One major enhancement in Joomfish is that it is possible to add translation filters to content element files. For example look at the full version of content.xml:

<?xml version="1.0" ?>
<joomfish type="contentelement">
<name>Contents</name>
<author>A. Kempkens</author>
<version>1.7</version>
<description>Definition for the core content component</description>
<reference type="content">
<table name="content">
<field type="referenceid" name="id" translate="0">ID</field>
<field type="titletext" name="title" translate="1">Title</field>
<field type="text" name="title_alias" translate="0">Title Alias</field>
<field type="htmltext" name="introtext" translate="1">Introtext</field>
<field type="htmltext" name="fulltext" translate="1">Fulltext</field>
<field type="text" name="images" translate="1">Images</field>
<field type="text" name="metakey" translate="1">Metakey</field>
<field type="text" name="metadesc" translate="1">Metadesc</field>
<field type="created_date" name="created" translate="0">Created</field>
<field type="modified_date" name="modified" translate="0">Modified</field>
<field type="checked_out_by" name="checked_out" translate="0">Check out by</field>
<field type="checked_out_date" name="checked_out_time" translate="0">Check out date</field>
<filter>c.state >= 0</filter>
</table>
</reference>
<translationfilters>
<category>catid</category>
<author>created_by</author>
<keyword>title</keyword>
<published>published</published>
</translationfilters>

</joomfish>

The section in red tells Joomfish to implement a category name filter where the database content table field representing the category id is *catid* and so on.

The builtin filters (that can be used for other content types) are:

filter type	description
category	filter based on category name
author	filter based on author name
keyword	filter based on keyword value. The field in this case is the field to search for the
published	filter based on whether a translation has been published (in a given language)

It is possible to add custom filters for 3rd party components. Consider the example of VirtueMart products. The contentelement file reads as follows:

<?xml version="1.0" ?> <joomfish type="contentelement"> <name>VirtueMart Product</name> <author>EI</author> <version>1.0</version> <description>Definition for VirtueMart component (Products)</description> <reference> <table name="vm_product"> <field type="referenceid" name="product_id" translate="0">ID</field> <field type="titletext" name="product_name" translate="1">Title</field> <field type="textarea" name="product_s_desc" translate="1">Short description</field> <field type="htmltext" name="product_desc" translate="1">Description</field> <field type="created_date" name="cdate" translate="0">Created</field> <field type="modified_date" name="mdate" translate="0">Modified</field> </table> </reference> <translationfilters> <vm_category>category_id</vm_category> </translationfilters> </joomfish>

You only need to add a file called **translationVm_categryFilter.php** into the content element folder and your translation filters are automatically implemented. As an example this file is attached.

Rules:

- The file and classname should start with "translation" and end with "Filter"
- The middle portion of the name should match the xml tag in the contentelement file
- 3 Methods should be implemented:

contructor	This can generally be modelled on other examples, changing the names accord
_createFilter	This is a portion of an SQL query that finds translations. The return value will be
createfilterHTML	This created the HTML part of the filter, changing for textbox, select list etc. Use th

Enjoy !!

Geraint Edwards 27 March 2006

```
 <?php
class translationVm_categoryFilter extends translationFilter
{
 function translationVm_categoryFilter ($contentElement){
  $this->filterNullValue=-1;
  $this->filterType=&quot;vm_category&quot;;
  $this->filterField = $contentElement->getFilter(&quot;vm_category&quot;);
  parent::translationFilter($contentElement);
 }

 function _createFilter(){
  global $database;
  if (!$this->filterField ) return &quot;&quot;;
  $filter=&quot;&quot;;
  if ($this->filter_value!=$this->filterNullValue){
   // get list of product_id in the appropriate category
   $sql = &quot;SELECT xref.product_id FROM #__vm_product_category_xref as xref&quot;
   .&quot; WHERE xref.category_id=$this->filter_value&quot;;
   $database->setQuery($sql);
   $prodids = $database->loadObjectList();
   $idstring = &quot;&quot;;
   foreach ($prodids as $pid){
    if (strlen($idstring)>0) $idstring.=&quot;,&quot;;
    $idstring.=$pid->product_id;
   }
   $filter = &quot;c.product_id IN($idstring)&quot;;
```

```
 }
 return $filter;
}
 /**
* Creates vm_category filter
*
* @param unknown_type $filtertype
* @param unknown_type $contentElement
* @return unknown
*/
function _createfilterHTML(){
 global $database;  if (!$this->filterField) return &quot;&quot;;
 $categoryOptions=array();
 $categoryOptions[] = mosHTML::makeOption( '-1',_JOOMFISH_ADMIN_CATEGORY_ALL );
$sql = &quot;SELECT DISTINCT cat.category_id, cat.category_name FROM #__vm_category
as cat,&quot;
    .&quot; #__&quot;.$this->tableName.&quot; as c, #__vm_product_category_xref as
xref&quot;
    .&quot; WHERE c.product_id=xref.product_id AND
xref.&quot;.$this->filterField.&quot;=cat.category_id ORDER BY cat.category_name&quot;;
 $database->setQuery($sql);
 $cats = $database->loadObjectList();
 $catcount=0;
 foreach($cats as $cat){
  $categoryOptions[] = mosHTML::makeOption( $cat->category_id,$cat->category_name);
  $catcount++;
 }
 $categoryList=array();
 $categoryList[&quot;title&quot;]=_JOOMFISH_ADMIN_CATEGORY;
 $categoryList[&quot;html&quot;] = mosHTML::selectList( $categoryOptions,
'vm_category_filter_value', 'class=&quot;inputbox&quot; size=&quot;1&quot;
onchange=&quot;document.adminForm.submit();&quot;', 'value', 'text', $this->filter_value );
return $categoryList; }}?>
```