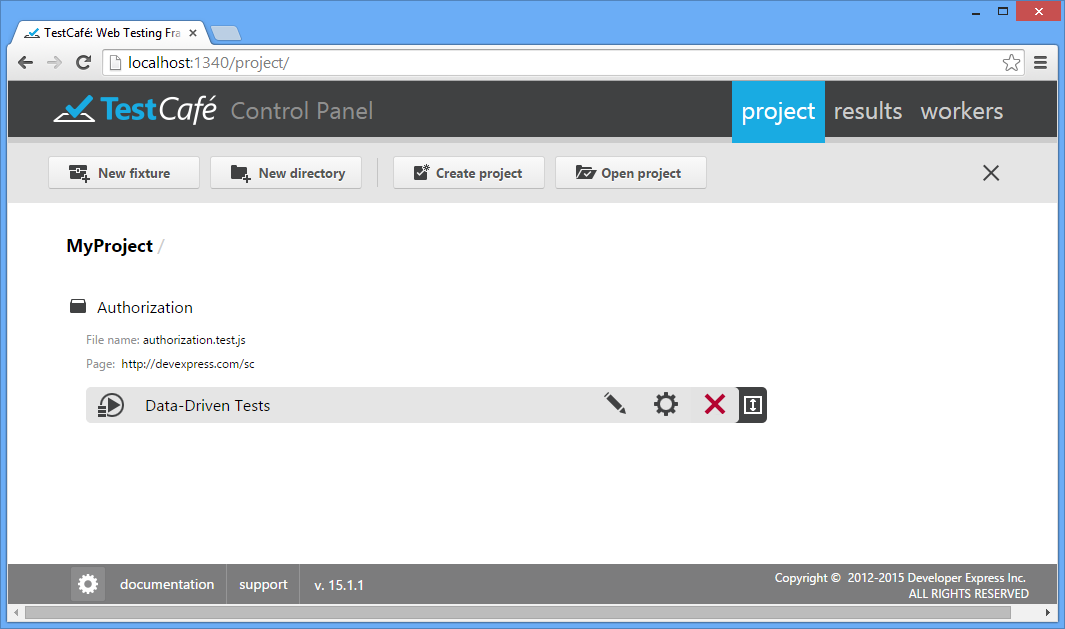


You can use the Open Group Button and Close Group Button buttons to open and close the group correspondingly.



Data-driven tests can be run either as the whole group (via the run group button) or each test individually.

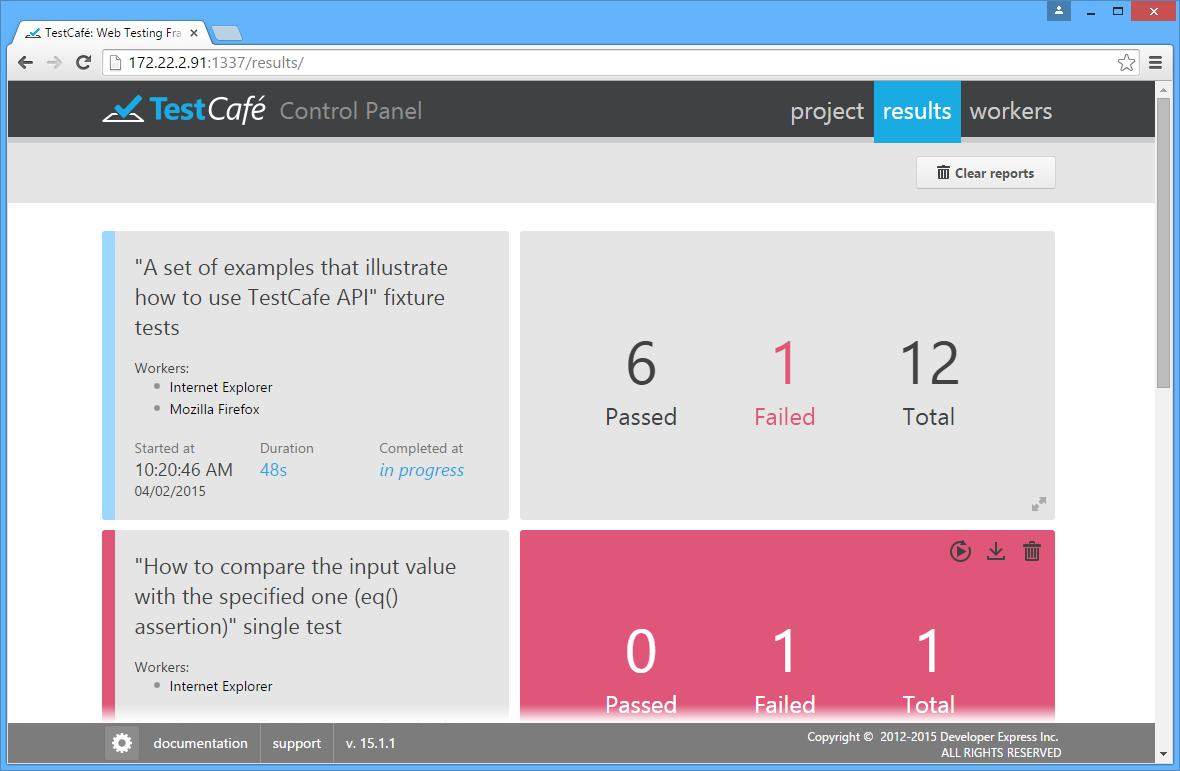
You can also edit the test code, modify its settings or delete the whole group of tests (deleting data-driven tests individually from the group is not permitted).



## Results Tab

Use the Results tab to review and analyze test results. When you switch to this tab, you will see the results of all executed tasks. Red and green colors will indicate failed tasks and successfully completed tasks, respectively. The tab will also show running and pending tasks marked in blue.

Note that a test is only considered processed after it has been executed in all specified browsers, so TestCafe will only update the UI after all selected workers have finished.

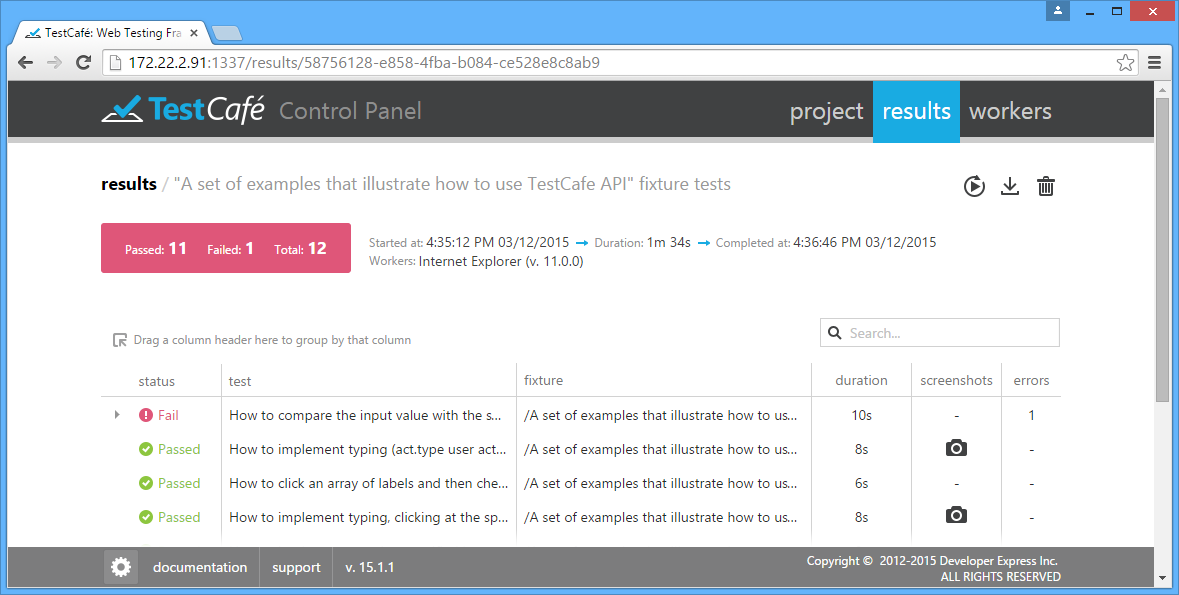


Every entry will display the start and end times of the task, all involved workers/browsers, and the number of tests that have failed and succeeded. You can also see detailed information about the task run (see [below](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Results_Tab_Detailed_View)).

### Detailed View

To see a detailed result report, click an entry in the Results tab. The tab will show the general information about the whole task run as well as detailed information about individual tests.

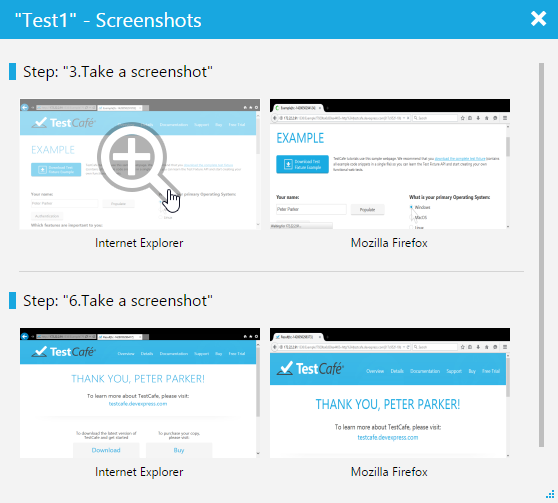
The detailed information is shown in the grid view. Each row in this grid corresponds to an individual test that was executed during the task run.



The grid has the following columns.

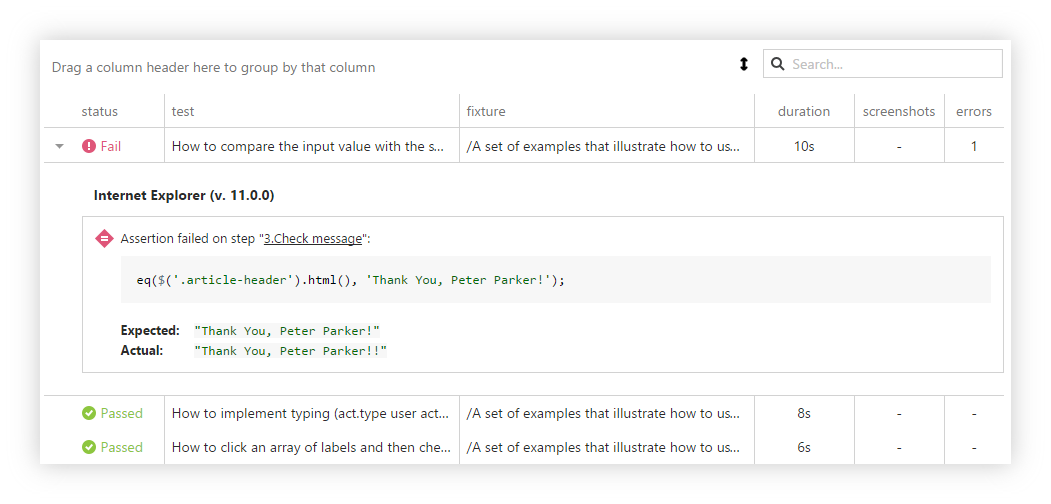
* **status** - The status of the test run: passed or failed.
* **test** - The name of the test.
* **fixture** - The name of the fixture that the test belong to.
* **duration** - The time spent on test execution.
* **screenshots** - If the test has the [act.screenshot](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/User_Actions/" \l "actscreenshot) action, this column refers to a collection of screenshots captured during the test run. To view the captured images, click the View Screenshot icon.
* **errors** - The number of errors occurred during the test run.

