# Title Manager

# Table of Contents

# Title Manager Documentation

## About this manual

This manual contains both User's Guide and Developer's Guide.

Release version: 1.1
Release date: 07.09.2009



## User's Guide

### Getting started

#### What is Title Manager?

Title Manager is a native Joomla 1.5 extension (component + plug-in) that allows you to easily manage the title of your Joomla-based web site. The 'title' of a web site is the text that appears in the user's browser when the web site is currently displayed. The title is also used in the Windows taskbar and browser tab to identify the browser window. Search engines use titles to identify what the current page is about, making it really important to use the right title.

The Joomla CMS does not currently provide a way for an administrator/manager to control the title of the web site. There is a default behavior for assembling the title. It almost always uses the name of the current menu item. In some cases, the name of the current article is also used. Although in rare cases this might be sufficient, it is more likely to result in inadequate titles shown to the web site client.

Title Manager provides an easy-to-use administrative interface that allows the web site administrator/manager to control the web site title without knowing about the internal Joomla architecture or having any programming skills.

#### Features

- **Pattern model** - very flexible model for defining titles.
- **Default title patterns starting/ending** - allows setting default title starting/ending part that is active for all title patterns.
- **Not bound to a list of 'supported components'** - this management system could be used with any Joomla component.

- **Integration with Advanced Contexts Manager** - provides the ability to define titles based on contexts. For more information on contexts and the Advanced Contexts Manager extension, click here.
- **Integration with Content Enhancer** - provides the ability to use variables and automatic replacements in titles. For more information on variables, replacements and the Content Enhancer product, click here.
- **Integration with Breadcrumbs Manager** - allows you to use breadcrumbs items in titles. For more information on breadcrumbs items and the Breadcrumbs Manager extension, click here.
- **Easily extendable** - created using a simple plug-in architecture to allow further user customization.
- **Easy to use administrative interface** - standard Joomla-style administrative interface.
- **No changes in core files** - allows you to update your Joomla installation without worrying that something may break.

## Installation and licensing

### How to install

All of our Joomla extensions use the standard Joomla mechanism for installation and upgrade. After you download the product archive, you must install it from the Joomla administration panel as you would install any other extension. For more information, please look in the Joomla! manual.

To use the product you will also need to install ionCube loader. Installation is described here. After the ionCube loader is working you will need to get a license for the product.

Note, that some of our products consist of more than one extension. For example, a component is often used with a plug-in that handles its functionality or a module. To make it easier for you, we have created a special mechanism for installing all of the extensions from one archive. Don't be surprised if you find components, modules or plug-ins that you have not explicitly installed.

### Installing IonCube

All of our PHP-based products (Joomla! extensions included) use the ionCube PHP Encoder for source code protection and licensing. In order to use our products you will need an appropriate version of the ionCube Loader to be installed on your hosting server. Note, that the ionCube Loader is completely free. You don't have to pay for it.

If you use shared hosting, you may have the ionCube loader installed by default. If it is not installed, you have two choices. If the server configuration allows it, you may be able to install the loader by yourself. If not, you should ask the administrators of your server

to do it for you. IonCube is a popular product and your hosting should have no concerns about installing it.

More information on installing ionCube loaders can be found here.

## How to uninstall

Uninstalling one of our Joomla extension products is done through the standard Joomla interface. Note, however, that most of our products consist of more than one Joomla extension. Even when you install a product from a single archive it may internally install multiple extensions. We have created a special mechanism to allow this in order to ease your installation process. When uninstalling such product you must uninstall the main component of the product. Doing this will remove all related plug-ins, modules and components as well.

Note that, even when you uninstall the product in the way that was described, some files may not be removed. There are some common files that we use in a number of our products and there is no reliable way to know if they are used by another product or not. These files are located in folder "joomla_root/administrator/com_es_joomla_common". If you are sure you do not use any of our products, you may remove this folder.

## Licensing

To use the product you will need a license file. If the license file is missing, a license acquire page will be displayed when you try to access the main component of the product. This page contains instructions which will guide you through the license generation process. It is a simple process and only takes seconds to complete.

#### License generation types.

There are two ways to generate a license - automated and manual.

##### Automated license generation.

The automated procedure is the recommended way to get a license. It is embedded in our products so that you can use it directly from your web site. To use it your server needs to be connected to the internet. The procedure compiles data (domain and IP address) for your server and sends it to our server. Our server responds with a license file. The license file is downloaded automatically in the right location.

##### Manual license generation.

If your server is not connected to the Internet or there is a problem with the automated procedure, you can always get a license file manually. This is done through our web site, so you still need Internet connection, but your server doesn't.

You can manually generate a license from here.

The manual process in our web site ends with a download link to a license file. You will need to put it in the right location on your server. This location is specified in the "No license found" page displayed by Joomla when you try to access the extension and don't have a license. This location is almost always the Joomla root folder.

#### License types.

There are two types of licenses that we use for our products - evaluation and full.

#### Evaluation license.

An evaluation license is a fully functional license for a limited time period. It is meant only for evaluating the product and will stop working after that period has passed. After that you can always purchase the product and continue using it without loosing any data or settings. You can obtain a free evaluation license for any of our products. **We strongly recommend** you to try our products before you purchase. This way you can be sure they work as you expect.

#### Full license.

The full license is the license that you will be able to acquire once you have purchased the product. To get it you will need to supply the purchase key that was sent to you in the "Successful purchase" e-mail.

**Important:** you may only request a limited number of full license files with your key. That depends on the license of the product you purchased. Be sure to use all your keys only from the real web site (hosting server), not from test installations.

If you have any questions or difficulties with the license generation process, please contact us.

## How it works

### Basics

To be able to define how a title of a page is assembled, Title Manager uses patterns. A pattern consists of one or more ordered pattern items.

**Example pattern items**:

- name of the current article
- name of the web site
- other predefined values
- user defined values

Each title item is evaluated to one or more text values (title parts). For example, the "[name of the web site]" item is evaluated to "My Site" (assuming the name of your site is "My Site").

**Example pattern:** [name of the web site] -> [name of article]
If we assume that your site's name is "My Site" and the current page displays the article "What's new?", you will get the following title:

**Result title:** My Site -> What's New

The pattern is evaluated for each page in your site to assemble the title. It is possible to have different patterns for different pages from your web site.

## Patterns

#### Patterns explained

The title pattern defines the parts of which the title of the web site is assembled. We call those parts "pattern items" (or "title items").

When a page from your site is requested, the active title pattern is transformed to text by evaluating its items. The resulting text is then set as title of the page.

The pattern model has two types of patterns - Global and Special.

#### Global patterns

Global patterns are ordered and the first appropriate pattern is used when displaying a page. Whether a pattern is "appropriate" is defined by its strict property and the current value of its items. For more information, see "Strict patterns". A non-strict pattern is always appropriate.

