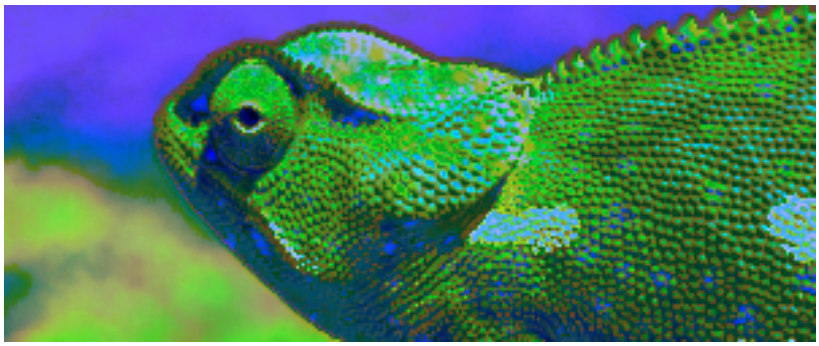


Blue Chameleon CMS
(Composing content)

For links outside this document,
download the relevant chapter or the

Blue Chameleon Content Management System full documentation.

May 8, 2012

Contents

1 Configuring and administration **13**

1.1 System interface . . . 13
1.2 Users and usergroups . . . 14
1.2.1 Users . . . 14
1.2.1.1 User groups that a user belongs to . . . 15
1.2.1.2 User management through time . . . 16
1.2.2 User groups . . . 16
1.2.3 User groups access rights . . . 16
1.3 Blue Chameleon Content Management System user rights . . . 17
1.3.1 Rights on headlines . . . 17
1.3.2 Rights on Articles . . . 18
1.3.3 Rights on page objects . . . 18
1.3.4 Rights on Images . . . 18
1.3.5 Rights on Search . . . 18
1.3.6 Rights on URLs . . . 18
1.3.7 Rights on columns . . . 19
1.3.8 Rights on publishing . . . 19
1.3.9 Rights on Data Base Publishing . . . 20
1.3.10 Rights on Editions . . . 20
1.3.11 Rights on Background Frames . . . 20
1.3.12 Rights on Downloads . . . 20
1.3.13 Rights on Site management . . . 21
1.3.14 Rights on Article Models . . . 21
1.3.15 Rights on Headline Models . . . 21
1.3.16 Rights on Notice boards . . . 21
1.3.17 Rights on System log . . . 22
1.3.18 Rights on Scripts . . . 22
1.3.19 Administrator rights . . . 22
1.3.20 Rights on Hermetic groups . . . 22
1.3.21 Information level . . . 23
1.3.22 Rights on Books . . . 23
1.3.23 Rights on Book Models . . . 23
1.3.24 Java option menu . . . 23
1.3.25 Connection speed . . . 23
1.3.26 Rights on User management . . . 24
1.3.27 Rights on Menus, Menu Templates . . . 24

1.3.28 Rights on Packages . . . 24
1.3.29 Rights on Forms . . . 24
1.4 Columns . . . 25
1.4.1 Publisher and objects . . . 25
1.4.2 Creating columns . . . 25
1.4.3 Managing columns . . . 26
1.4.3.1 Changing columns . . . 26
1.4.3.2 Updating columns . . . 26
1.4.3.3 Backups . . . 27
1.4.4 Blue Chameleon directory structure . . . 28
1.4.4.1 Virtual columns . . . 28
1.5 Publisher settings . . . 28
1.5.1 File types . . . 29
1.6 FTP profiles . . . 30

2 Composing content **33**
2.1 Principles . . . 33
2.2 Headlines models . . . 33
2.2.1 Creating a new headline model . . . 34
2.2.2 Updating a headline model . . . 35
2.2.3 Contents of a headline model . . . 36
2.2.3.1 Enriching a headline model with frames . . . 37
2.2.3.1.1 Frame templates used by a headline model . . . 39
2.2.3.1.2 Updating frame number . . . 40
2.2.3.2 Meta Tags . . . 40
2.3 Frame templates . . . 41
2.3.1 Creating and updating frame templates . . . 41
2.3.2 "List" frames . . . 42
2.3.3 "Leaf" frames . . . 43
2.3.4 Frame style . . . 44
2.4 Headlines : creation and editing . . . 45
2.4.1 Editing a headline : a basic example . . . 45
2.4.2 Editing a leaf frame . . . 46
2.4.3 Editing a list frame . . . 47
2.4.3.1 Example for a leaf frame as used by a list . . . 48
2.5 Layouts . . . 50
2.6 Articles and article models . . . 50
2.7 Managing pages . . . 50
2.7.1 Publishing pages . . . 50
2.7.1.1 Multiple publish . . . 51
2.7.1.2 Scheduled publishing . . . 52
2.7.2 Properties of headlines . . . 53
2.7.3 Other management options for pages . . . 54

