**Use of Strings in Selenium WebDriver Examples**

[**https://www.seleniumeasy.com/java-tutorials/string-examples-using-selenium-webdriver**](https://www.seleniumeasy.com/java-tutorials/string-examples-using-selenium-webdriver)

[Home](https://www.seleniumeasy.com/) >> [Java Tutorials for Selenium](https://www.seleniumeasy.com/java-tutorials) >> [Use of Strings in Selenium WebDriver Examples](https://www.seleniumeasy.com/java-tutorials/string-examples-using-selenium-webdriver)

In java Strings are special and very interesting topic and are most commonly used. String is nothing but sequence of characters such as "hello", Strings should be enclosed with double quotes(" ").

String is not a primitive type like int or float. Instead string is a class which is present in java.lang package which creates string objects.

Declaration and initialization of String is same as primitive types such as int or double but String first letter is always capital as it is a class but not keyword like int and 's' is not primitive type, rather it is reference type

**String s="Hello Java";** *//This is String literal*

In Java, Below is an how we declare and initialize a String

package com.test;

public class StringInitExample {

public static void main(String[] args) {

String s; //declaring String

s="String in Java"; // initializing String with sequence of characters

System.out.println(s);

}

}

Strings are also created By using new keyword

**String str1 = new String("Hello Java");** //Here it is String object

String str2 = new String();  
str = "Hello, String";

**string.equals()**

Syntax: boolean equals(Object obj);

equals() method compares two references and returns true only if two references are pointing to same object but in String class equals method compares based on content of the string. If the content is same in two different objects, it returns true.

And '==' is used to compare references only. "==" operator will return true if two object reference are same otherwise "==" will return false.

Example: This example makes you clear about .equals() method and == operator.

public class StringEqualsDemo {

public static void main(String[] args) {

String s= new String("Hello");

String s1=new String("Hello");

String s3= new String("Apple");

String s4= new String("APPLE");

System.out.println(s.equals(s1));// returns true

System.out.println(s==s1);// returns false

System.out.println(s3.equals(s4));// returns false

System.out.println(s3.equalsIgnoreCase(s4));// returns true

}

}

**equalsIgnoreCase(String string)** : It works same as equals() method, but it doesn’t consider the case while comparing strings. In the above example String s3 and s4 returns true when we use equalsIgnoreCase() as it ignores the case during comparison.

**Strings are immutable** :-  
Immutable means changes made to existing object will not be affected to the object. If we want to make any changes to the existing object, we need to create a new object, so that the changes are affected to the new object.

Example:

public class StringImmutable {

public static void main(String[] args) {

String s= new String("Hello");

s.concat("Java");

System.out.println(s);// prints Hello

}

}

In the above program we are concatenating the string object but concatenation is not done to the object 's' and prints just 'Hello' but not 'Hello Java'.

The same can be achieved by using the below :

String s= new String("Hello");

String s1;

s1=s.concat("Java");

System.out.println(s);// prints Hello

System.out.println(s1);// prints Hello Java

Difference between String literal and creating String object with new operator:

String s="Hello"; Here compiler creates a single String object in String Constant Pool with the value "Hello"

Let's see some of the most commonly used java string methods in Selenium webdriver automation :-

**How to compare Strings in Webdriver?**

In Selenium webdriver, we get text using 'webelement.getText()' and use this string to verifyText / compare two strings and validate / assert in our automation.

