Feature: Multiple site support

Background:
Given a global administrator named "Greg"
And a blog named "Greg's anti-tax rants"
And a customer named "Wilson"
And a blog named "Expensive Therapy" owned by "Wilson"

Scenario: Wilson posts to his own blog
Given I am logged in as Wilson
When I try to post to "Expensive Therapy"
Then I should see "Your article was published."

Scenario: Greg posts to a client's blog
Given I am logged in as Greg
When I try to post to "Expensive Therapy"
Then I should see "Your article was published."

All code in https://github.com/Diana0604/moodle-local-custombehattests
How does behat code look like…?

Given the following "courses" exist:
<table>
<thead>
<tr>
<th>fullname</th>
<th>shorthand</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course fullname</td>
<td>C_shortname</td>
<td>0</td>
</tr>
</tbody>
</table>

And I log in as "admin"
And I am on site homepage
Distribution of code

@tag11 @tag2
Feature: Title of feature
   Description of feature

   Background:
      Background steps run for all scenarios

@javascript
Scenario: First scenario run with javascript
   Steps for first scenario

Scenario: Second scenario will be round as background process
   Steps for second scenario
Mink extension

And I log in as "admin"

```php
/** *
 * Logs in the user. There should exist a user with the same value as username and password.
 */

@Given I log in as :username

public function i_log_in_as($username) {
    // Visit login page.
    $this->getSession()->visit($this->locate_path('login/index.php'));

    // Enter username and password.
    $this->execute('behat_forms::i_set_the_field_to', array('Username', $this->escape($username)));
    $this->execute('behat_forms::i_set_the_field_to', array('Password', $this->escape($username)));

    // Press log in button, no need to check for exceptions as it will checked after this step execution.
    $this->execute('behat_forms::press_button', get_string('login'));
}
```
Mink extension

```php
<?php
class my_bheat_class extends Behat\MinkExtension\Context\RawMinkContext {

/**
 * Description of first function
 */

protected function get_some_info($arg) {
    // manipulate $arg to get $info
    return $info;
}

/**
 * @Given I do step one on :param
 */

public function do_this_step($param) {
    // some code
    $var_needed = get_some_info($param);
    // some more code
}
```
Login test

All defined steps are stored in /admin/tool/behat/index.php
Login test

In a few steps we’re capable of creating this simple test

Feature: Simple Login
In order to interact with moodle
I need to log in

@javascript
Scenario: Log in with the predefined admin user
    Given I log in as "admin"
Login test

In a few steps we’re capable of creating this simple test

Feature: Simple Login
In order to interact with moodle
I need to log in

Scenario: Log in with the predefined admin user
    Given I log in as "admin"
    And I change window size to "large"
    Then I should see "You are logged in as Admin User" in the "page-footer" "region"
## Manage filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Active?</th>
<th>Order</th>
<th>Apply to</th>
<th>Settings</th>
<th>Uninstall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email protection</td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Uninstall</td>
<td></td>
</tr>
<tr>
<td>Display emoticons as images</td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Settings</td>
<td>Uninstall</td>
</tr>
<tr>
<td>Glossary auto-linking</td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Uninstall</td>
<td></td>
</tr>
<tr>
<td>Multi-Language Content</td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Settings</td>
<td>Uninstall</td>
</tr>
<tr>
<td>TeX notation</td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Settings</td>
<td>Uninstall</td>
</tr>
<tr>
<td>HTML tidy</td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Uninstall</td>
<td></td>
</tr>
<tr>
<td>Convert URLs into links and images</td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Settings</td>
<td>Uninstall</td>
</tr>
<tr>
<td><strong>MathType by WIRIS</strong></td>
<td>Disabled</td>
<td></td>
<td>Content</td>
<td>Settings</td>
<td>Uninstall</td>
</tr>
</tbody>
</table>

Turn filter on
Turn filter on

\[ \frac{1}{2} \]
Turn filter on

We can use their functions and the test works fine

```javascript
@javascript
Scenario: Turn filter on using moodle methods
   And I navigate to "Plugins" in site administration
   And I click on "Manage filters" "link"
   And I click on "On" "option" in the "MathType by WIRIS" "table_row"
   And I open my profile in edit mode
   And I click on "HTML" "button"
   And I set the field "Description" to "<math><mfrac><mn>1</mn><mn>2</mn></mfrac></math>"
   And I click on "HTML" "button"
Then "//img[@alt='1 half']" "xpath_element" should exist
```
Let’s try to write it in a more understandable way...

@javascript
Scenario: Turn filter on using moodle methods
   And I navigate to "Plugins" in site administration
   And I click on "Manage filters" "link"
   And I turn "MathType by WIRIS" filter on
   And I open my profile in edit mode
   And I click on "HTML" "button"
   And I set the field "Description" to mathml "1 half"
   And I click on "HTML" "button"
Then I should see MathType image "1 half"
Turn filter on

Let’s try to write it in a more understandable way...