3 Object management **57**
3.1 Images . . . 57
3.1.1 Uploading an image . . . 57
3.1.2 Image folders . . . 59
3.1.3 Managing images . . . 59
3.1.4 Integrating images . . . 59
3.2 Links . . . 60
3.2.1 Creating a URL link . . . 61
3.2.2 Managing URL links . . . 62
3.2.3 Integrating links . . . 62
3.3 Scripts . . . 64
3.3.1 Creating a script - Management . . . 64
3.3.2 Integrating a frame script . . . 65
3.3.3 Integrating a link style script . . . 66
3.4 Downloads . . . 67
3.4.1 Creating a download . . . 67
3.4.2 Managing downloads . . . 68
3.4.3 Integrating downloads . . . 68
3.5 Menus . . . 69
3.5.1 Menu scripts . . . 69
3.5.2 Menu models . . . 71
3.5.3 Creation of a menu . . . 71
3.5.3.1 Filling of a menu . . . 72
3.5.3.2 Menu result and integration . . . 74
3.5.4 More complex menus . . . 75
3.5.4.1 A menu script that handles children menu items . . . 75
3.5.4.2 A menu with children menu items . . . 75
3.6 Forms . . . 78
3.6.1 Form models . . . 78
3.6.1.1 Form model management - parameters . . . 79
3.6.2 Defining forms . . . 79
3.6.2.1 Updating a form . . . 80
3.6.3 Composing a form . . . 80
3.6.3.1 Editing a form page . . . 81
3.6.3.2 Form page element types . . . 82
3.6.3.3 Editing a form page element . . . 83
3.6.3.4 After form is submitted . . . 84
3.6.3.5 A form result . . . 84
3.6.3.6 A form with several pages . . . 84
3.6.4 Integrating a form . . . 85
3.7 Variable files . . . 86
3.7.1 Example of varfile contents . . . 86
3.7.2 Varfiles as variables . . . 87

4 Other elements **89**
4.1 Site management . . . 89
4.1.1 Uploading files . . . 89
4.1.2 Creating directories . . . 90
4.1.3 Managing files and directories . . . 91
4.1.4 Executing `SQL` . . . 91
4.1.5 Export and Import . . . 93
4.1.5.1 Generating export files - importing them . . . 93
4.1.5.2 Export file contents . . . 94
4.2 Model defaults . . . 96
4.2.1 Model defaults and headlines . . . 96
4.3 Books . . . 97
4.3.1 Book models . . . 97
4.3.2 Creating books . . . 97
4.3.3 Modifying a book . . . 99
4.4 Notice boards . . . 100
4.4.1 Creating and managing notice boards . . . 100
4.4.2 Association with a column . . . 100
4.4.3 Notice board entries . . . 101
4.5 Editions . . . 102
4.5.1 General edition management . . . 103
4.5.2 Publishing editions . . . 104
4.6 Options . . . 104
4.6.1 Version . . . 104
4.6.2 System Logs . . . 105
4.6.3 Expiration report . . . 106
4.7 Searching for content . . . 106
4.8 Management of objects . . . 108

5 Packages **109**
5.1 Mandatory scripts . . . 109
5.1.1 `PackageInstall.phs` . . . 109
5.1.2 `PackageGroupModify.phs`, `PackageGroupModify1.phs` . . . 110
5.1.2.1 Result : package group rights . . . 110
5.1.3 `PackageMain.phs` . . . 113
5.1.3.1 Result : package main page . . . 114
5.1.4 `PackageUninstall.phs` . . . 116
5.2 Package general management . . . 117
5.2.1 Installing a package . . . 117
5.2.2 Updating a package . . . 117
5.2.3 Uninstalling a package . . . 118
5.3 Using object types and objects to compose headlines . . . 118
5.3.1 Enabling a leaf frame to handle packages and an object . . . 119
5.3.2 `PackageObjTypeList.phs` . . . 120
5.3.3 `PackageObjectList.phs` . . . 122
5.3.4 `PackageObjectName.phs` . . . 122

5.3.5 `PackageObjectView.phs` . . . 124
5.4 Using attributes . . . 125
5.4.1 Enabling a frame to handle attributes . . . 125
5.4.2 Scripts for attribute . . . 126
5.5 Featuring multiple objects on the same headline . . . 127
5.5.1 A leaf frame that handles multiple objects . . . 128
5.5.2 Packages : defining more of them . . . 130
5.5.3 Object and attribute scripts for this case (guidelines) . . . 130
5.5.3.1 The `PackageObjectView.phs` script for this example . . . 132

6 Annex **135**
6.1 Leaf frame variables . . . 135
6.1.1 "Table" variables . . . 137
6.1.2 "Select" variables . . . 137
6.1.3 "`SQL` select" variables . . . 138
6.2 System script glossary . . . 138
6.2.1 Headline model scripts . . . 138
6.2.2 Leaf frame model scripts . . . 139
6.3 Icon glossary . . . 141
6.4 Form page elements : detailed . . . 143
6.4.1 "Active" elements . . . 143
6.4.1.1 Checkboxes . . . 143
6.4.1.2 Radiobuttons . . . 144
6.4.1.3 Listboxes . . . 144
6.4.1.4 Input field and text area . . . 145
6.4.1.5 Form label - 'with check box' . . . 146
6.4.2 "Passive" elements . . . 146
6.4.2.1 Hidden field . . . 146
6.4.2.2 Form image . . . 146
6.4.2.3 Form label - 'normal' . . . 146
6.4.2.4 Form separator . . . 147

List of Figures

1.1 The Publisher's main page. . . . 13
1.2 Defining a new user. . . . 15
1.3 Groups a user belongs to. . . . 15
1.4 Defining a user group. . . . 16
1.5 The most important access rights as defined for a user group. . . . 17
1.6 Creating a new column. . . . 26
1.7 Updating a column. . . . 27
1.8 The properties of your Publisher. . . . 29
1.9 Defining a FTP profile. . . . 31

