

```

1  @inherits SpecFlow.Plus.Runner.Reporting.CustomTemplateBase<TestRunResult>
2
3  @using System
4  @using System.Collections.Generic
5  @using System.Linq
6  @using System.Globalization
7  @using TechTalk.SpecRun.Framework
8  @using TechTalk.SpecRun.Framework.Results
9  @using TechTalk.SpecRun.Framework.TestSuiteStructure
10 @helper GetReportBar(TestItemExecutionResult test)
11 {
12     <td class="testview-item"
13         data-sr-exectime="@GetSeconds(test.ExecutionTime)"
14         data-sr-rescode="@((int)test.ResultType)" data-sr-order="@test.ExecutionOrder"
15         data-sr-acctime="@GetSeconds(test.ActExecutionTime)">
16         <a
17             class="bar @("color" + test.ResultType)"
18             style="height: @GetBarSize(tr => tr.ExecutionTime.DurationMilliseconds,
19             test, 0.0, 60)px;"
20             title="@GetTestBarTooltip(test)"
21             href="#@GetTestAnchor(test)">&nbsp;</a>
22     </td>
23 }
24 @helper GetTimelineBar(DateTime startTime, DateTime endTime, double msecPerPixel,
25 TestItemExecutionResult test)
26 {
27     int endPixel = Math.Max((int)Math.Round((endTime -
28         Model.ExecutionTime.StartTime).TotalMilliseconds / msecPerPixel), currentPixel + 4);
29     int size = endPixel - currentPixel;
30     currentPixel = endPixel;
31     <td>
32         <a
33             class="bar @(test == null ? "startupBar" : "color" +
34             test.ResultType.ToString())"
35             style="width: @(size - 1)px;" 
36             @if (test != null)
37             {
38                 @:title="@GetTestBarTooltip(test)"
39                 @:href="#@GetTestAnchor(test)"
40             }
41             else
42             {
43                 @:title="test thread startup"
44             }
45         >&nbsp;</a>
46     </td>
47 }
48 @helper GetSummaryHeader(string titleHeader, bool showDuration = false)
49 {
50     <tr>
51         @if (titleHeader != null)
52         {
53             <th>@titleHeader</th>
54         }
55         <th colspan="2">Success rate</th>
56         <th>Tests</th>
57         <th>Succeeded</th>
58         <th>Failed</th>
59         <th>Pending</th>
60         @if (showDuration)
61         {
62             <th>Duration</th>
63         }
64     </tr>
65 }
66 @helper GetSummaryRow(TestCollectionResultSummary summary, string title, string href,
67 TimeSpan? executionDuration = null)
68 {
69     <tr>
70

```

```

63         @if (title != null)
64         {
65             <td><a href="#@href">@title</a></td>
66         }
67     @RenderTestExecutionSummaryRowTail(summary, executionDuration)
68 </tr>
69 }
70
71 @helper RenderBar(TestNodeResultType testNodeResultType, int count, int total,
72 Func<TestNodeResultType, string> titleFactory)
73 {
74     <td>
75         <a class="bar @("color" + testNodeResultType)"
76             style="width: @GetPixelBarWidth(total, count)px;" 
77             title="@titleFactory(testNodeResultType)"
78             @if (testNodeResultType.GetGroup() == TestNodeResultTypeGroup.Failure)
79             {
80                 @:href="#error_summary"
81             }
82         ></a>
83     </td>
84 }
85
86 @helper RenderSummaryBars(IEnumerable<KeyValuePair<TestNodeResultType, int>> data, int
87 total, Func<TestNodeResultType, string> titleFactory)
88 {
89     <table class="timelineview" cellpadding="0" cellspacing="0">
90         <tr>
91             @foreach (var resultCount in data)
92             {
93                 @RenderBar(resultCount.Key, resultCount.Value, total, titleFactory)
94             }
95         </tr>
96     </table>
97 }
98
99 @helper RenderTotalSummaryRowTail(TestCollectionResultSummary summary, TimeSpan?
100 executionDuration = null)
101 {
102     @RenderSummaryRowTail(summary.Total, summary.ResultCounts, summary.TotalMessage,
103     summary.GetText, executionDuration)
104 }
105
106
107 @helper RenderTestExecutionSummaryRowTail(TestCollectionResultSummary summary, TimeSpan?
108 executionDuration = null)
109 {
110     @RenderSummaryRowTail(summary.Total, summary.TestExecutionResultCounts,
111     summary.TotalMessage, summary.GetText, executionDuration)
112 }
113
114 @helper RenderSummaryRowTail(int total, IDictionary<TestNodeResultType, int>
115 resultCounts, string totalMessage, Func<TestNodeResultType, string> getText, TimeSpan?
116 executionDuration = null)
117 {
118     int succeeded = resultCounts.Where(rc =>
119         rc.Key.IsInGroup(TestNodeResultTypeGroup.Success)).Sum(rc => rc.Value);
120     int failed = resultCounts.Where(rc =>
121         rc.Key.IsInGroup(TestNodeResultTypeGroup.Failure)).Sum(rc => rc.Value);
122     int pending = resultCounts.Where(rc =>
123         rc.Key.IsInGroup(TestNodeResultTypeGroup.Pending)).Sum(rc => rc.Value);
124
125     <td class="percentage">
126         @if (succeeded + failed + pending > 0)
127         {
128             @:@GetRoundedSuccessPercentage(succeeded, failed, pending)%
129         }
130     else
131     {
132         @:n/a

```

```

121         }
122     </td>
123     <td class="bartd">
124         @RenderSummaryBars(GetOrderedBarChartData(resultCounts), total, getText)
125     </td>
126     <td>@totalMessage</td>
127     <td>@succeeded</td>
128     <td>@failed</td>
129     <td>@pending</td>
130     if (executionDuration != null)
131     {
132         <td>@executionDuration.Value</td>
133     }
134 }
135
136 @helper TestItemLinks(TestItem testItem, int level)
137 {
138     if (level == 0)
139     {
140         <a href="#@GetTestNodeAnchor(testItem, "t", 0)">@testItem.Type:
141         @testItem.Title</a>
142     }
143     else
144     {
145         <a href="#@GetTestNodeAnchor(testItem, "t", 0)">@testItem.Title</a>
146     }
147     var tiResult = GetTestItemResult(testItem);
148     if (tiResult != null)
149     {
150         foreach (var retry in tiResult.Executions.Skip(1))
151         {
152             <a href="#@GetTestNodeAnchor(testItem, "t",
153             retry.TestItemExecutionIndex)">retry #@retry.TestItemExecutionIndex</a>
154         }
155     }
156     @helper TestNodeLinks(TestNode testNode, int level)
157     {
158         if (testNode is TestItem)
159         {
160             @TestItemLinks((TestItem)testNode, level)
161         }
162         if (testNode is TestCollection)
163         {
164             <span>@testNode.Type: @testNode.Title</span>
165             <ul class="subNodeLinks">
166                 @foreach (var subTestNode in ((TestCollection)testNode).SubNodes)
167                 {
168                     <li>
169                         @TestNodeLinks(subTestNode, level + 1)
170                     </li>
171                 }
172             </ul>
173         }
174     @functions
175     {
176         double GetRoundedSuccessPercentage(int succeeded, int failed, int pending)
177         {
178             double absolute = succeeded + failed + pending;
179             double percent = succeeded / absolute;
180             double scaledRoundedPercent = Math.Round(percent * 100);
181             return scaledRoundedPercent;
182         }
183         string GetFixtureTitle(TestNode fixtureNode)
184         {
185             return fixtureNode.IsDefaultTestTarget ? fixtureNode.Title : string.Format("{0}
186             (target: {1})", fixtureNode.Title, fixtureNode.TestTarget);
186         }

```

```

187
188     TimeSpan CalculateDuration(TestNode testNode)
189     {
190         TimeSpan executionDuration = TimeSpan.Zero;
191
192         var testNodeResults = Model.TestExecutionResults.Where(tr =>