Syntax to get text in selenium :

WebElement element= driver.findElement(By.id(“someid”));

String strText = element.getText();

**Example to compare String in webdriver**:-

WebElement strValue = driver.findElement(By.id("your id"));

String strExpected = "Text to compare";

String strActual = strValue.getText();

System.out.println(strActual);

if (strExpected.equals(strActual)) {

System.out.println("Strings are equal");

} else {

System.out.println("Strings are NOT equal");

}

Example to compare List of Strings in Webdriver, We will get list of WebElements from select dropdown and compare with expected String Array: -

int count = 0;

//Expected values in dropdown

    String[] values = {"Month", "JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"};

    WebElement dropdownElement = driver.findElement(By.id("locator"));

    Select select = new Select(dropdownElement);

//select.getoptions() returns all options belonging to select tag

    List<WebElement> options = select.getOptions();

    for (WebElement option : options) {

        for (int i = 0; i < values.length; i++) {

            if (option.getText().equals(values[i])) {

                count++;

            }

        }

    }

    if (count == values.length) {

        System.out.println("values matched..");

    } else {

        System.out.println("values not matched");

    }

**String split(): -**

Syntax:  
String[] split(String regex);  
String[] split(String regex, int limit)  
Split method separates the input string with the specified regular expression / delimiter and returns string array.

Example: -

package com.test;

public class StringSplitExample {

public static void main(String[] args) {

String s = "Java String Split Example";

String[] str1 = s.split(" ");

for (int i = 0; i < str1.length; i++) {

System.out.println(str1[i]); // output

//Java

//String

//Split

//Example

}

//Split with limit -

//it will split the string based on the limit we specify

String[] str2 = s.split(" ", 2);

for (int i = 0; i < str2.length; i++) {

System.out.println(str2[i]); // output

// Java

//String Split Example

}

}

}

**String contains(): -**

Syntax: boolean contains(charSequence s);

This method returns boolean value as true if the specified string or sequence of characters passed as parameter matches to the given string object. otherwise it returns false. Remember contains() method is case sensitive.

If the CharSequence is null then it method throws 'NullPointerException'.

**Java String contains() example :-**

package com.test;

public class StringContainsExample {

public static void main(String args[]){

String s="Hello Java";

System.out.println(s.contains("Hello Java"));// returns true

System.out.println(s.contains("Java"));//true as Java is present in the given String

System.out.println(s.contains("hello Java"));//false - as contains() method is case sensitive.

System.out.println(s.contains("o"));//true

System.out.println(s.contains("aa"));//false as 'aa' is not present,

//if else example

String str = "Java String example";

if(str.contains("example")){

   System.out.println("Success - String found");

}

else{

   System.out.println("Failed - String not found");

}

   }

}

**String Contains Example with Selenium WebDriver**

public boolean isStrPresent(String str) {

allElementsList=driver.findElements(By.xpath("xpath"));

for(WebElement strElement:allElementsList) {

String strValue = strElement.getText();

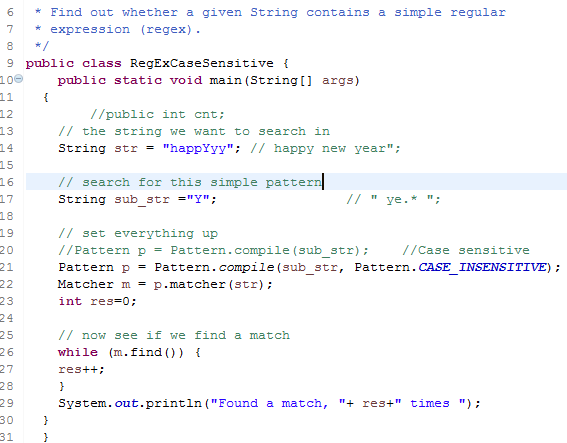
if(strValue.contains(str))

return true;

}

return false;

}



**String concate() : -**

Syntax: String concate(String str);

In order to concatenate multiple strings, we use concat() method in Java.

We can also do String concatenation using + Operator, using StringBuilder and StringBuffer class to join Strings in Java.

Example:

package com.test;

public class StringConcatExample {

public static void main(String args[]) {

//using string.concat

String s1="Hello";

String s2="all";

System.out.println(s1.concat(s2)); // Helloall --

//adds second object's string to first object's end of string without any spaces.