2.1 Creating a headline model. . . . 34
2.2 Here, a headline model can be updated and its children frame templates be chosen. . . . 36
2.3 Thanks to this, the pages created on this model will have meta-tags called Author, LastUpdate and a Page Title. . . . 41
2.4 Creating a new headline. . . . 45
2.5 First edit ; as both frames use only one template ("Basic text frame" and "Side Menu List"), so clicking on the icon shows them immediately. . . . 46
2.6 Filling in with information the leaf's various variables. . . . 47
2.7 A preview of the page. . . . 48
2.8 Populating a list frame. . . . 49
2.9 Publishing several headlines in one single time. . . . 51
2.10 Configuring and viewing data related to headline "Home Page". . . . 53
2.11 Managing headlines in a file system way. . . . 55

3.1 Uploading an image. . . . 58
3.2 Creating a folder for images. . . . 59
3.3 From here, all images and folders can be globally managed. . . . 60
3.4 Integrating an uploaded image. . . . 61
3.5 Creating a link. . . . 62
3.6 Putting a previously-defined URL link. . . . 63
3.7 Creating a script to be used in a leaf frame. . . . 65
3.8 Uploading a file that will be used for downloading purposes. . . . 67
3.9 Integrating a download, with a 'download' variable. . . . 69
3.10 Creating a menu model. . . . 71
3.11 Creating a menu "Basic top menu" based upon model 'Top Menu'. . . . 72
3.12 Creating a form model. . . . 78

3.13 Adding parameters to a form model. 79
3.14 Creating a form based on the example model. 79

4.1 Uploading a file to be used in headline contents. 90
4.2 Creating a new directory, directly under the «#SQLWWWHOME#» root. . . . 91
4.3 Editing a text-based file. 92
4.4 Outputting the contents of a table. 93
4.5 Exporting the current column. 94
4.6 An example of export file contents. 95
4.7 Creating a model default for a headline model. 96
4.8 Creating a book model. 98
4.9 Creating a book. 98
4.10 From here, book elements can be managed. 100
4.11 Creating a general notice board. 101
4.12 Manually adding two entries to this notice board. 102
4.13 Checking one's own activity for last week. 106
4.14 Performing a search on objects which publishing name begin by 'Josh'. . 107
4.15 From here, any object can be managed. 108

5.1 Modifying this package's rights for a particular user group. 110
5.2 The main page for package Festivals. 114
5.3 The first install of the package "Festivals". 117
5.4 The process of selecting an object type and an object for a leaf frame that is package-enabled. 119
5.5 A selection of object types. 121
5.6 Finally choosing the object. 123
5.7 A page design that calls for multiple objects to be featured. 128
5.8 The 3-object frame after all relevant choices of objects and attributes have been made. 132

6.1 Composing a checkbox selection for a form. 144

List of Tables

5.1 Scripts, functions and variables as used for attribute setting 126

6.1 Leaf frame variables 136
6.2 Leaf frame variables (continued) 137
6.3 System scripts for headline, list and container models 139
6.4 System scripts for leaves 140
6.5 System scripts for leaves (continued) 141
6.6 Blue Chameleon Content Management icons 142
6.7 Blue Chameleon Content Management icons (continued) 143

Chapter 2

Composing content

Now that matters related to Blue Chameleon Content Management System configuring and administration have been dealt with at the previous Chapter, it is time to see how webpages themselves are composed.

2.1 Principles

Blue Chameleon Content Management System is based upon those simple principles :

- a website is a collection of pages called "**articles**" and "**headlines**" ;
- these articles and headlines are created using *Article models* and *Headlines models* ;
- these templates include *frame templates* ;
- articles and headlines are enriched with objects such as images and links and, when complete, are published on the website.

Frame templates consist in :

- containers, which can contain any type of frame - children frames are in fixed number, decided during modification of the frame template ;
- lists, which can also contain any type of frame - children frames are in variable number, added or subtracted when headline or article is edited ;
- leaves, which cannot contain children frames, but exclusively provide Variables to provide various content input when headline or article is edited.

2.2 Headlines models

User group rights for handling Headline Models are detailed at 1.3.15.

2.2.1 Creating a new headline model

Fig.2.1 shows how a headline model can be created.

