Test Cases Writing Guidelines

Why

Setting some rules about how to writing Sage HR manual test cases will make it easier to perform our job always within our quality standards.

We want

- Style and syntax consistency;
- The best readability;
- High clarity, conciseness and completeness;

- Automation oriented.

We don´t want

- Assumptions;
- Redundancies;
- Check UI in functional tests.

Content

Test Name

- The test name should be descriptive and legible, with words separated by spaces.

Example:

`[TestName("Timeoff event-based expiration")]`

Categories

AutoStatus

Only one, always first, before the tags.

Tags

As many as we need for identifying and filter the test cases.

Must be ordered from the most generic to the more specific.

Example:

`[Category(AutoStatus.Automated), Category(Tag.CI_Regression), Category(Tag.TimeOff), Category(Tag.Calculations), Category(Tag.Accruals), Category(Tag.Annually)]`

`[Category(AutoStatus.Manual_Assist), Category(Tag.Product), Category(Tag.Smoke), Category(Tag.CoreHR), Category(Tag.EmployeeData), Category(Tag.Documents)]`

Bugs

In case of bug, tag “Bug” should come just after the AutoStatus.

Example:

`[Category(AutoStatus.Manual_Assist), Category (Tag.Bug), Category(Tag.Product), Category(Tag.Smoke),`

`Category(Tag.CoreHR), Category(Tag.EmployeeData)]`
`[Issue("CHR-15488")]`

In the last line of the test, add the KnownBugLog helper with the corresponding Jira´s ticket link.

Example:

`TestHelpers.KnownBugLog("https://cakehr.atlassian.net/browse/CHR-3618");`

Description

- Should contain the summary and the purpose of the test.

Example:

`[Describe("Ensure that, for an event-based policy with expiration, an employee with a child date out of the expiration period, will not see the policy in time off details, only in time off request")`

Setup

- Setup steps should explicitly indicate all pre-conditions (settings - everything different from default - and data creation) needed for the execution of the test case.

Example:

```
Setup("Log in as the admin");
Setup("Create Employee1");
Setup("Create a policy 'Non Labour Based' with following values: Allocate '366' 'days' 'per year and accrue' 'annually' and that stops accrual with 'Sickday' policy. Add it to Employee1");
Setup("For 'Employee1', submit a time-off request of 1 day in policy 'Sickday' and approve it");
Setup("Go to time off page of Employee1 and confirm 'Yearly Allowance' of 'Non Labour Based' is '365 days'");
```

Note that the 5 setup steps above are just the actions needed to build the scenario for the test itself - edit the time off request. After the setup, the test really begins:

`Step("View as 'Employee1'");`

`Step("Edit the time off request changing the policy from 'Sickday' to 'Vacation'")`;

Common Setups

- The first pre-condition in the majority of the tests: `Setup("Log in as the admin");`
- Employee creation: `Setup("Create Employee");`

Steps

- Steps should contain essential information to perform the action functionally, keeping the business logic.
- Only for UI Test, we should add the elements used to perform the action.

Action Description

We prefer:

`Step("Delete the Schedule Group");`

If a step needs more information to be clear for all testers, we can add it - but we should not name elements or make a "click description", as "click on 'Save' button", unless it´s UI test case.

Options below should be used **only for UI tests**:

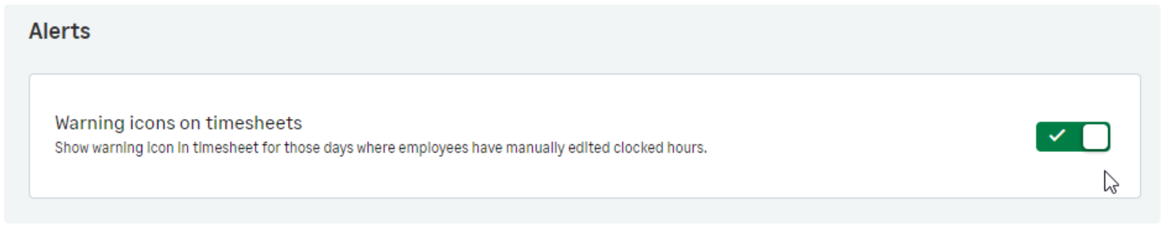
`Step("Delete the Schedule Group by clicking bin icon");`

`Step("Click on bin icon to delete the Schedule Group");`

`Step("Click on bin icon");`

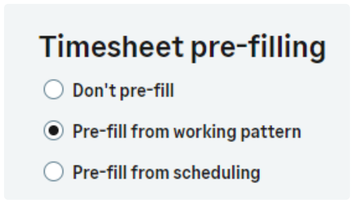
Common Actions

- When the action requires view as the employee (from employee profile): `Step("View as Employee1...);`
- Only when an action requires a real log in we use: `Step("Log in as Employee1...);`
- When must specify that needs to sign back in as the admin: `Step("Sign back in as admin...);`
- When an action requieres a real log out we use `Step (Log out as Employee1....);`
- When we want to enable an option `("In 'Alerts' section enable 'Warning icons on timesheets' option and save it");`


Alerts

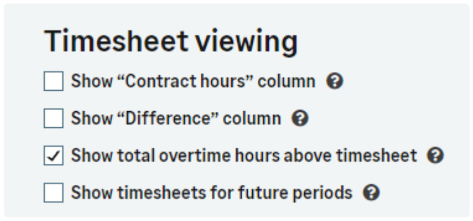
Warning icons on timesheets
Show warning icon in timesheet for those days where employees have manually edited clocked hours. ✓

- When we want to select an option `("In 'Timesheet pre-filling' section select 'Pre-fill from working pattern' option and save it");`