When you click the View Screenshot icon in the screenshot column, the screenshot collection is displayed in a new window. The images are grouped by test steps. Each group contains screenshots for all workers engaged in the test.



#### Analyzing Errors

The detailed view allows you to see additional information for failed tests. When you click a failed test, the grid view shows an extended message.

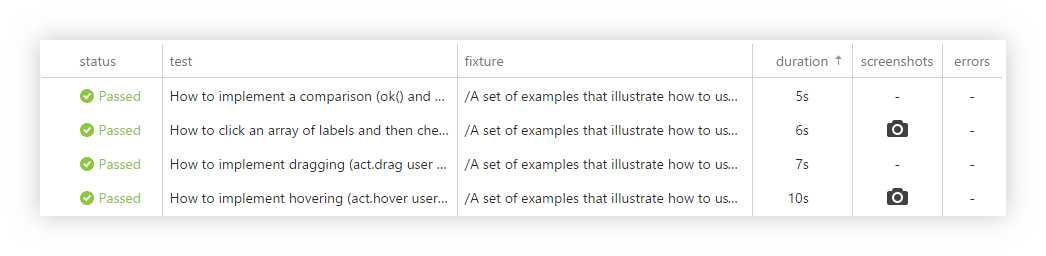


This message typically contains the cause of the fail and the step number where the fail occurs. You can easily switch to this step in the [code editor](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Fixtures_Test_Code_Editing) by clicking the reference in the message.

If the [Take a screenshot if a test fails](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Fixtures_Running_Tests) option was enabled during the test run, you will also see a screenshot of the tested application below the extended message. This image helps you find causes of the error more quickly.

#### Sorting Results

By default, test results are sorted by the status column. You can sort the results by any column by clicking the column header. The sort icon Sort Icon will appear next to the column caption. The image below demonstrates sorting test results by the duration column.



#### Grouping Results

You can group the test results by a specific column. To do this, drag the column header to **Drag a column header here to group by that column**.

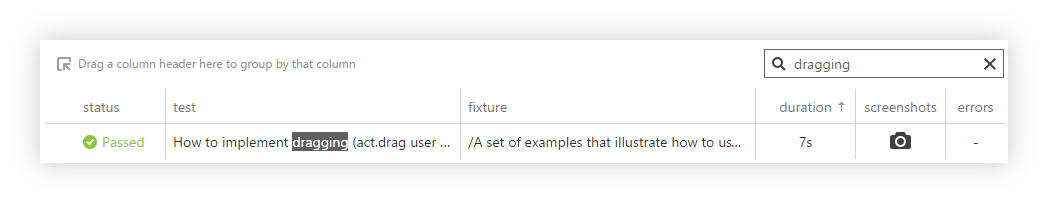


The records with identical column values will be arranged into groups. In the image below, the results are grouped against the Fixture column. To ungroup results, drag the column header back to the grid view.



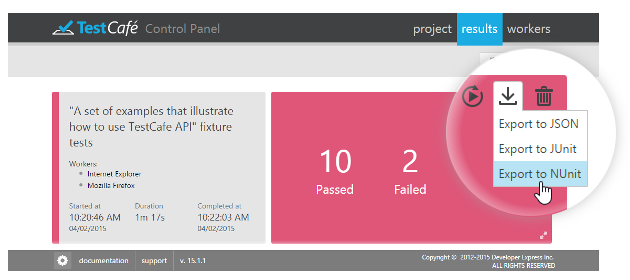
#### Searching Results

You can search through the results by using the search box with the Search-icon icon. When you type a search text into this box, the grid’s content is filtered according to what you have typed. For example, if you type “dragging”, the Result tab will show only the test results that contain that word. This makes finding report entries faster.

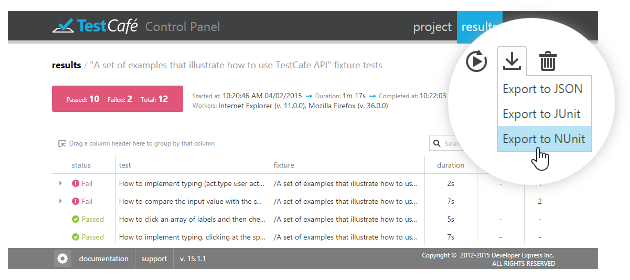


### Exporting Reports

You can export test results to JSON, JUnit and NUnit formats. To do this, click the Export Report button in the top right corner of the test report and select the desired format. The report file will be saved to your web browser's default download location.



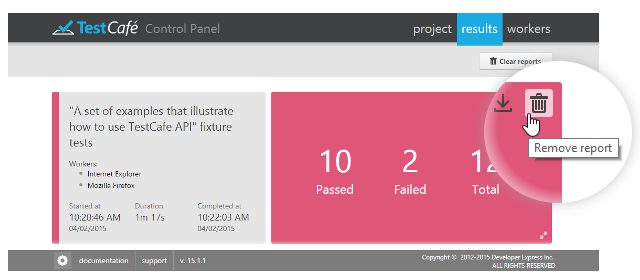
The Export Format button is also available in the [detailed view](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Results_Tab_Detailed_View) of the Results tab.



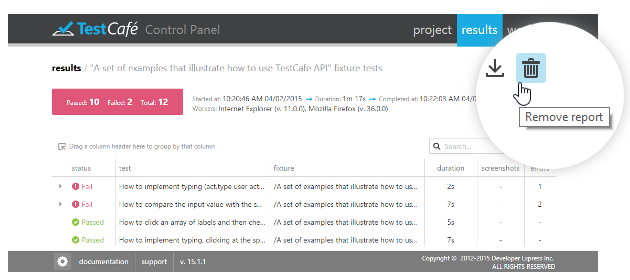
### Removing Reports

By default, TestCafe stores reports for all executed tasks and displays them in the Results tab. You can delete unnecessary reports when you need.

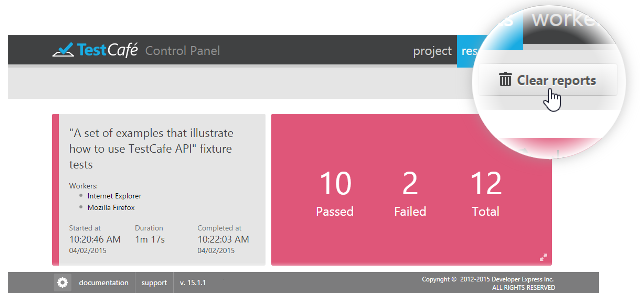
To delete a certain report, click the Remove report button in the top right corner of the test report. TestCafe will ask for confirmation to delete a report. If you select Yes, TestCafe will remove the report from the Results tab and completely delete the report file from your computer.



The Remove report button is also available in the [detailed view](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Results_Tab_Detailed_View) of the Results tab.

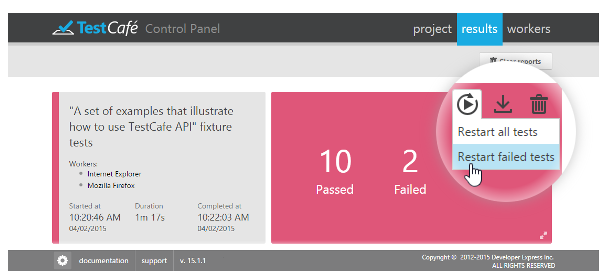


To remove all reports, click the Clear reports **Clear reports** button in the top right corner of the Results tab. TestCafe will permanently delete all reports from the Results tab and your computer.

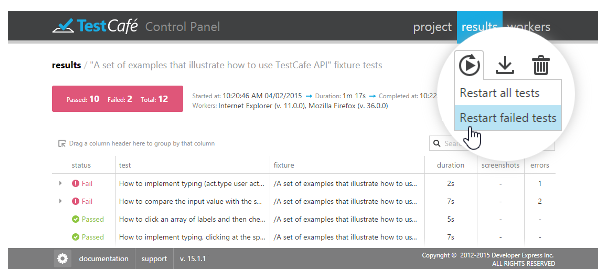


### Restarting Tests

You can restart a test run right from the **Results** tab. To do this, click the Restart Task button and select Restart all tests or Restart failed tests.



The Restart Task button is also available in the [detailed view](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Results_Tab_Detailed_View) of the Results tab.



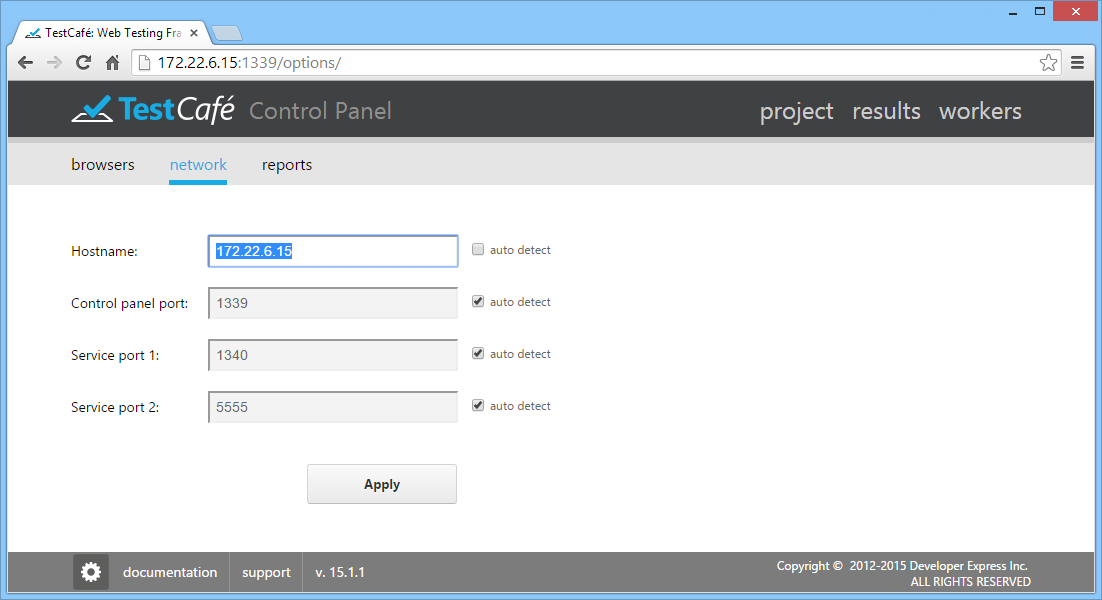
## Workers Tab

The Workers tab lists local and remote browser windows (workers) connected to the current TestCafe instance. For workers busy running tests, the execution progress is displayed.