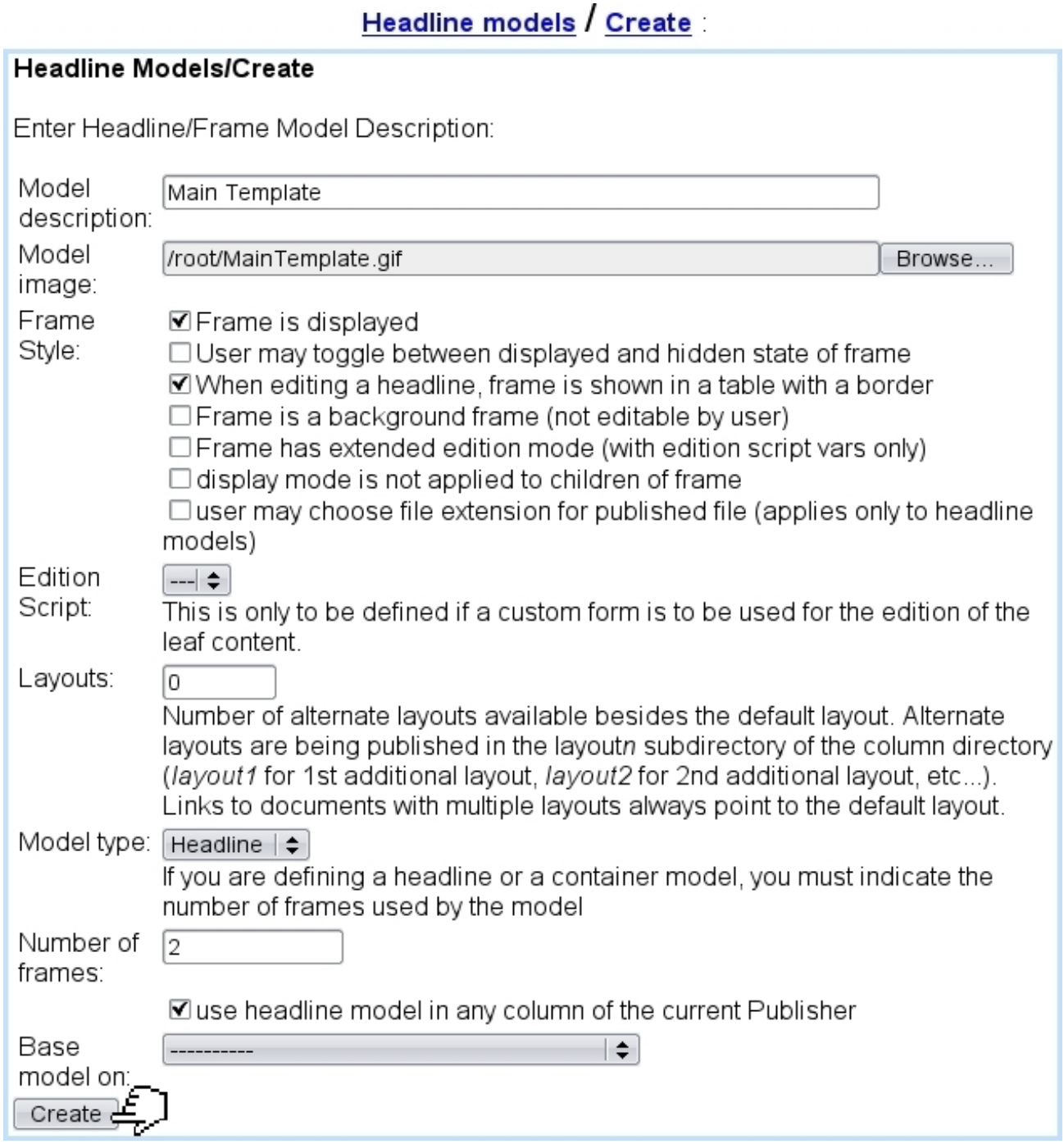


Figure 2.1: Creating a headline model.

This page shows many provides many elements :

- a name for this new headline model, as well as an image (200x150 pixels is ideal) that will represent it ;

- options for frame style (2.3.4) ;
- an edition script ;
- number of *alternate* layouts (2.5) ;
- the model type : here, "headline" is chosen but other choices include 'container', 'list', 'leaf' (2.3) and 'book' (see 4.3.1 for this particular object) ;
- the number of frames (if any) this headline will contain ;
- if headlines already exist, it is possible to choose one upon which the new headline will be based.

Upon click on the Create button at the bottom of this page, one is redirect to a new page where the text of the headline model is inputed, as described as follows.

Most of this data (headline model name, image...) can afterwards be updated through **Headline models** / **Properties**, as explained next.

2.2.2 Updating a headline model

As featured on Fig.2.2, it is possible to update a few of a headline model's information, such as its name, model image, frame style, number of layouts and column availability.

Buttons at the bottom of the page respectively allow to Update changes done to the form and, if the case, validate Image Upload. Another button leads to the configuring of the Meta Tags (2.2.3.2) that headlines created from this model will inherit.

What is more, a link at the bottom of the page allows to select templates that may be used for child frames, as explained in 2.2.3.1.1.

