VelocityReport Plugin - Feature #253

wrap #parse in a $plugin.parse to allow passing of parameters

03/21/2011 11:37 AM - Patrick Ruhsert

Status:	Closed	**Start date:**	03/21/2011
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Browser (if web client):			

Description

Hello,

I have the following situation: I want to print a report on a record that has related data from several different tables. I have managed to create a simple list layout that dynamically creates rows and cells on the dataproviders I provider. The logic is basically this

#foreach($record in $foundset)
<tr>
#foreach($field in $dataproviders)
<td>$!record[$field.column_name]</td>
#end
</tr>
#end

in the example above, $dataproviders is a dataset with the names of the dataproviders I want (among other things).

Now back to my report. I would like to pass an array of relation names along with the dataproviders for each relation to my report and iterate over the relations. Then, with #parse I could use ONE list template for the related details, no matter what relation that is. Currently, I don't see how I can do that. If, however, I could pass parameters to the #parse (for example the related foundset and the dataproviders I want), then I would only need two simple HTML files. Now, I think, I need to create a list HTML for every relation I want to include.

Would that be possible?

Thanks
Patrick

History

#1 - 03/21/2011 01:10 PM - Patrick Talbot

- Status changed from New to Closed

#parse include files share the same context as the main template.
But each variable that is set at the moment of the #parse include is going to be available to the sub template.

Now what you could do is have an array of objects with relation name + dataproviders.
Say something like:

```
var relations = [];
relations.push( { related: 'relation_to_whatever', providers: ['field1', 'field2', 'field3'] } );
relations.push( { related: 'relation_to_another', providers: ['another1', 'another2'] } );
```

Then in the main template iterate on this array of objects, setting a unique object to that relation object:

```
#foreach($record in $foundset)
    #foreach($relation in $relations)
        #parse('sub_report.html')
    #end
#end
```

Inside the sub report, you should then be able use:

```
#foreach($provider in $relation.providers)
    $!record[$related][$provider] or $!record.get($related).get($provider)
#end
```

I haven't tested it but this logic should give you the result you're looking for...
If this is not working for you, please feel free to reopen the issue.

#2 - 03/21/2011 02:11 PM - Patrick Ruhsert

Oh yes! Excellent. Wasn't aware of the fact that the parsing is done with the whole context at hand. Doing my subreport like this and it works like a charm! Thanks for the pointer.

<table cellspacing="0">
<thead>
<tr>
#foreach($key in $relation.i18n_keys)
<td>$i18n.get($key)</td>
#end
</tr>
</thead>
<tbody>
#foreach($relatedRecord in $record[$relation.relation_name])
<tr class="$alternator">
#foreach($relatedField in $relation.dataproviders)
<td>$!relatedRecord[$relatedField]</td>
#end
</tr>
#end
</tbody>
</table>