Selenium with PHP Cheatsheet

A comprehensive cheat sheet for using Selenium with PHP for web application testing, covering setup, basic commands, element interaction, waits, and best practices.

Setup and Basic Commands

Installation

PHP WebDriver Installation (facebook/webdriver) with Composer

```
composer require facebook/webdriver
```

Selenium Server Setup

1. Download Selenium Server (selenium.dev/downloads)
2. Start the server:

```
java -jar selenium-server-standalone-<version>.jar
```

ChromeDriver Setup

1. Download ChromeDriver (chromedriver.chromium.org/downloads) - ensure compatibility with your Chrome version.
2. Place ChromeDriver in a directory accessible by your system path.

Basic Browser Operations

Opening a Browser

```
use Facebook\WebDriver\Remote\RemoteWebDriver;
use Facebook\WebDriver\WebDriverBy;

$host = 'http://localhost:4444/wd/hub';
// Selenium Standalone
$driver = RemoteWebDriver::create('http://localhost:4444', DesiredCapabilities::chrome());
```

Closing a Browser

```
$driver->quit();
```

Navigating to a URL

```
$driver->get('https://www.example.com');
```

Getting Current URL

```
$currentUrl = $driver->getCurrentURL();
echo $currentUrl;
```

Back, Forward, Refresh

```
$driver->navigate()->back();
$driver->navigate()->forward();
$driver->navigate()->refresh();
```

Getting Page Title

```
$title = $driver->getTitle();
echo $title;
```

Finding Elements

Locating Elements

```
use Facebook\WebDriver\WebDriverBy;

// By ID
$element = $driver->findElement(WebDriverBy::id('elementId'));

// By Name
$element = $driver->findElement(WebDriverBy::name('elementName'));

// By Class Name
$element = $driver->findElement(WebDriverBy::className('elementClass'));

// By CSS Selector
$element = $driver->findElement(WebDriverBy::cssSelector('#elementId > .elementClass'));

// By XPath
$element = $driver->findElement(WebDriverBy::xpath('//div[@id="elementId"]/p'));

//By Tag Name
$element = $driver->findElement(WebDriverBy::tagName('h1'));

//By Link Text
$element = $driver->findElement(WebDriverBy::linkText('Clickable Link'));

//By Partial Link Text
$element = $driver->findElement(WebDriverBy::partialLinkText('Clickable'));
```

Element Interaction and Waits

Working with Web Elements

Clicking an Element

```
$element->click();
```

Sending Keys to an Element

```
$element->sendKeys('Some text');
```

Getting Text from an Element

```
$text = $element->getText();
echo $text;
```

Clearing a Field

```
$element->clear();
```

Interacting with Dropdowns

```
use Facebook\WebDriver\Support\UI\Select;

$dropdown = new Select($driver->findElement(WebDriverBy::id('dropdownId')));
$dropdown->selectByValue('optionValue');
$dropdown->selectByVisibleText('Option Text');
$dropdown->selectByIndex(1); //Index starts at 0
```

Interacting with Checkboxes

```
if (!$checkbox->isSelected()) {
    $checkbox->click();
}
```

Submitting a Form

```
$element->submit();
```

Checking if Element is Displayed

```
$isDisplayed = $element->isDisplayed();
if ($isDisplayed) {
    echo 'Element is displayed';
}
```

Waits

Implicit Wait

```
$driver->manage()->timeouts()->implicitlyWait(10); // seconds
```

Note: Sets a timeout for all subsequent findElement(s) calls.

Explicit Wait

```
use Facebook\WebDriver\WebDriverExpectedCondition;
use Facebook\WebDriver\WebDriverBy;
use Facebook\WebDriver\Remote\RemoteWebDriver;

$driver->get('https://example.com');
$wait = new WebDriverWait($driver, 10);

// Wait until the element is located
$element = $wait->until(
    WebDriverExpectedCondition::presenceOfElementLocated(
        WebDriverBy::id('myDynamicElement')
    )
);
```

Common conditions: `presenceOfElementLocated`, `visibilityOfElementLocated`, `elementToBeClickable`.

