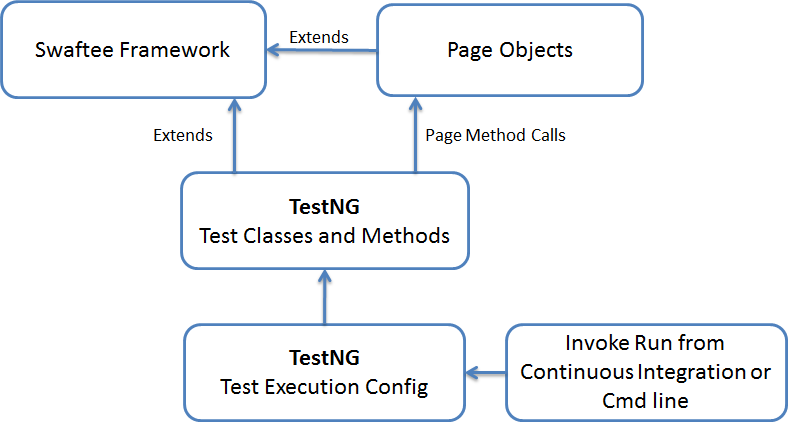
1. **SWAFTEE** -> Selenium Wrapper for Automating Frontend Testing and End to End
   * Automation Framework which is for across all the teams of XOME.
   * Basic framework -> Page Object Design pattern -> Skelton structure.
   * This is the lowest level of abstraction built above Selenium using its WebDriver API.
   * It contains all basic classes w.r.t Page Objects & Test scripts.
   * It is having basic functionalities and basic methods.
     + The above functionalities and methods are shared across team.
   * By using this framework we are creating automation scripts.
   * For reporting we are using zephyr API's.
2. **Java** & **Selenium** for Automation framework.
   1. multi platform support(desktop, mobile), cross browser support and enables us to program in different languages - java, python, perl, ruby etc
   2. Cascade and Liberty applications are automated using Selenium-TestNG in Java.

****

1. **TestNG Annotations**
   1. @Jira -> Will give testcaseID as TC
   2. @Test -> will give test description as description
   3. By using Priority parameter we can priorate the execution order.
      1. Whenever if you are executing a class this method will be executed first based upon the Priority.
   4. One test can be mapped to different test cases.
   5. Each testcases are built independently so there is NO dependent with other TC’s.
2. **Page Objects Repo:**
   * Separate class for each page.
   * Even separate class for each Tab.
   * Each page classes inherits Homepage.
   * Homepage is customized for specific application (Login Page -> Role change etc) and it extends AppPage.
3. **Page Test:**
   * Test classes for each page.
   * Each Test classes inherits BaseTest.
   * BaseTest is customized for specific application(Login Test, Waits, System time etc) and it extends AppTest.
4. **AppTest / AppPage**
   * + It is the Base classes for all teams of XOME.
     + We build our Page classes and Test Classes based upon this.
     + It have Skelton functions so we make a use of it in our automation scripts.
     + AppPage -> Any Page class should inherit AppPage.

* initElements -> Initialize all the elements.
* Take screenshots
* All typesWaits
* Any new method also we can include in AppPage so that other teams also can utilize it.

1. **DB connections:**
   1. Connect DB
   2. Database connection
   3. DOMXml Parser
2. **zephyr API's**
   1. It is rest assured API
   2. It will communicate to Jira and update the test result(Pass/Fail/Blocked).
   3. It will calls in ‘After Suite’ method in **AppTest** class.
   4. UpdateExecutionStatusInJira -> TestCase ID will be updated with execution status using TestCyleID.
   5. generatehtmlreport for emailing the report for a job.
3. **TestConstants:**
   1. All the constants are stored here w.r.t individual application.
4. **POM**
   1. To store the Repo for Page Objects and Test classes.
   2. Jar files handled here.
5. **SpotSlave Machines**
   1. on-demand machines
   2. When we run the job on-demand it will allocate the machine and start running.
6. **Automation code Branch maintaining:**
   1. Master will be updated one.
   2. We have separate branches individually.
   3. Each time once local changes done in individual branches then pull and Push the latest changes to Master branch.
   4. Master branch only we will run the execution so it will be latest and update one.
7. **Test Suits:**
   1. E2E Testing -> Every release
   2. Regression Testing -> Every release
   3. Prod Sanity Testing -> Daily
   4. BVT Testing -> Post Release