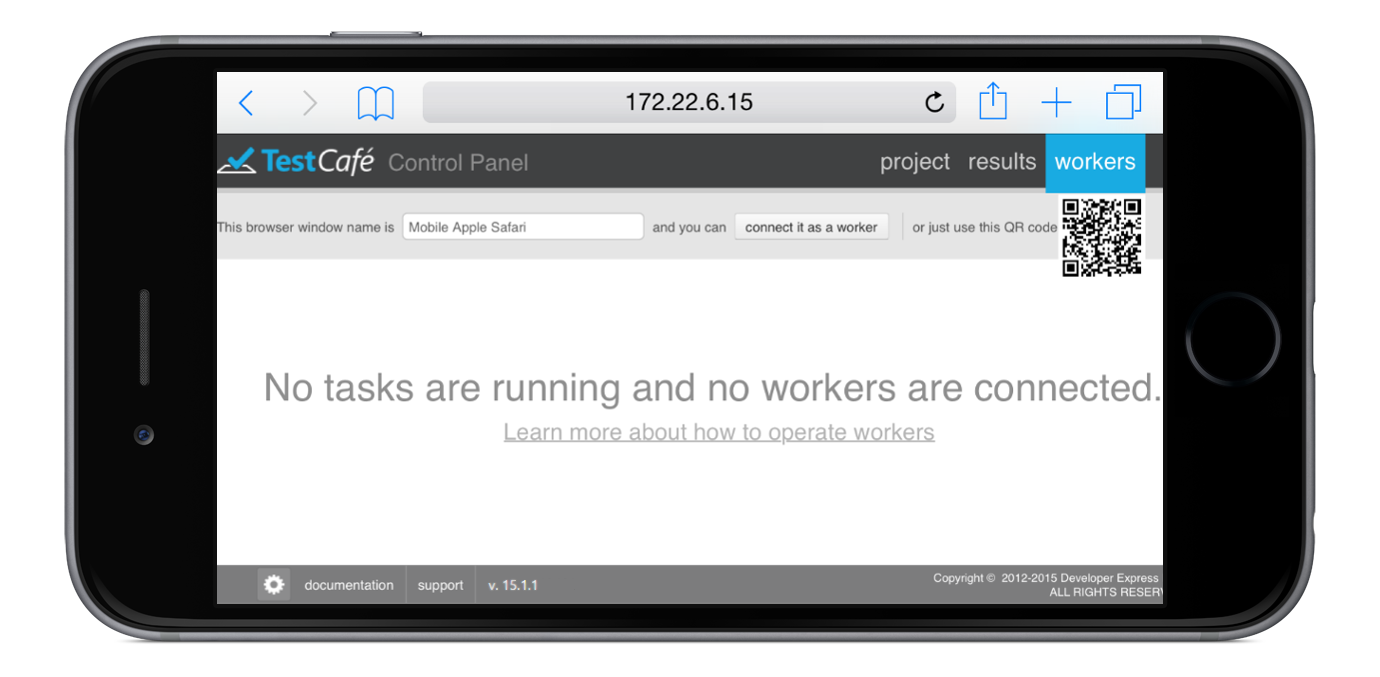
When opened on a remote computer or mobile device, the Workers tab allows you to connect this device as a remote worker to run tests on it.

### Connecting a Remote Device

Before you turn to your remote computer, tablet or smartphone, make sure that the machine that is running TestCafe has the **hostname** set to a value that allows remote access. You can check the currently specified value by looking at the TestCafe Control Panel URL. If it needs to be changed, refer to the [TestCafe Settings](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/" \l "TestCafe_Settings) topic.



Now open a browser on your remote device and navigate to the URL where the Workers tab is located.



On this page, click **connect it as a worker**. The page will now indicate that this browser has been registered and is waiting for a task.

**Important note!**

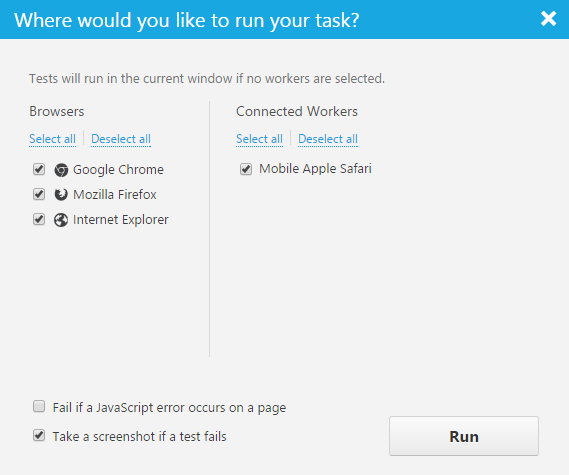
You can specify a Worker name in the input field after the phrase "This browser window name is". This name can be obtained from the test code by using the **this.\_\_workerName** property (refer to the [Test Run Metadata](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Common_Concepts#Test_Run_Metadata) topic).



At the same time, the Workers tab on the machine running TestCafe will display a newly registered target browser.

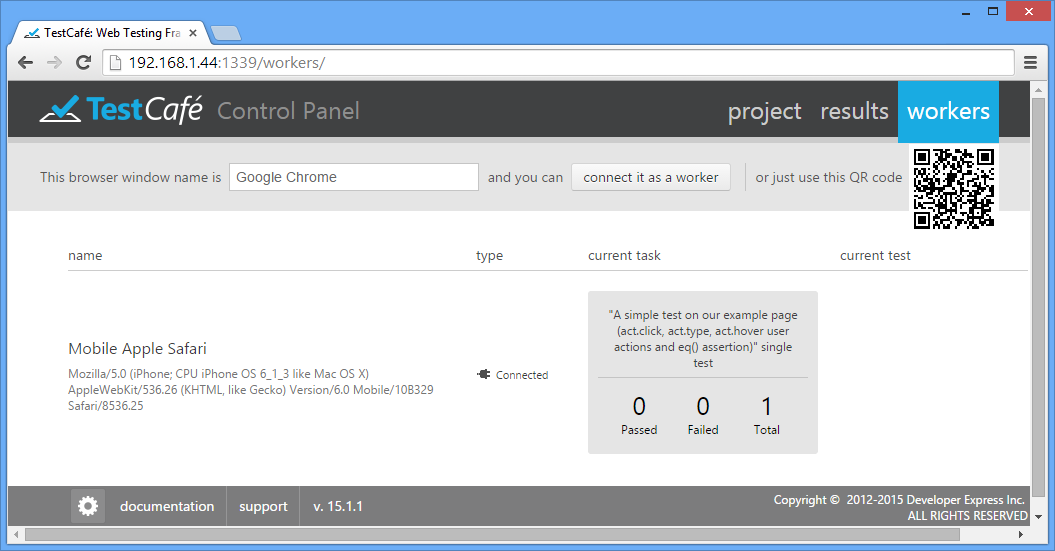


You can switch back to the Project tab, select a fixture or an individual test and press the **Run** button next to it as described in the previous section. The browser chooser dialog will now display your remote worker, so you can check it together with all other browsers and click **Run**.



### View Connected Workers

When tests are running, the Workers tab displays the test execution progress. For all workers (local and connected remotely), you will see the total number of tests, and how many have succeeded or failed so far.



All results will automatically be collected and presented in the [Results tab](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Results_Tab).

VISUAL TEST RECORDER

**Getting Started**

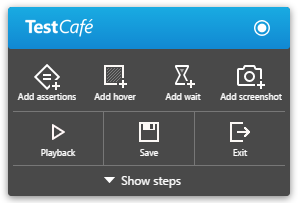
Once you clicked the Rec button, the Visual Test Recorder navigates you to the target website and logs all of your actions - from key presses to drag-and-drop operations. When the webpage is loaded for recording, the top left corner of the window contains the [toolbar](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/#Toolbar) and a [list of recorded steps](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/#Recorded_Steps), which allow you to quickly add user actions, save and playback the test, navigate through the recorded steps and modify them if needed.

After recording the appropriate action sequence, you can update your test in code (use the [Edit Test Code](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Fixtures_Test_Code_Editing) topic) in order to add assertions or to modify your test if you don't want to record it again from scratch.

**Toolbar**

TestCafe's toolbar allows you to add user actions to a test, save and playback the test, and review recorded steps. If the TestCafe toolbar obscures important website elements, you can freely drag it to any position within the browser window.

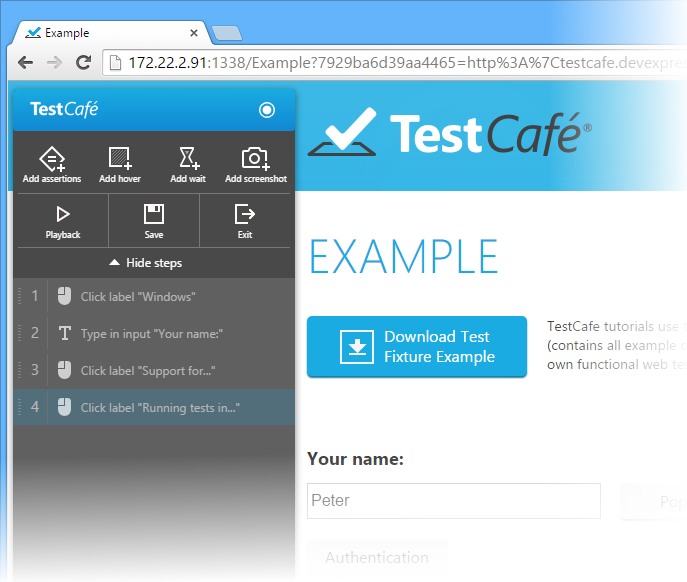
Each recorded user action is automatically added to the list of recorded steps. To make the toolbar more compact, you can hide or show [a list of recorded steps](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/#Recorded_Steps).



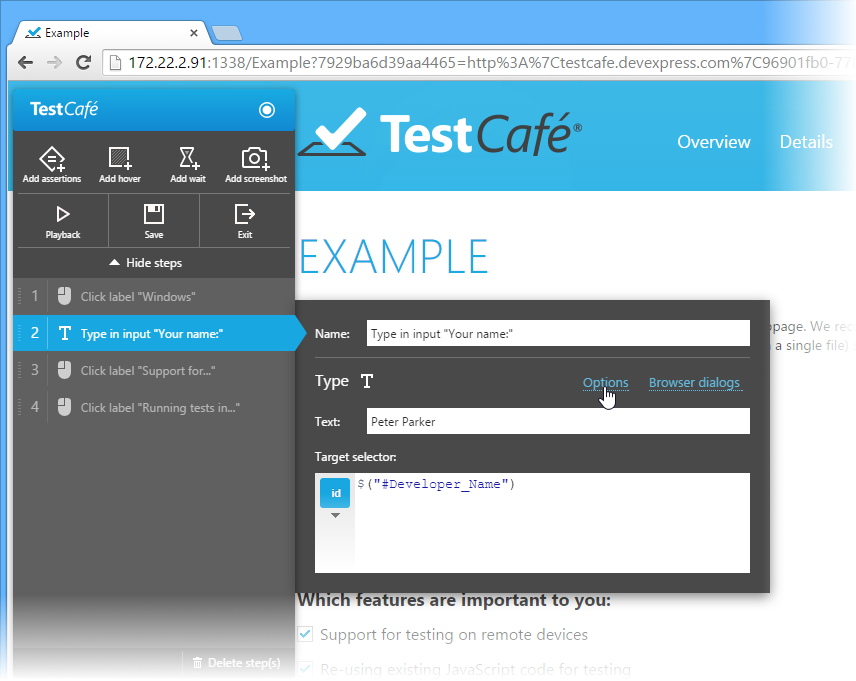
**Recorded Steps**

Each time a step has been recorded, it appears within the list of recorded steps and gets the selection. At this time, you can modify its options, swap it with other steps or delete this step if it was incorrectly recorded.