Advanced Interactions

Handling Popups and Frames

Handling Alerts

```
$alert = $driver->switchTo()->alert();
echo $alert->getText();
$alert->accept(); // Or $alert->dismiss();
```

Handling Iframes

```
// Switch to iframe by index
$driver->switchTo()->frame(0);

// Switch to iframe by name or ID
$driver->switchTo()->frame('frameName');

// Switch back to the main document
$driver->switchTo()->defaultContent();
```

Handling Multiple Windows

```
// Get current window handle
$currentWindow = $driver->getWindowHandle();

// Get all window handles
$windowHandles = $driver->getWindowHandles();

// Switch to a specific window
foreach ($windowHandles as $handle) {
    if ($handle != $currentWindow) {
        $driver->switchTo()->window($handle);
        break;
    }
}
```

Mouse and Keyboard Actions

Hover Action

```
use Facebook\WebDriver\WebDriverAction;
use Facebook\WebDriver\Interactions\WebDriverActions;

$builder = new WebDriverActions($driver);
$element = $driver->findElement(WebDriverBy::id('hoverElement'));
$builder->moveToElement($element)->perform();
```

Drag and Drop Action

```
use Facebook\WebDriver\WebDriverAction;
use Facebook\WebDriver\Interactions\WebDriverActions;

$builder = new WebDriverActions($driver);
$source = $driver->findElement(WebDriverBy::id('draggable'));
$target = $driver->findElement(WebDriverBy::id('droppable'));
$builder->dragAndDrop($source, $target)->perform();
```

Keyboard Actions

```
use Facebook\WebDriver\WebDriverKeys;

$element->sendKeys(WebDriverKeys::ENTER);
$element->sendKeys(WebDriverKeys::CONTROL . 'a'); //Select All
```

Common keys: `ENTER`, `TAB`, `CONTROL`, `SHIFT`, `ALT`.

Taking Screenshots

Taking a Screenshot

```
$driver->takeScreenshot('screenshot.png');
```

Saves the screenshot to the specified file path.

Taking a Screenshot of an Element

```
$element = $driver->findElement(WebDriverBy::id('elementId'));
$element->takeScreenshot('element_screenshot.png');
```

Saves the screenshot of specific element to the specified file path.

Testing Framework and Best Practices

Structuring Tests with PHPUnit

Basic PHPUnit Test Structure

```
use PHPUnit\Framework\TestCase;
use Facebook\WebDriver\Remote\RemoteWebDriver;
use Facebook\WebDriver\DesiredCapabilities;

class ExampleTest extends TestCase
{
    /**
     * @var RemoteWebDriver
     */
    protected $driver;

    public function setUp(): void
    {
        $host = 'http://localhost:4444/wd/hub';
        $capabilities = DesiredCapabilities::chrome();
        $this->driver = RemoteWebDriver::create($host,
$capabilities, 5000);
    }

    public function tearDown(): void
    {
        $this->driver->quit();
    }

    public function testExample()
    {
        $this->driver->get('https://www.example.com');
        $this->assertStringContainsString('Example Domain',
$this->driver->getTitle());
    }
}
```

Running Tests

```
./vendor/bin/phpunit
```

Best Practices and Tips

Page Object Model (POM) Basics

- Create classes representing web pages.
- Each class contains elements and methods to interact with the page.
- Reduces code duplication and improves maintainability.

Test Data Management

- Use data providers in PHPUnit to parameterize tests.
- Store test data in separate files (e.g., JSON, CSV).
- Avoid hardcoding data within tests.

Debugging and Logging

- Use `var_dump()` or `print_r()` for debugging.
- Implement logging to track test execution and errors.
- Use try-catch blocks to handle exceptions gracefully.

Retry Failed Tests

- PHPUnit has a plugin to automatically retry failed tests.
- This is useful for handling intermittent failures due to network issues.

Parallel Test Execution

- Tools like ParaTest can be used to run PHPUnit tests in parallel.
- Reduces the overall test execution time.