WEB システムのテストの自動化

attool

Ver. 1.10

Datagram Ltd.

2014 年 12 月 4 日

概 要

当資料は、WEB システムのテストの自動化の具体例を記述しています。

改版履歴

Ver. 1.00 2014 年 12 月 3 日
新規作成

Ver. 1.10 2014 年 12 月 4 日
改修

目 次

1　概要

1.1　現状の問題点

JUnit や PHPunit は、名前の Unit のとおり、単体テスト向けのものです。大規模なシステムですと単体試験ツールは有用ですが、中規模以下のシステムでは、余計な時間や作業を要してしまいます。また、いくら単体部品が良くても、それを組み合わせた後では、状況が異なりますので、テストが必要です。

1.2　対象用途

当資料で紹介する試験ツール（attool）は、以下の用途を対象としています。

- 結合試験
- 総合試験

attool は、人間の操作を基本としていますので、ソースコードに試験用のコードを加える必要はありません。

1.3　attool の仕組み

通常、WEB システムでは、人間が WEB ブラウザを操作して処理を遂行しています。当該試験ツール（attool）では、インラインフレーム内に被試験対象の WEB システムを表示させることで、JavaScript による自動テスト化を実現しています。

2　被試験対象の WEB 画面

図 1: 被試験対象の WEB 画面

図 1 は、被試験対象の WEB 画面です。テキスト入力エリアが 2 つあります。その入力エリアに値を入れて送信ボタンをクリックすると、図 2 のようになります。現在時刻と入力された値が下部に表示されます。

3　被試験対象 WEB 画面のソースコード

被試験対象 WEB 画面のソースコード (test01.php) は次のようになっています。

```
1  <?php
2  $txt01 = (empty($_POST['txt01'])) ? '(empty)' : $_POST['txt01'];
3  $txt02 = (empty($_POST['txt02'])) ? '(empty)' : $_POST['txt02'];
4  $dt    = date('Y-m-d H:i:s');
5  $stat  = 'Status : ' . $dt . ' : ' . $txt01 . ' : ' . $txt02 . ';';
6  ?>
7  <!DOCTYPE html>
8  <html>
9  <head>
```

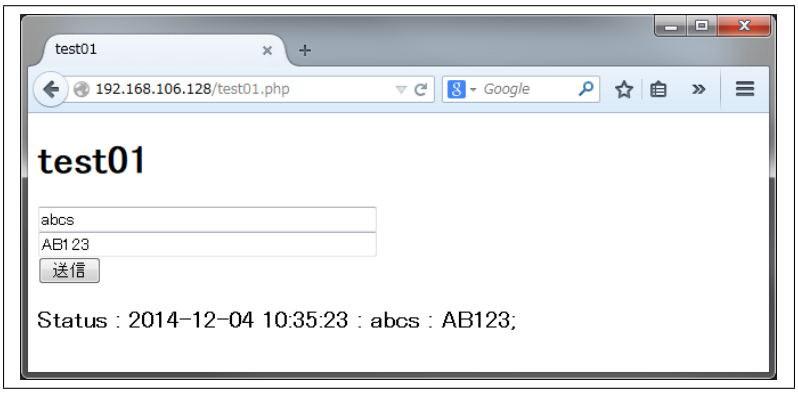

図 2: 被試験対象の動作

```
10  <title>test01</title>
11  </head>
12  <body>
13  <h1>test01</h1>
14  <form id="frm01" name="frm01" action="/test01.php" method="post">
15  <input id="txt01" name="txt01" type="text" size="40"
16   value="<?php echo $txt01; ?>" />
17  <br />
18  <input id="txt02" name="txt02" type="text" size="40"
19   value="<?php echo $txt02; ?>" />
20  <br />
21  <input id="btn01" name="btn01" type="submit" value="送信">
22  </form>
23  <div>
24  <h3><?php echo $stat; ?></h3>
25  </div>
26  </body>
27  </html>
```