#### Special patterns

Special patterns are not used in order, but are bound to specific places in the web site.

#### Using multiple patterns

It is possible to use different patterns for different pages in your web site. This is accomplished through one of the following methods:

- Strict patterns
- Static pattern items
- Pattern binding*

* Only available if the Advanced Contexts Manager product is installed.

#### Strict patterns

As you might already know, the general-type patterns are ordered. The strict property of the patterns is used to provide a way to skip a pattern and try the next in the list.

Strict patterns are patterns that require all of their items to be successfully evaluated. If any of the items could not be evaluated, the next pattern is tried. This continues until a pattern is found which is either non-strict or all of its items were evaluated successfully. If no appropriate pattern is found, the default Joomla title is used.

A good example of a pattern item that could not always be evaluated is the built-in "Current article name" item. It evaluates to the name of the currently displayed article. However, if the requested page is not an article page, the item could not be evaluated. If the pattern in which this item is used is marked as strict, it will be skipped, because it requires all items to be valid. If it is non-strict, the title will be assembled as if the failed item was not in the pattern.

#### Pattern binding

Pattern binding is one of the features that is designed for integration with the Advanced Contexts Manager product.

When you have both products installed and working, it is possible to bind a title pattern to one or more "contexts". The pattern will be used when those contexts are active. If more patterns are found for the currently active contexts, the pattern for the context with the highest priority is used.

Binding is the most flexible mechanism to define where a pattern is used. "Context" is a term used in the Advanced Contexts Manager product. It means a part (or section) of the web site that is defined by the user, using a rich set of conditions. For more information about Advanced Contexts Manager, see "Related products".

### Pattern items

#### Pattern items

The "pattern item" (or "title item") is a definition of how to get one or more text values. These text values are used as parts of the title.

A pattern consists of ordered pattern items. One pattern item could be used in many patterns, which is good in terms of reusability. The pattern item is the smallest, indivisible part of the web site's title.

Title Manager provides some built-in pattern items and also gives you the ability to define your own, custom items. You can see a list of the built-in pattern items here.

The most important thing in the definition of a custom pattern item is its value. The value is set using value type and parameters. Each value type has its own set of parameters that the user can set. For example, the built-in SQL value type has a "Query" parameter - the query that is to be executed to obtain the value. You can see a list of the built-in value types here.

#### Static pattern items

It is possible to define static pattern items. They are not explicitly used in patterns. Instead, they are automatically inserted in all patterns in a user-defined position. The position could be specified as "First", "Last" or "Custom", where "Custom" could be a positive or negative number indicating where in the pattern to insert the value.

You can set the static item value to be either a normal pattern item or a whole pattern. You can also restrict the patterns in which the static item will be used. If the currently used pattern is not selected, the static item will not be displayed.

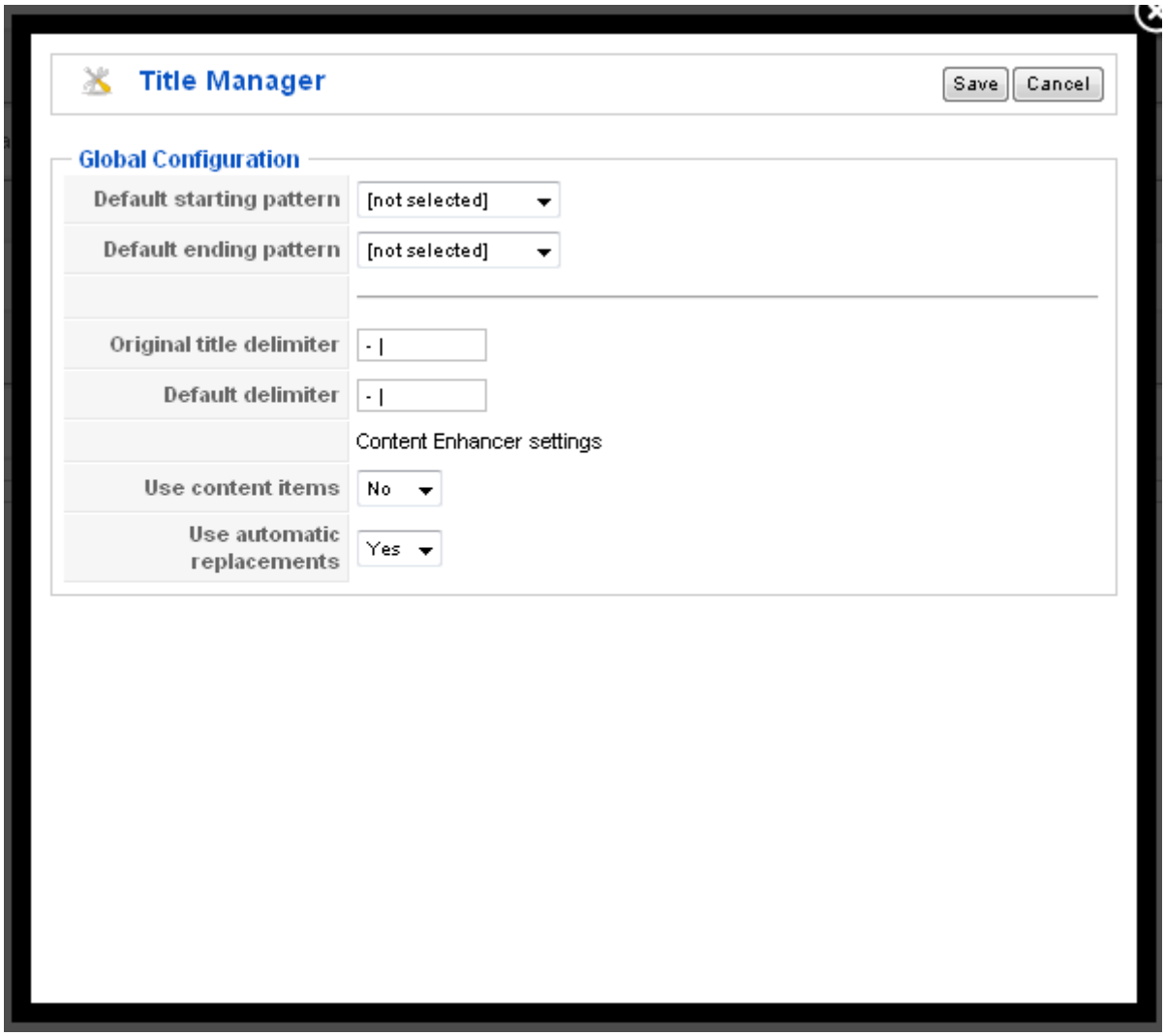
What makes static items very flexible is the possibility to bind them to URL or context*. You can directly copy and paste the query part of a Joomla URL to restrict the static item to the corresponding page only.