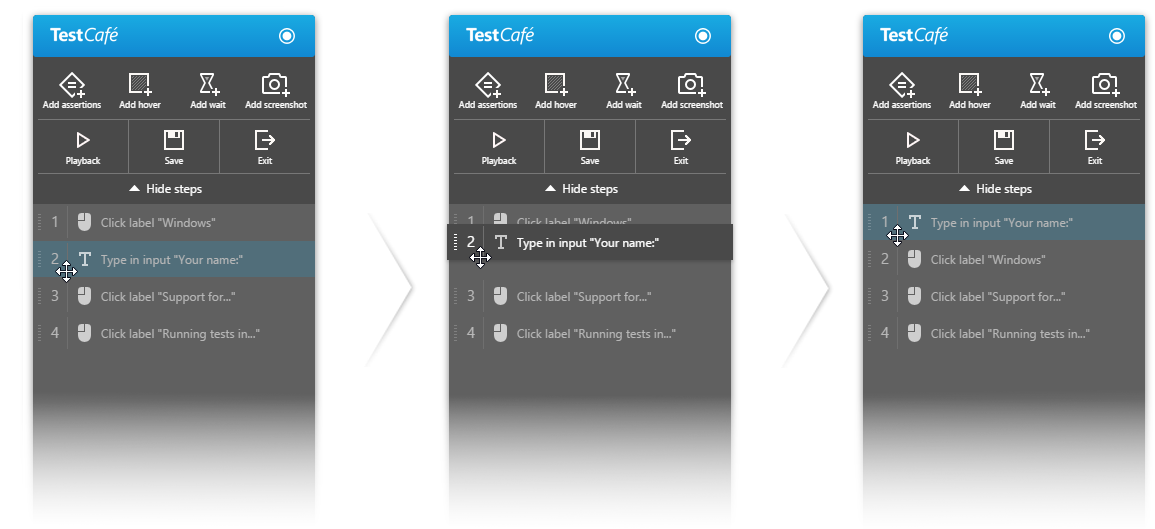
Each step in the list has an ordinal number, specific icon (that corresponds with the type of user action) and an automatically generated step name.



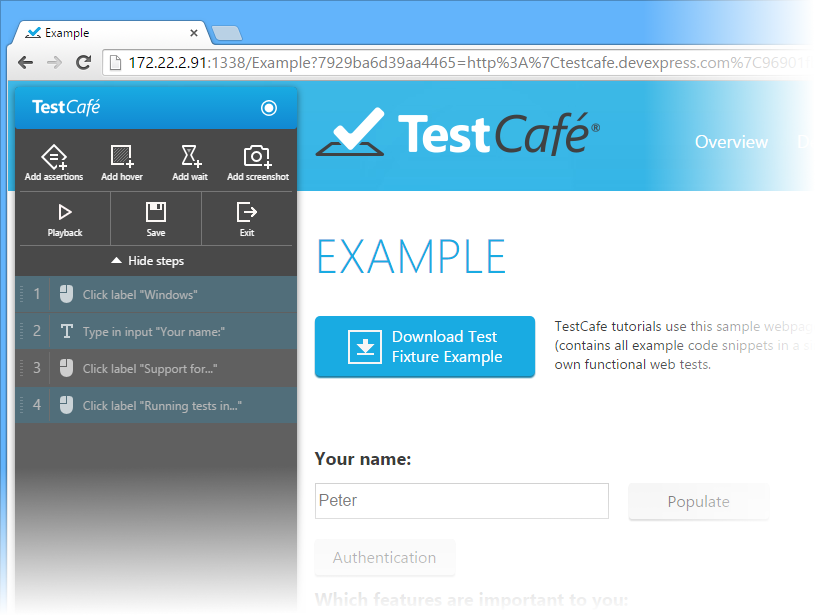
You can review and modify each step at any time while a test is recording. Click the step to open a dialog window, which allows you to modify the step options. Depending on the action type, you are allowed to modify the step name, element selector expression and other step parameters. Click **Options**to modify additional options like caret position, offsets and others. Use **Browser dialogs** to handle browser's native dialogs. The changes are saved automatically once you exit the dialog.



To swap steps within the list, drag the step and then drop it in the required position. TestCafe will automatically update ordinal numbers of steps within the list.



You can select one or several steps in the list at a time. A single mouse click selects a step and opens a dialog window. To select a step without opening a dialog window, use **Ctrl+Click**. While the **Ctrl** key is pressed, you can add one or several steps to the selection. To select a consecutive group of steps, click the first step, press and hold down the **Shift** key, and then click the last step.



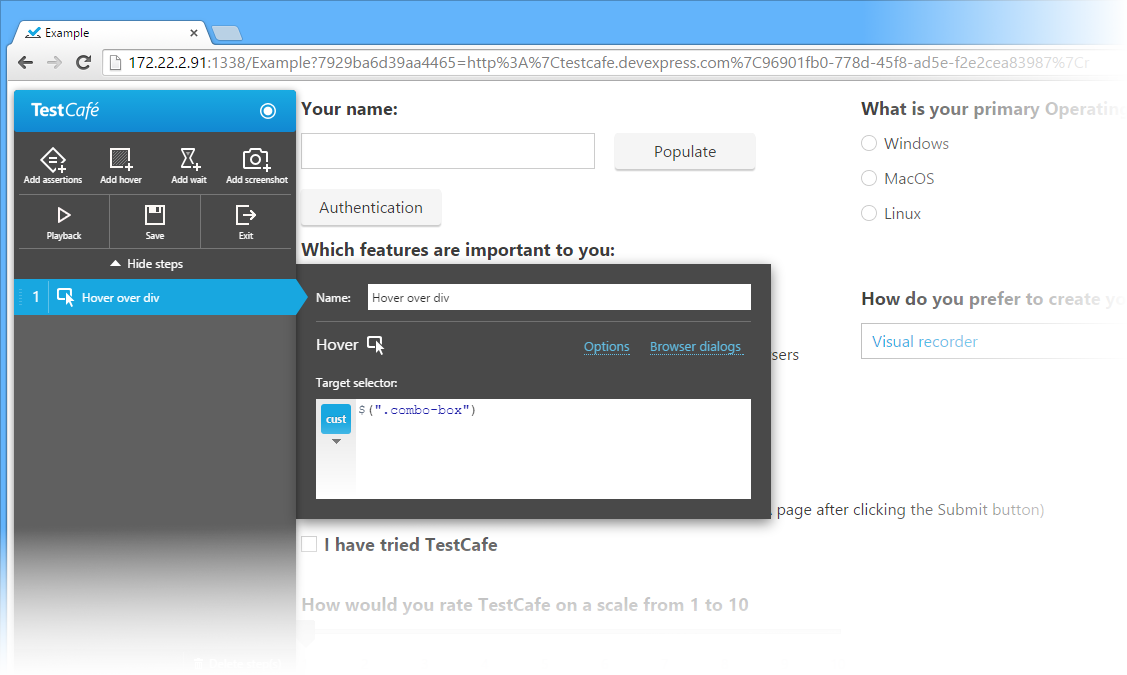
TestCafe allows you to delete individual steps or a group of steps by using the **Delete Step** button. To delete steps, select them as shown above and click the **Delete Step** button. TestCafe will automatically update step ordinal numbers.



**Hover Action**

To record the hover action, click the hover action button within the toolbar. Then, hover the mouse over the required element on the page and click it. Available elements become outlined with a blue line when hovered. You can click the **Esc** key to cancel the hover action.

As a result, the hover action will be added to [the list of recorded steps](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/#Recorded_Steps).

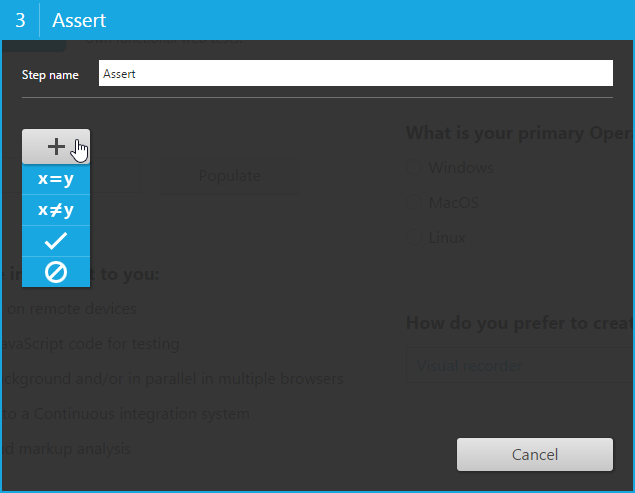


**Tip**

See also the [act.hover](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/User_Actions" \l "acthover) user action.

**Assertions**

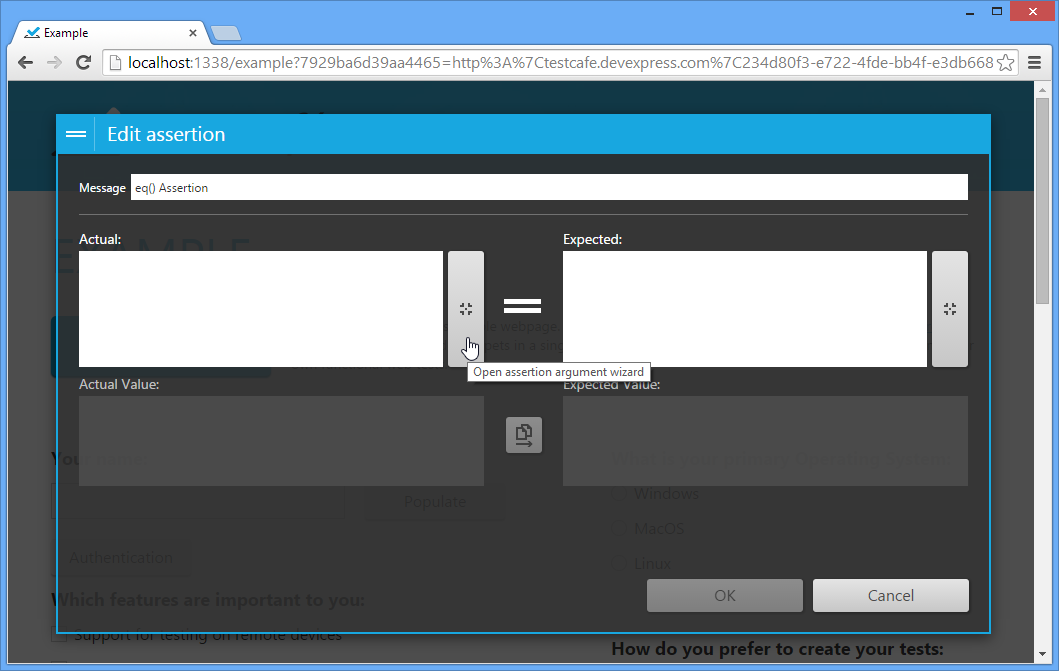
To record a step that contains one or more assertions, click the assertions button. Within the opened dialog, specify the step name and select the required type of assertion. TestCafe allows you to add assertions that will check whether an arbitrary expression resolves to true or false, and whether two objects are the same or different (use the [Assertions](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions) section).



