

Test Debugging Documentation

Owner: Adithya Solai (adithyasolai7@gmail.com)

Setup

Make sure to have your VSCode's launch.json file configured properly as described in the Software Setup documentation. Your VSCode's launch.json file should look something like this:

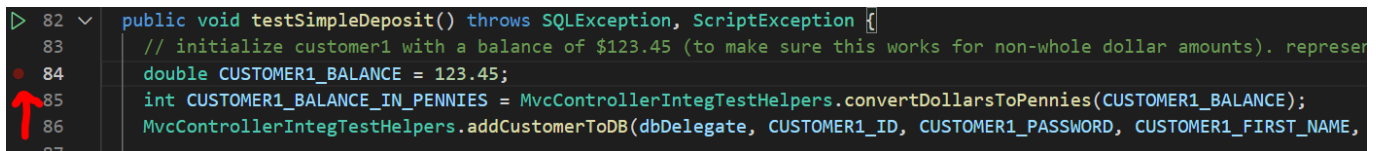
```
1  {
2    "version": "0.2.0",
3    "configurations": [
4      {
5        "type": "java",
6        "name": "TestudoBank Tests",
7        "request": "attach",
8        "hostname": "localhost",
9        "port": 8000
10     },
11     {
12       "type": "java",
13       "name": "Launch Current File",
14       "request": "launch",
15       "mainClass": "${file}"
16     },
17     {