193             tr.TestItemResult.TestNode == testNode);
194         foreach (var testNodeResult in testNodeResults)
195         {
196             executionDuration += testNodeResult.ExecutionTime.Duration;
197         }
198
199         return executionDuration;
200     }
201
202     double GetPixelBarWidth(double total, double value)
203     {
204         return Math.Round((value * 200 / total) - 1);
205     }
206
207     IEnumerable<KeyValuePair<TestNodeResultType, int>>
208     GetOrderedBarChartData(IEnumerable<KeyValuePair<TestNodeResultType, int>> source)
209     {
210         return source.Where(rc => rc.Value > 0).OrderByDescending(rc =>
211             rc.Key.GetGroup() == TestNodeResultTypeGroup.Success ? 1000 : (int)rc.Key);
212     }
213
214     @section ProjectInformation
215     {
216         <ul>
217             <li>Start Time: @Model.ExecutionTime.StartTime</li>
218             <li>Duration: @Model.ExecutionTime.Duration</li>
219             <li>Test Threads: @Model.TestThreads.Count</li>
220             @if (Model.FrameworkError != null)
221             {
222                 <li><div class="errorMessage">Execution framework error:<br/>
223                     @ (Model.FrameworkError.ToString())</div></li>
224             }
225         </ul>
226     }
227
228     @section TestResultView
229     {
230         <h2>Test Result View</h2>
231         <div id="testview" class="viewbox">
232             <div id="bar-control">
233                 <div id="bar-control-sort">
234                     <label>sort by:</label>
235                     <span class="option"><input type="radio" name="barSortOrder" value="execetime" />Time</span>
236                     <span class="option"><input type="radio" name="barSortOrder" value="acttime" />Act Time</span>
237                     <span class="option"><input type="radio" name="barSortOrder" value="order" />Execution</span>
238                     <span class="option"><input type="radio" name="barSortOrder" value="rescode" checked="checked" />Result</span>
239                     <span class="option"><input type="checkbox" name="barSortDesc" id="barSortDesc" checked="checked" />desc</span>
240                 </div>
241                 <div id="bar-control-heights">
242                     <label>heights:</label>
243                     <span class="option"><input type="radio" checked="checked" name="barHeight" value="execetime" />Time</span>
244                     <span class="option"><input type="radio" name="barHeight" value="acttime" />Act Time</span>
245                 </div>
246             </div>
247         </div>
248         <table class="vertical-scale" cellpadding="0" cellspacing="0">

```

```

245
246     <tr class="scale-max">
247         <td class="left-padding scale-max-label">&nbsp;</td>
248         <td colspan="@Model.TestExecutionResults.Count()">&nbsp;</td>
249         <td class="right-padding">&nbsp;</td>
250     </tr>
251     <tr class="scale-mid">
252         <td class="left-padding scale-mid-label">&nbsp;</td>
253         <td colspan="@Model.TestExecutionResults.Count()">&nbsp;</td>
254         <td class="right-padding">&nbsp;</td>
255     </tr>
256     <tr class="scale-min">
257         <td class="left-padding scale-min-label">&nbsp;</td>
258         <td colspan="@Model.TestExecutionResults.Count()">&nbsp;</td>
259         <td class="right-padding">&nbsp;</td>
260     </tr>
261 </table>
262 <div class="scrollable">
263     <table class="testview-items" cellpadding="0" cellspacing="0">
264         <tr class="testview-items-row">
265             <td class="left-padding">&nbsp;</td>
266             @foreach (var test in Model.TestExecutionResults.OrderBy(tr =>
267                 tr.ResultType))
268             {
269                 @GetReportBar(test);
270             }
271             <td class="right-padding">&nbsp;</td>
272         </tr>
273         <tr class="horizontal-scale">
274             <td class="left-padding">&nbsp;</td>
275             <td colspan="10">&nbsp;</td>
276             @for (int test10Index = 1; test10Index <
277                 Model.TestExecutionResults.Count() / 10; test10Index++)
278             {
279                 <td class="scale-10-label" colspan="10">@((test10Index * 10))</td>
280             }
281         </tr>
282     </table>
283 </div>
284 </div>
285 }
286 @section FeatureSummary
287 {
288     <h2>Feature Summary</h2>
289     <table class="testEvents">
290         @GetSummaryHeader("Feature", true)
291
292         @foreach (var fixtureNode in GetTextFixtures())
293         {
294             var fixtureSummary = GetSummary(fixtureNode);
295             string fixtureTitle = GetFixtureTitle(fixtureNode);
296             string testNodeAnchor = GetTestNodeAnchor(fixtureNode, "f");
297
298             var executionDuration = fixtureNode.SubNodes.Aggregate(TimeSpan.Zero, (acc,
299                 testNode) => acc + CalculateDuration(testNode));
300
301             <tr>
302                 <td><a href="#@testNodeAnchor">@fixtureTitle</a></td>
303                 @RenderTestExecutionSummaryRowTail(fixtureSummary, executionDuration)
304             </tr>
305         }
306     </table>
307 }
308 <!DOCTYPE html>
309 <html>
310     <head>
311         <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
312         <title>@Model.Configuration.ProjectName Test Execution Report</title>
313         <script type="text/javascript"
314             src="http://code.jquery.com/jquery-1.6.2.min.js"></script>

```

```
310 <script type="text/javascript">
311     /**
312      * jQuery.fn.sortElements
313      * -----
314      * #author James Padolsey (http://james.padolsey.com)
315      * #version 0.11
316      * #updated 18-MAR-2010
317      * #url
318      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
319      * -----
320      * #param Function comparator:
321      *   Exactly the same behaviour as [1,2,3].sort(comparator)
322      *
323      * #param Function getSortable
324      *   A function that should return the element that is
325      *   to be sorted. The comparator will run on the
326      *   current collection, but you may want the actual
327      *   resulting sort to occur on a parent or another
328      *   associated element.
329      *
330      * E.g. $('td').sortElements(comparator, function() {
331          return this.parentNode;
332      })
333      *
334      * The <td>'s parent (<tr>) will be sorted instead
335      * of the <td> itself.
336      */
337     jQuery.fn.sortElements = (function () {
338
339         var sort = [].sort;
340
341         return function (comparator, getSortable) {
342
343             getSortable = getSortable || function () { return this; };
344
345             var placements = this.map(function () {
346
347                 var sortElement = getSortable.call(this),
348                     parentNode = sortElement.parentNode,
349
350                     // Since the element itself will change position, we have
351                     // to have some way of storing it's original position in
352                     // the DOM. The easiest way is to have a 'flag' node:
353                 nextSibling = parentNode.insertBefore(
354                     document.createTextNode(''),
355                     sortElement.nextSibling
356                 );
357
358                 return function () {
359
360                     if (parentNode === this) {
361                         throw new Error(
362                             "You can't sort elements if any one is a descendant of another."