フォーム、入力エリアそしてボタンは ID で指定できるようになっています。

4 試験ツール attool

4.1 試験ツールのデータ例

試験ツールのデータ例 (testdata.js) は次のようになっています。

```
 1  //
 2  // Test Data
 3  //
 4  var td    = {
 5                ttl: 'Test for test01.php',
 6                man: true,
 7                url: 'test01.php',
 8                exe: {
 9                    0: {
10                        cnt: 2,
11                        dat: ['txt01', 'txt02'],
12                        val: ['abcd', 'ABCD'],
13                        frm: 'frm01'
14                    },
15                    1: {
16                        cnt: 2,
17                        dat: ['txt01', 'txt02'],
18                        val: ['efgh', 'EFGH'],
19                        btn: 'btn01'
20                    },
```

```
21                  2: {
22                      cnt: 1,
23                      dat: ['txt02'],
24                      val: ['Finish'],
25                      btn: 'btn01'
26                  }
27              }
28          };
29
30  // end of line
```

“man” を true にしますと、人間の目で確認しやすいように確認ウィンドウが表示さます。“man” を false にしますと、自動的に一気に処理を進めます。“url” は初期 URL です。“exe” 項目は配列になっています。順番（シーケンシャル）に処理を進めます。“cnt” は入力項目数を表しています。“dat” は ID 名です。“val” は入力される値です。“frm” は submit 対象のフォーム ID 名です。“btm” はクリックされるボタンの ID 名です。

4.2 試験ツール（attool）のソースコード

試験ツール（attool）のソースコード (attool.php) は次のようになっています。

```
 1  <?php
 2  $uid = date('Ymdhis');
 3  ?>
 4  <!DOCTYPE html>
 5  <html>
 6  <head>
 7  <title>attool</title>
 8  </head>
 9  <body>
10  <div>
11  <div id="ttl">title area</div>
12  <iframe id="tst01" name="tst01" onload="onl01()" width="100%" height="1000"></iframe>
13  </div>
14  <script src="./testdata.js?uid=<?php echo $uid; ?>"></script>
15  <script type="text/javascript">
16
17      var tst01 = document.getElementById('tst01'),
18          ttl   = '<?php echo $uid; ?> : ' + td['ttl'],
19          url   = td['url'],
20          seq   = 0,
21          msg   = '';
22
23      function prc01() {
24
25          var d = tst01.contentDocument, e = td['exe'][seq];
26
27          for (var i = 0; i < e['cnt']; i++) {
28              d.getElementById(e['dat'][i]).value = e['val'][i];
29          }
30
31          if (e['frm']) {
32              d.getElementById(e['frm']).submit();
33          } else if (e['btn']) {
34              d.getElementById(e['btn']).click();
35          }
36
37          seq++;
38      }
39
40      function exe_man() {
41          if (td['exe'][seq]) {
42
43              var e = td['exe'][seq];
44
45              for (var i = 0; i < e['cnt']; i++) {
```

```
46                  msg += e['dat'][i] + '=' + e['val'][i] + "\n";
47              }
48
49              if (e['frm']) {
50                  msg += e['frm'] + '.submit()' + "\n";
51              } else if (e['btn']) {
52                  msg += e['btn'] + '.click()' + "\n";
53              }
54
55              if (confirm(msg + "\n" + 'OK?')) {
56                  msg = '';
57                  prc01();
58              }
59          } else {
60              alert('The end.');
61          }
62      }
63
64      function exe() {
65          if (td['exe'][seq]) {
66              prc01();
67          }
68      }
69
70      function onl01() {
71          if (td['man']) {
72              exe_man();
73          } else {
74              exe();
75          }
76      }
77
78      document.getElementById('ttl').innerHTML = ttl;
79      tst01.src = url;
80
81  </script>
82  </body>
83  </html>
```

