Dependency Injection using PICO Container

Say we are doing web browser testing using Selenium in Cucumber. We have a feature file with several scenarios to test. We will write a ‘Step Definition’ for the feature file. You create a test runner and run the tests.

Now if we have many scenarios that involve many pages. It is better to include separate step definition files on each page. We follow something called **“Single Responsibility Principle”**. In this principle, we write one step definition class for just one web page.

Sometimes we may want to share data between the steps. To achieve that dependency Injection is done.

**Dependency injection** is a design pattern in which an object or function receives other objects or functions that it depends on. It is a programming technique that makes a class independent of its dependencies.

Cucumber Pico-container dependency helps to implement Dependency Injection.

Add the below dependency:

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-picocontainer</artifactId>

<version>7.18.1</version>

</dependency>

Let us consider this example for verifying Account Information.

Here, we will be creating an account, and after creating an account, we will verify the details in Account Information.

We will have to share the data between the steps. In this case, let us say the First Name and Last Name entered during the account creation step should be shared with verify Account information step. Here we have to perform dependency injection.

In order to perform dependency injection,

1. Create a Context class to create scenario context

**public** **class** Context {

HashMap<String,String> stepData;

**public** Context() {

stepData = **new** HashMap<>();

}

**public** **void** setContext(String key, String value) {

stepData.put(key, value);

}

**public** String getContext(String key) {

**return** stepData.get(key);

}

}

1. To store the data steps in context we have to add a constructor in all the page step definition classes.

**private** Context context;

**public** UserRegistrationSteps(Context context) {

**this**.context = context;

}

---------------------Setting context--------------------------------

context.setContext("firstname", fName);

context.setContext("lastname", lName);

----------------------------Getting Context ------------------------------

String fname = context.getContext("firstname");

String lName = context.getContext("lastname");