363                         );
364                     }
365
366                     // Insert before flag:
367                     parentNode.insertBefore(this, nextSibling);
368                     // Remove flag:
369                     parentNode.removeChild(nextSibling);
370
371                 };
372             });
373
374             return sort.call(this, comparator).each(function (i) {
375                 placements[i].call(getSortable.call(this));
376             });
377         });
378
379         return sort.call(this, comparator).each(function (i) {
380             placements[i].call(getSortable.call(this));
381         });
382     });
383
384     /**
385      * jQuery.fn.sortElements
386      * -----
387      * #author James Padolsey (http://james.padolsey.com)
388      * #version 0.11
389      * #updated 18-MAR-2010
390      * #url
391      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
392      * -----
393      * #param Function comparator:
394      *   Exactly the same behaviour as [1,2,3].sort(comparator)
395      *
396      * #param Function getSortable
397      *   A function that should return the element that is
398      *   to be sorted. The comparator will run on the
399      *   current collection, but you may want the actual
400      *   resulting sort to occur on a parent or another
401      *   associated element.
402      *
403      * E.g. $('td').sortElements(comparator, function() {
404          return this.parentNode;
405      })
406      *
407      * The <td>'s parent (<tr>) will be sorted instead
408      * of the <td> itself.
409      */
410     jQuery.fn.sortElements = (function () {
411
412         var sort = [].sort;
413
414         return function (comparator, getSortable) {
415
416             getSortable = getSortable || function () { return this; };
417
418             var placements = this.map(function () {
419
420                 var sortElement = getSortable.call(this),
421                     parentNode = sortElement.parentNode,
422
423                     // Since the element itself will change position, we have
424                     // to have some way of storing it's original position in
425                     // the DOM. The easiest way is to have a 'flag' node:
426                 nextSibling = parentNode.insertBefore(
427                     document.createTextNode(''),
428                     sortElement.nextSibling
429                 );
430
431                 return function () {
432
433                     if (parentNode === this) {
434                         throw new Error(
435                             "You can't sort elements if any one is a descendant of another."
436                         );
437                     }
438
439                     // Insert before flag:
440                     parentNode.insertBefore(this, nextSibling);
441                     // Remove flag:
442                     parentNode.removeChild(nextSibling);
443
444                 };
445             });
446
447             return sort.call(this, comparator).each(function (i) {
448                 placements[i].call(getSortable.call(this));
449             });
450         });
451
452         return sort.call(this, comparator).each(function (i) {
453             placements[i].call(getSortable.call(this));
454         });
455     });
456
457     /**
458      * jQuery.fn.sortElements
459      * -----
460      * #author James Padolsey (http://james.padolsey.com)
461      * #version 0.11
462      * #updated 18-MAR-2010
463      * #url
464      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
465      * -----
466      * #param Function comparator:
467      *   Exactly the same behaviour as [1,2,3].sort(comparator)
468      *
469      * #param Function getSortable
470      *   A function that should return the element that is
471      *   to be sorted. The comparator will run on the
472      *   current collection, but you may want the actual
473      *   resulting sort to occur on a parent or another
474      *   associated element.
475      *
476      * E.g. $('td').sortElements(comparator, function() {
477          return this.parentNode;
478      })
479      *
480      * The <td>'s parent (<tr>) will be sorted instead
481      * of the <td> itself.
482      */
483     jQuery.fn.sortElements = (function () {
484
485         var sort = [].sort;
486
487         return function (comparator, getSortable) {
488
489             getSortable = getSortable || function () { return this; };
490
491             var placements = this.map(function () {
492
493                 var sortElement = getSortable.call(this),
494                     parentNode = sortElement.parentNode,
495
496                     // Since the element itself will change position, we have
497                     // to have some way of storing it's original position in
498                     // the DOM. The easiest way is to have a 'flag' node:
499                 nextSibling = parentNode.insertBefore(
500                     document.createTextNode(''),
501                     sortElement.nextSibling
502                 );
503
504                 return function () {
505
506                     if (parentNode === this) {
507                         throw new Error(
508                             "You can't sort elements if any one is a descendant of another."
509                         );
510                     }
511
512                     // Insert before flag:
513                     parentNode.insertBefore(this, nextSibling);
514                     // Remove flag:
515                     parentNode.removeChild(nextSibling);
516
517                 };
518             });
519
520             return sort.call(this, comparator).each(function (i) {
521                 placements[i].call(getSortable.call(this));
522             });
523         });
524
525         return sort.call(this, comparator).each(function (i) {
526             placements[i].call(getSortable.call(this));
527         });
528     });
529
530     /**
531      * jQuery.fn.sortElements
532      * -----
533      * #author James Padolsey (http://james.padolsey.com)
534      * #version 0.11
535      * #updated 18-MAR-2010
536      * #url
537      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
538      * -----
539      * #param Function comparator:
540      *   Exactly the same behaviour as [1,2,3].sort(comparator)
541      *
542      * #param Function getSortable
543      *   A function that should return the element that is
544      *   to be sorted. The comparator will run on the
545      *   current collection, but you may want the actual
546      *   resulting sort to occur on a parent or another
547      *   associated element.
548      *
549      * E.g. $('td').sortElements(comparator, function() {
550          return this.parentNode;
551      })
552      *
553      * The <td>'s parent (<tr>) will be sorted instead
554      * of the <td> itself.
555      */
556     jQuery.fn.sortElements = (function () {
557
558         var sort = [].sort;
559
560         return function (comparator, getSortable) {
561
562             getSortable = getSortable || function () { return this; };
563
564             var placements = this.map(function () {
565
566                 var sortElement = getSortable.call(this),
567                     parentNode = sortElement.parentNode,
568
569                     // Since the element itself will change position, we have
570                     // to have some way of storing it's original position in
571                     // the DOM. The easiest way is to have a 'flag' node:
572                 nextSibling = parentNode.insertBefore(
573                     document.createTextNode(''),
574                     sortElement.nextSibling
575                 );
576
577                 return function () {
578
579                     if (parentNode === this) {
580                         throw new Error(
581                             "You can't sort elements if any one is a descendant of another."
582                         );
583                     }
584
585                     // Insert before flag:
586                     parentNode.insertBefore(this, nextSibling);
587                     // Remove flag:
588                     parentNode.removeChild(nextSibling);
589
590                 };
591             });
592
593             return sort.call(this, comparator).each(function (i) {
594                 placements[i].call(getSortable.call(this));
595             });
596         });
597
598         return sort.call(this, comparator).each(function (i) {
599             placements[i].call(getSortable.call(this));
600         });
601     });
602
603     /**
604      * jQuery.fn.sortElements
605      * -----
606      * #author James Padolsey (http://james.padolsey.com)
607      * #version 0.11
608      * #updated 18-MAR-2010
609      * #url
610      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
611      * -----
612      * #param Function comparator:
613      *   Exactly the same behaviour as [1,2,3].sort(comparator)
614      *
615      * #param Function getSortable
616      *   A function that should return the element that is
617      *   to be sorted. The comparator will run on the
618      *   current collection, but you may want the actual
619      *   resulting sort to occur on a parent or another
620      *   associated element.
