


## Consistent indentation

[Browse files](#) master parsonsbots committed yesterday1 parent [ac83440](#)commit [e28465227f4eb01d3aa525a546ca7f90ec135c90](#) Showing 2 changed files with 201 additions and 202 deletions.

Unified

Split

▼ 297  checker.php 

... @@ -1,182 +1,181 @@

1 &lt;?php

```
2
3 - class DedicatedIpChecker {
4 -
5 - /**
6 -  * Get user ID for an IP address
7 -  *
8 -  * @param string $ip
9 -  * @param array $userIps
10 -  * @param array $userIds
11 -  *
12 -  * @return string $response User ID for dedicated IP (if
    user ID doesn't exist for IP, pick an ID from existing
    user IDs)
13 -  */
14 -     protected function _getUserId($ip, $userIps,
    $userIds) {
15 -         if (!empty($userIps[$ip])) {
16 -             return $userIps[$ip];
17 -         }
18 -
19 -         if (strpos($ip, ':') !== false) {
20 -             $characters =
    'abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz';
21 -             $ipCharacters = strtolower($ip);
```

22

```
1 <?php
2 +     class DedicatedIpChecker {
3 +
4 +         /**
5 +          * Get user ID for an IP address
6 +          *
7 +          * @param string $ip
8 +          * @param array $userIps
9 +          * @param array $userIds
10 +          *
11 +          * @return string $response User ID for dedicated
    IP (if user ID doesn't exist for IP, pick an ID from
    existing user IDs)
12 +          */
13 +          protected function _getUserId($ip,
    $userIps, $userIds) {
14 +              if (!empty($userIps[$ip])) {
15 +                  return $userIps[$ip];
16 +              }
17 +
18 +              if (strpos($ip, ':') !== false) {
19 +                  $characters =
    'abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz';
20 +                  $ipCharacters =
    strtolower($ip);
```

21

```

23 -         for ($i = 0; $i <
strlen($ipCharacters); $i++) {
24 -             if (($numericValue =
strpos($characters, $ipCharacters[$i])) != false) {
25 -                 $ipCharacters[$i]
= $numericValue + 1 + ($numericValue & ($numericValue *
$numericValue));
26
                }
27
            }
28
29 -             $subnets =
array_filter(explode(':', $ipCharacters));
30 -             $key = array_sum($subnets) *
count($subnets);
31 -             } else {
32 -                 $subnets = explode('.', $ip);
33 -                 $key = ($subnets[0] != 0 ?
$subnets[0] : 1) * ($subnets[3] != 0 ? $subnets[3] : 1) +
$subnets[2] + $subnets[1];
34
                }
35
36 -             $response = ($userIds[$key]);
37 -             return $response;
38 -         }
39
40 - /**
41 -  * Parse and filter IP address list
42 -  *
43 -  * @param mixed [array/string] $ips
44 -  *
45 -  * @return array $response
46 -  */
47 -     protected function _parseIps($ips = array()) {
48 -         if (!is_array($ips)) {
49 -             $ips =
array_filter(preg_split("/[\\r\\n|\\n|\\r] <()~
{}|\\\\"'=?!*&#$,;_~|/", $ips));
50
                }
51

```

```

22 +         for ($i = 0; $i <
strlen($ipCharacters); $i++) {
23 +             if (($numericValue
= strpos($characters, $ipCharacters[$i])) != false) {
24 +                 $ipCharacters[$i] = $numericValue + 1 + ($numericValue &
($numericValue * $numericValue));
25 +                 }
26 +             }
27 +
28 +             $subnets =
array_filter(explode(':', $ipCharacters));
29 +             $key = array_sum($subnets)
* count($subnets);
30 +             } else {
31 +                 $subnets = explode('.',
$ip);
32 +                 $key = ($subnets[0] != 0 ?
$subnets[0] : 1) * ($subnets[3] != 0 ? $subnets[3] : 1) +
$subnets[2] + $subnets[1];
33
                }
34
35 +             $response = ($userIds[$key]);
36 +
            return $response;
37
        }
38
39 +     /**
40 +      * Parse and filter IP address list
41 +      *
42 +      * @param mixed [array/string] $ips
43 +      *
44 +      * @return array $response
45 +      */
46 +     protected function _parseIps($ips =
array()) {
47 +         if (!is_array($ips)) {
48 +             $ips =
array_filter(preg_split("/[\\r\\n|\\n|\\r] <()~
{}|\\\\"'=?!*&#$,;_~|/", $ips));
49 +
                }
50
51 +             $response = $this->_validateIps($ips);
52 +
            return $response;
53
        }
54

