Selenium 4: New Feature - Locate nearby elements

Java only

Ashis Raj

Selenium 4 brings Relative Locators (originally named Friendly Locators). This functionality was added to help you locate elements that are nearby other elements. These locators are helpful when it is not easy to construct a locator for the desired element, but easy to describe spatially where the element is in relation to an element that does have an easily constructed locator.

The available options are:

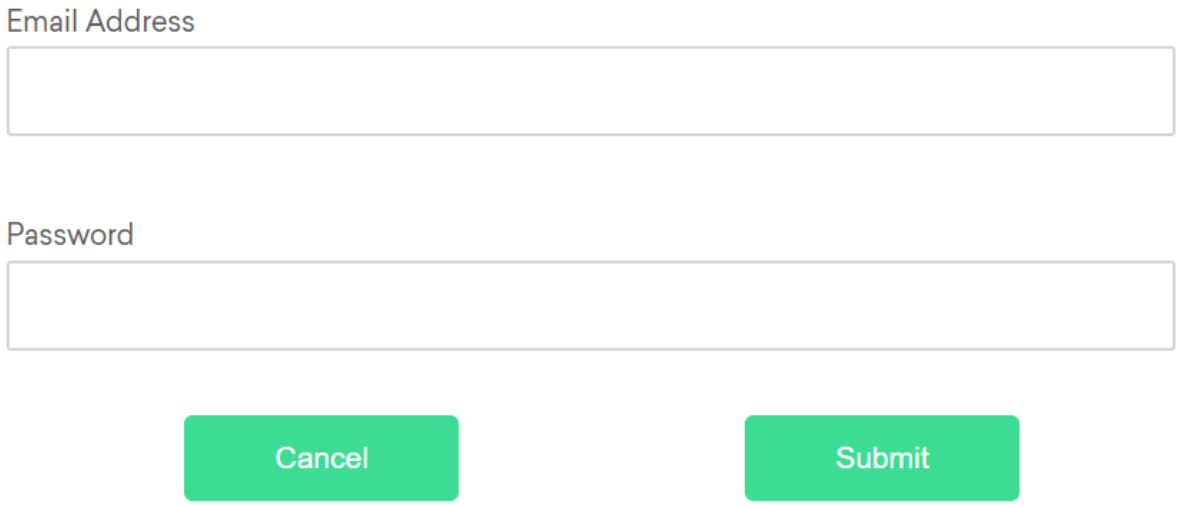
- above(): sought-after element appears above specified element
- below(): sought-after element appears below specified element
- toLeftOf(): sought-after element appears to the left of specified element
- toRightOf(): sought-after element appears to the right of specified element
- near(): sought-after element is at most 50 pixels away from specified element. There's also an overloaded method to allow you to specify the distance.

How it works

Selenium uses the JavaScript function getBoundingClientRect() to determine the size and position of elements on the page, and can use this information to locate neighboring elements (find the relative elements).

Relative locator methods can take as the argument for the point of origin, either a previously located element reference, or another locator. In these examples I'll be using locators only, but you could swap the locator in the final method with an element object and it will work the same.

Let us consider the below example for understanding the relative locators.



Available relative locators

Above

If the email text field element is not easily identifiable for some reason, but the password text field element is, we can locate the text field element using the fact that it is an “input” element “above” the password element.

```
By emailLocator =
RelativeLocator.with(By.tagName("input")).above(By.id("password"));
```

Below

If the password text field element is not easily identifiable for some reason, but the email text field element is, we can locate the text field element using the fact that it is an “input” element “below” the email element.

```
By passwordLocator =
RelativeLocator.with(By.tagName("input")).above(By.id("email"));
```

LeftOf

If the cancel button is not easily identifiable for some reason, but the submit button element is, we can locate the cancel button element using the fact that it is a “button” element to the “left of” the submit element.

```
By cancelLocator =
```

```
RelativeLocator.with(By.tagName("button")).toLeftOf(By.id("submit"));
```

RightOf

If the submit button is not easily identifiable for some reason, but the cancel button element is, we can locate the submit button element using the fact that it is a “button” element “to the right of” the cancel element.

```
By submitLocator =
RelativeLocator.with(By.tagName("button")).toLeftOf(By.id("cancel"));
```

Near

If the relative positioning is not obvious, or it varies based on window size, you can use the near method to identify an element that is at most 50px away from the provided locator. One great use case for this is to work with a form element that doesn’t have an easily constructed locator, but its associated input label element does.

```
By emailLocator =
RelativeLocator.with(By.tagName("input")).near(By.id("lbl-email"));
```

Chaining relative locators

You can also chain locators if needed. Sometimes the element is most easily identified as being both above/below one element and right/left of another.

```
By submitLocator =
RelativeLocator.with(By.tagName("button")).below(By.id("email").toRigh
tOf(By.id("cancel"));
```