**Automation Framework 1 – shen – June 21, 2018**

**Framework :**

**it is a structure that supports the software**

**Architecture**

**framework is like ground work**

**it should be applicable to any domain**

**in general its a concept**

**template or system**

**execution environment for automation**

**example of real time frameworks**

**highways**

**maps**

**any mall – you are here feature, helps in navigation**

**A test automation framework is a collection of interacting components facilitating the creation and execution of automated tests and the reporting of the results thereof.**

**Why do we need a framework?**

* **Design – framework – plug n play**
* **Easy to maintain - Minimizes the time needed to maintain**
* **Reusable code (avoid redundancy)**
* **Provision for new additions**
* **Improves readability, reusability**
* **Provides organized way or guideline on how to write the tests.**

**Advantages of a good framework:**

* **Minimizes errors**
* **Improves reporting**
* **Synchronization is possible**
* **Environment setup (changes)**
* **Allows multiple configurations**

**Common Components of AF**

* **Testing library**
* **Libraries for performing multiple test (end-to-end testing)**
* **Libraries that support BDD**
* **Stubs, mocks, virtual assets**
* **Solid test data approach**
* **Reporting mechanism**
* **Logging of error messages**

**Patterns generally used in an AF**

* **Wrapper classes(user defined classes) for reuse**
* **Centralized error handling**
* **Define classes with abstraction**
* **Allow inheritance (Reusable code)**

**Key components FW**

**Build(maven/ant) , source control(git/bitbucket/svn), continuous integration tool(Jenkins)**

**How to build a FW thumb rules**

1. **Determine what is needed?**
2. **What tests does the framework need to support?**
3. **What are the requirements with regards to reporting?**
4. **Who is responsible for creating tests and their level of expertise?**

**Types of Framework :**

**Data Driven**

**Keyword driver**

**Modular testing**

**Linear scripting**

**PageObject Model**

**Hybrid**

**BDD – Behavior driven development**

**TDD – test driven development**

**Common Components of AF**

* **Testing library : for reports, dependencies, mainly for different types of testing smoke, sanity, integration, regression**

**Group the test, priortise the test, provides mechanism to manage dependent test, enable or disable a test**

* **Libraries for performing multiple test (end-to-end testing)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirementid** | **Changerequirement**  **Yes/no RTM** | **Risk** | **testcaseid** | **Type of test** | **Automation ye/no** |
|  |  |  |  |  | |
|  |  |  |  |  | |

* **Libraries that support BDD**

**Cucumber , - features, based on the features test cases will be run**

**User acceptance criteria –BDD framework helps stake holders to under the different tests written**

**In this case BDD will be driving the tests**

* **Stubs, mocks, virtual assets :**

**Stubs : they are dummy services which provide same results**

**Mock services : programmatically we can handle the dependent method to be called ----junit mockito api ….when(shipping called) –then(run your own method) return success**

**Virtual assets : record and play services**

**Solid test data approach: test data preparation**

**Tester created data, production data, data from data analytics**

|  |  |  |
| --- | --- | --- |
| **username** | **password** | **expected** |
| **abcd** | **hjk** | **success** |
| **342352** |  | **failure** |

* **Reporting mechanism**

**Testng, itestlistener,…. Screenshot**

* **Logging of error messages Apache Log4j**
* **For class follow the appropriate design pattern**

**https://www.journaldev.com/1827/java-design-patterns-example-tutorial**

* **Dependency injection**
* **Util class – common classes**
* **Class for report**