Basic Web UI test

I can imagine the first thing you will want to do is to write a very simple and basic test - to heat up the engines. This page will guide you through the code:

```
@Execution(ExecutionMode.CONCURRENT)
```

Click here to expand...

This line tells the JUnit engine that the tests in this class were designed to run in parallel. Note! In order for this line to work, there is a property file named junit-platform.properties located at src/test/resources folder

```
public class BasicWebExampleTests extends BaseTest{
```

Click here to expand...

We extend the class BaseTest. BaseTest is a class that was created in the examples repo. It mainly holds the annotations and keeps this code clean.

```
@DriverCapabilities
```

Click here to expand...

There are 3 main annotations you will need to be familiar with in order to get your driver up without any issues: `@DriverCapabilities`, `@DriverUrl`, and `@DriverCommandExecutor`.

The `@DriverCapabilities` tells the VeriSoftDriver where to find the capabilities to use when starting up. We use dependency injection for injecting the WebDriver directly into the test. More on our dependency injection mechanism for the driver can be found here.

```
private DesiredCapabilities capabilities = new DesiredCapabilities();
    {
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--no-sandbox");
        options.addArguments("--headless");

        capabilities.setBrowserName("chrome");
        capabilities.setCapability("browserVersion", "113");
        capabilities.setCapability("driverVersion", "113");
        options.merge(capabilities);
    }
```

Click here to expand...

Capabilities are pretty straightforward. Just note that the framework will look for `platformName` for web and mobile testing. `platformName` in this example comes from the ChromeOptions object.

```
@Test
@DisplayName("Search Wikipedia test")
```

Click here to expand...

Both lines are JUnit lines. The `@DisplayName` line is very important since if you use this annotation, then most of the report mechanisms will use it as the test name, rather than the method name.

```
public void searchWikipedia(VerisoftDriver driver)  {
```

Click here to expand...

We use a dependency injection mechanism, which takes the `@DriverCapabilities, @DriverUrl`, and `@DriverCommandExecutor` (if present) and creates an instance of `VerisoftDriver`. The `VerisoftDriver` object is an instance of `WebDriver` and follows its interfaces and more. More about the VeriSoftDriver and VerisoftMobileDriver can be found here.

The next few lines are very basic Selenium code, so let's fast-forward a bit

```
// Note!! Verisoft Assert
Asserts.assertTrue(driver.getTitle().contains(phraseToAssert),
                "Page should contain the pharase " + phraseToAssert);
```

Click here to expand...

Asserts in the Verisoft framework do pretty much the same as regular asserts. The difference here is that we log the message to the various logging and reporting mechanisms that are listening to out Observer Report mechanism. More about the Report in the following line. You may choose to use a different Assert mechanism (and there are plenty of those in Java, but you will lose this nice logging feature.

```
Report.info("Got to this point - We are on the right page");
```

Click here to expand...

There are many places we want to write reports to - logging, report mechanism, perhaps a global logging system like Splunk, Jira,, etc. If we would write a line of code for each of the mechanisms we support, we would have more logging lines of code than actual code. So, we use the Observer pattern and use one single Report object to write to all of them. More about the Report object here.

Well, that's it! We made it through a basic web UI test. Here is the complete code, and you can also view it on Bitbucket

```
@Execution(ExecutionMode.CONCURRENT)
public class BasicWebExampleTests extends BaseTest{

    @DriverCapabilities
    private DesiredCapabilities capabilities = new DesiredCapabilities();
    {
        ChromeOptions options = new ChromeOptions();
        options.addArguments("--no-sandbox");
        options.addArguments("--headless");

        capabilities.setBrowserName("chrome");
        capabilities.setCapability("browserVersion", "113");
        capabilities.setCapability("driverVersion", "113");
        options.merge(capabilities);
    }


    @Test
    @DisplayName("Search Wikipedia test")
    public void searchWikipedia(VerisoftDriver driver)  {
        driver.get("https://www.wikipedia.org/");
        driver.findElement(By.id("searchInput")).sendKeys("Test Automation");
        new Actions(driver).sendKeys(Keys.ENTER).build().perform();

        String phraseToAssert = "Test automation";

        // Note!! Verisoft Assert
        Asserts.assertTrue(driver.getTitle().contains(phraseToAssert),
```

```
                            "Page should contain the pharase " + phraseToAssert);

        Report.info("Got to this point - We are on the right page");
    }
}
```