VelocityReport Plugin - Defect #618

Array in object not an array but java.util.AbstractList

12/31/2012 12:22 PM - Robert Ivens

Status:	Resolved	**Start date:**	12/31/2012
Priority:	High	**Due date:**	
Assignee:		**% Done:**	0%
Category:	velocity	**Estimated time:**	0.00 hour
Target version:	v3.0		
Browser (if web client):			

Description

I have a context where I have an array with objects and those objects contain an array. Velocity doesn't see this last array as a JavaScript array but as a java.util.AbstractList object.
Something I can't use in Velocity it seems.

This is my vr_getContext method:

```
function vr_getContext(request) {
    var context = {
            test1: {
                object1: {
                    object2: {
                        array1: [{ values: ['value1', 'value2'] }]
                    }
                }
            },
            test2: {
                array1: [{ values: ['value1', 'value2'] }]
            },
            test3: [['value1', 'value2']]
        };

    return plugins.Velocity.createResponse(context);
}
```

This my template:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="utf-8" />
    <title>Test</title>
</head>
<body>
<p>
<table>
    <tr><thead colspan="2">Test 1</thead></tr>
    <tr><td>test1.Object1</td><td>:$test1.object1</td></tr>
    <tr><td>test1.Object2</td><td>: $test1.object1.object2</td></tr>
    <tr><td>test1.Array1</td><td>: $test1.object1.object2.array1</td></tr>
    <tr><td>test1.Array1[0]</td><td>: $test1.object1.object2.array1[0]</td></tr>
    <tr><td>test1.Array1[0].values</td><td>: $test1.object1.object2.array1[0].values</td></tr>
    <tr><td>test1.Array1[0].values[0]</td><td>: $test1.object1.object2.array1[0].values[0]</td></t
r>
</table>
</p>
<p>
<table>
    <tr><thead colspan="2">Test 2</thead></tr>
```

```
    <tr><td>test2.Array1</td><td>: $test2.array1</td></tr>
    <tr><td>test2.Array1[0]</td><td>: $test2.array1[0]</td></tr>
    <tr><td>test2.Array1[0].values</td><td>: $test2.array1[0].values</td></tr>
    <tr><td>test2.Array1[0].values[0]</td><td>: $test2.array1[0].values[0]</td></tr>
</table>
</p>
<p>
<table>
    <tr><thead colspan="2">Test 3</thead></tr>
    <tr><td>test3</td><td>: $test3</td></tr>
    <tr><td>test3[0]</td><td>: $test3[0]</td></tr>
    <tr><td>test3[0][0]</td><td>: $test3[0][0]</td></tr>
</table>
</p>
</body>
</html>
```

This will return the following output:

```
Test 1
test1.Object1    :{object2: {array1: [{values: [value1, value2]}]}}
test1.Object2    : {array1: [{values: [value1, value2]}]}
test1.Array1    : [{values: [value1, value2]}]
test1.Array1[0]    : {values: [value1, value2]}
test1.Array1[0].values    : java.util.AbstractList$Itr@5baf4de3
test1.Array1[0].values[0]    : $test1.object1.object2.array1[0].values[0]

Test 2
test2.Array1    : [{values: [value1, value2]}]
test2.Array1[0]    : {values: [value1, value2]}
test2.Array1[0].values    : java.util.AbstractList$Itr@1cef0866
test2.Array1[0].values[0]    : $test2.array1[0].values[0]

Test 3
test3    : [[value1, value2]]
test3[0]    : [value1, value2]
test3[0][0]    : value1
```

Here you can see that test3 does return the array as an array when used inside another array but test1 and test2 show that when the second array is inside an object you get something completely different.

History

#1 - 01/02/2013 06:01 PM - Patrick Talbot

- Status changed from New to In Progress

It's weird that it would happen after a certain level of inclusion, this should be recursive.
I'll have another look.

#2 - 01/02/2013 10:29 PM - Patrick Talbot

- Status changed from In Progress to Resolved

"values" was actually a reserved word, because of the presence of a public Iterator<Object> getValues() method in the ScriptableWrapper, so it was returning an internal List.Iterator when called.
Changing the name of your variable to "vals" for example shows that it's not a recursion issue.

I've addedd getValues() a while ago because I thought it could be good to have, there's also a getProperties() method that you can access to iterate on all the properties of an object (meaning that "properties" is also a reserved keyword).

I've removed the getValues() as this can be easily achieved with a simple foreach, so you will be able to use 'values' inside an object with the next version.

I've kept getProperties() though as it can be convenient to loop through all the properties of a JavaScript object with #foreach($property in $obj.properties)