TA Guidelines

This document contains information and guidelines on how to use the test automation framework to setup your own project.

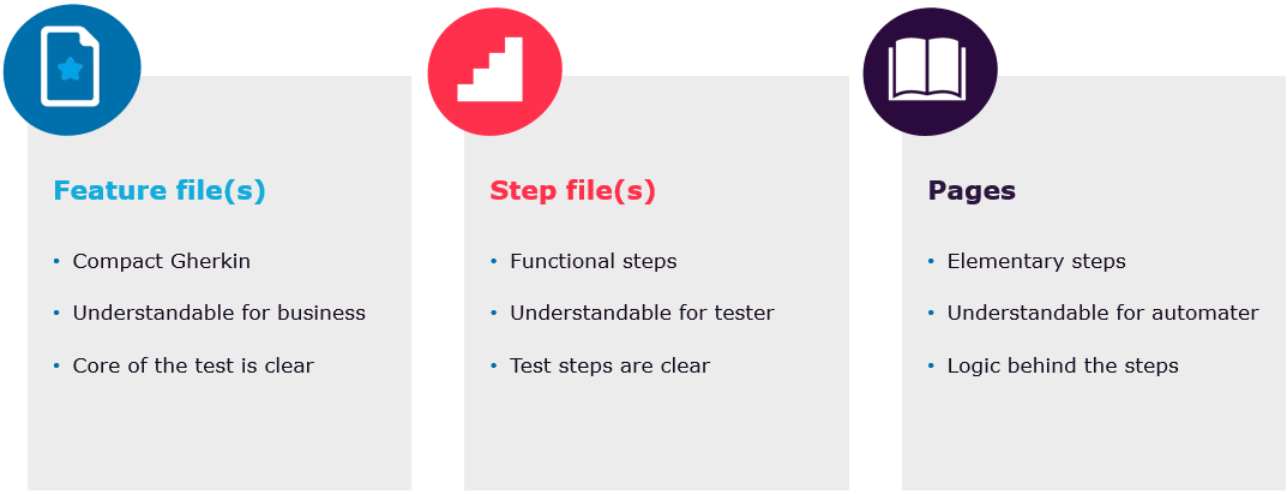
Architecture

All the java code written for test automation can be found within *src/test/java/com*. In this folder the java files have been categorized into a folder where the general framework is contained (*capgemini*) and a folder for the project specific files (*project*). For all further purposes of this documentation only the *project* folder is of relevance.

The *project* folder should contain the folders:

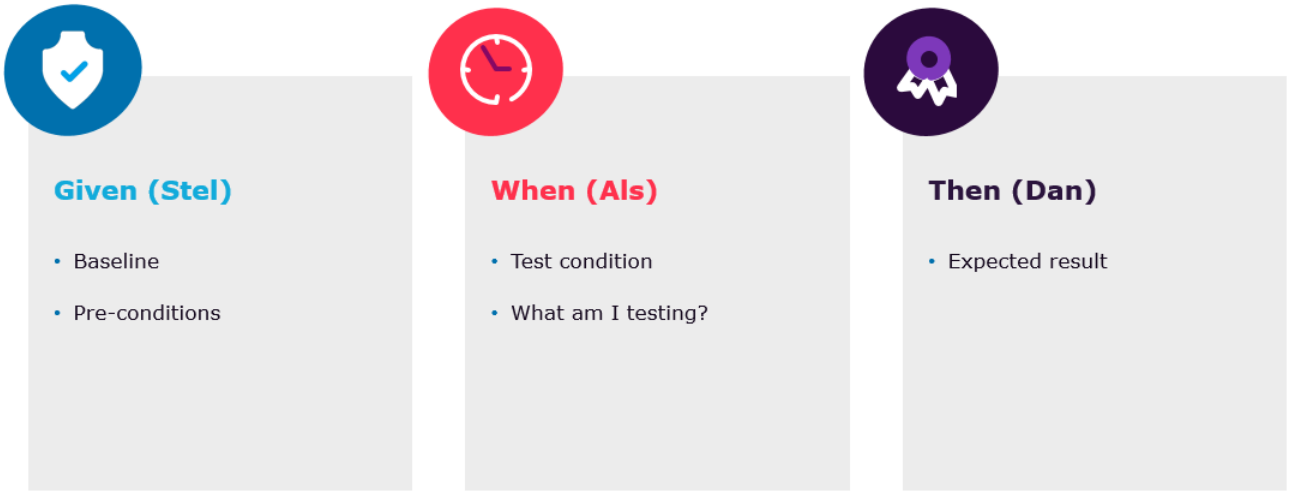
- features
- steps
- pages
- resources

As the content of the *project* folder suggests, the automated test has different levels: feature level, step level and page level. The *resources* folder contains the configuration files (subfolder *config*) and the objects/data types to help run the test. The following figure gives a summary of the different levels, where feature files are the highest level and pages are the lowest level.



Features

The features folder contains `.feature` files per functionality. Each feature contains different scenarios to test this functionality. Thus each scenario is a test case. Scenarios consist of a scenario name and Gherkin. Each scenario should clarify the core of the test and be readable for everyone. So a scenario should have compact Gherkin. The following figure shows some guidelines for proper Gherkin.



Steps

Each line of Gherkin in a scenario corresponds with a function in a step file. This works with hooks from Selenium Cucumber. Within each function in a step file, functional steps can be found. These functional steps are all just calls to functions on the page level.

Each step file has a corresponding feature file. That means that if you want to create a new step file: first create a feature file, add a Scenario and let IntelliJ create a new corresponding step file for you. I recommend creating a BaseSteps java file which can be extended by all your step files. In this BaseSteps file you can then add resources/pages which can be shared amongst the other step files.

Pages

Each functional step in the step files corresponds to a function on a page file. At the page level de elementary steps are taken such as "*click element x*", "*fill field y*" and "*wait for page to load*".

I recommend creating a BasePage and adding the browser and base functions to it. Every page file that extends the BasePage can interact with the *browser* and access the base functions defined in BasePage. An example of some elements of a BasePage can be found in the coding guidelines below.

Coding guidelines

- Selectors

  All elements on a page with which user interaction is needed must have selectors defined as such: *Name of element on screen + element type* + "Selector".

  Furthermore, all columns/elements selectors of resulting datatables must begin with "result".

  Selector code examples

```
private final By arrondissementListSelector =
By.cssSelector("select[name$=':'it9']");
private final By datumVanafInputSelector =
By.cssSelector("input[name$=':id4']");
private final By tonenButtonSelector =
By.cssSelector("[id$=':buttonTonen']");

private final By resultBegintijdSelector =
By.cssSelector("[id$=':c3']");
```

- BasePage
  The code example below gives an idea of how one might setup a BasePage. This is just a short example, but you can add as many functions as you want to the BasePage which can be used by all pages which extend it.

```
public class BasePage{

 //By making an object protected it is now accessible by all pages
that extend the BasePage
    protected OurWebDriver browser;

 private final By usermenuDropDownSelector =
By.cssSelector("[id$=':usermenu']");
 private final By logoutButtonSelector =
By.cssSelector("[id$=':pt_cmi1']");

 /**
   * Constructor for the BasePage.
  */
 public BasePage(){
     this.browser = BrowserFactory.getWebDriver();
 }

 /**
   * The logout function is available at all times
  */
 public void uitloggen(){
     browser.findElement(usermenuDropDownSelector).click();
     browser.waitForElement(logoutButtonSelector).click();
 }
}
```