* Only available if the Advanced Contexts Manager product is installed

## User interface

### Component settings

Global settings for the Title Manager product are available from the "Parameters" toolbar button.




**Default starting pattern** - you may choose a pattern that is automatically inserted in front of the active pattern. Note that a pattern could be configured not to use the default pattern.

**Default ending pattern** - you may choose a pattern that is automatically appended to the end of active pattern. Note that a pattern could be configured not to use the default pattern.

**Original title delimiter** - this sets the delimiter that Joomla uses for titles. It is used to break the default Joomla title to parts. You probably would never need to change this value.

**Default delimiter** - the delimiter used between different parts of the title. This is the default value. Each pattern may define its own delimiter.

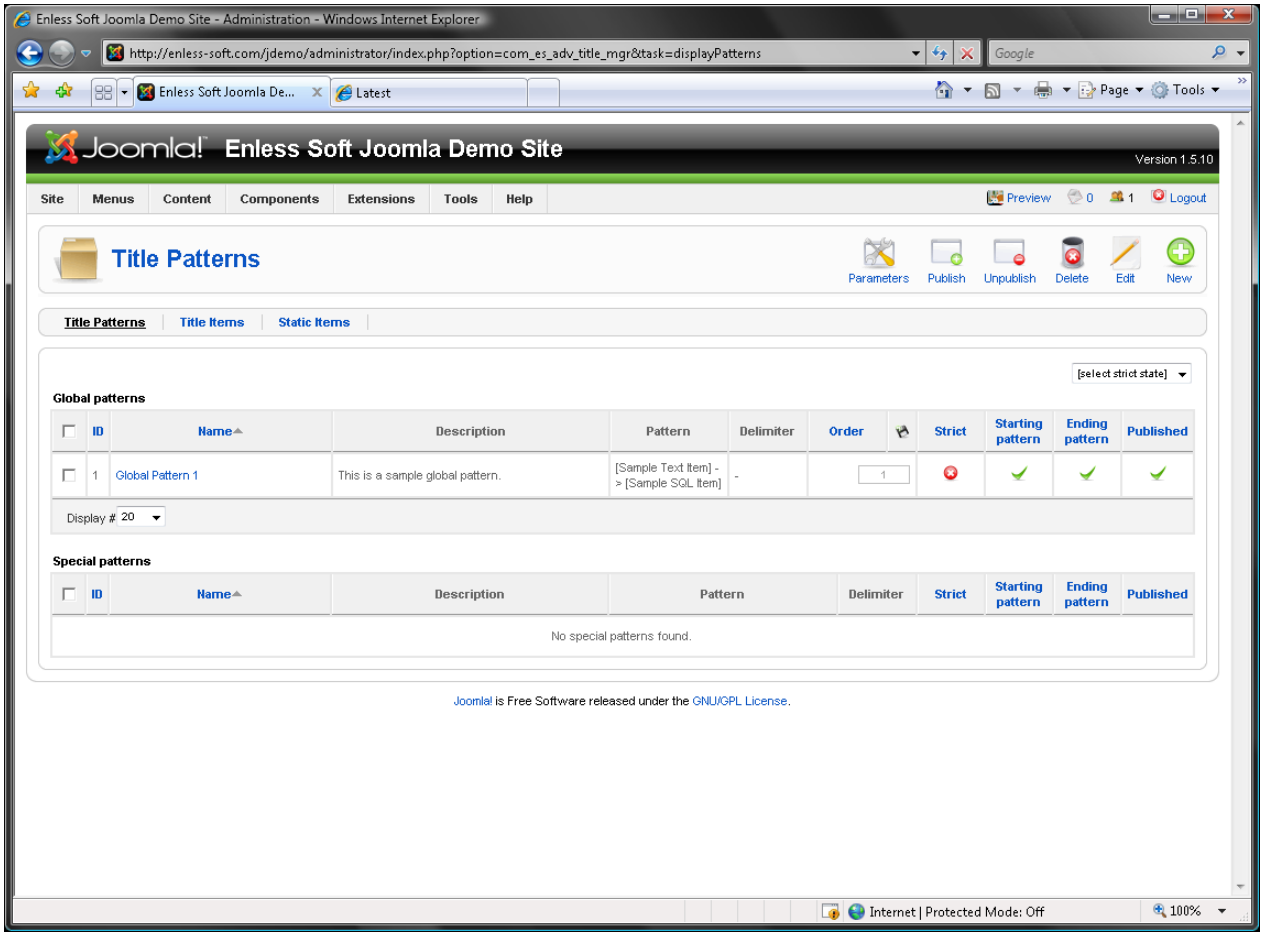
**Use content items** - whether to parse content items (variables) found in the title. This value is only used if the Content Enhancer product is installed.

**Use automatic replacements** - whether to apply automatic replacements to the title. This value is only used if the Content Enhancer product is installed.