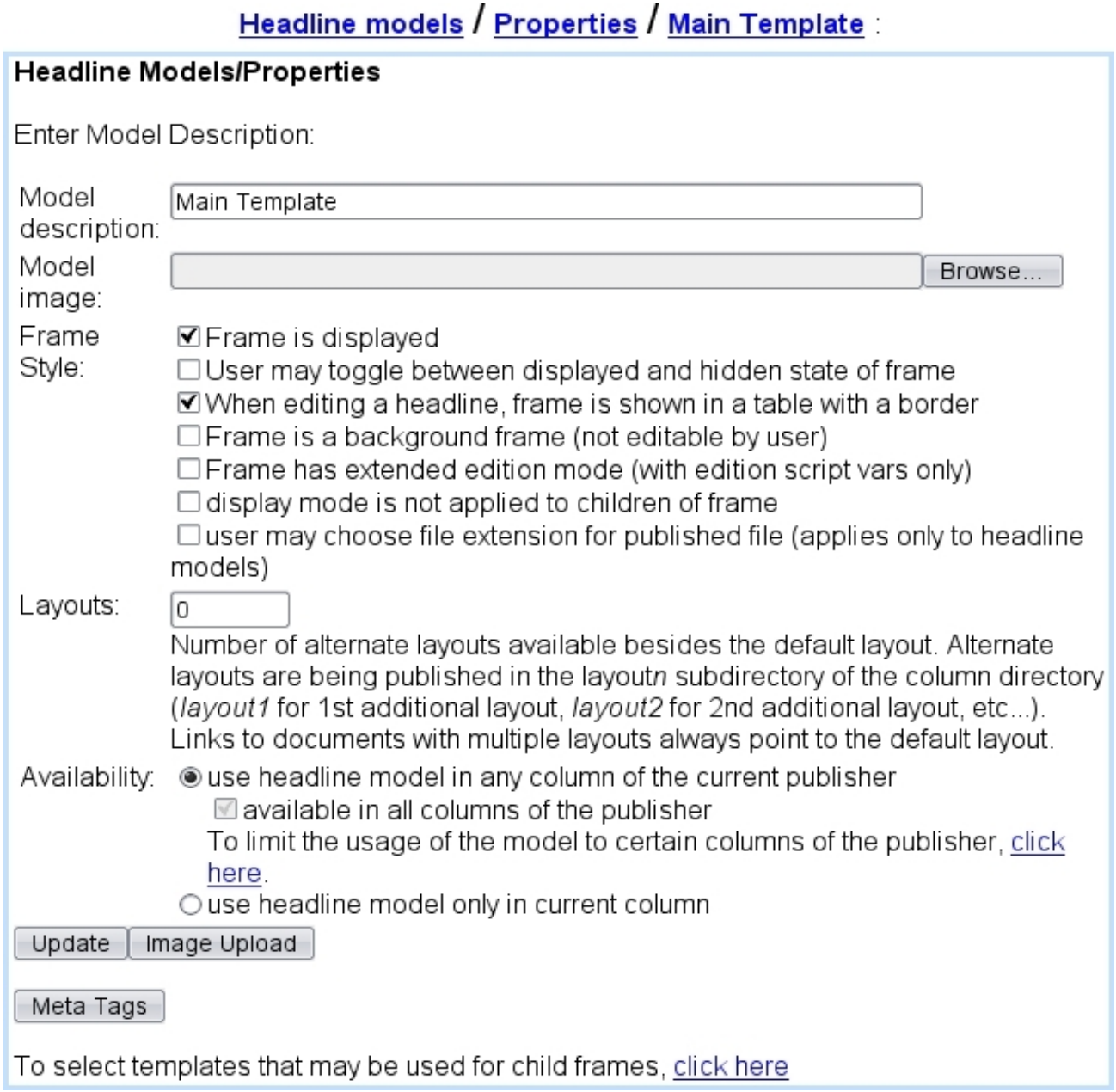
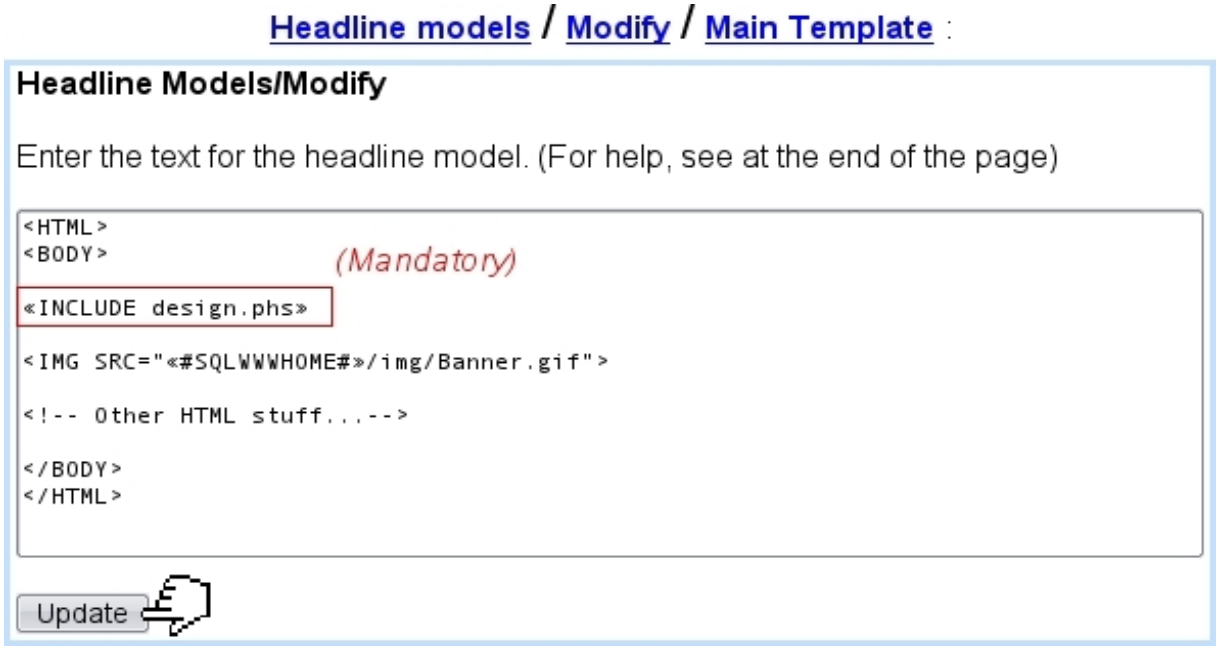


Figure 2.2: Here, a headline model can be updated and its children frame templates be chosen.

2.2.3 Contents of a headline model

Just after the validating as previously seen, or anytime through **Headline models** / **Modify**, the contents of a headline is inputed/altered.

The following loads for instance the picture "Banner.gif" that has been previously uploaded (see 4.1.1 on how to upload any file) to the `img/` directory :



The `design.phs` script must always be `INCLUDE`'d in any headline model's contents

Basically, here is composed, with `HTML` and a few `OSL` code the layout of the headlines that will be modelled after this headline model.

Of course, Blue Chameleon Content Management System allows to go further by equipping this headline model with frames.

