Allow macro content inside any other macro

This page describes a way of allowing arbitrary Confluence content, including macros, in the bodies of macros that don't normally allow this.

Introduction

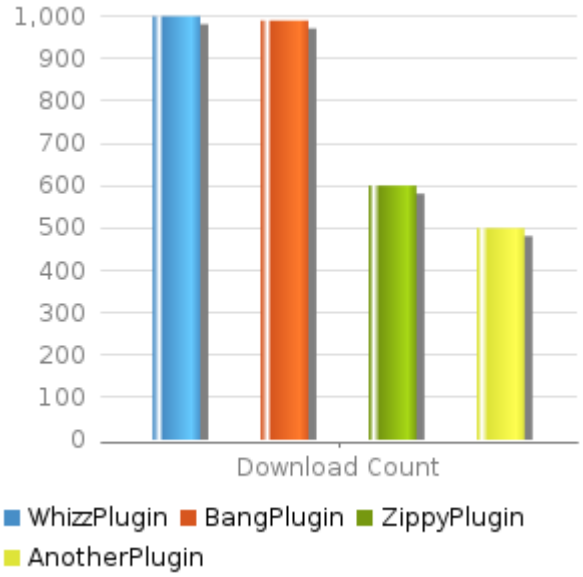
Confluence macros can be nested within other macros. For instance, the `chart` macro can take a static table as input:

Macro

Chart | type = bar | dataOrientation = horizontal

Plugin	Download Count
WhizzPlugin	1000
BangPlugin	990
ZippyPlugin	600
AnotherPlugin	500

Result



But the `chart` macro can also take output from a SQL macro:

Macro

Chart | type = bar | dataOrientation = horizontal

SQL Query | dataSource = PluginStats

```
SELECT name AS Plugin,
    downloadcount AS "Download Count"
FROM plugin
ORDER BY 2 DESC
LIMIT 5;
```

Result

In fact `chart` can take input from any other table-generating macro (e.g. CSV).

Like Unix pipes, you can string together and reuse macros. This **plugin composability** is one of Confluence's most powerful features.

This is great, but only works when the macro in question was designed to accept the "rich text" XHTML generated by another macro. For example, but the `SQL Query` macro can't take macro output as its input.

Dynamic Wrapper Macros

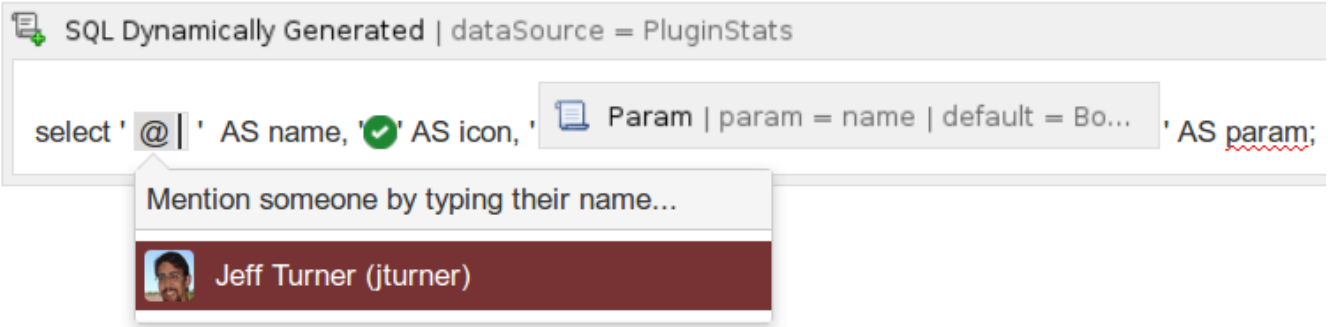
Why would we want to send dynamic content to a macro not expecting it? Here are a few situations:

- You are using the SQL macro to query a database, but want to use a HTTP parameter from the URL (param macro) in the query to make it dynamic. This is a simple alternative to the Self-Service Reports Plugin.
- You want to use SQL results in Javascript in a `html` macro. For instance, one might like to use a modern Javascript graphing library like Chart.js, fed with data from your SQL back-end.