## Patterns

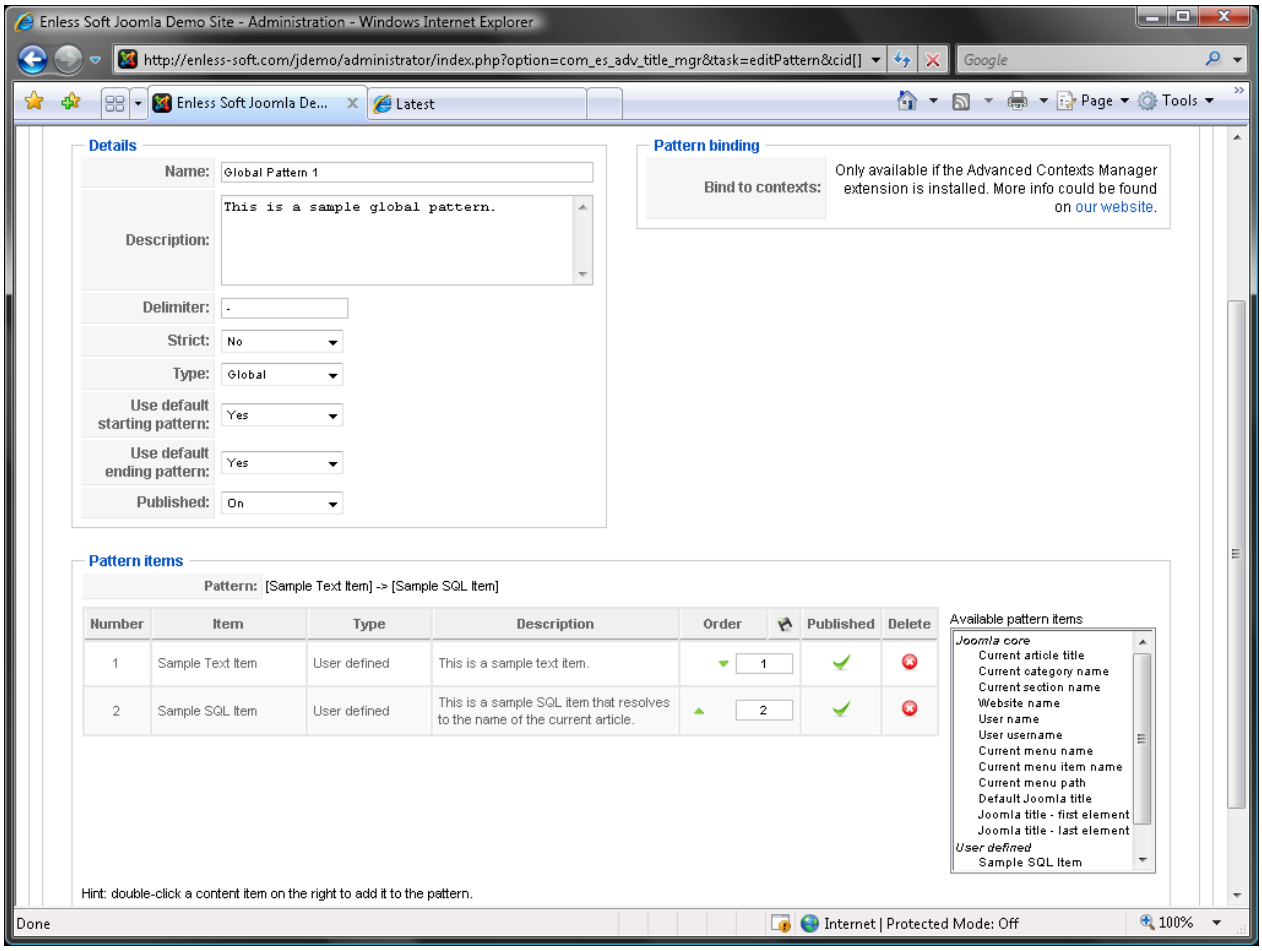
### Patterns overview page

This page displays all defined global and special patterns. Here you can reorder the global patterns.




### Add/edit pattern page

This is the page that is shown when you add or edit a pattern.




**Name** - the name of the pattern.

**Description** - the description of the pattern.

**Delimiter** - the delimiter that is used between different parts of this pattern. Leaving this field empty will cause the pattern to use the default delimiter, defined in the component settings page.

**Strict** - sets whether the pattern is strict or not. You can read more about strict patterns here.

**Type** - sets whether the pattern is global or special. You can read more about pattern types here.

**Use default starting pattern** - sets whether the default (global) starting pattern will be used when this pattern is active.

**Use default ending pattern** - sets whether the default (global) ending pattern will be used when this pattern is active.

**Published** - sets whether this pattern is published. Note, that if a pattern is only used for default starting/ending pattern it could be unpublished.

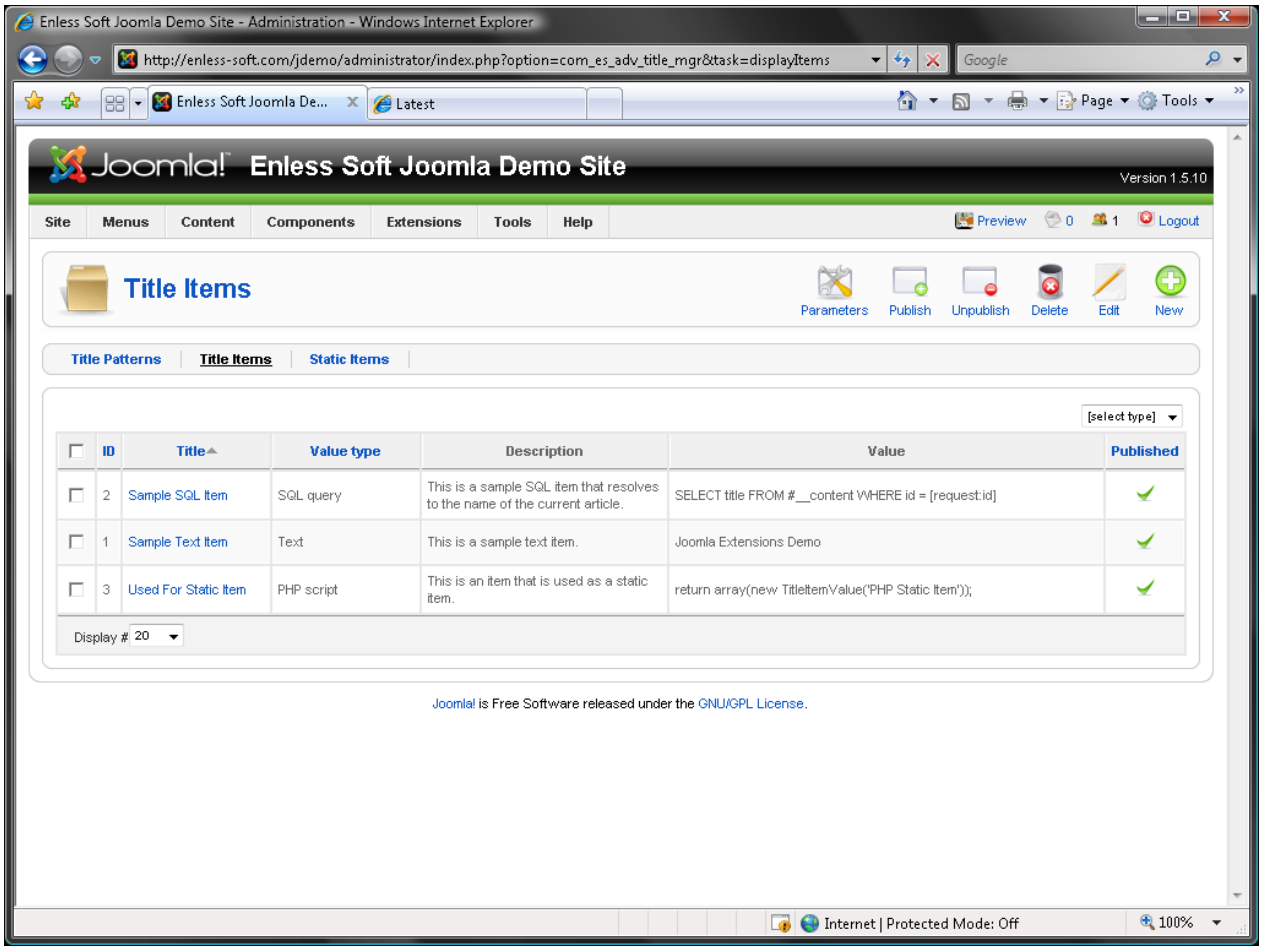
**Pattern** - shows the parts of the current pattern in the same order they will be used to assemble the web site title.