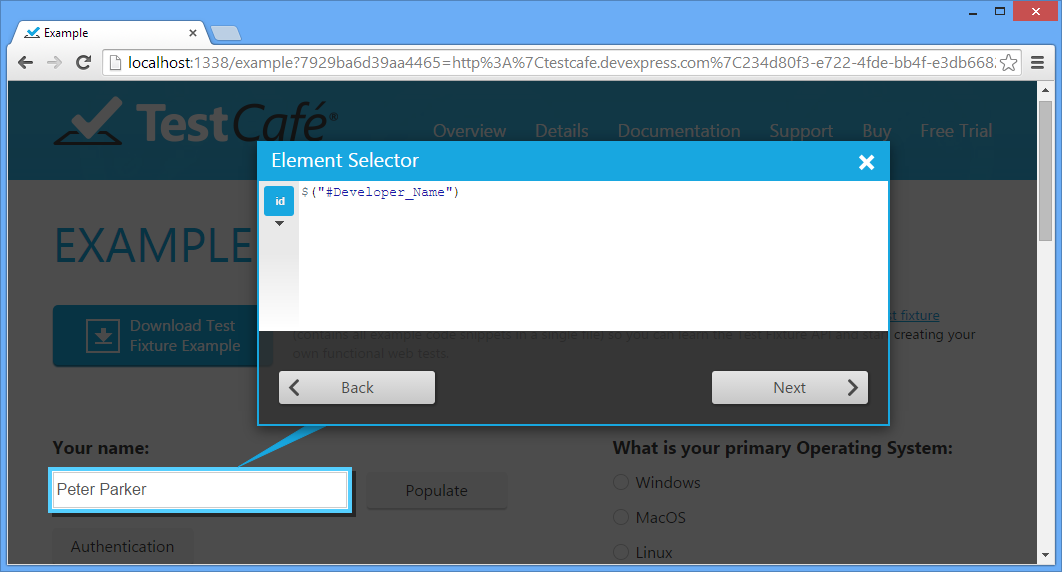
A text box at the top of an opened dialog window allows you to enter an assertion name that will later be reflected in the list of the step's assertions or shown within the TestCafe's Control Panel if a test will fail on a step that contains that assertion. Depending on the assertion type, you will be asked to specify the **Actual** and **Expected** expressions for the [eq](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions" \l "eq) and [notEq](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions" \l "notEq) assertions, or to fill up the **Expression** field for the [ok](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions#ok) and [notOk](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions" \l "notOk) assertions respectively. Values of expressions are displayed within read-only fields that are located below expression fields.



Click an expression field to open the editor that allows you to modify the expression manually. Alternatively, you can use the specific integrated wizard, which is opened using the button located on the right edge of each expression field. The wizard allows you to configure the expression visually step by step.

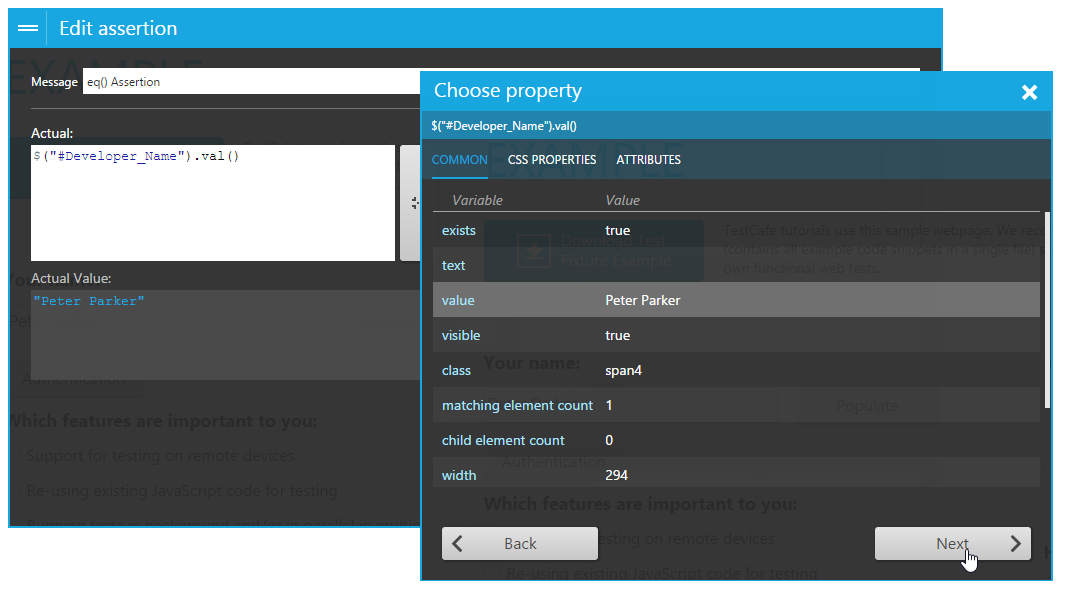


First, select an element on the web page and click it. Available elements become outlined with a blue line when hovered. Once you have clicked the required element, TestCafe opens a dialog window where you can modify the [element selector expression](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/#Selector_Expressions) for the element you just clicked. To continue, click **Next**.

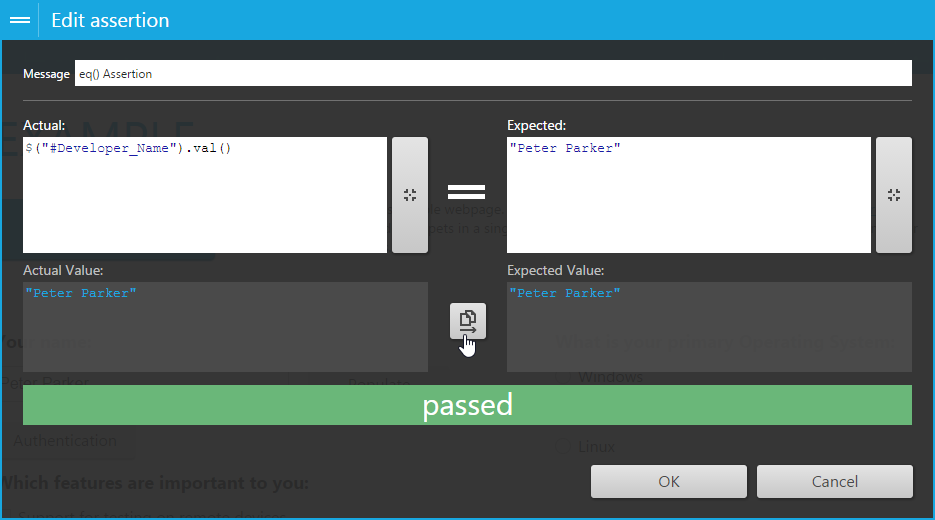


The opened Choose property dialog provides three tabs to choose common properties, CSS properties, or element attributes. Names of properties and attributes are displayed in the left column, while the right column shows their values. Select the required property or attribute, whose selector expression and value will be later shown within an expression and expression value fields correspondingly. Click **Finish** to close the wizard. If there is no appropriate property within the **Choose Property** dialog window, uncheck the selected item and click the **Finish** button to add the element's jQuery selector expression that you can later modify manually within the integrated expression editor.

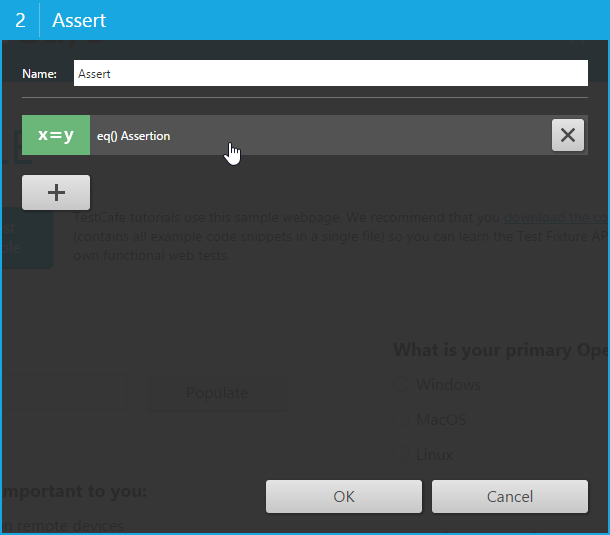
You can see that an expression field now contains the element selector expression. The blackout field below illustrates the expression value.



Note that for the [eq](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions" \l "eq) and [notEq](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions" \l "notEq) assertions, there is a specific button at the right edge of the actual expression value field. It allows you to copy the field's value to the corresponding field for the expected expression value.



When all required fields are specified, TestCafe automatically shows whether the assertion has passed or failed. Click the **OK** button to save the assertion. You can see that created assertion has been added to the list of assertions. Repeat the actions mentioned above to add one or more assertions to the step.



Click the **OK** button to close the assertions dialog and save the changes to the step.

**Tip**

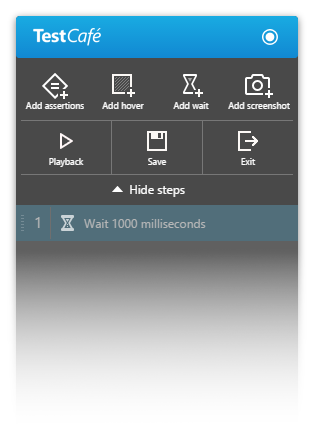
See also the [Assertions](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Assertions/) API section

**Wait Action**

To add a wait action, click the wait button button within the toolbar. The opened dialog allows you to specify the step name and the pause interval in milliseconds.



Click **Add Action** to save the step to [the list of recorded steps](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Visual_Test_Recorder/#Recorded_Steps).



**Tip**

See also the [act.wait](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/User_Actions" \l "actwait) and [act.waitFor](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/User_Actions" \l "actwaitFor) user actions.

**Automated Screenshots**

TestCafe can take a screenshot of the tested page at any moment during test run. Use the Screenshot action to record a step at which TestCafe captures the webpage. The resulting screenshot is stored in the test report, which you can view in the [Results tab](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Results_Tab).

To add the Screenshot action, use the Add Screenshot button.

