An Insider Look at BASIS Testing

The biggest bane of software development is releasing a product before it has been properly tested or containing an array of bugs caught only after the release. At BASIS, our number one priority is producing the best product that we possibly can. Although we use many tools to achieve this goal, the most crucial is our test suite that we designed specifically to test our code changes at all levels of production to ensure that our products are released with the highest quality. We run the suite automatically after a new build compiles, which begins within 8 minutes of any checkin to the SVN source code archive. Our current test suite has two parts - JUnit and testbed.

BASIS uses JUnit as our first level of defense to ensure that any new code development does not alter or affect other parts of the code currently working and operating within BBj® Services. JUnit, originally written by Erich Gamma and Kent Beck, provides an open source tool that comes standard with Java and is a simple framework for writing repeatable tests. JUnit is an instance of the xUnit architecture for testing frameworks and contains three distinct parts - assertions for testing expected results, test fixtures for sharing common test data, and test runners for running tests. By running the repeatable tests before we check in any code, JUnit proves that the new code integrated with the existing code will not produce any unexpected errors.

The second level of defense, better known as the testbed, contains the test server and client that we designed to establish that the actual product is fit for general release. The test server is responsible for running BBj Services, displaying the GUI, and executing the BBj test code. The test client's sole responsibility is to log and record the information obtained while the test server runs the code tests. This level of testing is similar to the way BASIS developers test their own BBj applications.

So how do we actually use these tools? Well, the first step is to compile BBj Services so that it runs properly on all the platforms that we support. Specifically, we take the code that we checked in during our "first level" tests, compile it on the oldest supported platform version, and then run BBj Services on the newest supported platform. For example, Microsoft's oldest supported OS is Windows XP while its newest released OS is Windows 8. Therefore, we compile on Windows XP and run BBj Services on Windows 8. The same would apply to Linux, UNIX, Solaris, and so on. This practice ensures that BBj Services will not require the latest OS specific features and tags that would limit backward compatibility. As a result, BBj Services can run on all platforms supported by the operating system vendor.

The second step in our testing process is to write the BBj code that will execute the new features. If you are thinking we just write a simple program to test the new additions to our product, you are half right. The first BBj program usually tests the new feature itself using our product documentation as a guide. The next several tests integrate the new feature within our old tests to increase the level of complexity. In fact, many of the BBj test programs appear as code samples in the online BASIS documentation.

Finally, we run BBj Services on the newest supported platforms with our test BBj code. If the tests have a user interface, the UI will appear on the test server with the test client logging the results of the tests. We then analyze the logs and review for corrections. After making any necessary changes, we rerun the tests and repeat the process until it is error-free.

Why go through of all this effort? This meticulous testing allows us to see the actual code and controls in action on the screen as if we were running it in a real-world production environment. As a company in the modern marketplace, BASIS is committed to producing the most modern, up-to-date, cutting edge software, and strives to include modern programming features. We take great pride in what we do and the test suite ensures that we can accomplish our goals while continuing to meet yours. ■

By Aaron Wantuck
Software Engineer

For more information on
- JUnit - refer to junit.org or download it from bit.ly/7ESsE1
- Writing JUnit tests - visit bit.ly/SSaSh2