When I click on "On" option in the "MathType by WIRIS" "table_row"
When I turn "MathType by WIRIS" filter on

And I set the field "Description" to "<math><mfrac><mn>1</mn><mn>2</mn></mfrac></math>"
And I set the field "Description" to mathml "1 half"

Then "//img[@alt='1 half']" "xpath_element" should exist
Then I should see MathType image "1 half"
Steps to program

Turn filter on
Turn filter on

Steps to program

1. **Reproduce moodle structure**
2. Apply secret technique
3. Run command `php admin/tool/behat/cli/util.php --enable`
Turn filter on

Structure of code

Mink Extension

- behat_base.php
- behat_moodle_steps.php
- behat_my_base.php
- behat_my_steps.php

- protected
- public
Turn filter on

Simplified version of our code

```php
<?php
/*
 * Protected methods
 */
require_once(__DIR__ . '/relative/path/to/lib/behat/behat_base.php');

class behat_custom_base extends behat_base {
    // We don't need any auxiliar methods at the moment
}
```

```php
<?php
/*
 * Public methods to interact with behat
 */
require_once(__DIR__ . '/behat_custom_base.php');

class behat_custom_filter extends behat_custom_base {
    // Public methods to interact with behat
}
```

behat_custom_base.php

behat_custom_filter.php
Turn filter on

Steps to program

1. Reproduce moodle structure
2. **Apply secret technique**
3. Run command `php admin/tool/behat/cli/util.php --enable`

```plaintext
When I click on "On" "option" in the "MathType by WIRIS" "table_row"
When I turn "MathType by WIRIS" filter on

And I set the field "Description" to "&lt;math&gt;&lt;mfrac&gt;&lt;mn&gt;1&lt;/mn&gt;&lt;mn&gt;2&lt;/mn&gt;&lt;/mfrac&gt;&lt;/math&gt;"
And I set the field "Description" to mathml "1 half"

Then "/img[@alt='1 half']" "xpath_element" should exist
Then I should see MathType image "1 half"
```
Steps to program

- **Steps definitions**
  - Type: All the available steps definitions
  - Area: All the available steps definitions
  - Contains: click

Click on the element of the specified type which is located inside the second element. When I click on "ELEMENT_STRING" choose... in the "ELEMENT_CONTAINER_STRING" choose... behat_general: i_click_on_in_the()
Turn filter on

Steps to program

1. Reproduce moodle structure
2. **Apply secret technique**
3. Run command php admin/tool/behat/cli/util.php --enable

```bash
When I click on "On" "option" in the "MathType by WIRIS" "table_row"
When I turn "MathType by WIRIS" filter on
```
Turn filter on

Simplified version of our code

```
/*
 * Click on the element of the specified type which is located inside the second element.
 */

public function i_click_on_in_the($element, $selectortype, $nodeelement, $nodeselectortype) {
    $node = $this->get_node_in_container($selectortype, $element, $nodeselectortype, $nodeelement);
    $this->ensure_node_is_visible($node);
    $node->click();
}
```

behat_general.php

```
/*
 * Turns MathType filter on in 'Manage Filters' menu
 * @Given I turn :arg filter on
 *
 * @throws Behat\Mink\Exception\ElementNotFoundException If $xpathnumber is not correctly computed.
 */

public function i_turn_filter_on($filtername) {
    $node = $this->get_node_in_container("option", "On", "table_row", $filtername);
    $this->ensure_node_is_visible($node);
    $node->click();
}
```

behat_custom_filter.php
Turn filter on

Steps to program

1. Reproduce moodle structure
2. **Apply secret technique**
3. Run command `php admin/tool/behat/cli/util.php --enable`
Turn filter on

Simplified version of our code

```php
class behat_custom_base extends behat_base {
    const MATHML = "<math><mfrac><mn>1</mn><mn>2</mn></mfrac></math>";

    /**
     * Return mathml given by alternative text
     *
     * @param string $alt -> alternative text
     * @throws Exception If alternative is not recognized
     */
    protected function alternative_to_mathml($alt) {
        if ($alt == "1 half") {
            return self::MATHML;
        }
        throw new Exception("Alternative text not recognized");
    }
}
```

```php
class behat_custom_filter extends behat_custom_base {

    /**
     * @Given I set the field $field to mathml $alt
     */
    public function i_set_the_field($field, $alt) {
        $mathml = $this->alternative_to_mathml($alt);
        $this->execute('behat_forms:i_set_the_field_to', array($field, $mathml));
    }

    /**
     * @Then I should see MathType image $alt
     */
    public function i_should_see_image($alt) {
        $element = "/img[@alt="$alt"]";
        // Getting node selector and locator
        list($selector, $locator) = $this->transform_selector("xpath_element", $element);
        // Will throw an ElementNotfoundException if it does not exist.
        $this->find($selector, $locator);
    }
}
```
Turn filter on

Simplified version of our code

```php
<?php
require_once(__DIR__ . '/../././././lib/behat/behat_base.php');

class behat_custom_base extends behat_base {
    const MATHML = "<math><mfrac><mn>1</mn><mn>2</mn></mfrac></math>";
    
    * Return mathml given by alternative text
    * 
    * @param string $alt -> alternative text
    * @throws Exception If alternative is not recognized
    */
    protected function alternative_to_mathml($alt){
        if($alt == "1 half") return self::MATHML;
        throw new \Exception('Alternative text not recognized');
    }
}
```
Let’s try to write it in a more understandable way...

@javascript
Scenario: Turn filter on using moodle methods
  And I navigate to "Plugins" in site administration
  And I click on "Manage filters" "link"
  And I turn filter on "MathType by WIRIS"
  And I open my profile in edit mode
  And I click on "HTML" "button"
  And I set the field "Description" to mathml "1 half"
  And I click on "HTML" "button"
  Then I should see MathType image "1 half"
Turn filter on

Steps to program

1. Reproduce moodle structure
2. Apply secret technique
3. Run command `php admin/tool/behat/cli/util.php --enable`
Thank you

diana@wiris.com

All code in https://github.com/Diana0604/moodle-local-custombehattests