**Important notes**

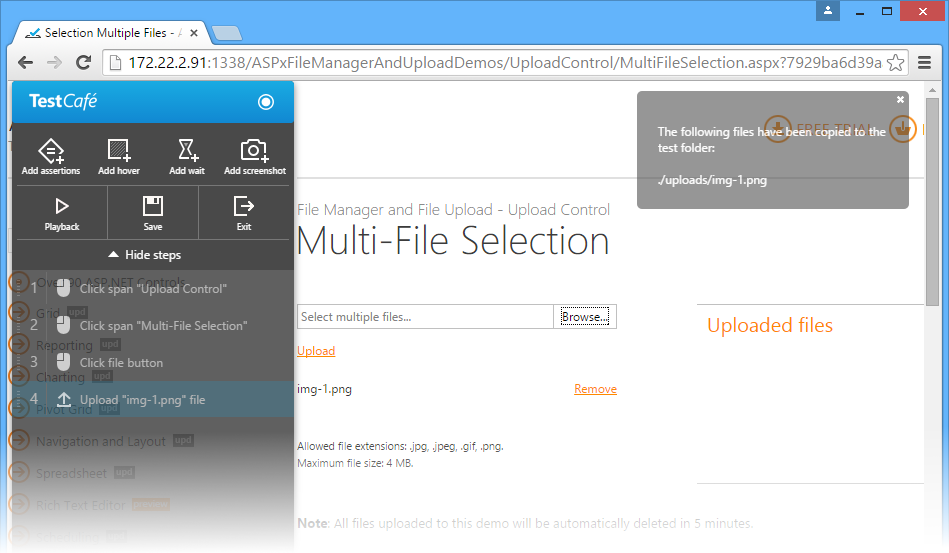
* The automated screenshot feature is not available on the Linux operating system.
* Screenshots cannot be taken when a test is running in a [remote worker](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel#Using_TestCafe_Control_Panel_Workers_Tab_Connecting_a_Remote_Device).

**Tip**

See also the [act.screenshot](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/User_Actions" \l "actscreenshot) user action.

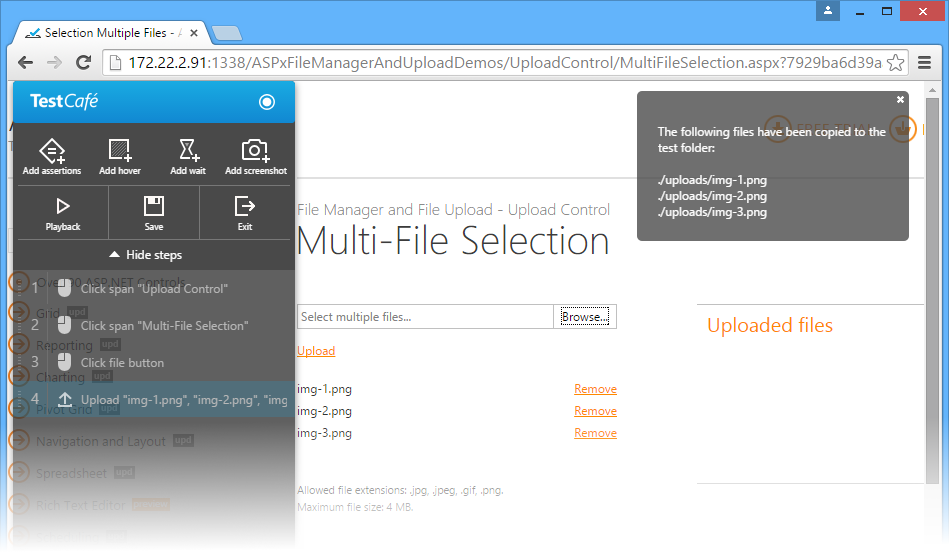
**File Uploading**

When you upload a file during the recording of a test, TestCafe automatically copies the file to the Upload folder that is located within the folder that contains the test file. The popup window in the top right corner of the screen informs you that the uploaded file has been successfully copied to a specific path, or that an error has occurred during file uploading (access is denied, local disk is out of space, etc.). If a file with the same name already exists, it will be rewritten. Successfully uploaded files will then be used during the test run.



Note that you are not allowed to modify the name and the path of the uploaded file during the recording of a test. You can modify them later manually using code.

In case the tested webpage allows you to implement multi-file selection during uploading, TestCafe will record this action as a single step showing relative paths to all uploaded files within the popup window.



**Tip**

See also the [act.upload](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/User_Actions" \l "actupload) user action.

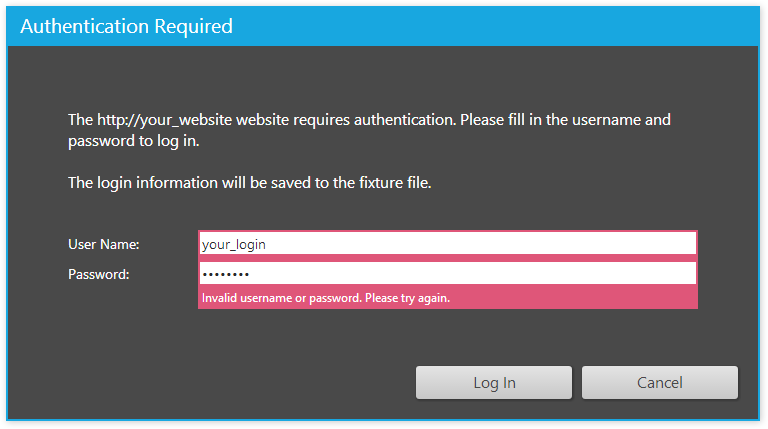
**Http Authentication**

If the page requests Windows or Basic Http authentication, TestCafe intercepts this action and opens its own authentication form where you can enter user credentials. If the entered login and password are correct, TestCafe saves them to the fixture (\*test.js) file within the **@auth** directive (see the [Http Authentication](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Common_Concepts#Http_Authentication) topic).

Note that in case of Windows authentication, TestCafe additionally requires domain and workstation (PC) names. By default, these names are automatically received from the machine where TestCafe is installed.



If the entered credentials are invalid, TestCafe will ask you to re-enter the login and password.



When the credentials are saved, the visual recorder continues the recording.

If the authentication is cancelled (using the Cancel button), TestCafe provides the following scenarios:

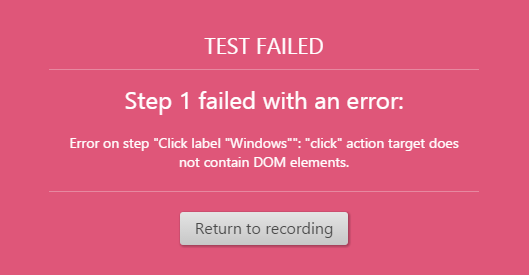
* If the authentication was invoked during the first step recording, TestCafe will return you to the Control Panel.
* If the authentication was invoked during the second step recording, TestCafe will delete the last recorded step and restart recording.
* If several steps were recorded before the authentication, TestCafe will show a corresponding message, delete the last recorded step, play back the steps and return you to the recording.
* If TestCafe detects incorrect test steps before the authentication, it will delete the last recorded step and redirect the recording to the initial test page, so that you can correct errors and replay the test.

**Playback**

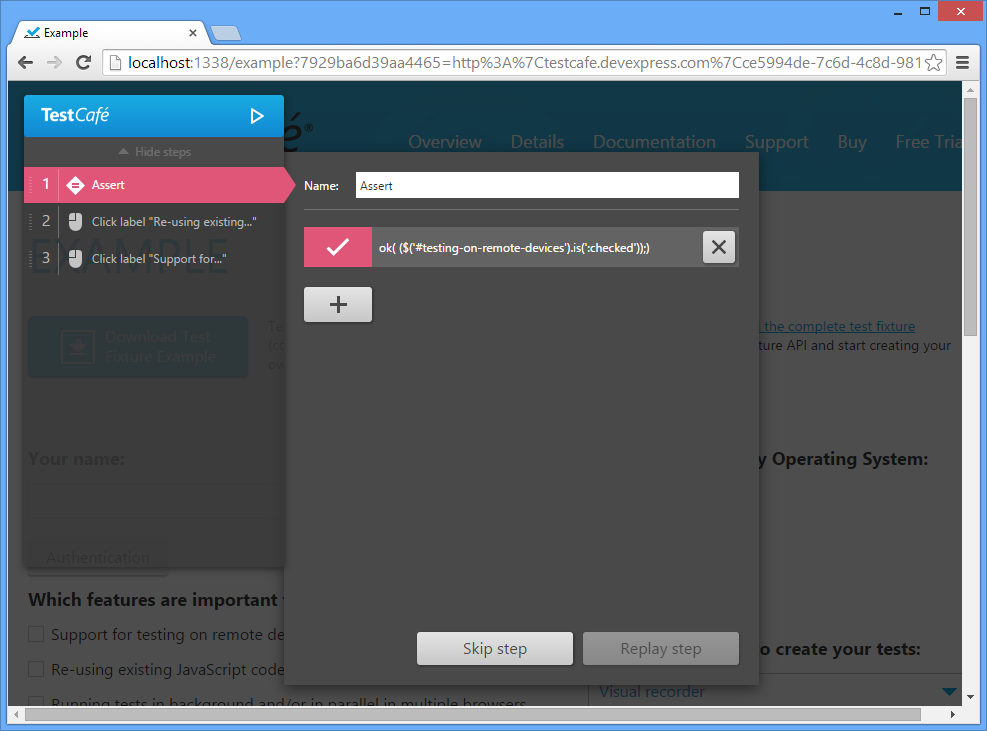
During a [test recording](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/#Using_TestCafe_Control_Panel_Projects_Tab_Fixtures_Test_Recording), you can play back a test by clicking the play back the test button.

Once you click the button, TestCafe will play back all recorded steps in the order they were recorded for a specific web page within the same browser window.

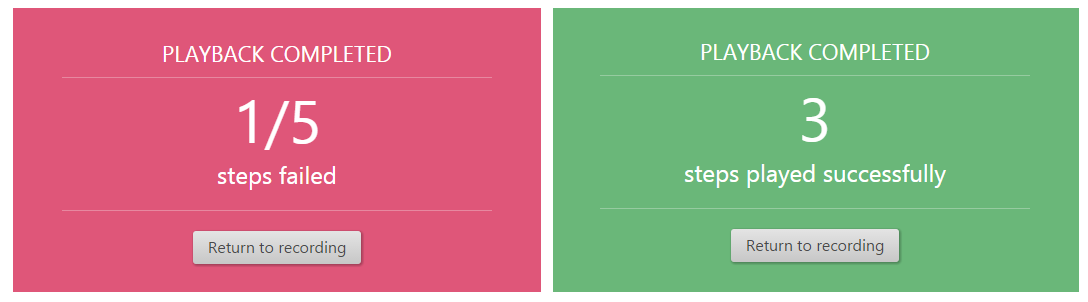
If an error occurs within a test step, TestCafe considers this step to be critical, stops playback and opens a popup window that shows information about the error. To close the window, click **Return to recording**. TestCafe will mark the failed step with red and open the step's options dialog where you can fix the error.



If a non-critical error (for example, errors within assertions) has occurred at any step, TestCafe will stop the playback at the failed step, select the failed step in the list of recorded steps and show the dialog with the supposed error. Note that you can edit the step right within the dialog window and click the **Replay Step** button to playback this step once again. Click **Skip step** to continue the playback from the next step. If you skip the failed step without fixing it, then this step will be marked with red until you fix it or restart the playback.

