Framework Part 4 - Test Strategy- Control Tests Execution- Run Parallel Tests

next topics of data driven testing and parameterization.

 I am going to show some more extension with how to drive the data from JSON.

JSON become very famous in driving the test data from external sources than typical Excels or CSV files. JSON is some light weighted.

before that, first let's start with data provider - data provider does in testNG, that is basically help you to drive the data and pass the multiple data sets.

In the submitOrderTest class – I created one method Get data, which is responsible to feed the data for any test case.

So if you give annotation(@) called data provider, so TestNG automatically thinks that

this guy will give the data what needed for all the tests in that particular class."

let's say you want to run this test with the two different data sets ?

So if you want to run the two sets, then simply create an two dimensional array ( new object [] [] { { } , { } } )which accepts multiple sets. inside this curly brace you will write values

for one dataset. We want two data sets that;s why we given 2 curly brances.

So your method, wherever you want to data drive that method receives that two dimensional array. And in the first iteration it'll take first dataset and it'll run the test. And again, it'll pick the another dataset and it'll run the test again. So that means it will run two times with two different data sets when you send this two dimensional array back to your method.

…object is something which is apparent data type for all this. And it's a generic data type which accepts any kind of data type.

Req : So basically you are saying I have two different data login sets, I want to run my test with two different sets.

we will attach this data provider to the test. How to attach? @Test(dataProvider=”getData”)

---- THIS is new method in the submitorderTest we send 2 different data sets two login

@DataProvider

Public object[][] getData()

{

Return new object[][] {{“hari@gmail.com”,”hari777”,”ZARA COAT 3”}, {“QW@gmail.com”,”231998”, “ADIDAS ORIGINAL”}}

}

In main submit Order Test

@Test(dataProvider=”getData”,groups={“Purchase”})

Public void submitorder(String email, string password, strong productName)

Copy the errorvalidationtest and paste it in the testsuites ( change the name to Purchase )

Then run this file

173. Integration of Hashmap to Data provider to send the data as one Hash object

there are three parameters in data provider(email,password,productname) what you are sending to your test. So you are catching those three parameters in the method here. But tomorrow if you are test to have 15 parameters, what will you do?

…data provider allowed to return hash map

You can also send hash maps inside this arrays. Now instead of sending this data here, so put all this in key value pairs and directly send the hash map.

we are sending any value, It could be object string, doesn't matter. So I'm placing object comma object. So that means, I'm clearly telling my key value pair can be any data type.

@DataProvider

Public object[][] getData()

{

HashMap<String,String> map = new HashMap<String,String>();

map.put(“email”, “[hari@gmail.com](mailto:hari@gmail.com)”);

map.put(“password”,”hari777”);

map.put(“product”, “ZARA COAT 3”);

HashMap<String,String> map1 = new HashMap<String,String>();

Map1.put(“email”, “[QW@gmail.com](mailto:QW@gmail.com)”);

Map1.put(“password”,”231998”);

Map1.put(“product”, “ADIDAS ORIGINAL”);

Return new object[][] {{map}, {map1}};

}

In submitorderTest

Public void submitorder(HashMap<String,String> input)

… input.get(“email”),input.get(“password”)

174. How to read the data from Json files and create the list of Hashmaps for testing

 how to drive this data from JSON ?

create new package in src/test ( rahulshettyacademy.data)

inside this package create one new file(PurchaseOrder.json)

now you write your own JSON in the above created file.

[

{

“email” : “[hari@gmail.com](mailto:hari@gmail.com)”,

“password”: “hari777”,

“product”: “ZARA COAT 3”

},

{

“email” : “[shetty@gmail.com](mailto:shetty@gmail.com)”,

“password”: “Iamking@000”,

“product”: “ADIDAS ORIGINAL”

}

]

So now write one utility which actually scans this JSON and create hash maps out of it.

So now manually we created hash maps here. If you observe two hash maps we created,

So instead these hash maps now should automatically create by scanning this JSON.

This is an external data file what we push in our project and you have to write some method which can scan this JSON and create hash maps out of it. So once you have hash maps in place, then you can nicely plug in here.

we know that data is stored now in JSON file.

Now let me create one utility. So in the same package I am crating class file ( DataReader)

So here you can write “n” number of utilities to make scan your JSON or to retrieve the values based on your requirement.

In Java there is one method which will read file. That means if you just pass JSON file,

it will scan the entire content of your JSON and convert that into one string variable.

 if you see that it is not detected, then go ahead and include that dependency

in your pom dot XML file. with the existing dependencies what we have,

that is automatically included.

to convert this string to hash map, then with existing dependencies, what you have,

it's not possible. For that you have to go and get a new dependency called Jackson data bind. This is the one of the dependency which can help you to convert the spring content into hash map.

Mvn repo -> Jackson datablind -> copy the dependency -> paste it in pom.xml file

In Data Reader class

**public** List<HashMap<String, String>> getJsonDataToMap() **throws** IOException

{

// read json to string

String jsonContent = FileUtils.~~readFileToString~~(**new** File(System.*getProperty*("user.dir")+//src//test//java//rahulshettyacademy//data//PurchaseOrder.json),

StandardCharsets.***UTF\_8***));

//string to HashMap - jackson Datablind

ObjectMapper mapper = **new** ObjectMapper();

List<HashMap<String, String>> data = mapper.readValue(jsonContent, **new** TypeReference<List<HashMap<String, String>>>(){

});

**return** data;

}

Copy all the above lines in basetest and modify the below line in the base test

**public** List<HashMap<String, String>> getJsonDataToMap(String filePath) **throws** IOException

{

// read json to string

String jsonContent = FileUtils.*readFileToString*(**new** File(filePath), StandardCharsets.***UTF\_8***);

In submit order test class

@DataProvider

**public** Object[][] getData() **throws** IOException

{

List<HashMap<String,String>> data = getJsonDataToMap(System.*getProperty*("user.dir")+"//src//test//java//rahulshettyacademy//data//PurchaseOrder.json");

// retrive the item for the list

**return** **new** Object[][] {{data.get(0)}, {data.get(1)}};

}

175. How to Create Screenshot Utility in Base Test class for catching Failed tests

 how to build a screenshot utility. whenever a test fails, it should automatically take a screenshot. And then we should implement one reporting, so that screenshot will attach to that our HTML report.

So to take a screenshot first you have to cast your driver to screenshot mode,

Driver is something which have control on your browser and you want to take a screenshot out of it, So you can cast a screenshot to this driver object.

In the BaseTest class – create one method

**public** String getScreenshot(String testCaseName) **throws** IOException

{

TakesScreenshot ts = (TakesScreenshot)driver;

File Source = ts.getScreenshotAs(OutputType.***FILE***);

File file = **new** File(System.*getProperty*("user.dir")+"//reports//"+ testCaseName +".png");

FileUtils.*copyFile*(Source, file);

**return** System.*getProperty*("user.dir")+"//reports//"+ testCaseName + ".png";

}