621      *
622      * E.g. $('td').sortElements(comparator, function() {
623          return this.parentNode;
624      })
625      *
626      * The <td>'s parent (<tr>) will be sorted instead
627      * of the <td> itself.
628      */
629     jQuery.fn.sortElements = (function () {
630
631         var sort = [].sort;
632
633         return function (comparator, getSortable) {
634
635             getSortable = getSortable || function () { return this; };
636
637             var placements = this.map(function () {
638
639                 var sortElement = getSortable.call(this),
640                     parentNode = sortElement.parentNode,
641
642                     // Since the element itself will change position, we have
643                     // to have some way of storing it's original position in
644                     // the DOM. The easiest way is to have a 'flag' node:
645                 nextSibling = parentNode.insertBefore(
646                     document.createTextNode(''),
647                     sortElement.nextSibling
648                 );
649
650                 return function () {
651
652                     if (parentNode === this) {
653                         throw new Error(
654                             "You can't sort elements if any one is a descendant of another."
655                         );
656                     }
657
658                     // Insert before flag:
659                     parentNode.insertBefore(this, nextSibling);
660                     // Remove flag:
661                     parentNode.removeChild(nextSibling);
662
663                 };
664             });
665
666             return sort.call(this, comparator).each(function (i) {
667                 placements[i].call(getSortable.call(this));
668             });
669         });
670
671         return sort.call(this, comparator).each(function (i) {
672             placements[i].call(getSortable.call(this));
673         });
674     });
675
676     /**
677      * jQuery.fn.sortElements
678      * -----
679      * #author James Padolsey (http://james.padolsey.com)
680      * #version 0.11
681      * #updated 18-MAR-2010
682      * #url
683      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
684      * -----
685      * #param Function comparator:
686      *   Exactly the same behaviour as [1,2,3].sort(comparator)
687      *
688      * #param Function getSortable
689      *   A function that should return the element that is
690      *   to be sorted. The comparator will run on the
691      *   current collection, but you may want the actual
692      *   resulting sort to occur on a parent or another
693      *   associated element.
694      *
695      * E.g. $('td').sortElements(comparator, function() {
696          return this.parentNode;
697      })
698      *
699      * The <td>'s parent (<tr>) will be sorted instead
700      * of the <td> itself.
701      */
702     jQuery.fn.sortElements = (function () {
703
704         var sort = [].sort;
705
706         return function (comparator, getSortable) {
707
708             getSortable = getSortable || function () { return this; };
709
710             var placements = this.map(function () {
711
712                 var sortElement = getSortable.call(this),
713                     parentNode = sortElement.parentNode,
714
715                     // Since the element itself will change position, we have
716                     // to have some way of storing it's original position in
717                     // the DOM. The easiest way is to have a 'flag' node:
718                 nextSibling = parentNode.insertBefore(
719                     document.createTextNode(''),
720                     sortElement.nextSibling
721                 );
722
723                 return function () {
724
725                     if (parentNode === this) {
726                         throw new Error(
727                             "You can't sort elements if any one is a descendant of another."
728                         );
729                     }
730
731                     // Insert before flag:
732                     parentNode.insertBefore(this, nextSibling);
733                     // Remove flag:
734                     parentNode.removeChild(nextSibling);
735
736                 };
737             });
738
739             return sort.call(this, comparator).each(function (i) {
740                 placements[i].call(getSortable.call(this));
741             });
742         });
743
744         return sort.call(this, comparator).each(function (i) {
745             placements[i].call(getSortable.call(this));
746         });
747     });
748
749     /**
750      * jQuery.fn.sortElements
751      * -----
752      * #author James Padolsey (http://james.padolsey.com)
753      * #version 0.11
754      * #updated 18-MAR-2010
755      * #url
756      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
757      * -----
758      * #param Function comparator:
759      *   Exactly the same behaviour as [1,2,3].sort(comparator)
760      *
761      * #param Function getSortable
762      *   A function that should return the element that is
763      *   to be sorted. The comparator will run on the
764      *   current collection, but you may want the actual
765      *   resulting sort to occur on a parent or another
766      *   associated element.
767      *
768      * E.g. $('td').sortElements(comparator, function() {
769          return this.parentNode;
770      })
771      *
772      * The <td>'s parent (<tr>) will be sorted instead
773      * of the <td> itself.
774      */
775     jQuery.fn.sortElements = (function () {
776
777         var sort = [].sort;
778
779         return function (comparator, getSortable) {
780
781             getSortable = getSortable || function () { return this; };
782
783             var placements = this.map(function () {
784
785                 var sortElement = getSortable.call(this),
786                     parentNode = sortElement.parentNode,
787
788                     // Since the element itself will change position, we have
789                     // to have some way of storing it's original position in
790                     // the DOM. The easiest way is to have a 'flag' node:
791                 nextSibling = parentNode.insertBefore(
792                     document.createTextNode(''),
793                     sortElement.nextSibling
794                 );
795
796                 return function () {
797
798                     if (parentNode === this) {
799                         throw new Error(
800                             "You can't sort elements if any one is a descendant of another."
801                         );
802                     }
803
804                     // Insert before flag:
805                     parentNode.insertBefore(this, nextSibling);
806                     // Remove flag:
807                     parentNode.removeChild(nextSibling);
808
809                 };
810             });
811
812             return sort.call(this, comparator).each(function (i) {
813                 placements[i].call(getSortable.call(this));
814             });
815         });
816
817         return sort.call(this, comparator).each(function (i) {
818             placements[i].call(getSortable.call(this));
819         });
820     });
821
822     /**
823      * jQuery.fn.sortElements
824      * -----
825      * #author James Padolsey (http://james.padolsey.com)
826      * #version 0.11
827      * #updated 18-MAR-2010
828      * #url
829      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
830      * -----
831      * #param Function comparator:
832      *   Exactly the same behaviour as [1,2,3].sort(comparator)
833      *
834      * #param Function getSortable
835      *   A function that should return the element that is
836      *   to be sorted. The comparator will run on the
837      *   current collection, but you may want the actual
838      *   resulting sort to occur on a parent or another
839      *   associated element.
840      *
841      * E.g. $('td').sortElements(comparator, function() {
842          return this.parentNode;
843      })
844      *
845      * The <td>'s parent (<tr>) will be sorted instead
846      * of the <td> itself.
847      */
848     jQuery.fn.sortElements = (function () {
849
850         var sort = [].sort;
851
852         return function (comparator, getSortable) {
853
854             getSortable = getSortable || function () { return this; };
855
856             var placements = this.map(function () {
857
858                 var sortElement = getSortable.call(this),
859                     parentNode = sortElement.parentNode,
860
861                     // Since the element itself will change position, we have
862                     // to have some way of storing it's original position in
863                     // the DOM. The easiest way is to have a 'flag' node:
864                 nextSibling = parentNode.insertBefore(
865                     document.createTextNode(''),
866                     sortElement.nextSibling
867                 );
868
869                 return function () {
870
871                     if (parentNode === this) {
872                         throw new Error(
873                             "You can't sort elements if any one is a descendant of another."