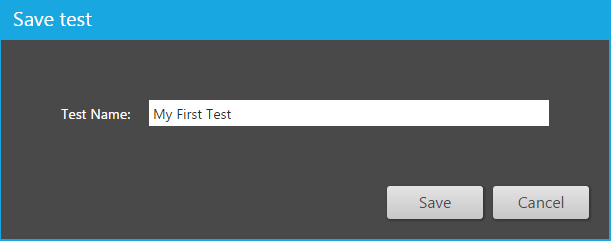


At the end, TestCafe will display a popup window illustrating how many steps failed and how many succeeded. Click **Return to recording** to continue.



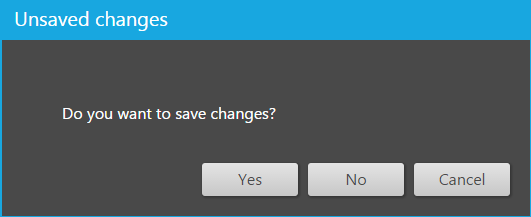
**Test Saving**

To save a test without leaving the recorder, click the save the test button. Within the opened dialog window, TestCafe will prompt you for a test name and then return to recording. Note that the save the test button is not available if there are no unsaved changes in the test.

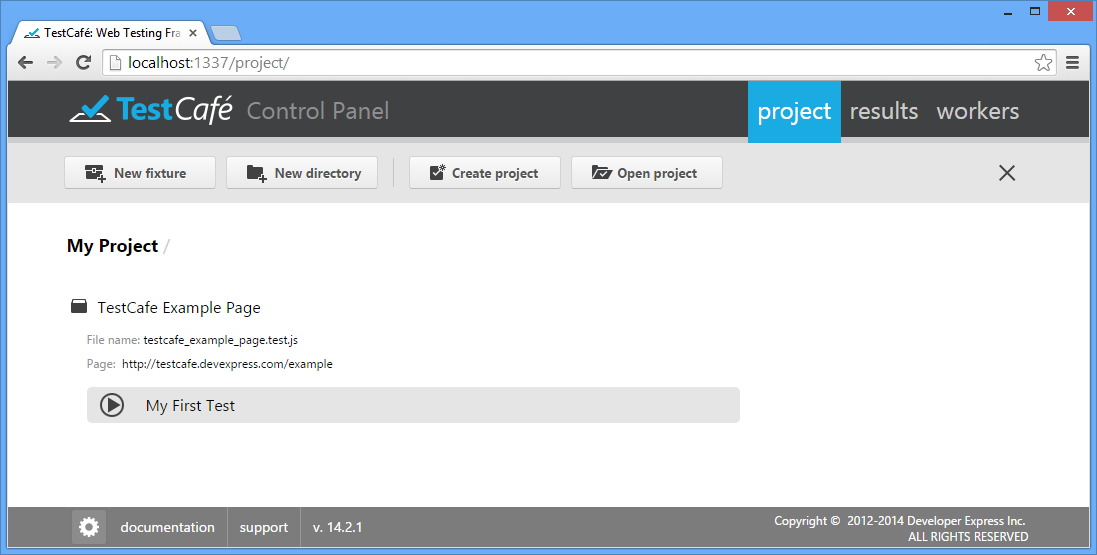


If you are going to stop test recording and then return to the Control Panel, click the exit button button. If the test has been saved, TestCafe will return you to the Control Panel without showing that dialog window.

If the test wasn't saved, TestCafe will inform you that there are unsaved changes in the test and will then open a dialog window. Click **Yes** to save the test. If you have already named the test previously using the **Save** toolbar button, TestCafe will save the changes to the same test file. If the test is being saved for the first time, it will ask you to enter the test name. The **No** button allows you to leave the recorder without saving. The **Cancel** button returns you to the test recording.

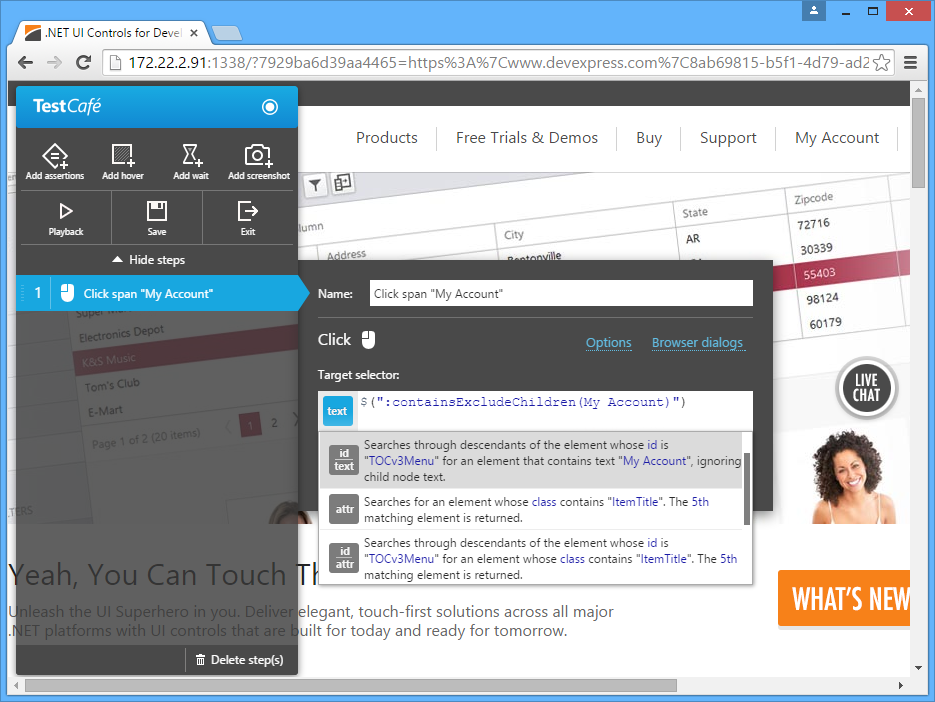


Once you save the test, it will appear in the Control Panel within the fixture.

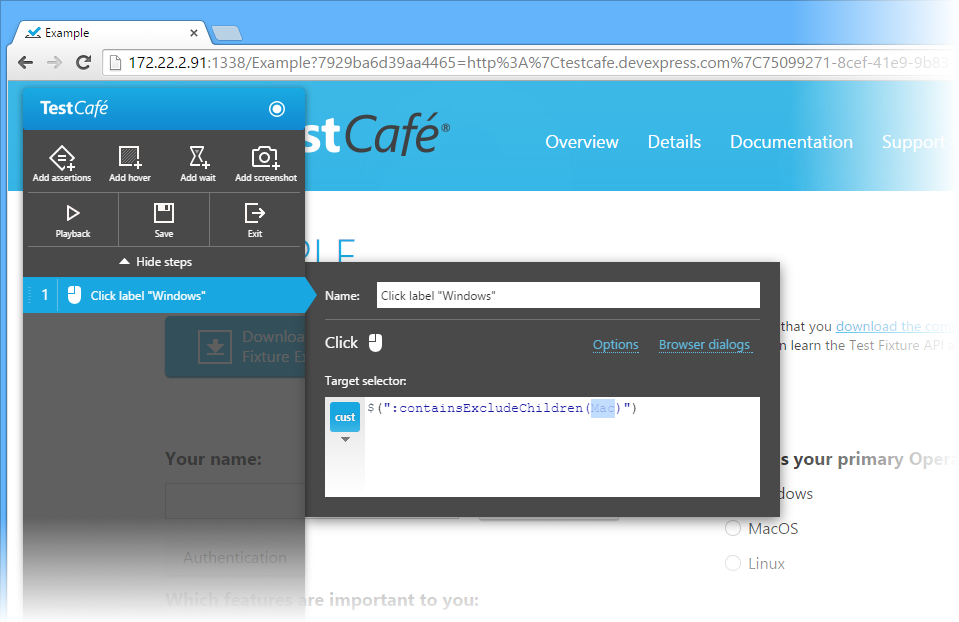


**Selector Expressions**

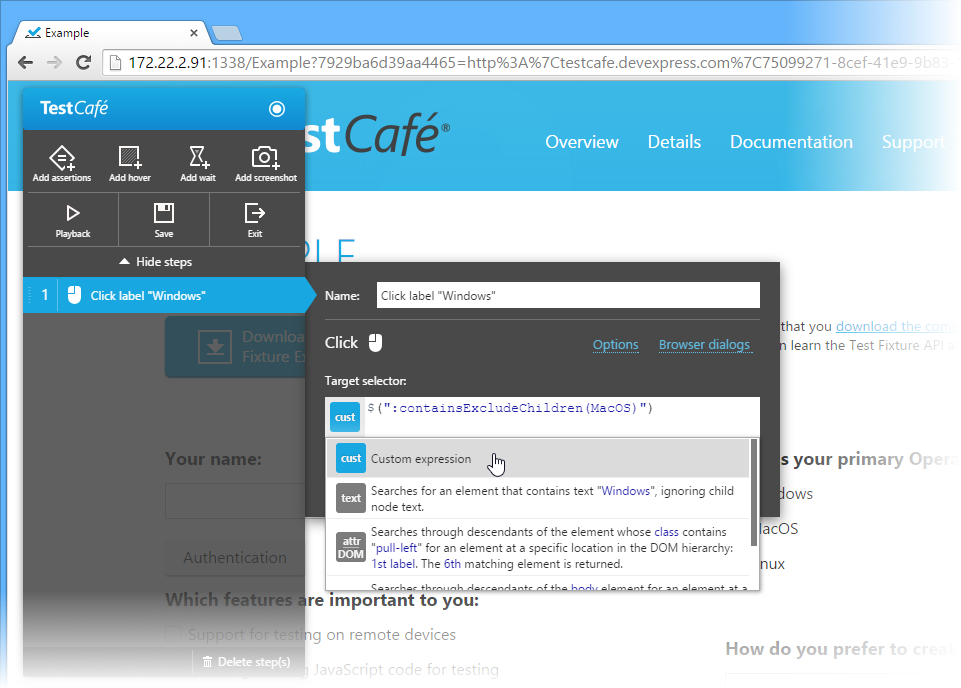
The target element selector is the essential action setting. It uniquely identifies and refers to the [action target element](https://testcafe.devexpress.com/Documentation/API_Reference/Test_Fixture_API/Miscellaneous#User_action_target) within the page. To help you be more precise in selecting the target element, TestCafe offers a set of predefined selector expressions. The expressions are built as a combination of specifically designed patterns. You can identify which patterns are used for a selector by the icon it displays. Each selector also displays a textual description of the pattern logic to be applied. Click a selector icon to open a popup list of available selector expressions and choose one that suits your needs best.



You are able to compose a custom selector expression as well.



Once you have changed the default selector code, a new - custom - selector is added to the list of available selectors.



**Shortcuts**

Use the following shortcuts when recording tests.