**Available pattern items** - shows all available items that could be used in the pattern.

## Pattern items

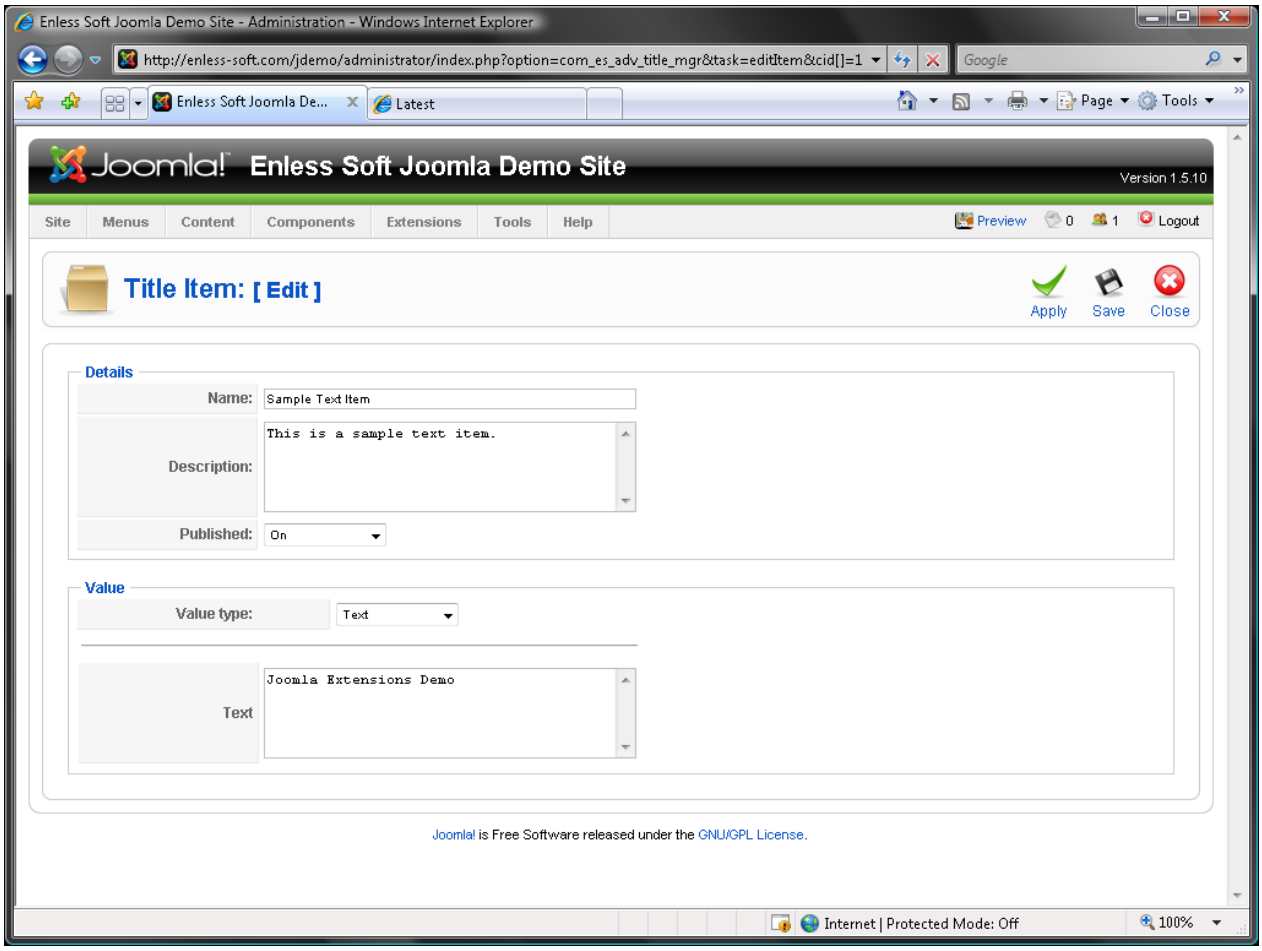
### Pattern items overview page

This page displays all defined custom pattern items.




### Add/edit pattern item page

This is the page that is shown when you add or edit a custom pattern item.




**Name** - the name of the item. This name is displayed in the "Available items" list in the "Add/edit pattern" page.

**Description** - the description of the item.

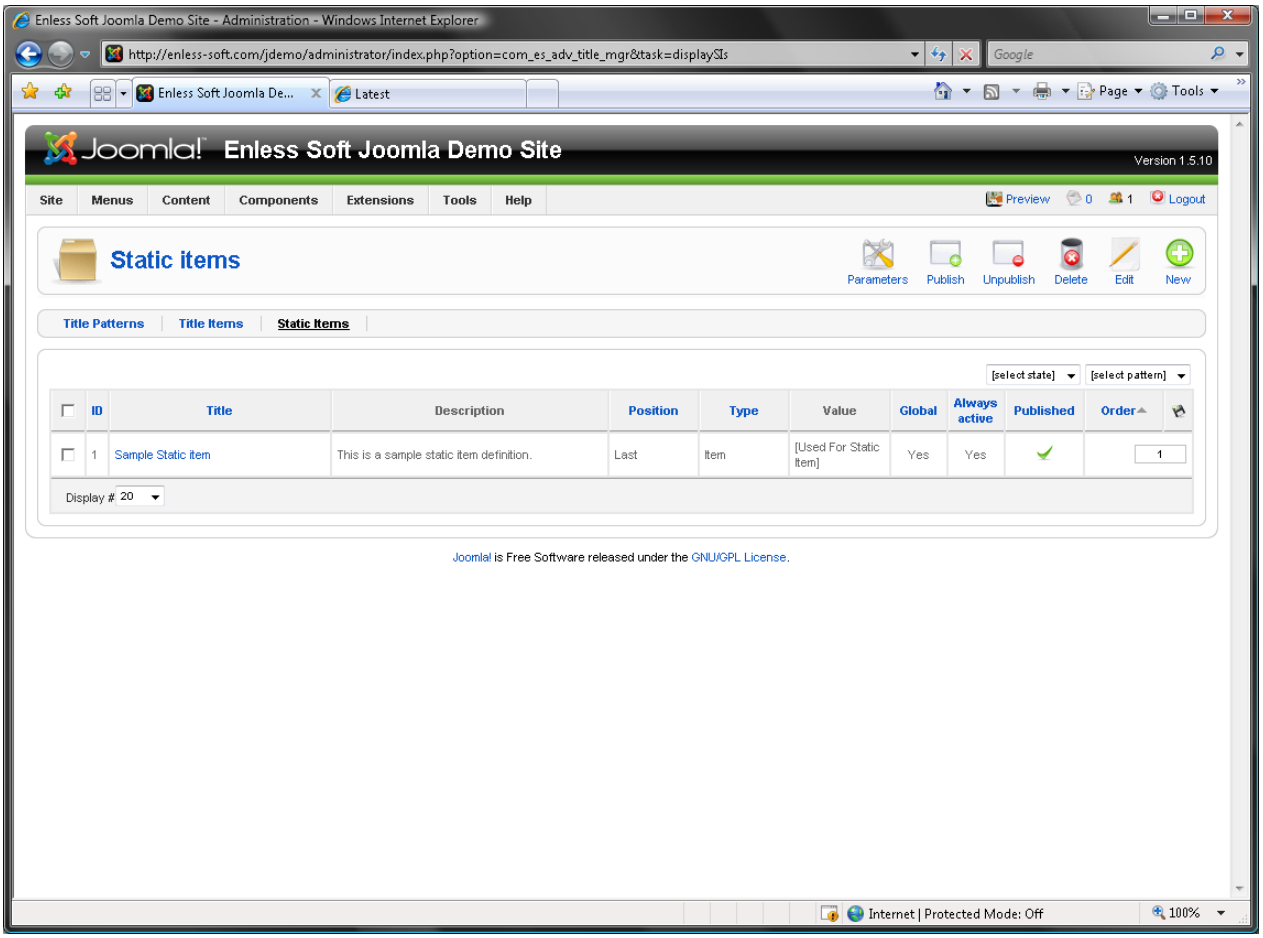
**Published** - sets whether the item is currently published.