testdata.js に書かれている内容を実行するプログラムです。

5 実行例

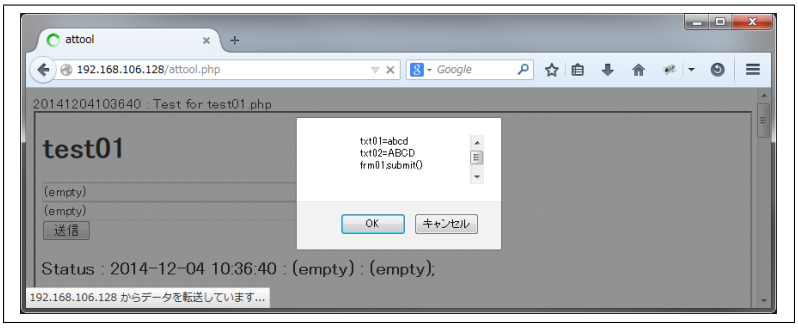

図 3: 実行例#1

図 3 から図 6 は testdata.js の内容を実行したものです。“man” が true なので確認ウィンドウが表示されています。実行する前に設定される値が表示されます。

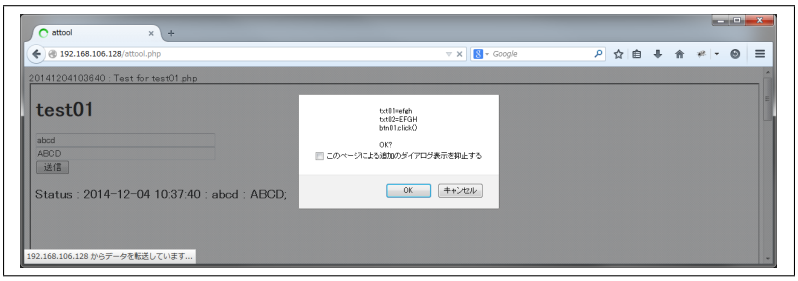

図 4: 実行例#2

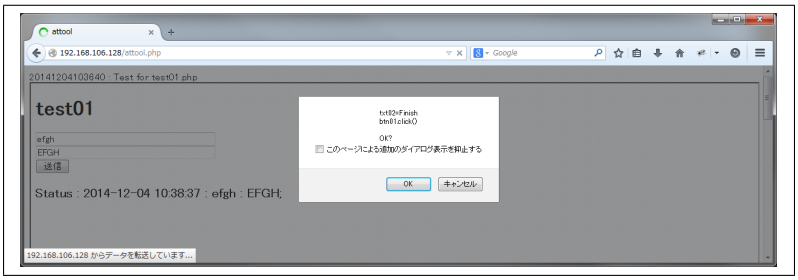

図 5: 実行例#3

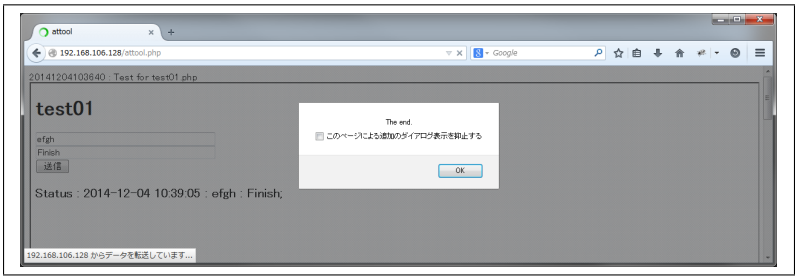

図 6: 実行例#4

6 考察

6.1 人間の目での確認 or ログでの確認

“man” が true の場合、実行結果は人間の目で確認できます。“man” が false の場合、実行結果は、おそらくログデータや DB の状態を診て判断されることと思います。

6.2 PDCA サイクル

WEB 開発は必ず変更や修正が入ります。この場合、試験をやり直す必要があります。また、修正が一回では済まないことが多々あります。このツールを使うことで PDCA サイクル[1] が速く回ります。ソースコードに変更が入った場合や再試験の時に、当ツールは役立つと思います。

6.3 負荷試験／耐久試験

“man” を false にした場合、目にも止まらぬ速させ処理が続行されます。これはサーバ側の大きな負担となります。負荷試験にも最適です。またこれは疲れ知らずなので、耐久試験にも使用可能です。

[1] PDCA cycle : plan-do-check-act cycle