```

```

55 + /**
56 +  * Validate IPv4 address
57 +  *
58 +  * @param string $ip
59 +  *
60 +  * @return mixed [boolean/string] $response
61 +  */
62 +     protected function _validateIpv4($ip) {
63 +         $response = false;
64 +         $splitIpSubnets = explode('.',
$ip);
65 +
66 +         if (count($splitIpSubnets) === 4)
67 +         {
68 +             foreach ($splitIpSubnets
as $splitIpSubnet) {
69 +                 if (
70 +
71 + !is_numeric($splitIpSubnet) ||
72 +
73 + strlen($splitIpSubnet) > 3 ||
74 +
75 + $splitIpSubnet > 255 ||
76 +
77 + $splitIpSubnet < 0
78 +
79 + ) {
80 +
81 +
82 +
83 +
84 +
85 +
86 +
87 +
88 +
89 +
90 +
91 +
92 +
93 +
94 +
95 +
96 +
97 +
98 +
99 +
100 +
101 +
102 +
103 +
104 +
105 +
106 +
107 +
108 +
109 +
110 +
111 +
112 +
113 +
114 +
115 +
116 +
117 +
118 +
119 +
120 +
121 +
122 +
123 +
124 +
125 +
126 +
127 +
128 +
129 +
130 +
131 +
132 +
133 +
134 +
135 +
136 +
137 +
138 +
139 +
140 +
141 +
142 +
143 +
144 +
145 +
146 +
147 +
148 +
149 +
150 +
151 +
152 +
153 +
154 +
155 +
156 +
157 +
158 +
159 +
160 +
161 +
162 +
163 +
164 +
165 +
166 +
167 +
168 +
169 +
170 +
171 +
172 +
173 +
174 +
175 +
176 +
177 +
178 +
179 +
180 +
181 +
182 +
183 +
184 +
185 +
186 +
187 +
188 +
189 +
190 +
191 +
192 +
193 +
194 +
195 +
196 +
197 +
198 +
199 +
200 +
201 +
202 +
203 +
204 +
205 +
206 +
207 +
208 +
209 +
210 +
211 +
212 +
213 +
214 +
215 +
216 +
217 +
218 +
219 +
220 +
221 +
222 +
223 +
224 +
225 +
226 +
227 +
228 +
229 +
230 +
231 +
232 +
233 +
234 +
235 +
236 +
237 +
238 +
239 +
240 +
241 +
242 +
243 +
244 +
245 +
246 +
247 +
248 +
249 +
250 +
251 +
252 +
253 +
254 +
255 +
256 +
257 +
258 +
259 +
260 +
261 +
262 +
263 +
264 +
265 +
266 +
267 +
268 +
269 +
270 +
271 +
272 +
273 +
274 +
275 +
276 +
277 +
278 +
279 +
280 +
281 +
282 +
283 +
284 +
285 +
286 +
287 +
288 +
289 +
290 +
291 +
292 +
293 +
294 +
295 +
296 +
297 +
298 +
299 +
300 +
301 +
302 +
303 +
304 +
305 +
306 +
307 +
308 +
309 +
310 +
311 +
312 +
313 +
314 +
315 +
316 +
317 +
318 +
319 +
320 +
321 +
322 +
323 +
324 +
325 +
326 +
327 +
328 +
329 +
330 +
331 +
332 +
333 +
334 +
335 +
336 +
337 +
338 +
339 +
340 +
341 +
342 +
343 +
344 +
345 +
346 +
347 +
348 +
349 +
350 +
351 +
352 +
353 +
354 +
355 +
356 +
357 +
358 +
359 +
360 +
361 +
362 +
363 +
364 +
365 +
366 +
367 +
368 +
369 +
370 +
371 +
372 +
373 +
374 +
375 +
376 +
377 +
378 +
379 +
380 +
381 +
382 +
383 +
384 +
385 +
386 +
387 +
388 +
389 +
390 +
391 +
392 +
393 +
394 +
395 +
396 +
397 +
398 +
399 +
400 +
401 +
402 +
403 +
404 +
405 +
406 +
407 +
408 +
409 +
410 +
411 +
412 +
413 +
414 +
415 +
416 +
417 +
418 +
419 +
420 +
421 +
422 +
423 +
424 +
425 +
426 +
427 +
428 +
429 +
430 +
431 +
432 +
433 +
434 +
435 +
436 +
437 +
438 +
439 +
440 +
441 +
442 +
443 +
444 +
445 +
446 +
447 +
448 +
449 +
450 +
451 +
452 +
453 +
454 +
455 +
456 +
457 +
458 +
459 +
460 +
461 +
462 +
463 +
464 +
465 +
466 +
467 +
468 +
469 +
470 +
471 +
472 +
473 +
474 +
475 +
476 +
477 +
478 +
479 +
480 +
481 +
482 +
483 +
484 +
485 +
486 +
487 +
488 +
489 +
490 +
491 +
492 +
493 +
494 +
495 +
496 +
497 +
498 +
499 +
500 +
501 +
502 +
503 +
504 +
