2.Page UI Map

Introduction

A collection of page sets, which in turn contain UI elements. The UI map is the medium for translating between UI specifier strings, page elements, and UI elements.

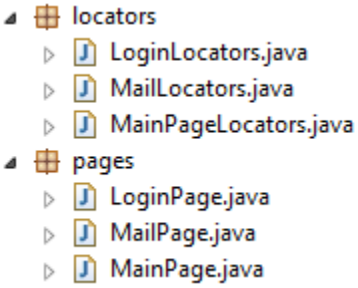
UI Map

The main idea of a UI Map is to group together all the locators of a given page, the same as we group together the methods for a given page for Page Objects.

There are two approaches for this idea:

1. Separate package and classes
   Construct a new package in the current project structure and call it "Locators" for example.
   Create a separate class for each Page Object class.
   Group in each class the element locators from that given page.

   The structure may look like this:

   A Locator class will contain the element identification. Example:

   ```
   /*Class that defines the set of locators for Login page*/
   public class LoginLocators {
    public static By usernameField = By.id("username");
    public static By passwordField = By.name("passwd");
    public static By loginBtn = By.id(".save");
   }
   ```

2. The same class as the Page Object class
   Another way, not that recomanded is to define the UI Map in the same class as the Page Object. In the short term development this can be a good approach, but for the future, when multiple objects and locators are defined, it will be harder to handle it in this way. If there are some parametrized variables that can be used for multiple Page Objects, this model won't help.

   Example:

```
/*Login Page Object class that groups together the page main actions and locators*/
public class LoginPage {

 public static By usernameField = By.id("username");
 public static By passwordField = By.name("passwd");
 public static By loginBtn = By.id(".save");

 public LoginPage enterUsername(String username){
  PageDriver.fillText(usernameField, username);
  return this;
 }
 public LoginPage enterPassword(String password){
  PageDriver.driver.findElement(passwordField).sendKeys(password);
  return this;
 }
 public LoginPage authenticate(){
  PageDriver.driver.findElement(loginBtn).click();
  return this;
 }
}
```

In constructing the UI Map there are some cases on different projects where the UI Map can be constructed with parametrized variable, and just reuse the defined UI Map for different process and the variable here would be the name of the element.

For example, in case where we can define a general UI locator for a given object, let's say a button, and call the method in different pages with a different button name:

```
/*general used button*/
 public static By button(String btnName) {
  return By.xpath("//div[@class='aq-container-content']/following::input[@value='" +
btnName + "']");
 }

//call the button in a Test class where Submit is the name of the button
button("Submit");
```