18       "type": "java",
19       "name": "Launch TestudoBankApplication",
20       "request": "launch",
21       "mainClass": "net.codejava.TestudoBankApplication",
22       "projectName": "SpringBootFormHandling"
23     }
24   ]
25 }
```

Setting Breakpoints

First, decide if you would like to debug a test in the Unit Test suite

(`src\test\java\net\testudobank\tests\MvcControllerTest.java`) or the Integration Test suite (`src\test\java\net\testudobank\tests\MvcControllerIntegTest.java`).

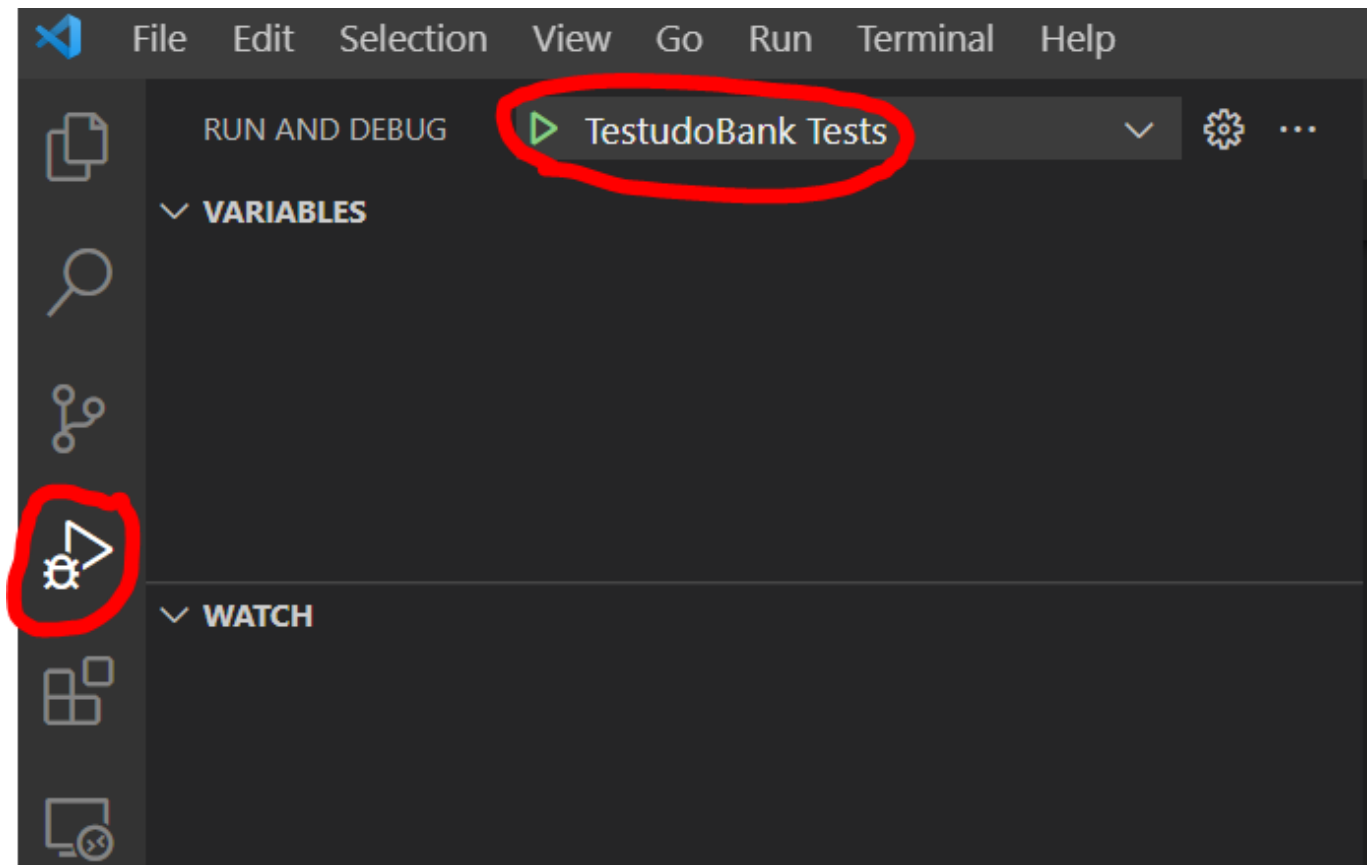
Set a breakpoint in that test's code at the desired spot you would like to start debugging by clicking your mouse to the left of the line as indicated below:

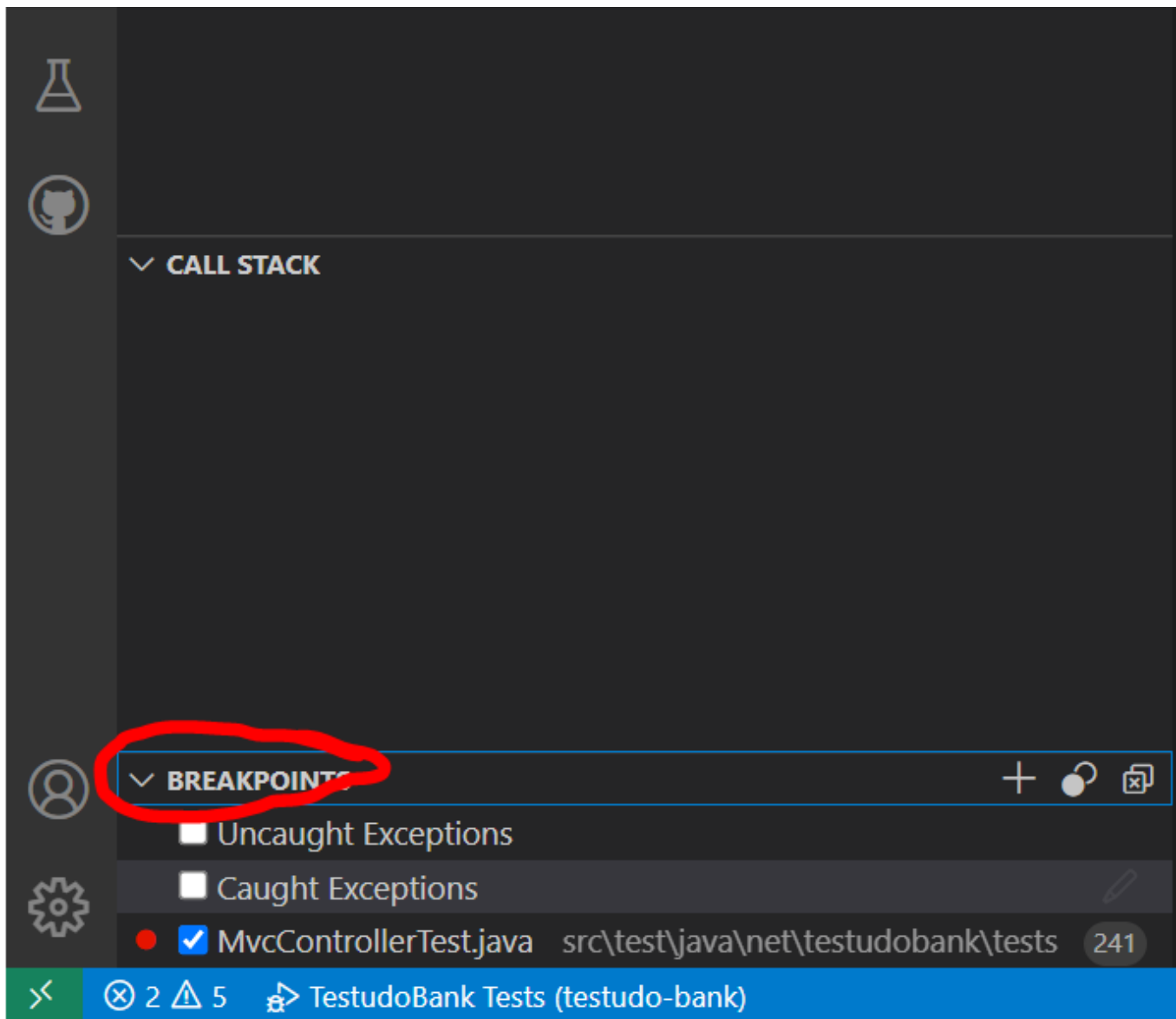


```
82 public void testSimpleDeposit() throws SQLException, ScriptException {  
83     // initialize customer1 with a balance of $123.45 (to make sure this works for non-whole dollar amounts). represent  
84     double CUSTOMER1_BALANCE = 123.45;  
85     int CUSTOMER1_BALANCE_IN_PENNIES = MvcControllerIntegTestHelpers.convertDollarsToPennies(CUSTOMER1_BALANCE);  
86     MvcControllerIntegTestHelpers.addCustomerToDB(dbDelegate, CUSTOMER1_ID, CUSTOMER1_PASSWORD, CUSTOMER1_FIRST_NAME,  
87
```