505 +
506 +
507 +
508 +
509 +
510 +
511 +
512 +
513 +
514 +
515 +
516 +
517 +
518 +
519 +
520 +
521 +
522 +
523 +
524 +
525 +
526 +
527 +
528 +
529 +
530 +
531 +
532 +
533 +
534 +
535 +
536 +
537 +
538 +
539 +
540 +
541 +
542 +
543 +
544 +
545 +
546 +
547 +
548 +
549 +
550 +
551 +
552 +
553 +
554 +
555 +
556 +
557 +
558 +
559 +
560 +
561 +
562 +
563 +
564 +
565 +
566 +
567 +
568 +
569 +
570 +
571 +
572 +
573 +
574 +
575 +
576 +
577 +
578 +
579 +
580 +
581 +
582 +
583 +
584 +
585 +
586 +
587 +
588 +
589 +
590 +
591 +
592 +
593 +
594 +
595 +
596 +
597 +
598 +
599 +
600 +
601 +
602 +
603 +
604 +
605 +
606 +
607 +
608 +
609 +
610 +
611 +
612 +
613 +
614 +
615 +
616 +
617 +
618 +
619 +
620 +
621 +
622 +
623 +
624 +
625 +
626 +
627 +
628 +
629 +
630 +
631 +
632 +
633 +
634 +
635 +
636 +
637 +
638 +
639 +
640 +
641 +
642 +
643 +
644 +
645 +
646 +
647 +
648 +
649 +
650 +
651 +
652 +
653 +
654 +
655 +
656 +
657 +
658 +
659 +
660 +
661 +
662 +
663 +
664 +
665 +
666 +
667 +
668 +
669 +
670 +
671 +
672 +
673 +
674 +
675 +
676 +
677 +
678 +
679 +
680 +
681 +
682 +
683 +
684 +
685 +
686 +
687 +
688 +
689 +
690 +
691 +
692 +
693 +
694 +
695 +
696 +
697 +
698 +
699 +
700 +
701 +
702 +
703 +
704 +
705 +
706 +
707 +
708 +
709 +
710 +
711 +
712 +
713 +
714 +
715 +
716 +
717 +
718 +
719 +
720 +
721 +
722 +
723 +
724 +
725 +
726 +
727 +
728 +
729 +
730 +
731 +
732 +
733 +
734 +
735 +
736 +
737 +
738 +
739 +
740 +
741 +
742 +
743 +
744 +
745 +
746 +
747 +
748 +
749 +
750 +
751 +
752 +
753 +
754 +
755 +
756 +
757 +
758 +
759 +
760 +
761 +
762 +
763 +
764 +
765 +
766 +
767 +
768 +
769 +
770 +
771 +
772 +
773 +
774 +
775 +
776 +
777 +
778 +
779 +
780 +
781 +
782 +
783 +
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 +
```

```

99 -         $splitIpSubnets = explode(':', $ip);
100
101 -         if (count($splitIpSubnets) === 8) {
102 -             foreach ($splitIpSubnets as
103 $splitIpSubnet) {
104                 if (strlen($splitIpSubnet)
105 > 4) {
106                     return false;
107
108                 }
109
110             }
111
112             $response = $ip;
113
114         }
115
116         return $response;
117     }
118
119     /**
120     * Validate IP address list
121     *
122     * @param array $ips
123     *
124     * @return array $response
125     */
126     protected function _validateIps($ips) {
127         foreach ($ips as $key => $ip) {
128             if (
129
130                 empty($ip) ||
131
132                 strlen($ip) < 7 ||
133                 !($ip = trim($ip, '.')) ||
134                 !($ip = trim($ip, ':')) ||
135
136                 (
137
138                     strpos($ip, ':::')
139 === false &&
140
141                     substr_count($ip,
142 ':') > 4 &&
143
144                     ($ips[$key] =
145 $this->_validateIpv6($ip)) === false
146                 ) ||
147
148                 (
149                     strpos($ip, ':')
150 === false &&
151
152                     ($ips[$key] =
153 $this->_validateIpv4($ip)) === false
154                 ) {
155                 unset($ips[$key]);
156             }
157         }
158     }