**Value type** - Sets the value type of the current item. The content below the horizontal line differ for different value types. For more information on the available value types, please read here.

## Static items

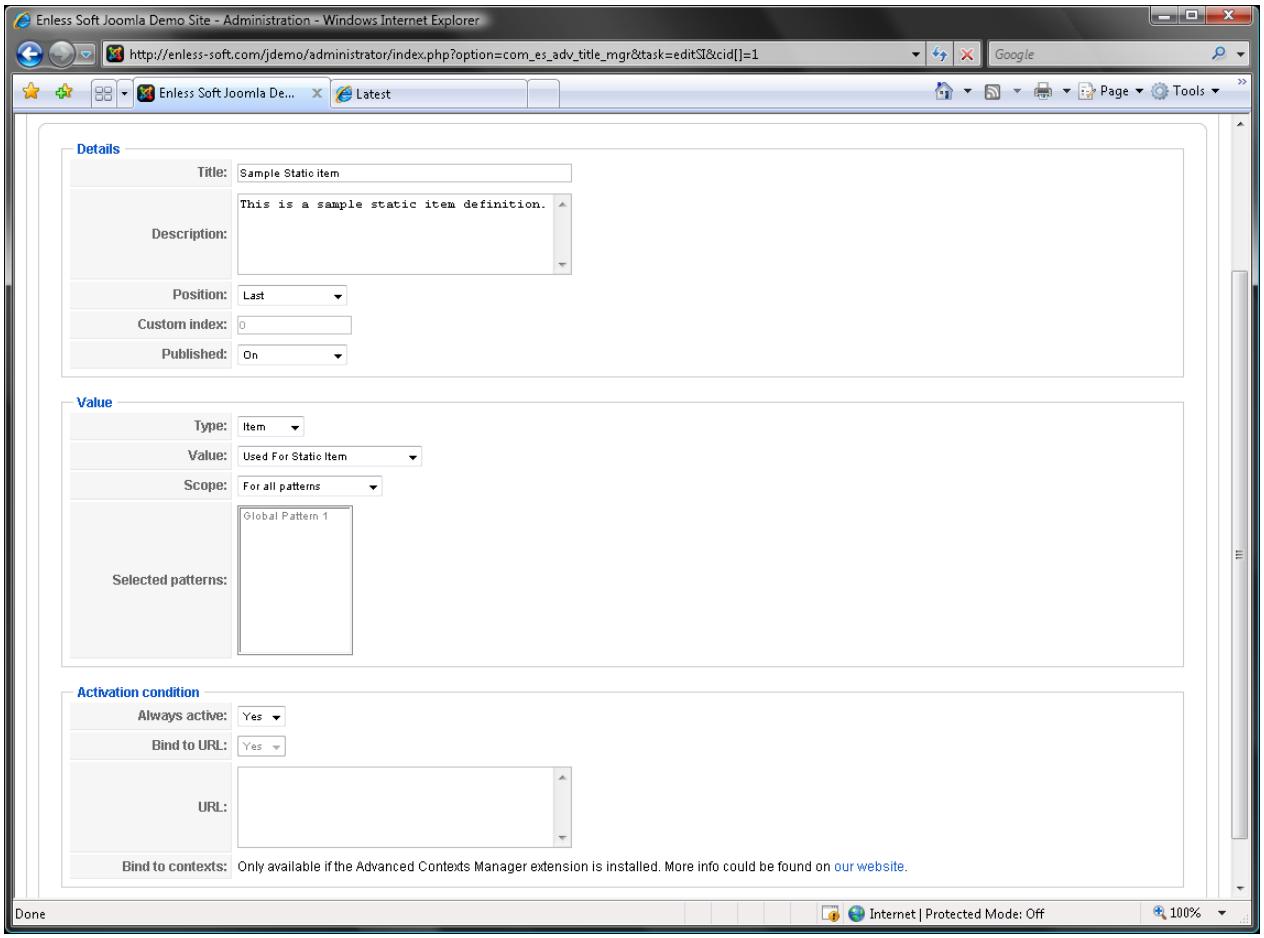
### Static items overview page

This page displays all defined static items.




**Add/edit static item**

This is the page that is shown when you add or edit a static item.




**Title** - the title (name) of the static item. Used only in the administrative interface.

**Description** - the description of the static item.

**Position** - the position in which the item is to be inserted in the active pattern. Possible values are "First", "Last" and "Custom". If "Custom" is selected, the index entered in the "Custom index" parameter is used.

**Custom index** - the position in which the item is to be inserted in the active pattern.

**Published** - sets whether the static item is published.

**Type** - sets the value type of the static item. It could be either Item or Pattern.

**Value** - sets the value for the static item. If the value type is "Item", a list of defined items is displayed. If the value type is "Pattern", a list of all patterns is displayed.

**Scope** - sets the activity scope for the static item. Possible values are: "For all patterns" and "For selected patterns".

**Selected patterns** - the patterns for which the static item is active. This property is used only if the "Scope" parameter is set to "Selected patterns".