Running the Debug Session

Navigate to the Run & Debug tab on VSCode's dashboard at the left. It should now look like the image below:





Make sure the “TestudoBank Tests” configuration is selected at the top. Make sure your breakpoints are visible in the bottom tab.

Then, run one of the following commands in your terminal to set up the Debugger listener.

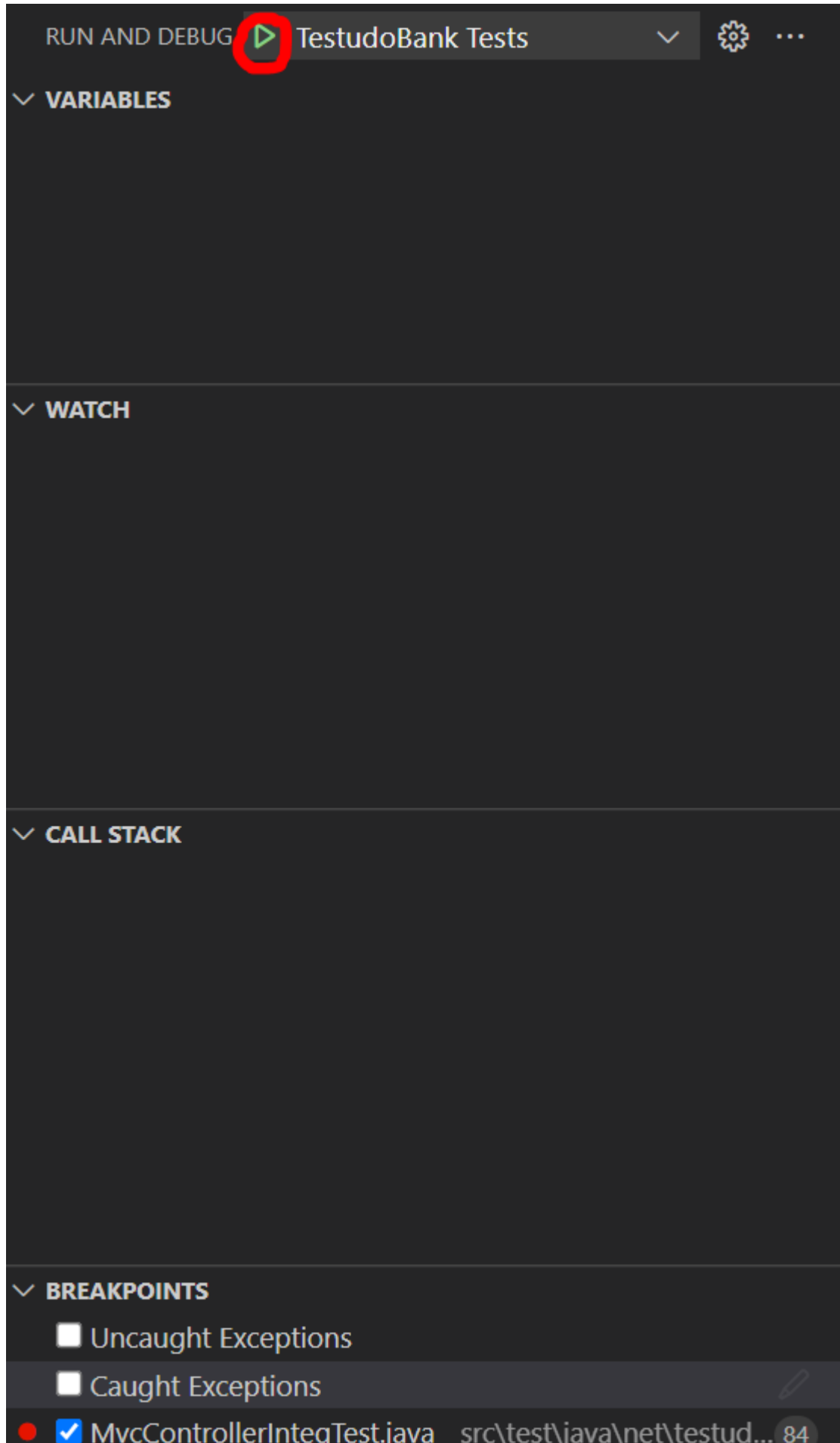
If you are debugging a Unit Test: `mvnDebug -DforkMode=never test -Dtest=MvcControllerTest`

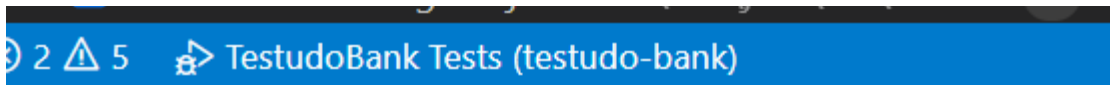
If you are debugging an Integration Test: `mvnDebug -DforkMode=never test -Dtest=MvcControllerIntegTest`

You should see the following output in your terminal:

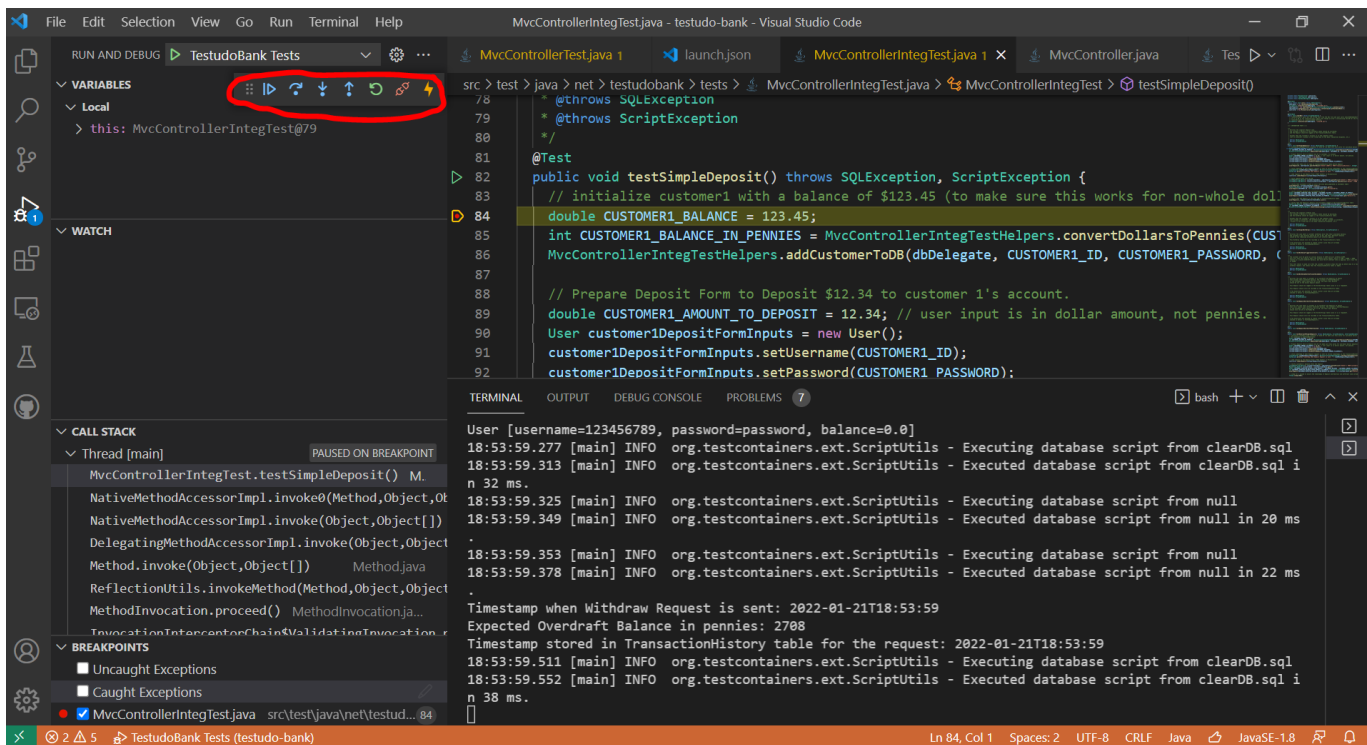
```
pratb@DESKTOP-FSG5HP4 MINGW64 ~/CMSC389G-STIC/testudo-bank (adithyasolai/integration-tests)
$ mvnDebug -DforkMode=never test -Dtest=MvcControllerIntegTest
Preparing to execute Maven in debug mode
Listening for transport dt_socket at address: 8000
```

Finally, navigate back to the Run & Debug tab, and click the Green Run Button at the top:





Wait a few seconds for Maven to reach your breakpoint while it is running the entire test suite. Your screen should look like this afterwards:



Debugging Tools

Once you are in the Debug Mode shown in the above image, you have some very useful tools at your disposal.

Use the Toolbar highlighted in Red in the image above to **Step Over** and **Step Into** lines of code. The Step Into feature is extremely useful for diving into function calls to determine exactly where a test breaks.

Another useful tool is right-clicking inside the “Watch” tab on the left dashboard and selecting “Add Expression”. With this feature, you can add expressions that will be re-calculated each time you Step Over a line of code. This is useful when you want to track how the contents of certain variables change over the course of a test. An example expression would be something like

`(int)customer1Data.get("Balance")` to verify the contents of an SQL query result in an integration test.

The “Variables” tab on the left dashboard is also very useful. It will re-populate automatically with local variables and their contents every time you Step Over a line of code. This is another easy way to track the contents of variables over the course of your test code.