Currently MBF.Tests & MBF.TestAtuomation make web service calls using MBF framework. If service is not available MBF throws a failure exception because which all corresponding cases fail.

The idea here is to introduce a mock layer b/w MBF.Tests, MBF.TestAutomation and MBF. This new layer depending on the availability of the service either gets the required output from real world or generates a mock output object from the output repository and returns it.

MBF.Tests

Or

MBF.TestAutmation

MBF

# Mock Layer

Mock Layer

Test case schema

Test case simulator

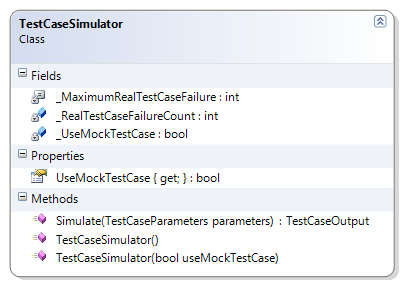
Output Repository

(Binary format)

## Test case simulator

### TestCaseSimulator

Takes a test case object and executes it, this class implements failover logic for web test cases. This class provides flexibility to execute a set of test cases in mock/real case scenario. If real case fails, it failovers and return mock output. And if count of failed test case is more than maximum limit, it permanently switches to mock case scenario.



|  |  |
| --- | --- |
| **Name** | **Description** |
| Simulate | * If “UseMockTestCase” is ‘true’   + Get the mock result (object) from “MockOutputRepository”.   + Create an instance of “TestCaseOutput” with mock result (object).   + Return instance of “TestCaseOutput”. * Else   + Execute the real world function pointer and get the result (object).     - If execution fails (On exception. Only WebException & SoapException?)       * Increment “\_RealTestCaseFailureCount”       * If “\_RealTestCaseFailureCount” > “\_MaximumRealTestCaseFailure”, set \_UseMockTestCase to true.       * Get the mock result (object) from “MockOutputRepository”.   + Create an instance of “TestCaseOutput” with output result (object).   + Return instance of “TestCaseOutput”. |
| UseMockTestCase | Gets a value indicating whether to use a mock test case. |
| \_RealTestCaseFailureCount | Tracks the count of real test case that has failed. |
| \_MaximumRealTestCaseFailure (constant) | Maximum number of real test failures allowed before permanently switching to execution of mock test case executions (3). |

Usage snippet:

\*

TestCaseSimulator \_TestCaseSimulator = new TestCaseSimulator();

// FetchResultsSyncTest is unique test case identifier.

// TestNcbiQBlastDna is a function pointer to real test case.

TestCaseParameters parameters = new TestCaseParameters("FetchResultsSyncTest", null, TestNcbiQBlastDna, testCaseParameters);

object resultsObject = \_TestCaseSimulator.Simulate(parameters).Result;

Assert.IsNotNull(resultsObject);

// List<BlastResult> is expected output type

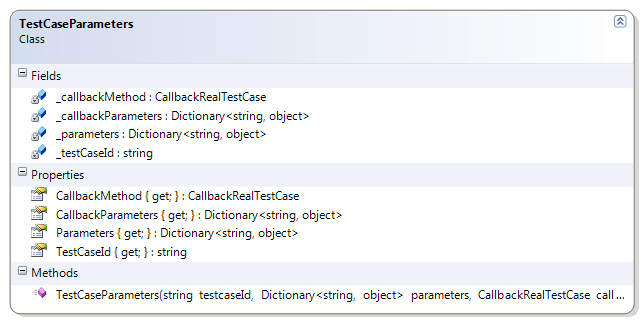
List<BlastResult> results = resultsObject as List<BlastResult>;

// Validation of actual output goes …

\*

### TestCaseParameters

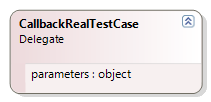
This class contains the parameters required by the simulator to execute a test case.



|  |  |
| --- | --- |
| **Name** | **Description** |
| CallbackMethod | Real world test case callback method (Invoker passes a function pointer that has to be called to execute a real world scenario). |
| CallbackParameters | Gets real test case parameters. |
| Parameters | Gets test case parameters. |
| TestCaseId | Gets unique test case identifier. |

### CallbackRealTestCase

This delegate represents a function pointer to be called to execute a real test case scenario. Take an object as parameter and returns an object as output.



\*

public delegate TestCaseOutput CallbackRealTestCase(Dictionary<string, object> parameters);

E.g.,

static TestCaseOutput TestNcbiQBlastDna(Dictionary<string, object> parameters)

{

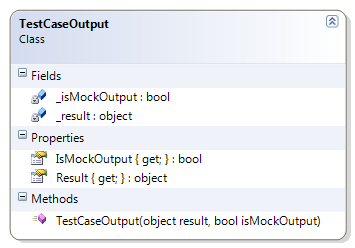
// Write real test code here.

}

\*

### TestCaseOutput

This class contains the output of test case executed by TestCaseSimulator.



|  |  |
| --- | --- |
| **Name** | **Description** |
| IsMockOutput | Gets a value indicating whether the output is mock. |
| Result | Gets output of the test case. |

## Test case schema

Schema containing the map b/w testcaseid and the corresponding output file in Output Repository.

<TestCases>

<TestCase>

<TestCaseID></TestCaseID>

<MockOutputPath></MockOutputPath>

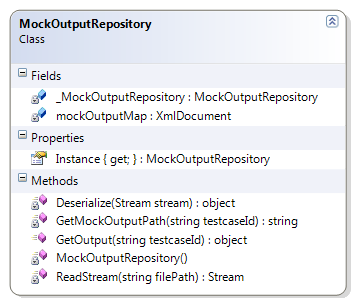
</TestCase>

</TestCases>

## Output Repository (Binary Format)

### MockOutputRepository

This class represents a repository of mock output object. On request reads maps a testcaseId to output file path, deserialize the stream in file and return the mock output object. This class implements a singleton pattern.



|  |  |
| --- | --- |
| **Name** | **Description** |
| GetOutput | 1. Find the path of file containing the mock output of testcaseid using Test case schema. 2. Read the output stream 3. Deserialize the stream to ‘object’. 4. Return the object. |
| Instance | Gets the singleton instance of MockOutputRepository. |

Any object that can go in this repository must implement Serialization at the framework level.

Note: Currently BlastResult does not implement serialization.

# Assumption

1. A test case will test one unit of functionality.

E.g., Execution of blast service requires

* Submitting a request.
* Polling for status.
* Read the output.
* Parse the output.

The above sets of operations are executed in a single unit to get the BlastResult as output.

The callback delegate will do all the above required operations but in mock scenario, none of the above will be executed, it just reads the mock output from a source (file) and returns it.