874                         );
875                     }
876
877                     // Insert before flag:
878                     parentNode.insertBefore(this, nextSibling);
879                     // Remove flag:
880                     parentNode.removeChild(nextSibling);
881
882                 };
883             });
884
885             return sort.call(this, comparator).each(function (i) {
886                 placements[i].call(getSortable.call(this));
887             });
888         });
889
890         return sort.call(this, comparator).each(function (i) {
891             placements[i].call(getSortable.call(this));
892         });
893     });
894
895     /**
896      * jQuery.fn.sortElements
897      * -----
898      * #author James Padolsey (http://james.padolsey.com)
899      * #version 0.11
900      * #updated 18-MAR-2010
901      * #url
902      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
903      * -----
904      * #param Function comparator:
905      *   Exactly the same behaviour as [1,2,3].sort(comparator)
906      *
907      * #param Function getSortable
908      *   A function that should return the element that is
909      *   to be sorted. The comparator will run on the
910      *   current collection, but you may want the actual
911      *   resulting sort to occur on a parent or another
912      *   associated element.
913      *
914      * E.g. $('td').sortElements(comparator, function() {
915          return this.parentNode;
916      })
917      *
918      * The <td>'s parent (<tr>) will be sorted instead
919      * of the <td> itself.
920      */
921     jQuery.fn.sortElements = (function () {
922
923         var sort = [].sort;
924
925         return function (comparator, getSortable) {
926
927             getSortable = getSortable || function () { return this; };
928
929             var placements = this.map(function () {
930
931                 var sortElement = getSortable.call(this),
932                     parentNode = sortElement.parentNode,
933
934                     // Since the element itself will change position, we have
935                     // to have some way of storing it's original position in
936                     // the DOM. The easiest way is to have a 'flag' node:
937                 nextSibling = parentNode.insertBefore(
938                     document.createTextNode(''),
939                     sortElement.nextSibling
940                 );
941
942                 return function () {
943
944                     if (parentNode === this) {
945                         throw new Error(
946                             "You can't sort elements if any one is a descendant of another."
947                         );
948                     }
949
950                     // Insert before flag:
951                     parentNode.insertBefore(this, nextSibling);
952                     // Remove flag:
953                     parentNode.removeChild(nextSibling);
954
955                 };
956             });
957
958             return sort.call(this, comparator).each(function (i) {
959                 placements[i].call(getSortable.call(this));
960             });
961         });
962
963         return sort.call(this, comparator).each(function (i) {
964             placements[i].call(getSortable.call(this));
965         });
966     });
967
968     /**
969      * jQuery.fn.sortElements
970      * -----
971      * #author James Padolsey (http://james.padolsey.com)
972      * #version 0.11
973      * #updated 18-MAR-2010
974      * #url
975      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
976      * -----
977      * #param Function comparator:
978      *   Exactly the same behaviour as [1,2,3].sort(comparator)
979      *
980      * #param Function getSortable
981      *   A function that should return the element that is
982      *   to be sorted. The comparator will run on the
983      *   current collection, but you may want the actual
984      *   resulting sort to occur on a parent or another
985      *   associated element.
986      *
987      * E.g. $('td').sortElements(comparator, function() {
988          return this.parentNode;
989      })
990      *
991      * The <td>'s parent (<tr>) will be sorted instead
992      * of the <td> itself.
993      */
994     jQuery.fn.sortElements = (function () {
995
996         var sort = [].sort;
997
998         return function (comparator, getSortable) {
999
1000            getSortable = getSortable || function () { return this; };
1001
1002            var placements = this.map(function () {
1003
1004                var sortElement = getSortable.call(this),
1005                    parentNode = sortElement.parentNode,
1006
1007                    // Since the element itself will change position, we have
1008                    // to have some way of storing it's original position in
1009                    // the DOM. The easiest way is to have a 'flag' node:
1010                nextSibling = parentNode.insertBefore(
1011                    document.createTextNode(''),
1012                    sortElement.nextSibling
1013                );
1014
1015                return function () {
1016
1017                    if (parentNode === this) {
1018                        throw new Error(
1019                            "You can't sort elements if any one is a descendant of another."
1020                            );
1021
1022                    }
1023
1024                    // Insert before flag:
1025                    parentNode.insertBefore(this, nextSibling);
1026                    // Remove flag:
1027                    parentNode.removeChild(nextSibling);
1028
1029                };
1030            });
1031
1032            return sort.call(this, comparator).each(function (i) {
1033                placements[i].call(getSortable.call(this));
1034            });
1035        });
1036
1037        return sort.call(this, comparator).each(function (i) {
1038            placements[i].call(getSortable.call(this));
1039        });
1040    });
1041
1042    /**
1043      * jQuery.fn.sortElements
1044      * -----
1045      * #author James Padolsey (http://james.padolsey.com)
1046      * #version 0.11
1047      * #updated 18-MAR-2010
1048      * #url
1049      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
1050      * -----
1051      * #param Function comparator:
1052      *   Exactly the same behaviour as [1,2,3].sort(comparator)
1053      *
1054      * #param Function getSortable
1055      *   A function that should return the element that is
1056      *   to be sorted. The comparator will run on the
1057      *   current collection, but you may want the actual
1058      *   resulting sort to occur on a parent or another
1059      *   associated element.
1060      *
1061      * E.g. $('td').sortElements(comparator, function() {
1062          return this.parentNode;
1063      })
1064      *
1065      * The <td>'s parent (<tr>) will be sorted instead
1066      * of the <td> itself.
1067      */
1068     jQuery.fn.sortElements = (function () {
1069
1070         var sort = [].sort;
1071
1072         return function (comparator, getSortable) {
1073
1074             getSortable = getSortable || function () { return this; };
1075
1076             var placements = this.map(function () {
1077
1078                 var sortElement = getSortable.call(this),
1079                     parentNode = sortElement.parentNode,
1080
1081                     // Since the element itself will change position, we have
1082                     // to have some way of storing it's original position in
1083                     // the DOM. The easiest way is to have a 'flag' node:
1084                 nextSibling = parentNode.insertBefore(
1085                     document.createTextNode(''),
1086                     sortElement.nextSibling
1087                 );
1088
1089                 return function () {
1090
1091                     if (parentNode === this) {
1092                         throw new Error(
1093                             "You can't sort elements if any one is a descendant of another."
1094                             );
1095
1096                     }
1097
1098                     // Insert before flag:
1099                     parentNode.insertBefore(this, nextSibling);
1100                     // Remove flag:
1101                     parentNode.removeChild(nextSibling);
1102
1103                 };
1104             });
1105
1106             return sort.call(this, comparator).each(function (i) {
1107                 placements[i].call(getSortable.call(this));
1108             });
1109         });
1110
1111         return sort.call(this, comparator).each(function (i) {
1112             placements[i].call(getSortable.call(this));
1113         });
1114     });
1115
1116     /**
1117      * jQuery.fn.sortElements
1118      * -----
1119      * #author James Padolsey (http://james.padolsey.com)
1120      * #version 0.11
1121      * #updated 18-MAR-2010
1122      * #url
1123      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
1124      * -----
1125      * #param Function comparator:
1126      *   Exactly the same behaviour as [1,2,3].sort(comparator)
1127      *
1128      * #param Function getSortable
1129      *   A function that should return the element that is
1130      *   to be sorted. The comparator will run on the
1131      *   current collection, but you may want the actual
1132      *   resulting sort to occur on a parent or another
1133      *   associated element.