**Always active** - sets whether the pattern is always active or is only active when the specified conditions are met.

**Bind to URL** - sets whether the pattern is bound to specific URL (e.g. the URL condition is used).

**URL** - the query part of the URL to which the pattern is bound.

**Bind to contexts** - sets whether the static item is bound to specific contexts (e.g. the context condition is active). This parameter is only available if the Advanced Contexts Manager product is installed.

**Contexts** - selects the contexts to which the static item is bound. his parameter is only available if the Advanced Contexts Manager product is installed.

## Appendixes

### Available item value types

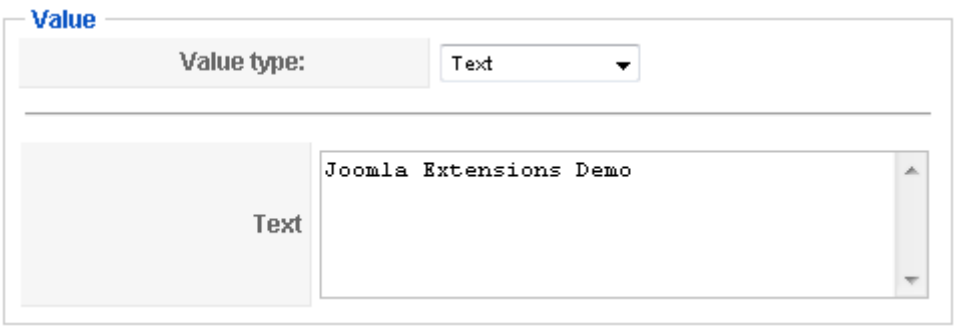
#### List of built-in item value types

The following value types are built-in:

- Text value type
- SQL value type
- PHP value type

#### Text value type

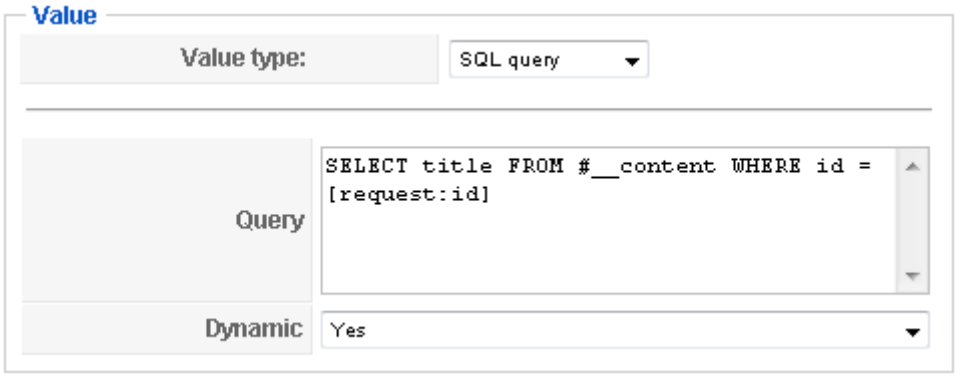
The text value type is defined by entering plain text. This text is used directly in the title. Multiple parts can be specified. Each part should be on a new line in the textbox.




The parameters entered in the screenshot would result in a single title part being added to the title - "Joomla Extensions Demo".

#### SQL value type

The SQL value type is defined by entering an SQL query. The query must return one value in each row. If more values are returned, the first value in the row is used.



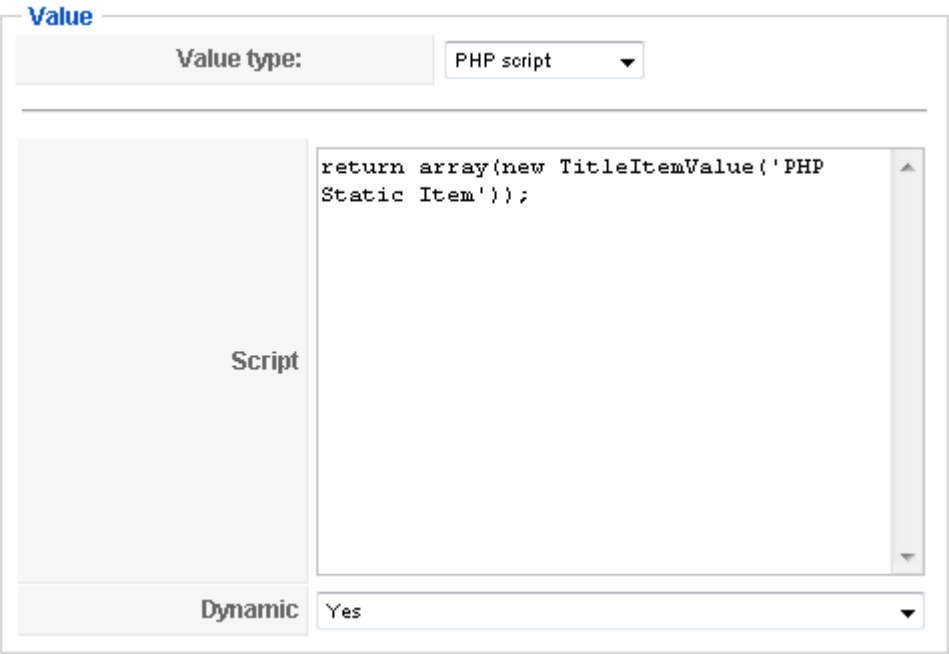

**Query** - the query to execute.

**Dynamic** - sets whether the query is dynamic. A "dynamic" query is a query that may return different values in different executions.

As you can see in the screenshot, it is possible to use environment data in your SQL queries. "[request:id]" evaluates to the value of the "id" parameter from the current HTTP request (POST or GET). If you use such syntax you must set the "Dynamic" property yo "Yes".

#### PHP value type

The PHP value type uses a custom PHP to evaluate its value.




**Script** - the PHP script to be executed. The script must return an array of TitleItemValue objects.

**Dynamic** - sets whether the entered script is dynamic. A "dynamic" script is a script that may return different values in different executions.

### Available pattern items

#### List of built-in pattern items

The following pattern items are built-in:

- current article title
- current category name
- current section name
- web site name
- name of the logged user
- username of the logged user
- current menu name
- current menu item name
- current menu path
- default Joomla title
- default Joomla title - first item
- default Joomla title - last item

### Integration with other extensions

To extend further its functionality, Title Manager is integrated with other products we offer.

**1. Content Enhancer.**
Integration with Content Enhancer adds the following functionality:

- use variables in your custom title items
- use automatic replacements in title patterns

**2. Advanced Contexts Manager**
Integration with Advanced Contexts Manager adds the following functionality:

- bind pattern to a contexts
- define static items only for specified contexts

**3. Breadcrumbs Manager.**
Integration with Breadcrumbs Manager adds the following functionality:

- use breadcrumbs items in title patterns

More information about these products could be found here.

**Related products**

If you need more information about the products related to Title Manager, you can find it in our web site.