* CTRL+R: Records a step containing one or more assertions
* CTRL+SPACE: Records a **Hover** action
* CTRL+Q: Records a **Wait** action
* CTRL+M: Records the **Screenshot** action
* CTRL+D: Playbacks the test step by step from the beginning
* CTRL+S: Saves the test
* CTRL+E: Ends recording
* CTRL+G: Completes an action; used when working with input fields to indicate that an action, such as typing, is completed

窗体顶端

Search

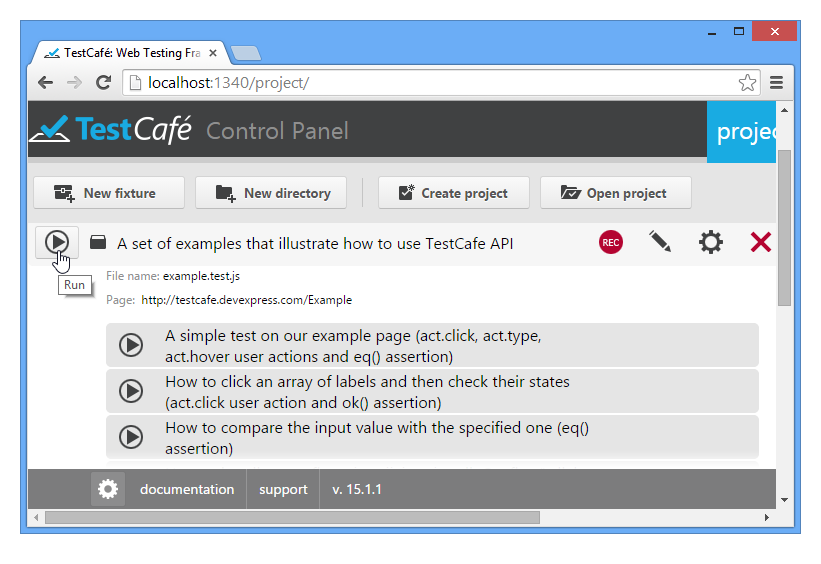
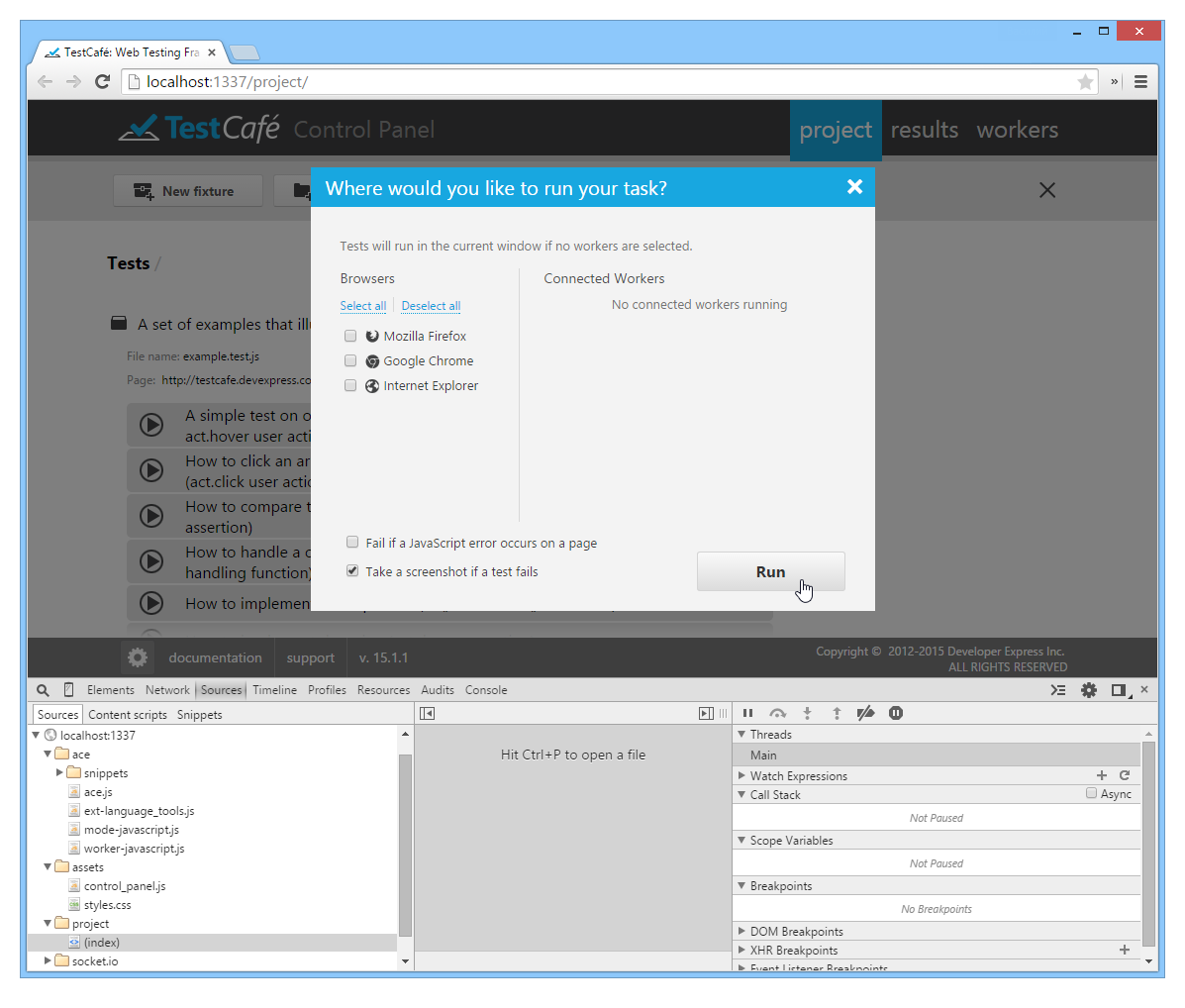
窗体底端

TEST DEBUGGING

The majority of modern browsers ship with integrated Web Developer tools, including JavaScript debuggers. And since TestCafe tests run JavaScript code, you can use your browser's built-in debugger to analyze your test code. To set up a breakpoint in your code, use the **debugger** keyword.

* debugger;

Most browsers allow for breaking script execution only when developer tools are invoked. Therefore, we suggest the following workflow when you are debugging tests.

1. Launch the target browser and click the **Run** button for the desired test, fixture or folder.  
   
2. This invokes the **Run Test** dialog. Make sure that no check box in the **Browsers** and **Connected Workers** sections is selected so that the test will run in the current window. Invoke **Web Developer Tools** and click **Run**.  
   

The test will start and when the breakpoint is reached, test execution will be suspended.



# CONTINUOUS INTEGRATION

At this point, you have already created, executed and debugged your tests. They are now ready to be used for your project's quality assurance. Also, if you use a Continuous Integration system, you may wish to take advantage of automatic test execution, instead of running tests manually using the UI.

## Workflow

Taking into account that Continuous Integration is a process running on a dedicated machine, you will need to follow these steps.

* **Copy your tests to the server**.  
  Since tests are JavaScript files, this is a simple file copy-paste operation.
* **Setup TestCafe on the server**.  
  This only requires a single line in the command prompt.
* **Write a Node.js application that runs the tests and logs the results**.  
  The TestCafe provides a dedicated Continuous Integration API allowing you to select and run tests, track their execution, and specify target browsers. You can also take advantage of Node.js to analyze the test results and log them as required.
* **Call the application from your Continuous Integration system**.  
  Once again, this only requires a single line in the command prompt.

## Setup on the Server

Using TestCafe with a Continuous Integration system suggests programmatic usage instead of UI access. This means that you can use a simpler setup path: install TestCafe directly as an **npm** module. For this purpose, first switch to the directory in which you will keep the applications that launch tests. Then, install the module using the following command line.

npm install testcafe@15.1.x

Note that you can use [TestCafe Settings](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/" \l "TestCafe_Settings) when you install TestCafe on your personal computer. However, you can skip this step if you specify these settings programmatically, as shown in the next section.

## Sample Testing Application

First, create a new JS file in the directory in which you installed the TestCafe (or any directory if you enabled global access) and then create a Node.js application.

Below, you will find a very simple Node.js application that you can use to run tests from a specified directory. Test execution results are logged into the console.

* var TestCafe = require('testcafe');
* var testCafe = new TestCafe (
* {
* controlPanelPort : 1337,
* servicePort : 1338,
* testsDir : 'D:\\TestCafe-13.1\\tests',
* browsers : {
* 'Mozilla Firefox': {
* path : "C:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe",
* cmd : "-new-window"
* }
* }
* });
* var runOptions = {
* workers: testCafe.listConnectedWorkers(),
* browsers: testCafe.listAvailableBrowsers(),
* emulateCursor: true
* };
* testCafe.runTests(runOptions, function () {
* testCafe.on('taskComplete', function (report) {
* console.log(report);
* });
* });

Test execution results can also be saved in the Nunit, Junit and JSON formats that are supported by most Continuous Integration systems (CruiseControl, TeamCity, Hudson/Jenkins, etc.). You can specify the test report's format and path using the **reportFormat** and **reportPath** properties of the [runTests](https://testcafe.devexpress.com/Documentation/API_Reference/Continuous_Integration_API/TestCafe_Methods" \l "runTests) method's **options** parameter. Test results will be saved when test execution is completed, before the [taskComplete](https://testcafe.devexpress.com/Documentation/API_Reference/Continuous_Integration_API/TestCafe_Events" \l "taskComplete) event is raised.

Alternatively, you can handle [taskComplete](https://testcafe.devexpress.com/Documentation/API_Reference/Continuous_Integration_API/TestCafe_Events" \l "taskComplete) and [taskUpdated](https://testcafe.devexpress.com/Documentation/API_Reference/Continuous_Integration_API/TestCafe_Events" \l "taskUpdated) events to save the test results.

For more information, refer to the [Comprehensive Reference on Continuous Integration API](https://testcafe.devexpress.com/Documentation/API_Reference/Continuous_Integration_API)

## Running the Application

Whatever continuous integration system you choose, you will run your tests using shell commands. Use the standard syntax for Node.js applications - Node.js executable with the application file as the command line parameter. (If Node.js is not installed globally, you will need to copy its executable to the directory that contains your application files.)

node testcafe\_getting\_started.js

## Remote Workers

To connect a remote computer or mobile device to the TestCafe that's running on a Continuous Integration server, navigate the browser on your device to the following URL :

http://<hostname>:<controlPanelPort>/worker/add/<workerName>

**hostname**: Continuous Integration server's hostname. This value should match the hostname you specified in the [TestCafe Settings](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/" \l "TestCafe_Settings) or in the [TestCafe constructor](https://testcafe.devexpress.com/Documentation/API_Reference/Continuous_Integration_API/TestCafe_Constructor).  
**controlPanelPort**: TestCafe's Control Panel port value. As described above, this value can either be specified in the [TestCafe Settings](https://testcafe.devexpress.com/Documentation/Using_TestCafe/Control_Panel/" \l "TestCafe_Settings) or passed as a TestCafe object constructor parameter.  
**workerName** - the name that you wish to assign to the connected worker.

When you navigate to the URL above, your remote browser will automatically become a connected worker.



Technically, the newly registered worker will be included in the list returned by the listConnected Workers() method. Note also that TestCafe raises specially designed events each time a worker is connected or disconnected, so you can respond as needed.