2.2.3.1 Enriching a headline model with frames

The great flexibility of the Content Management System allows to make a headline model contain frames that are chosen amongst the list of defined frame templates. Afterwards, during the editing of the headline itself, those frames will be edited individually.

Code 1 shows how the example headline model's content is made to contain two frames, one besides the other.

There, a system script called `frame.phs` is called with the argument `_FrameIndex` corresponding to the number of current frame to insert. There are therefore as many of those `INCLUDE`s to be put as the 'Number of frames:' that was defined during headline model creation (Fig.2.1).

This code serves as to provide two frames for the headline ; now have to be defined *which* frame templates will be proposed by the headline model.

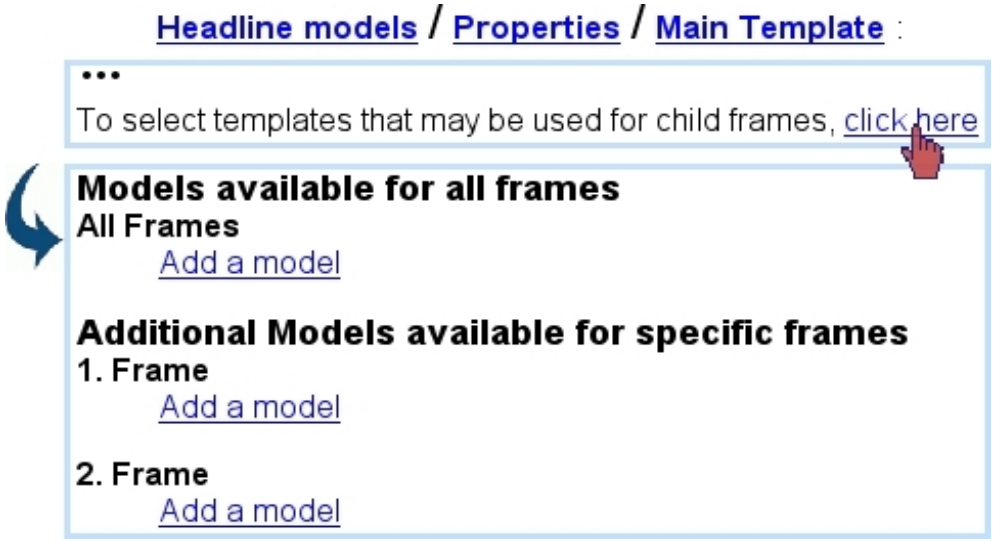
Code 1 This headline model will allow two frames :

Headline models / **Modify** / **Main Template** :

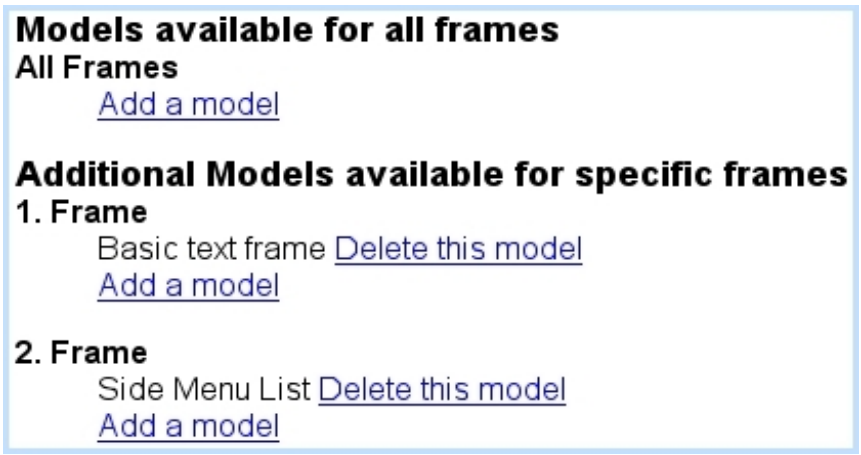
```
<HTML>
(...)
<BODY>

«INCLUDE design.phs»

<IMG SRC="«#SQLWWWHOME#»/img/Banner.gif">

<TABLE>
 <TR>
  <TD>
   «INCLUDE frame.phs;_ParentId=«_FSId»;_FrameIndex=1»
  </TD>
 </TR>
 <TR>
  <TD>
   «INCLUDE frame.phs;_ParentId=«_FSId»;_FrameIndex=2»
  </TD>
 </TR>
</TABLE>
(...)
```

2.2.3.1.1 Frame templates used by a headline model

The page displaying a headline model's properties (2.2.2) also allows to define which frame templates it can use :



The **Add a model** link then shows a list of frames (containers, lists and leaves). One is selected for each numbered frame, or for 'All Frames', which means any of the numbered frames will be able to use it. Once a selection is done, something like this is obtained :