```

```

98 +         $splitIpSubnets = explode(':',
99 $ip);
100
101 +         if (count($splitIpSubnets) === 8)
102 {
103 +             foreach ($splitIpSubnets
104 as $splitIpSubnet) {
105                 if
106 (strlen($splitIpSubnet) > 4) {
107                     return
108 false;
109
110                 }
111
112             }
113
114             $response = $ip;
115
116         }
117
118         return $response;
119     }
120
121     /**
122     * Validate IP address list
123     *
124     * @param array $ips
125     *
126     * @return array $response
127     */
128     protected function _validateIps($ips) {
129         foreach ($ips as $key => $ip) {
130             if (
131
132                 empty($ip) ||
133                 strlen($ip) < 7 ||
134                 !($ip = trim($ip,
135 '.')) ||
136                 !($ip = trim($ip,
137 ':')) ||
138
139                 (
140
141                     strpos($ip, ':::') === false &&
142
143                     substr_count($ip, ':') > 4 &&
144
145                     ($ips[$key] = $this->_validateIpv6($ip)) === false
146                 ) ||
147
148                 (
149                     strpos($ip, ':') === false &&
150
151                     ($ips[$key] = $this->_validateIpv4($ip)) === false
152                 ) {
153                 unset($ips[$key]);
154             }
155         }
156
157         $response = implode("\n",
158 array_unique($ips));