- **Content Enhancer**
- **Advanced Contexts Manager**
- **Breadcrumbs Manager**

# Developer's Guide

## Plug-in development

### Plug-in development

Although a very simple and easy in terms of general usage, the Title Manager product is designed to be extendable. Note that using the extension points requires basic PHP programming skills. You will also need to be familiar with the basics of how the Title Manager product works.

#### Why writing an extension?

One possible reason for writing an extension is that the logic for assembling the titles you need is rather complex. In this case it might be easier and/or cleaner solution to implement an extension than to make use of the default functionality.

Another reason for extending might be if you are an author of a component and you want to present your users with a way to have sensible titles when using it.

#### Extension points.

There are two extension points currently provided:

- pattern items providers - a general extension approach. Allows you to create plug-ins that provide pattern items of your choice. These items could be used as parts of your title patterns.
- item value types - allows creating your own value types that can be used as values of the title items.

## Items providers

#### What is an items provider?

An items provider lets you create your own custom title items. Those items will be listed in the title pattern definition page and you will be able to use them in your patterns.

Title Manager does not handle the definition of your provider's title items, it only calls a method to get all available. It is up to you to provide interface for items definition, if necessary. When an item must be evaluated, another method of your plug-in is called. This architecture leaves you entirely responsible for what your items are, but still allows you to use the pattern model with them.

There are two items provider plug-ins that are included in the distribution of the product. The first handles the custom items you define in the user interface of Title Manager and the other handles the Joomla-based built-in items.

Learn how to write an items provider plug-in.

#### How to write an items provider

To create an item provider plug-in, you must implement the ITitleItemsProvider interface.

The naming of the files and the extension class must follow certain rules.

**Pattern:**
Name of the implementing class: **TitleItemsProvider<plugin_name>**
PHP file that contains the class: **<plugin_name>.php**

**Example:**
Name of the implementing class: **TitleItemsProviderMyProvider**
PHP file that contains the class: **MyProvider.php**

**File locations:**

The .php file should be put in the following folder:
**joomla_root/administrator/components/com_es_adv_title_mgr/plugins/items**

#### ITitleItemsProvider interface

```
interface ITitleItemsProvider extends IPluginObject {

	//Must return an array of TitleItem objects
	function getItems();

	//Must return an array of TitleItemValue objects. The $params argument
is currently not used.
	function getItemValues($item_name, $params);

	//Must return the name of the provider
	function getProviderName();

	//Must return a string that is shown as a value for the item in the
administrative interface. Does not affect the end-user interface in any way.
	function getItemValueString($item_id, $short = false);

	//Must return a string that is shown as a label for the item in the
administrative interface. Does not affect the end-user interface in any way.
	function getItemLabelString($item_id);

	//Must return a string that is shown as a description for the item in
the administrative interface. Does not affect the end-user interface in any
way.
	function getItemDescriptionString($item_id);

}
```

## Item value types

#### What is an item value type?

Value types are used in the definition of pattern items. Each value type has some parameters. The value type together with its parameters is the actual definition of how to get the title item value in runtime. If you create a new value type, it will be available in the "Add/edit pattern item" page.

Learn how to write a value type plug-in.

#### How to write a value type

To create a value type, you must implement the ITitleValueTypeProvider interface and define your item parameters. Optionally, you can create language file for your plug-in.

The naming of the files and the extension class must follow certain rules.

**Pattern:**
Name of the implementing class: **TitleValueTypeProvider<plugin_name>**
PHP file that contains the class: **<plugin_name>.php**
XML file with parameters: **<plugin_name>.xml**
Language file: **<lang>.<plugin_name>.ini**

**Example:**
Name of the implementing class: **TitleValueTypeProviderTest**
PHP file that contains the class: **Test.php**
XML file with parameters: **Test.xml**
Language file: **en-GB.Test.ini**

**File locations:**
The .php and .xml files should be put in the following folder:
**joomla_root/administrator/components/com_es_adv_title_mgr/plugins/value_types**

The .ini language file should be put in:
**joomla_root/administrator/components/com_es_adv_title_mgr/plugins/value_types/language/<lang>**

where <lang> is the language folder for the language, e.g. "en-GB".

**Note**: practically, any functionality of a value type could be implemented using the built-in PHP value type. However, it might be a much cleaner and/or easy-to-use approach to create a value type plug-in.

#### ITitleValueTypeProvider interface

```
interface ITitleValueTypeProvider extends IPluginObject {
        /**
         * Must return a user-friendly label for the value type.
         * This label is displayed in the administrative interface.
         */
        function getValueTypeLabel();

        /*
         * Must return a description of the provider.
         */
        function getValueTypeDescription();

        /**
         * This method is invoked to get the title parts corresponding to the
given parameters.
         * @return An array of TitleItemValue objects.
         * @param JParameter $params - the parameters defined by the user when
defining the title item
         */
        function getValue($params);
```

```
        /**
         * Must return a string that is as overview of the value.
         * @param JParameter $params - the parameters defined by the user when
defining the title item
         */
        function getValueString($params);

        /**
         * Must return a boolean indicating whether the returned value for the
given parameters is dynamic.
         * A value is dynamic if the getValue method may return different
value for the same parameters.
         * @param JParameter $params - the parameters defined by the user when
defining the title item
         */
        function isDynamic($params);
}
```

## Other classes

#### The TitleItem class.

```
class TitleItem {
        var $title = null;
        var $name = null;
        var $description = null;

        function __construct($title, $name) {
                $this->title = $title;
                $this->name = $name;
        }
}
```

#### The TitleItemValue class.

```
class TitleItemValue {
        var $value = null;

        function __construct($value) {
                $this->value = $value;
        }
}
```

## Common plug-in classes

#### The ReturnedByPluginObject class.

```
class ReturnedByPluginObject {
        var $id = 0;
        var $provider_id = null;
```

```
        function __construct($id, $provider_id) {
        }
        function getCompoundID() {
        }
        static function getObjectCompoundID($provider_id, $object_id) {
        }
        static function parseObjectCompoundID($html_id, &$plugin_id,
&$object_id) {
        }
}
```

## Common plug-in interfaces

#### IPluginObject interface.

An interface implemented by all plug-in classes.

```
interface IPluginObject {
        /**
         * Must return a string used for identification of the plugin.
         */
        public function getPluginID();

        /**
         * Must return whether the plugin should be used under the current
circumstances.
         */
        function isAvailable();
}
```

#### IListablePluginObject interface.

An interface that is used for plug-ins that are rendered in a list.

```
interface IListablePluginObject extends IPluginObject {
        /**
         * Must return a user-friendly label for the plug-in.
         */
        function getLabel();

        /**
         * Must return a user-friendly description of the plug-in.
         */
        function getDescription();

        /**
         * Must return any additional HTML attributes that will be added to
the
```