Selenium Actions

In Selenium, l'interfaccia `Actions` è utilizzata per eseguire azioni complesse, come operazioni di trascinamento e rilascio, clic e attesa simultanei, tastiere virtuali, ecc. Questa interfaccia si trova nel package `org.openqa.selenium.interactions` ed è spesso utilizzata in combinazione con `WebDriver`.

Ecco alcuni dei metodi principali dell'interfaccia `Actions` di Selenium:

1. **`click(WebElement element)`**:
    - Esegue un clic sull'elemento specificato.

```
WebElement button = driver.findElement(By.id("myButton"));
Actions actions = new Actions(driver);
actions.click(button).perform();
```

2. **`doubleClick(WebElement element)`**:
    - Esegue un doppio clic sull'elemento specificato.

```
WebElement element = driver.findElement(By.id("myElement"));
Actions actions = new Actions(driver);
actions.doubleClick(element).perform();
```

3. `contextClick(WebElement element)`:
    - Esegue un clic destro (contesto) sull'elemento specificato.

```
WebElement element = driver.findElement(By.id("myElement"));
Actions actions = new Actions(driver);
actions.contextClick(element).perform();
```

4. **`sendKeys(CharSequence... keys)`**:
    - Invia una sequenza di tasti. Può essere utilizzato per inviare combinazioni di tasti come Ctrl+A, Ctrl+C, Ctrl+V, ecc.

```
Actions actions = new Actions(driver);
actions.sendKeys(Keys.CONTROL, "a").sendKeys(Keys.CONTROL, "c").perform();
```

5. `moveToElement(WebElement toElement)`:
    - Sposta il mouse sopra l'elemento specificato.

```
WebElement element = driver.findElement(By.id("myElement"));
Actions actions = new Actions(driver);
actions.moveToElement(element).perform();
```

6. **`dragAndDrop(WebElement source, WebElement target)`**:
    - Esegue un'operazione di trascinamento e rilascio dalla posizione sorgente alla posizione di destinazione.

```
WebElement sourceElement = driver.findElement(By.id("sourceElement"));
WebElement targetElement = driver.findElement(By.id("targetElement"));
Actions actions = new Actions(driver);
actions.dragAndDrop(sourceElement, targetElement).perform();
```

7. `dragAndDropBy(WebElement source, int xOffset, int yOffset)`:
    - Esegue un'operazione di trascinamento e rilascio dalla posizione sorgente di un offset specificato.

```
WebElement sourceElement = driver.findElement(By.id("sourceElement"));
Actions actions = new Actions(driver);
actions.dragAndDropBy(sourceElement, 100, 50).perform();
```

8. **`release()`**:
    - Rilascia il pulsante del mouse. È spesso utilizzato dopo un'operazione di trascinamento.

```
Actions actions = new Actions(driver);
actions.clickAndHold(sourceElement).moveToElement(targetElement).release().perform();
```

9. `clickAndHold(WebElement element)`:
    - Esegue un clic e tiene premuto il pulsante del mouse sull'elemento specificato.

```
WebElement element = driver.findElement(By.id("myElement"));
Actions actions = new Actions(driver);
actions.clickAndHold(element).perform();
```

10. **`build()`**:
    - Compila tutte le azioni definite fino a quel momento, creando un oggetto `CompositeAction`. Può essere utile quando si eseguono azioni multiple in sequenza.

```
Actions actions = new Actions(driver);
actions.moveToElement(element1).click().moveToElement(element2).click().build().perform();
```

11. **`perform()`**:
    - Esegue tutte le azioni definite fino a quel momento.

```
Actions actions = new Actions(driver);
actions.moveToElement(element).click().perform();
```

Questi sono solo alcuni dei metodi principali forniti dall'interfaccia `Actions` di Selenium. L'utilizzo di `Actions` è particolarmente utile per automatizzare scenari che coinvolgono interazioni complesse con il mouse e la tastiera durante i test delle applicazioni web.