The solution is to create a wrapper user macro that accepts rich text input, strips out any XHTML tags, and feeds the resulting plain text to the macro you're interested in.

sqlquery_dynamic

The first example lets the SQL Query macro take dynamic input. Here is a silly example, with SQL that emits XHTML for a Confluence @user reference, (/) tick and a custom `param` user macro:



rendering as:

name	icon	param
@Jeff Turner		bob

> This is a silly example not least because of the SQL injection attack using `param` like this involves.

The macro definition looks like this:

Edit User Macro

Macro Browser Information

Macro Name* sqlquerydynamic

Macro names are converted to lower case

Visibility
- (●) Visible to all users in the Macro Browser
- ( ) Visible only to system administrators in the Macro Browser

Macro Title* SQL Dynamically Generated

Description Subset of the SQL Query macro that accepts nested macros in the SQL body.

Categories
Administration
Communication
Confluence content
Development
External content

Icon URL /images/icons/add_user_macro_16.png

Absolute URL (e.g. http://mysite.com/myimage.png), or relative to the Confluence base URL (e.g. /images/logo/confluence_48_white.png)

Documentation URL

Definition of User Macro

Macro Body Processing
- ( ) No macro body
- ( ) Escaped
- ( ) Unrendered

  You should use this option for bodies that are processed within the template before being output. Ensure that HTML is ultimately output by the template.
- (●) Rendered

  The body will be rendered so most HTML entered will be passed to the template unmodified but Confluence specific mark up such as macro definitions will be rendered.

With macro body:

```
## @param dataSource:title=Data source name|type=string|required=true|desc=Type or select a data source
configured by your Confluence administrator.
## @param output:title=Output format|type=enum|default=html|enumValues=html,xhtml,wiki,unrenderedWiki
## @param table:title=Show result as a table|type=boolean|default=true
## @param printsql:title=Print SQL after results|type=boolean|default=true
## @param autototal:title=Auto row total|type=boolean|default=false|desc=Adds a row to the end of the table
that totals numeric columns.
## @param columnTypes:title=Column Types|type=string|desc=Comma separated list of type indicators. Column type
determines sorting and other characteristics. See https://bobswift.atlassian.net/wiki/display/TBL
/Common+Table+Capabilities

## Given rendered macro output (HTML), strips HTML formatting elements, leaving just the content.
#macro(clean $body)
#foreach($i in [1..10])
## (?:..) is a non-capturing group.
#set($body=$body.replaceAll('(?smi)<span(?: [^>]*)?>(.*?)</span>', '$1'))
#set($body=$body.replaceAll('(?smi)<p(?: [^>]*)?>(.*?)</p>', ' $1'))
#set($body=$body.replaceAll('(?smi)<div(?: [^>]*)?>(.*?)</div>', '$1'))
#set($body=$body.replaceAll('(?smi)<p(?: [^>]*)?>(.*?)</p>', ' $1'))
#end
#set($body=$body.replaceAll('<br ?/>', '
'))
#set($body=$body.replaceAll('&apos;', "'"))
#set($body=$body.replaceAll('&lt;', "<"))
#set($body=$body.replaceAll('&gt;', ">"))
#set($body=$body.replaceAll('&quot;', '"'))
#set($body=$body.replaceAll('&;', '"'))
$body#end
##
<ac:structured-macro ac:name="sql-query">
  <ac:parameter ac:name="output">$paramoutput</ac:parameter>
  <ac:parameter ac:name="dataSource">$paramdataSource</ac:parameter>
  <ac:parameter ac:name="atlassian-macro-output-type">INLINE</ac:parameter>
  <ac:parameter ac:name="table">$paramtable</ac:parameter>
  #if($paramcolumnTypes)<ac:parameter ac:name="columnTypes">$paramcolumnTypes</ac:parameter>#end
  <ac:parameter ac:name="rowStyles">border-bottom:black 2px solid;border-top:black 2px solid;,background:
#ffffee,background:#fff</ac:parameter>
  <ac:parameter ac:name="autoTotal">$paramautototal</ac:parameter>
  <ac:parameter ac:name="noDataMessage">no results</ac:parameter>
  <ac:plain-text-body><![CDATA[#clean($body)]]></ac:plain-text-body>
</ac:structured-macro>

#if($paramprintsql == true)
<small>Rendering SQL: $generalUtil.htmlEncode($body)</small><br>
#end
```