```

```

140         }
141
142     -         $response = implode("\n",
array_unique($ips));
143     -         return $response;
144     -     }

145
146     - /**
147     -  * Verify dedicated IPs
148     -  *
149     -  * @param array $ips
150     -  * @param array $userIps
151     -  *
152     -  * @return $response
153     -  */
154     -     public function verify($ips, $userIps) {
155     -         $ips = $this->_parseIps($ips);
156     -         $userIds = array_values($userIps);
157     -
158     -         if (!empty($ips)) {
159     -             $userIdsFormatted = false;
160     -
161     -             do {
162     -                 $userIds =
array_merge($userIds, $userIds);
163     -
164     -                 if
(count($userIds) > 255255) {
165     -                     $userIdsFormatted = true;
166     -                 }
167     -             } while ($userIdsFormatted ==
false);
168     -         }
169
170     -         $ips = explode("\n", $ips);
171
172         +         return $response;
173     }
174
175     + /**
176     +  * Verify dedicated IPs
177     +  *
178     +  * @param array $ips
179     +  * @param array $userIps
180     +  *
181     +  * @return $response
182     +  */
183     +     public function verify($ips, $userIps) {
184     +         $ips = $this->_parseIps($ips);
185     +         $userIds = array_values($userIps);
186     +
187     +         if (!empty($ips)) {
188     +             $userIdsFormatted = false;
189     +
190     +             do {
191     +                 $userIds =
array_merge($userIds, $userIds);
192     +
193     +                 if
(count($userIds) > 255255) {
194     +                     $userIdsFormatted = true;
195     +                 }
196     +             } while ($userIdsFormatted ==
false);
197     +         }
198
199     +         $ips = explode("\n", $ips);
200
201         +         foreach ($ips as $key => $ip) {
202         +             $ips[$ip] = $this->_getId($ip, $userIps, $userIds);