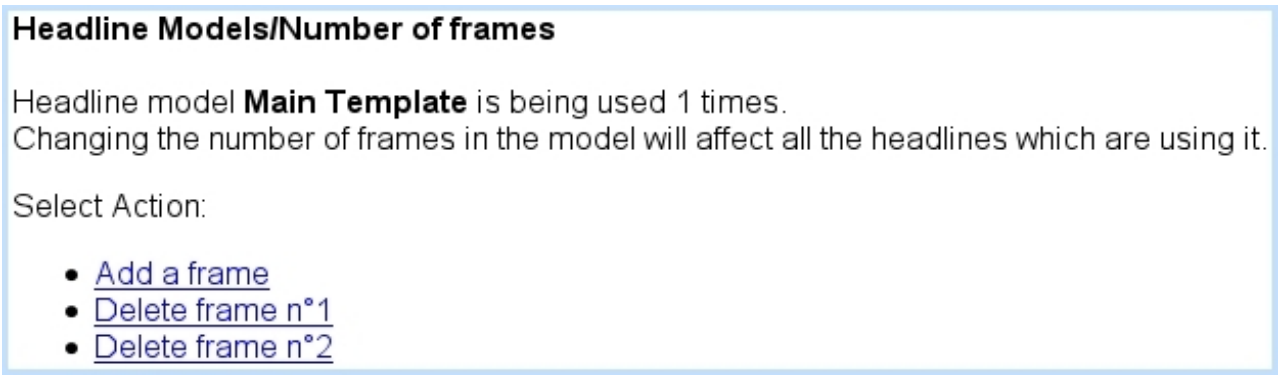


If a mistake was made, a **Delete this model** link allows to remove any frame template from the list of frame templates this headline model can use.

This selection of frame templates also happen for containers and lists, which likewise accept children frames. .

2.2.3.1.2 Updating frame number

During development of a website, it can happen that a headline model has to allow a new frame, or maybe has no use for a specific frame anymore. Such updates can be proceeded with through **Headline models** / **Frame number**, where links such as **Add a frame** or **Delete frame n°...** enable to perform these actions :



2.2.3.2 Meta Tags

These serve as to provide information on a headline (such as the author, the last modify date,....) and will be inserted automatically into the "header" part of `HTML` code of the headline. They are inherited from the headline model on which the headline is based on.

For Meta Tags to be inserted, the headline model contents must include the following call, in the page's header :

Headline models / **Modify** / **Some headline model** :

```
<HTML>
<HEAD>
...
«INCLUDE metatags.phs»
...
</HEAD>
(...)
```

Meta-Tags in use can be configured while at updating a headline model's properties, as featured in Fig.2.3.

There, the information give is twofold :

- the document properties, with two already-existing tags Author and LastUpdate ; they, as well as any other tag that is Create'd, can be modified and deleted

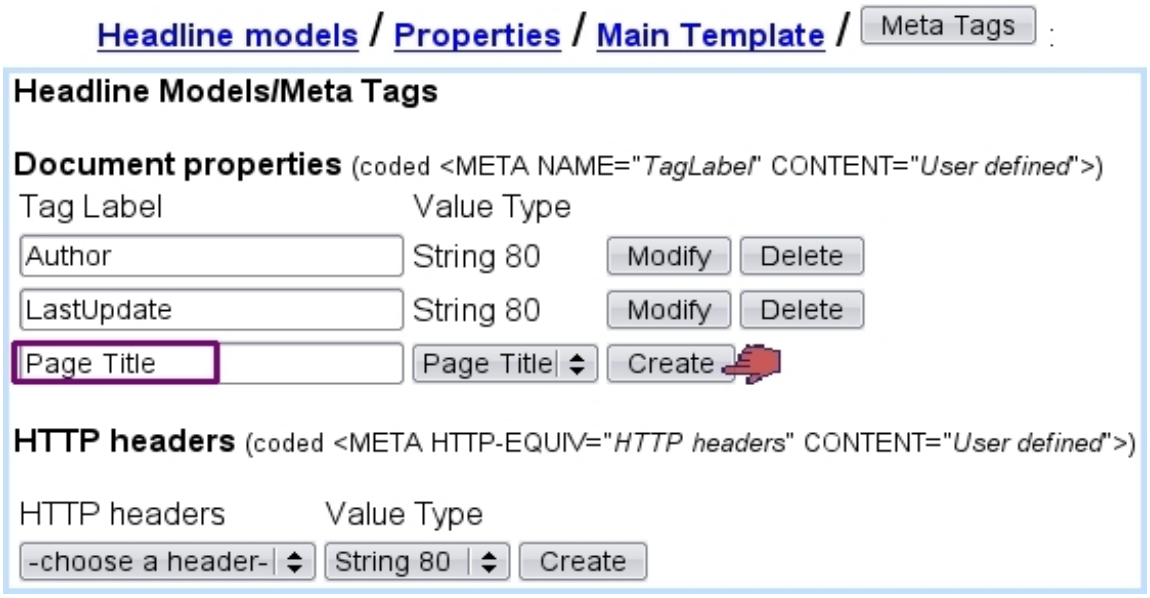


Figure 2.3: Thanks to this, the pages created on this model will have meta-tags called Author, LastUpdate and a Page Title.

through the eponymous buttons. 'Type' as defined for a tag creation must handle the tag information given afterwards, either a String (of maximum length 80, 255 or 512 characters) or a Page Title (the example here) ;

- the `HTTP` headers, with the same value types as above, giving the following choice for header :
  - Cache-control
  - Content-Encoding
  - Content-Language
  - Content-Type
  - Data
  - Expires
  - Pragma

2.3 Frame templates

Now that the way of creating frame-using headline models has been thoroughly explained, it is time to see how frame templates, i.e. container-, list- and leaf models are created.

2.3.1 Creating and updating frame templates

Frames are created in the exact same way as for headline models (Fig.2.1) i.e. through **Headline models** / **Create** except that now 'container', 'list' or 'leaf' is chosen from the 'Model type:' menu.

Updating a frame also happens in a similar way as for headline models, i.e. through **Headline models** / **Properties**.