//using + operator to concatenate String

String first = "Raj";

String last = "Chris";

String name = first + " " + last;

System.out.println(name);

//using StringBuilder

StringBuilder strBuilder = new StringBuilder(14);

strBuilder.append(first).append(" ").append(last);

System.out.println(strBuilder.toString());

//using StringBuffer

StringBuffer strBuffer = new StringBuffer(15);

strBuffer.append(first).append(" ").append(last);

System.out.println(strBuffer.toString());

}

}

String concatenation using '+' operator works fine if we have to join fixed size of Strings like one or two, but if you have to join some thousands of String then it effects the performance.

For huge operations, we should prefer using StringBuilder for concatenation of multiple Strings.

**String length():-**

Syntax: int length();

returns the length of the given string i.e, number of characters including space. The maximum size a string can have 231-1.

Example:

String s="Hello Java";

System.out.println(s.length());// returns integer number as 10

In the above example, the method calculates the length of the string including white spaces.

If you want to get the count for the number of characters in a string excluding the spaces, then we use string replace method to remove white space.

Example :

String s="Hello Java";

System.out.println(s.replace(" ", "").length());// returns integer number as 9

**String isEmpty(): -**

Syntax: boolean isEmpty();

Checks whether the given string is empty or having sequence of characters. if the string is empty then returns true else false.

String s2="";

String s="Hello Java";

System.out.println(s2.isEmpty());// returns true

System.out.println(s.isEmpty());// returns false

**String replace(): -**

Syntax: String replace(char oldchar, char new char);  
This method replaces the character with first parameter to second parameter and returns the modified String object.

Example:

String str="Hello Java";

System.out.println(str.replace('a', 'b'));//returns the modified string as 'Hello Jbvb'

**String replaceAll(): -**

Syntax: String replaceAll(String oldString, String newString);

This method replaces the old string with specified given string.  
String s1="Java Programming Language";  
System.out.println(s1.replaceAll("Java", "Object Oriented"));// returns the String as Object Oriented Programming Language  
System.out.println(s1.replaceAll("Hello", "Object Oriented"));// returns the given String itself because the specified old String value is not existing in the given String object

**String replaceFirst(): -**

Syntax: String replaceFirst(String old, String new);

this method replaces the old string with new string , but with the first occurrence of the string object.  
Example:

String s1="the sentence doesn't end with because, because because is a conjunction";

System.out.println(s1.replaceFirst("because", "what"));

**String startsWith(): -**

Syntax: boolean startsWith(String startString);

This method returns true if the specified string is starts with the given string object.

Example:

String str="Hello Java";

String s1="Java Programming Language";

System.out.println(s1.startsWith("Java"));//true

System.out.println(str.startsWith("Java"));// false

**startsWith(String, int):**

Syntax:boolean startsWith(String startString, int startIndex);

checks for the specified string is existing in string object with specified start index.

Example:

String str="Hello Java";

String s1="Java Programming Language";

System.out.println(s1.startsWith("Java",5));//false

System.out.println(str.startsWith("Java",6));//true

**String toLowerCase(): -**

This method converts the given string into lower case.  
Example:

String str="Hello JAVA";

System.out.println(str.toLowerCase());//returns the converted lowercase string 'hello java'

**String toUpperCase(): -**

This method converts the given string into uppercase  
Example:

String str="Hello Java";

System.out.println(str.toUpperCase());// returns 'HELLO JAVA'

**String trim(): -**

Returns the given String excluding the spaces before and after the string.

Example:

String str=" Hello Java ";

System.out.println(str.length());// 12

String str1=str.trim();

System.out.println(str1);//Hello Java

System.out.println(str1.length());//10

**String substring(): -**

This method returns the string or sequence of characters starts from the index position specified in the parameter.  
Example:

String str="Hello Java";

System.out.println(str.substring(4));// returns 'o Java'

**substring(int startIndex, int endIndex): -**   
This method returns the string or sequence of characters in between the start index position and end index position specified in the parameter.  
Example:

String str="Hello Java";

System.out.println(str.substring(2,8));// returns 'llo Ja'