Timesheet pre-filling

○ Don't pre-fill
● Pre-fill from working pattern
○ Pre-fill from scheduling

- When we want to tick/untick an option `("In 'Timesheet viewing' section tick 'Show total overtime hours above timesheet' option and save it");`


Timesheet viewing

☐ Show "Contract hours" column
☐ Show "Difference" column
☑ Show total overtime hours above timesheet
☐ Show timesheets for future periods

Subject and Verb Tense

We always use the imperative form:

`Step("Change the Inactivity threshold to 5 minutes and save");`

Confirms

Subject and Verb Tense

We use the passive voice and the present tense:

`Confirm("'Save' button is displayed");` instead of `Confirm("We see the 'Save' button");`

`Confirm("The fields are no longer editable");` instead of `Confirm("The fields will be no longer editable");`

Comments

Extra information that must be taken into account or steps & confirms that must be temporally skipped.

Examples:

`// When running the test in non-bissextil years, we must  deduct 1 from all numbers regarding allocation and allowance (in steps and confirms)`

`// This test takes place over 2 days`

Context

- We only must specify the change of context (from admin to employee or the opposite), no need to specify it in every step.
- The context changes actions must be specified isolated in one line.

We want:

```
Setup("Log in as the admin);
Step("Go to /Settings/Time-offs");
Step("Create a policy...");
Step("View as Employee1");
Step("Go to dashboard and request a time off);
Step(Sign back as admin);
Step(Go to...)
```

We don´t want:

```
Step("Log in as Admin and go to /Settings/Time-offs);
Step("As Admin, create a policy...");
Step("View as Employee1 and go to the dashboard");
Step("As employee, request a time off);
Step(Sign back as Admin and go to...);
```

Syntax

URL, pages, sections

When performing the step, write all path that user must follow manually, separated by "/" and with first letter capitalized

Examples:

`Go to Settings/Time-off/Policies`

`Go to Dashboard`

If the path includes an id, we specified the page, then the path.

Examples:

```
Go to Employee2/Personal
```

```
Go to Employee2/Time-off
```

In case of employee profile,

```
Go to Employee2 profile
```

Naming

PascalCase

For every name composed of more than one word, we must use PascalCase concept.

Employees

Employees' names should describe its functions as much as possible.

If there´s only one employee, use only “Employee”, “DirectManager”...

If there are or more than one, add a number: ‘Employee1’, ‘Employee2’...

Use ‘Employee[n]’ to employees that won´t have the status changed.

Use ‘DirectManager1’ to the employee that will be a direct manager. In case you need more than one, use 'Direct Manager[n].

Use ‘TeamManager1' to the employee that will be a team manager. In case you need more than one, use ‘TeamManager[n]’.

Example:

```
Create four employees: DirectManager1, TeamManager1, Employe
e1 and Employee2.
```

Other Data

As above, other data creation will be the name of the element itself followed by a number:

Location1, Location2....

Team1, Team2...

Position1, Position2...

Group1, Group2...

Workflow1, Workflow2...

Descriptive Approach

For some test cases, some data created may need a descriptive name to facilitate checking and validations.

```
Step("In Settings/Recipes, add a new recipe:Name 'EmailTimeoffCanceledAttachmentAdded'");
```

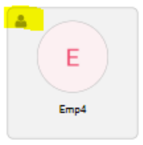
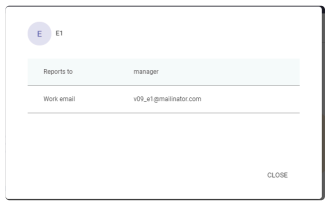
Quotes

Notifications copies or other names or strings must be between single quotes:

`Confirm("A new message is shown: 'There are no new asset assigned to this category'");`

UI Elements

ℹ️ Dear QA, you don´t know and don´t find how to name something? Let´s define it here 🙂

Name	Where	Image
open employee icon	org-chart	
basic info card	/org-chart Search Bar Organization	 E E1 Reports to manager Work email v09_e1@mailinator.com CLOSE
search bar	header	
stop accrual section	create / edit policy	
stopping policy	policy added to stop accrual section	
policy to stop	policy thas has a stopping policy added to stop accrual section	

References

When we refer to a specific element or to a value/string to be added or edited, we must mention it between single quotation marks.

```
Step("... check numbers in
'Yearly Allowance' and 'Accrued in current period'...");
```

Other Rules

Numbers

Number as quantity will be used as “n”: 1, 2, 56…

Date

We usually can´t use specific dates, as test cases are atemporal.

Most common references to dated will use ‘today', ‘tomorrow’, ‘yesterday’,'current', ‘next …', 'last…’.

Examples of date definitions:

For current date:

```
Step("Set Start date on [current day]/[current
month]/[current year]...");
```

For other dates:

```
Step("Set Start date on [01]/[next month]/[next year]...");
```

```
Step("Set Start date on [01]/[December]/[next year]...");
```

For Edge cases:

```
Step("Set Start date from next 20-Dec to next 10-Jan...);
```

```
Second week,, ten days from today, the day past tomorrow...
```

Time

Based on 24h system, format: hh:mm,

```
Step(".. at 17:30...);
```

Random

We want to avoid “random actions” even if the action allows it, so that the person running the test does not feel confused or lost.

Example:

- We don’t want:

Even if the test would works with this setup

`Setup("Edit ClockIn  and Clock Out for today and save changes");`

or

`Setup("Edit ClockIn  and Clock Out randomly for today and save changes");`

- We want:

To specify the action like that

`Setup("Edit ClockIn to:22:00 hours  and Clock Out to: 22:02 for today and save changes");`