1134      *
1135      * E.g. $('td').sortElements(comparator, function() {
1136          return this.parentNode;
1137      })
1138      *
1139      * The <td>'s parent (<tr>) will be sorted instead
1140      * of the <td> itself.
1141      */
1142     jQuery.fn.sortElements = (function () {
1143
1144         var sort = [].sort;
1145
1146         return function (comparator, getSortable) {
1147
1148             getSortable = getSortable || function () { return this; };
1149
1150             var placements = this.map(function () {
1151
1152                 var sortElement = getSortable.call(this),
1153                     parentNode = sortElement.parentNode,
1154
1155                     // Since the element itself will change position, we have
1156                     // to have some way of storing it's original position in
1157                     // the DOM. The easiest way is to have a 'flag' node:
1158                 nextSibling = parentNode.insertBefore(
1159                     document.createTextNode(''),
1160                     sortElement.nextSibling
1161                 );
1162
1163                 return function () {
1164
1165                     if (parentNode === this) {
1166                         throw new Error(
1167                             "You can't sort elements if any one is a descendant of another."
1168                             );
1169
1170                     }
1171
1172                     // Insert before flag:
1173                     parentNode.insertBefore(this, nextSibling);
1174                     // Remove flag:
1175                     parentNode.removeChild(nextSibling);
1176
1177                 };
1178             });
1179
1180             return sort.call(this, comparator).each(function (i) {
1181                 placements[i].call(getSortable.call(this));
1182             });
1183         });
1184
1185         return sort.call(this, comparator).each(function (i) {
1186             placements[i].call(getSortable.call(this));
1187         });
1188     });
1189
1190     /**
1191      * jQuery.fn.sortElements
1192      * -----
1193      * #author James Padolsey (http://james.padolsey.com)
1194      * #version 0.11
1195      * #updated 18-MAR-2010
1196      * #url
1197      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
1198      * -----
1199      * #param Function comparator:
1200      *   Exactly the same behaviour as [1,2,3].sort(comparator)
1201      *
1202      * #param Function getSortable
1203      *   A function that should return the element that is
1204      *   to be sorted. The comparator will run on the
1205      *   current collection, but you may want the actual
1206      *   resulting sort to occur on a parent or another
1207      *   associated element.
1208      *
1209      * E.g. $('td').sortElements(comparator, function() {
1210          return this.parentNode;
1211      })
1212      *
1213      * The <td>'s parent (<tr>) will be sorted instead
1214      * of the <td> itself.
1215      */
1216     jQuery.fn.sortElements = (function () {
1217
1218         var sort = [].sort;
1219
1220         return function (comparator, getSortable) {
1221
1222             getSortable = getSortable || function () { return this; };
1223
1224             var placements = this.map(function () {
1225
1226                 var sortElement = getSortable.call(this),
1227                     parentNode = sortElement.parentNode,
1228
1229                     // Since the element itself will change position, we have
1230                     // to have some way of storing it's original position in
1231                     // the DOM. The easiest way is to have a 'flag' node:
1232                 nextSibling = parentNode.insertBefore(
1233                     document.createTextNode(''),
1234                     sortElement.nextSibling
1235                 );
1236
1237                 return function () {
1238
1239                     if (parentNode === this) {
1240                         throw new Error(
1241                             "You can't sort elements if any one is a descendant of another."
1242                             );
1243
1244                     }
1245
1246                     // Insert before flag:
1247                     parentNode.insertBefore(this, nextSibling);
1248                     // Remove flag:
1249                     parentNode.removeChild(nextSibling);
1250
1251                 };
1252             });
1253
1254             return sort.call(this, comparator).each(function (i) {
1255                 placements[i].call(getSortable.call(this));
1256             });
1257         });
1258
1259         return sort.call(this, comparator).each(function (i) {
1260             placements[i].call(getSortable.call(this));
1261         });
1262     });
1263
1264     /**
1265      * jQuery.fn.sortElements
1266      * -----
1267      * #author James Padolsey (http://james.padolsey.com)
1268      * #version 0.11
1269      * #updated 18-MAR-2010
1270      * #url
1271      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js
1272      * -----
1273      * #param Function comparator:
1274      *   Exactly the same behaviour as [1,2,3].sort(comparator)
1275      *
1276      * #param Function getSortable
1277      *   A function that should return the element that is
1278      *   to be sorted. The comparator will run on the
1279      *   current collection, but you may want the actual
1280      *   resulting sort to occur on a parent or another
1281      *   associated element.
1282      *
1283      * E.g. $('td').sortElements(comparator, function() {
1284          return this.parentNode;
1285      })
1286      *
1287      * The <td>'s parent (<tr>) will be sorted instead
1288      * of the <td> itself.
1289      */
1290     jQuery.fn.sortElements = (function () {
1291
1292         var sort = [].sort;
1293
1294         return function (comparator, getSortable) {
1295
1296             getSortable = getSortable || function () { return this; };
1297
1298             var placements = this.map(function () {
1299
1300                 var sortElement = getSortable.call(this),
1301                     parentNode = sortElement.parentNode,
1302
1303                     // Since the element itself will change position, we have
1304                     // to have some way of storing it's original position in
1305                     // the DOM. The easiest way is to have a 'flag' node:
1306                 nextSibling = parentNode.insertBefore(
1307                     document.createTextNode(''),
1308                     sortElement.nextSibling
1309                 );
1310
1311                 return function () {
1312
1313                     if (parentNode === this) {
1314                         throw new Error(
1315                             "You can't sort elements if any one is a descendant of another."
