black

Quick Start Guide

black is a library that simplifies the process of working with FitNesse/SLiM . It provides an API for manipulating Query Tables and Table Tables in a more intuitive way.
It also takes the heavy lifting out of sharing data between FitNesse tables.

Query Table

When you have a list of domain objects that represent the results of a query, you can't simply return them from a Query Table fixture. You need to transform the domain objects into a complex data structure consisting of a List of Lists of Lists.

The black QueryTable can take a list of domain objects and transform it into the format required by FitNesse/SLiM.

```
List<AnyCLass> objects = AListOfAnyClass();

var table =
  new QueryTable<AnyClass>().For(objects);
```

That's all there is to Query Tables in black. This exact code will work for Query Tables, Ordered Query Tables, and Subset Query Tables without any change.

Table Table

The FitNesse/SLiM TableTable is a flexible way of representing any table layout, including rows with different numbers of cells.

Like the Query Table, the Table Table uses a complex arrangement of Lists of Lists to represent the table and it's data.

To work with a Table Table in your fixture you need to read the List based representation, and to return results you need to produce the same list based data structure.

The black TableTable object converts FitNesse/SLiM tables into a representation that allows direct manipulation of cells, and allows conversion back into the FitNesse/SLiM format for returning results.

In the doTable method of your TableTable fixture, create a black TableTable passing it the table received from FitNesse.

```
var table = new TableTable(fromFitNesse);
```

Accessing Rows and Cells

You can then access the rows within the table

```
TableRow row = table.Rows[0];
```

The cells within a row

```
TableCell cell = row.Cells[0];
```

Access a cell directly

```
TableCell cell = table.Cells(1,1);
```

Access a value in a cell directly

```
String cellValue = table.CellValues(1,1);
```

Check and Flag Results

Cells can be marked as Pass/Fail/Ignore/Report/Error

```
cell.Pass(<message>);
cell.Fail(<message>);
cell.Ignore(<message>);
cell.Report(message);
cell.Error(message);
```

Cells can be checked against an existing value, and the Pass/Fail status set automatically

```
cell.CheckAgainst(value);
```

Shared Tables

A common pattern in FitNesse is the creation of data in one table that serves as an input for another table. E.g. a table of business objects that are the data on which a QueryTable will operate.

Fixtures need to use static data e.g. Singletons. black provides an easy mechanism for creating, working with, and if necessary disposing of shared tables.

Create or get the instance of a table for storing objects of type AnyClass.

```
var table =
  SharedTables.Table<AnyClass>();
```

Add a domain object to a shared table

```
table.Add(
  new AnyClass { AnyProperty = "value" }
);
```

Dispose of a shared table

```
SharedTables.DisposeOf<AnyClass>();
```

Tables can also be accessed via an alias, allowing multiple tables of the same type of object

```
var table =
  SharedTables.Table<AnyClass>("Alias");
```

Dispose of all shared tables

```
SharedTables.DisposeOfAll();
```