```

<pre> 171 172 -         foreach (\$ips as \$key =&gt; \$ip) { 173 -             \$ips[\$ip] = \$this-&gt;_getUserId(\$ip, 174 -             \$userIps, \$userIds); 175 -             unset(\$ips[\$key]); 176         } 177 -         \$response = array_merge(\$ips, \$userIps); 178 -         return \$response; 179     } 180 - } 181 - } 182 ?&gt; </pre>	<pre> 173 +             unset(\$ips[\$key]); 174 +         } 175 176 +         \$response = array_merge(\$ips, 177 +         \$userIps); 178 +         return \$response; 179     } 180 } 181 ?&gt; </pre>
---	--

▼ 106 data.php

<pre> ... @@ -1,56 +1,56 @@ 1 &lt;?php 2 - /** 3 -  * IP list to check separated by new line 4 -  * 5 -  * Reserved IP addresses below are used for demo purposes 6 -  * 7 -  */ 8 -     \$ipListToCheck = implode("\n", array( 9 -         '2001:db8:abcd:0008:847g:3e2:1088:8888', 10 - 11 -         '2001:db8:abcd:0008::ffff:ffff', 12 -         '2001:db8:abcd:0008:0000:0000:8888:ffff', 13 - 14 -         '2001:db8:abcd:0008::1234:8888:FFFF', 15 - 16 -         '172.16.88.10', 17 -         '172.16.8.3', 18 -         '172.16.200.3', 19 -         '172.16.201.4', 20 -         '192.168.0.0', 21 -         '192.168.0.1', 22 -         '192.168.0.2', 23 -         '192.168.0.3', 24 -         '192.168.0.4', 25 -         '192.168.40.4', 26 -         '192.168.49.4', 27 -         '10.5.30.103', 28 -         '10.5.30.105', 29 -         '10.5.30.106', 30 -         '10.5.31.108', 31 -         '_.10.5.30.199_*', // Invalid IP formats 32 -         '10.5.89.44', 33 -         '10.8.8.100', 34 -         '10.8.8.101', 35 -         '10.8.8.102', 36 -         '10.8.8.103' 37 -     )); 38 - /** 39 -  * List of dedicated IP addresses that belong to each 40 -  * user (IP =&gt; ID format) </pre>	<pre> 1 &lt;?php 2 + /** 3 +  * IP list to check separated by new line 4 +  * 5 +  * Reserved IP addresses below are used for demo 6 +  * purposes 7 +  * 8 +  */ 9 +     \$ipListToCheck = implode("\n", array( 10 +         '2001:db8:abcd:0008:847g:3e2:1088:8888', 11 +         '2001:db8:abcd:0008::ffff:ffff', 12 +         '2001:db8:abcd:0008:0000:0000:8888:ffff', 13 +         '2001:db8:abcd:0008::1234:8888:FFFF', 14 +         '172.16.88.10', 15 +         '172.16.8.3', 16 +         '172.16.200.3', 17 +         '172.16.201.4', 18 +         '192.168.0.0', 19 +         '192.168.0.1', 20 +         '192.168.0.2', 21 +         '192.168.0.3', 22 +         '192.168.0.4', 23 +         '192.168.40.4', 24 +         '192.168.49.4', 25 +         '10.5.30.103', 26 +         '10.5.30.105', 27 +         '10.5.30.106', 28 +         '10.5.31.108', 29 +         '_.10.5.30.199_*', // Invalid IP 30 +         '10.5.89.44', 31 +         '10.8.8.100', 32 +         '10.8.8.101', 33 +         '10.8.8.102', 34 +         '10.8.8.103' 35 +     )); 36 + /** 37 +  * List of dedicated IP addresses that belong to 38 +  * each user (IP =&gt; ID format) </pre>
---	--

```

38 - * Users should have access to see their own user ID
39 - *
40 - * Reserved IP addresses below are used for demo purposes
41 - *
42 - */
43 -     $userIps = array(
44 -         '2001:db8:abcd:0008:0000:0000:8888:ffff' =>
         '5c9518c1-0ad8-4e41-80a1-5bd54221ccee',
45 -         '192.168.0.2' => '5c95396e-97c8-49af-abb3-
17e04221ccee',
46 -         '192.168.0.3' => '5c95396e-97c8-49af-abb3-
17e04221ccee',
47 -         '10.5.31.108' => '5c95354f-43c8-4976-b69d-
12334221ccee',
48 -         '172.16.88.10' => '5c953514-e9e8-4cca-954c-
12334221ccee',
49 -         '172.16.200.3' => '5c953514-e9e8-4cca-954c-
12334221ccee',
50 -         '10.5.89.44' => '5c952170-f908-45e9-af43-
6bf64221ccee',
51 -         '10.5.30.199' => '5c9518c1-0ad8-4e41-80a1-
5bd54221ccee',
52 -         '172.16.8.3' => '5c951454-0418-4ca5-80f8-
5b6b4221ccee',
53 -         '192.168.40.4' => '5c951242-56a9-401a-8d74-
5b7b4221ccee',
54 -         '10.8.8.102' => '5c95396e-97c8-49af-abb3-
17e04221ccee'
55 -     );
56     ?>

```

```

38 + _____ * Users should have access to see their own user
ID
39 + _____ *
40 + _____ * Reserved IP addresses below are used for demo
purposes
41 + _____ *
42 + _____ */
43 +     _____ $userIps = array(
44 +         _____
         '2001:db8:abcd:0008:0000:0000:8888:ffff' => '5c9518c1-0ad8-
4e41-80a1-5bd54221ccee',
45 +         _____ '192.168.0.2' => '5c95396e-97c8-
49af-abb3-17e04221ccee',
46 +         _____ '192.168.0.3' => '5c95396e-97c8-
49af-abb3-17e04221ccee',
47 +         _____ '10.5.31.108' => '5c95354f-43c8-
4976-b69d-12334221ccee',
48 +         _____ '172.16.88.10' => '5c953514-e9e8-
4cca-954c-12334221ccee',
49 +         _____ '172.16.200.3' => '5c953514-e9e8-
4cca-954c-12334221ccee',
50 +         _____ '10.5.89.44' => '5c952170-f908-
45e9-af43-6bf64221ccee',
51 +         _____ '10.5.30.199' => '5c9518c1-0ad8-
4e41-80a1-5bd54221ccee',
52 +         _____ '172.16.8.3' => '5c951454-0418-
4ca5-80f8-5b6b4221ccee',
53 +         _____ '192.168.40.4' => '5c951242-56a9-
401a-8d74-5b7b4221ccee',
54 +         _____ '10.8.8.102' => '5c95396e-97c8-
49af-abb3-17e04221ccee'
55 +     _____ );
56     ?>

```

0 comments on commit `e284652`