1316                             );
1317
1318                     }
1319
1320                     // Insert before flag:
1321                     parentNode.insertBefore(this, nextSibling);
1322                     // Remove flag:
1323                     parentNode.removeChild(nextSibling);
1324
1325                 };
1326             });
1327
1328             return sort.call(this, comparator).each(function (i) {
1329                 placements[i].call(getSortable.call(this));
1330             });
1331         });
1332
1333         return sort.call(this, comparator).each(function (i) {
1334             placements[i].call(getSortable.call(this));
1335         });
1336     });
1337
1338     /**
1339      * jQuery.fn.sortElements
1340      * -----
1341      * #author James Padolsey (http://james.padolsey.com)
1342      * #version 0.11
1343      * #updated 18-MAR-2010
1344      * #url
1345      https://raw.github.com/jamespadolsey/jQuery-Plugins/master/sortElements/jquery.sortElements.js</a
```

```

377
378     };
379
380     })();
381 </script>
382 <script type="text/javascript">
383     jQuery.fn.setBarSizes = (function () {
384         return function (metricName, maxBarSize, min) {
385             var max = Math.max.apply(Math, $.makeArray($(this).map(function () {
386                 return Number($(this).attr('data-sr-' + metricName));
387             })));
388             var scale = 1.0;
389             while (max > 0.0 && max <= 10.0) {
390                 scale = scale * 10.0;
391                 max = max * 10.0;
392             }
393             max = (Math.ceil(max / 2) * 2) / scale;
394
395             this.each(function () {
396                 var barAnchor = $(this).find('a');
397                 var actual = $(this).attr('data-sr-' + metricName);
398                 var newHeight = Math.max(Math.round(maxBarSize * (actual - min)
399                               / (max - min)), 2);
400                 barAnchor.css({ height: newHeight });
401             });
402
403             var unit = "";
404             if (metricName.substr(metricName.length - 4, 4) === "time")
405                 unit = "s";
406
407             $('#testview .scale-min-label').each(function () {
408                 $(this).text(min.toString() + unit);
409             });
410             $('#testview .scale-max-label').each(function () {
411                 $(this).text(max.toString() + unit);
412             });
413             var mid = max / 2;
414             $('#testview .scale-mid-label').each(function () {
415                 $(this).text(mid.toString() + unit);
416             });
417         });
418     })();
419
420     function getComparer(metricName, isDesc) {
421         return function (a, b) {
422             var aNumber = Number($(a).attr('data-sr-' + metricName));
423             var bNumber = Number($(b).attr('data-sr-' + metricName));
424             var result = aNumber > bNumber ? 1 : (aNumber < bNumber ? -1 : 0);
425             if (isDesc)
426                 result = -1 * result;
427
428             if (result == 0 && metricName != "order")
429                 result = getComparer("order", false)(a, b);
430
431             return result;
432         };
433     }
434
435     var currentSort = "";
436     function doSort(allowToggleDesc) {
437         var newSort = $("input[name='barSortOrder']:checked").val();
438         if (allowToggleDesc && currentSort == newSort) {
439             $('#barSortDesc').click();
440             doSort(false);
441             return;
442         }
443         currentSort = newSort;
444         $('#testview td.testview-item').sortElements(getComparer(newSort,
445             $('#barSortDesc').is(':checked')));
446     }

```

```
443
444
445     function doSetHeights(allowSort) {
446         var selectedMetric = $("input[name='barHeight']:checked").val();
447         $('#testview td.testview-item').setBarSizes(selectedMetric, 60, 0.0);
448
449         if (allowSort && currentSort != selectedMetric) {
450             var $radios = $("input[name='barSortOrder']");
451             $radios.filter('[value=' + selectedMetric + ']').attr('checked', true);
452             $("#barSortDesc").attr("checked", [true]);
453             doSort(false);
454         }
455     }
456
457     $(document).ready(function () {
458         $("input[name='barSortOrder']").click(function () { doSort(true); return true; });
459         $("input[name='barSortDesc']").change(function () { doSort(false); });
460         $("input[name='barHeight']").change(function () { doSetHeights(true); });
461
462         doSort(false);
463         doSetHeights(false);
464
465         $("div.scrollable").css({ 'overflow': 'auto' });
466     });
467 
468
469 <style type="text/css">
470     body
471     {
472         color: #000000;
473         font-family: Arial, Liberation Sans, DejaVu Sans, sans-serif;
474         line-height: 130%;
475     }
476     h1 {
477         font-family: Trebuchet MS, Liberation Sans, DejaVu Sans, sans-serif;
478         font-size: 170%;
479         font-weight: bold;
480     }
481     h2 {
482         font-family: Trebuchet MS, Liberation Sans, DejaVu Sans, sans-serif;
483         font-size: 130%;
484         font-weight: bold;
485         margin-bottom: 5px;
486     }
487     h3 {
488         font-family: Trebuchet MS, Liberation Sans, DejaVu Sans, sans-serif;
489         font-size: 120%;
490         font-weight: bold;
491         margin-bottom: 5px;
492     }
493     a.bar
494     {
495         text-decoration: none;
496         display: block;
497         line-height: 1px;
498     }
499     .description
500     {
501         font-style: italic;
502     }
503     .log
504     {
505         width: 600px;
506         white-space: pre-wrap;
507         display: block;
508         margin: 0px;
509     }
```

```
510
511     .errorMessage
512     {
513         width: 600px;
514         color: Red;
515         font-weight: bold;
516     }
517     .stackTrace
518     {
519         width: 600px;
520         white-space: pre-wrap;
521         font-style: italic;
522         color: Red;
523         display: block;
524     }
525     table.testEvents
526     {
527         border: solid 1px #e8eef4;
528         border-collapse: collapse;
529     }
530     table.testEvents td
531     {
532         vertical-align: top;
533         padding: 5px;
534         border: solid 1px #e8eef4;
535     }
536     table.testEvents th
537     {
538         padding: 6px 5px;
539         text-align: left;
540         background-color: #e8eef4;
541         border: solid 1px #a9bfd6;
542     }
543     .comment
544     {
545         font-style: italic;
546         font-size: smaller;
547     }
548     .startupBar
549     {
550         background-color: #EEEEEE;
551         cursor: default;
552     }
553     .colorSucceeded
554     {
555         background-color: #90ED7B;
556     }
557     .colorPending
558     {
559         background-color: #D47BED;
560     }
561     .colorNothingToRun
562     {
563         background-color: #CCCCFF;
564     }
565     .colorInconclusive
566     {
567         background-color: #7BEDED;
568     }
569     .colorCleanupFailed
570     {
571         background-color: #FFCCCC;
572     }
573     .colorRandomlyFailed
574     {
575         background-color: #EDB07B;
576     }
577     .colorFailed
578     {
```

```
579         background-color: #ED5F5F;
580     }
581     .colorInitializationFailed
582     {
583         background-color: #FF0000;
584     }
585     .colorFrameworkError
586     {
587         background-color: #FF0000;
588     }
589     ul.subNodeLinks
590     {
591         padding-left: 20px;
592         margin: 0px;
593     }
594     ul.subNodeLinks li
595     {
596         list-style: none;
597     }
598
599     /* views general */
600     div.scrollable
601     {
602         /*overflow: auto; - thsi has to be set from js, because of an IE9 bug */
603     }
604     div.viewbox
605     {
606         position: relative;
607         border: 3px solid #e8eef4;
608     }
609     div.viewbox table
610     {
611         border: 0px;
612     }
613
614     /* testview */
615     #testview
616     {
617         padding-top: 23px;
618     }
619
620     table.testview-items td
621     {
622         vertical-align: bottom;
623         padding: 0px 1px 0px 1px;
