Consolidate Tests by Making Them Data-Driven

Paul D. Sheriff

Business/IT Consultant PDS Consulting

psheriff@pdsa.com www.pdsa.com

Module Overview

- Create a table to hold test data
- Use data from table for testing
- Add code to base class
  - Connect to table to retrieve data
  - Build a DataTable
- Loop through DataTable rows

Data-Driven Testing

Eliminates creating multiple test methods

Create a table to hold inputs and output

Retrieve rows from table

Use test data (input and output) to call method

```sql
CREATE SCHEMA tests;

CREATE TABLE
  tests.FileProcessTest

(
    FileName varchar(255) NULL,
    ExpectedValue [bit] NOT NULL,
    CausesException [bit] NOT NULL
)
```

- Create schema for test tables
- File Process Table
- File name to test
- Expected value (true or false)
- Causes an exception?

Demo

Create test table

Add data

```
public void FileExistsTestFromDB() {
  LoadDataTable(sql, conn);


  foreach(row in DataTable) {
    // Test each row of data
    // Record success or failure
  }
  if(failure) {
    Assert.Fail();
  }
}
```

- Create new test method
- Load a DataTable from SQL Server

- Loop through all rows
- Test each row of data
- Record success or failure

- If any one test fails
- Fail the test

Demo

Create data-driven method

MyClasses.runsettings

```
<Parameter name="ConnectionString"
value="Server=Localhost;Database=Sandbox
;Integrated Security=Yes" />
```

◀ **Add connection string**

FileExistsTestFromDB() Method

```
string conn = TestContext.Properties

   ["ConnectionString"].ToString();
```

◀ **Retrieve connection string**

Demo

Add connection string to runsettings file

Module Summary

- Data-driven tests can eliminate many methods
- Add more tests without modifying code
- Use try...catch for exceptions
- Record messages with data values

Up Next:

Automating Unit Tests with Command Line Utility