624     }
625     td.right-padding, td.left-padding
626     {
627         width: 25px;
628         min-width: 25px;
629     }
630     table.testview-items a.bar
631     {
632         width: 5px;
633     }
634     table.testview-items tr.testview-items-row
635     {
636         height: 60px;
637     }
638
639     /* scale */
640     table.vertical-scale
641     {
642         position: absolute;
643         top: 23px;
644         left: 0px;
645         width: 100%;
646         z-index: -100;
647     }
```

```
648     table.vertical-scale td, tr.horizontal-scale td
649     {
650         font-size: 60%;
651         line-height: normal;
652     }
653     table.vertical-scale tr.scale-max, table.vertical-scale tr.scale-mid
654     {
655         height: 30px;
656     }
657     tr.horizontal-scale, table.vertical-scale tr.scale-min
658     {
659         height: 12px;
660     }
661
662     td.scale-max-label, td.scale-mid-label, td.scale-min-label
663     {
664         border-top: solid 1px #E6E6E6;
665         text-align: left;
666         vertical-align: top;
667     }
668     td.scale-10-label
669     {
670         border-left: solid 1px #E6E6E6;
671         text-align: left;
672         vertical-align: bottom;
673         padding-left: 1px;
674     }
675     tr.scale-mid td, tr.scale-min td, tr.scale-max td
676     {
677         border-top: solid 1px #E6E6E6;
678     }
679
680
681     /* bar-control */
682     #bar-control
683     {
684         font-size: 60%;
685         line-height: normal;
686         position: absolute;
687         right: 0px;
688         top: 0px;
689     }
690     #bar-control label
691     {
692         font-weight: bold;
693         vertical-align: middle;
694     }
695     #bar-control .option
696     {
697         vertical-align: middle;
698         text-transform: lowercase;
699     }
700     #bar-control input[type="checkbox"]
701     {
702         padding: 0 2px 0 3px;
703     }
704     #bar-control input
705     {
706         vertical-align: top;
707         height: 12px;
708         margin: 0px;
709         padding: 0px;
710     }
711     #bar-control div
712     {
713         float: right;
714         margin: 3px 5px 3px 5px;
715     }
716
```

```

717     /* timeline view */
718     #timelineview
719     {
720         padding-top: 5px;
721     }
722     table.timelineview a
723     {
724         height: 20px;
725     }
726     table.timelineview td
727     {
728         vertical-align: bottom;
729         padding: 0px 1px 0px 0px;
730         border: 0px;
731     }
732     tr.thread-items-row
733     {
734         height: 25px;
735     }
736     tr.thread-items-row td
737     {
738         vertical-align: bottom;
739     }
740     td.thread-label
741     {
742         padding: 0px 6px 0px 6px;
743         text-align: right;
744         line-height: 18px;
745         vertical-align: bottom;
746     }
747     th.thread-label
748     {
749         padding: 3px 6px 0px 6px;
750         line-height: 18px;
751         text-align: left;
752         vertical-align: bottom;
753     }
754     .bartd {
755         border-left: 0px solid white !important;
756     }
757     .percentage {
758         border-right: 0px solid white !important;
759     }
760     </style>
761 </head>
762 <body>
763     <h1>@Model.Configuration.ProjectName Test Execution Report</h1>
764     @RenderSection("ProjectInformation")
765
766     <h2>Result: @Model.Summary.ConcludedResultMessage</h2>
767     <table class="testEvents">
768         @GetSummaryHeader(null)
769         @RenderTotalSummaryRowTail(Model.Summary)
770     </table>
771
772     <h2>Test Timeline Summary</h2>
773     @{
774         double msecPerPixel = Model.ExecutionTime.DurationMilliseconds /
775             (Model.TestExecutionResults.Count() * 7);
776         var secScale = Math.Max(1.0, Math.Round((msecPerPixel / 1000 * 70) / 2) * 2);
777         var scaleItemCount = (int)Math.Floor(Model.ExecutionTime.DurationSeconds /
778             secScale) + 1;
779         var pixelScale = secScale * 1000 / msecPerPixel;
780     }
781     <div id="timelineview" class="viewbox">
782         <div class="scrollable">
783             <table cellpadding="0" cellspacing="0">
784                 <tr>
785                     <th class="thread-label" colspan="2">thread</th>

```

```

784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999

```

```

835             @RenderTestExecutionSummaryRowTail(testSummary)
836         </tr>
837         if (!string.IsNullOrEmpty(testResult.Error))
838         {
839             <tr>
840                 <td style="padding-left: 20px;"><div class="errorMessage">Error:<br/>@(testResult.Error)</div></td>
841                 <td colspan="6">
842                     @foreach (string tag in testResult.TestNode.Tags)
843                     {
844                         if (tag.Contains("Bug") || tag.Contains("bug") || tag.Contains("BUG") || tag.Contains("FileImport"))
845                         {
846                             @(tag);
847                             <br />
848                         }
849                     }
850                 </td>
851             </tr>
852         }
853     }
854 </table>
855
856 <h2>Scenario Summary</h2>
857 @foreach (var fixtureNode in GetTextFixtures())
858 {
859     <a name="@GetTestNodeAnchor(fixtureNode, "f")" />
860     <h3>@fixtureNode.Type: @GetFixtureTitle(fixtureNode)</h3>
861     if (!string.IsNullOrEmpty(fixtureNode.Description))
862     {
863         <div class="description"><pre>@fixtureNode.Description</pre>
864         </div>
865     }
866     <table class="testEvents">
867         @GetSummaryHeader("Test", true)
868         @foreach (var testNode in fixtureNode.SubNodes)
869         {
870             var testSummary = GetSummary(testNode);
871             <tr>
872                 <td>
873                     @TestNodeLinks(testNode, 0)
874                 </td>
875
876                 @RenderTestExecutionSummaryRowTail(testSummary,
877                     CalculateDuration(testNode))
878
879             </tr>
880         }
881     </table>
882 }
883
884 <h2>Execution Details</h2>
885 @foreach (var test in Model.TestExecutionResults.OrderBy(tr =>
886     tr.ExecutionOrder))
887 {
888     var testItem = test.TestItemResult.TestNode;
889     <a name="@GetTestAnchor(test)" />
890     <h3>@testItem.Type: @GetTestTitle(test)</h3>
891     if (!string.IsNullOrEmpty(testItem.Description))
892     {
893         <div class="description">
894             <pre>@testItem.Description</pre>
895         </div>
896     }
897     if (testItem.Tags.Any())
898     {
899         <div class="description">
900             tags: @string.Join(", ", testItem.Tags)
901         </div>

```

```

900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
}
<ul>
    <li>Status: @test.ResultType</li>
    <li>Start time: @test.ExecutionTime.StartTime</li>
    <li>Execution time (sec): @test.ExecutionTime.DurationSeconds</li>
    <li>Thread: #@test.ThreadId</li>
    @if (!string.IsNullOrEmpty(test.Result.Error))
    {
        <li>Error: @(test.Result.Error)</li>
    }
</ul>

<table class="testEvents">
    <tr>
        <th>Steps</th>
        <th>Trace</th>
        <th>Result</th>
    </tr>
    @foreach (var traceEvent in test.Result.TraceEvents)
    {
        if (!IsRelevant(traceEvent))
        {
            continue;
        }
        var relatedNode = GetTestNode(traceEvent);
        <tr>
            <td>
                <pre class="log">@(GetBusinessMessages(traceEvent))</pre>
            </td>
            <td>
                <!-- [@traceEvent.Type: @relatedNode.Type - @relatedNode.Title] -->
                <pre
                    class="log">@Raw(FormatTechMessages(traceEvent.TechMessages.TrimEnd()))</pre>
                @if (!string.IsNullOrEmpty(traceEvent.Error))
                {
                    <div
                        class="errorMessage">@Raw(FormatTechMessages(traceEvent.Error))</div>
                    <pre
                        class="stackTrace">@Raw(FormatTechMessages(traceEvent.StackTrace.TrimEnd()))</pre>
                }
            </td>
            <td>@traceEvent.ResultType in
                @GetSeconds(Math.Round(traceEvent.Duration.TotalSeconds,
                3)) s</td>
        </tr>
    }
</table>
</body>
</html>

```