According to frame template type, management can vary :

- 'container' being the only type of frame that contains a `fixed` number of other frames, a frame number has to be given during creation. The including of children frames (2.2.3.1), their selection (2.2.3.1.1) and the updating of frame number (2.2.3.1.2) happen similarly as for headline models ;
- for 'list' type of frame, only the selection of used children frames must be done ;
- for 'leaf' type, no child frame exist ; but on the other hand, suitable variables must be given.

2.3.2 "List" frames

A list frame template is quite particular : indeed, during the editing of a headline (see 2.4.3), it will allow, thanks to certain icons, to add (or remove) as many list elements as needed ; those list elements are frames.

For this, the list frame template's contents are set as shown in Code 2.

Code 2 A simple list frame.

Headline models / **Modify** / **Side Menu List** :

```
«VAR NEW _ItemCount»
«INCLUDE ListItems.phs;_ListFrameItems="_ItemCount"»
«IF _ItemCount>0»
 «SQLREPEAT» <!- element «#SQLREPEAT# »->
  «INCLUDE frame.phs;_ParentId=«_FSId»;_FrameIndex=«#SQLREPEAT#»»
 «/SQLREPEAT 1;«_ItemCount»»
«ENDIF»
```

There, the number of items (frames) in the list is counted ; then, if there are frames, through the `SQLREPEAT` loop, each of them is `INCLUDE`'d.

It is then necessary to create frames (e.g. 'Simple List Element') that will be supported by the list frame.

2.3.3 "Leaf" frames

A leaf frame is an end frame in itself (like the leaves of a tree) and does not contain any frame.

Nonetheless, it can be richly configured with the help of *variables*, something which does not exist with containers and lists. Indeed, going on **Headline models** shows a **Variables** link, that, upon click, shows only leaf-type frames.

Variables inputed there will then make appear, when a headline containing this frame is edited, various kind of inputs such as text lines, text fields, images, select menus... upon which information will be given.

For more details about objects for leaf frames and their variables, see Chapter 3.

Code 3 shows an example for the contents of a leaf and the related variables.

Code 3 Contents and variables for a simple leaf.

Headline models / **Modify** / **Concert photos** :

```
<B>«Title»</B> («Date»)<BR>
«Text»

<TABLE>
 <TR>
  <TD>
   «INCLUDE image.phs;pImage="_Pict1";Html="border='0'"»
  </TD>
  <TD>
   «INCLUDE image.phs;pImage="_Pict2";Html="border='0'"»
  </TD>
 </TR>
</TABLE>
```

Headline models / **Variables** / **Concert photos** :

```
Title;Title;1;
Date;Date;1;
Text;Text;2;
_Pict1;Picture 1;3;
_Pict2;Picture 2;3;
```

There, the leaf contents use several variables (`«Date»`, `«Title»`, ...) to output on the final page a title, a descriptive text as well as two pictures. These will be inputed during frame editing using the defined **Variables** : see 2.4.2 to view how frame editing will look like.

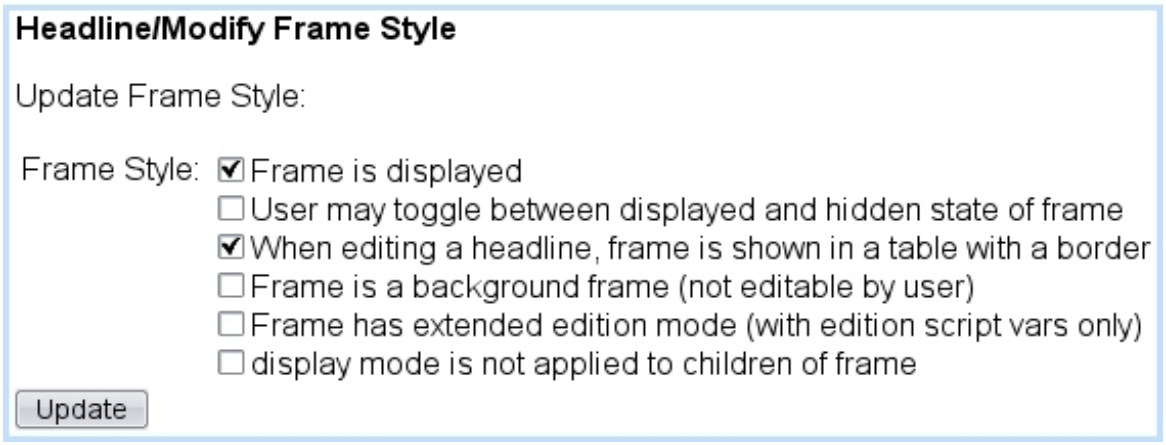
For the full list and description of leaf frame variables, please check the Annex (6.1).

2.3.4 Frame style

Frame style options are featured during creation/edit of a headline or frame, plus anytime when a headline's contents themselves are edited (2.4), through the  icon.

For any frame, they are as follows :

 :



Checkboxes there decide whether :

- frame is displayed ;
- user has the possibility to hide/show this frame (ticking this will make a  icon appear for this frame) ;
- whether frame contents are shown with a border for easier view ;
- frame is a background frame ;

User group rights for handling Background Frames are detailed at 1.3.11.

- frame has extended edition mode (concerning frames that handle 'edit script' variables) ;
- display mode is not applied to children of frame (this is the default choice)

2.4 Headlines : creation and editing

User group rights for Headlines to be handled are detailed at 1.3.1.

Once headline models have been created, a headline is simply created as featured on Fig.2.4.

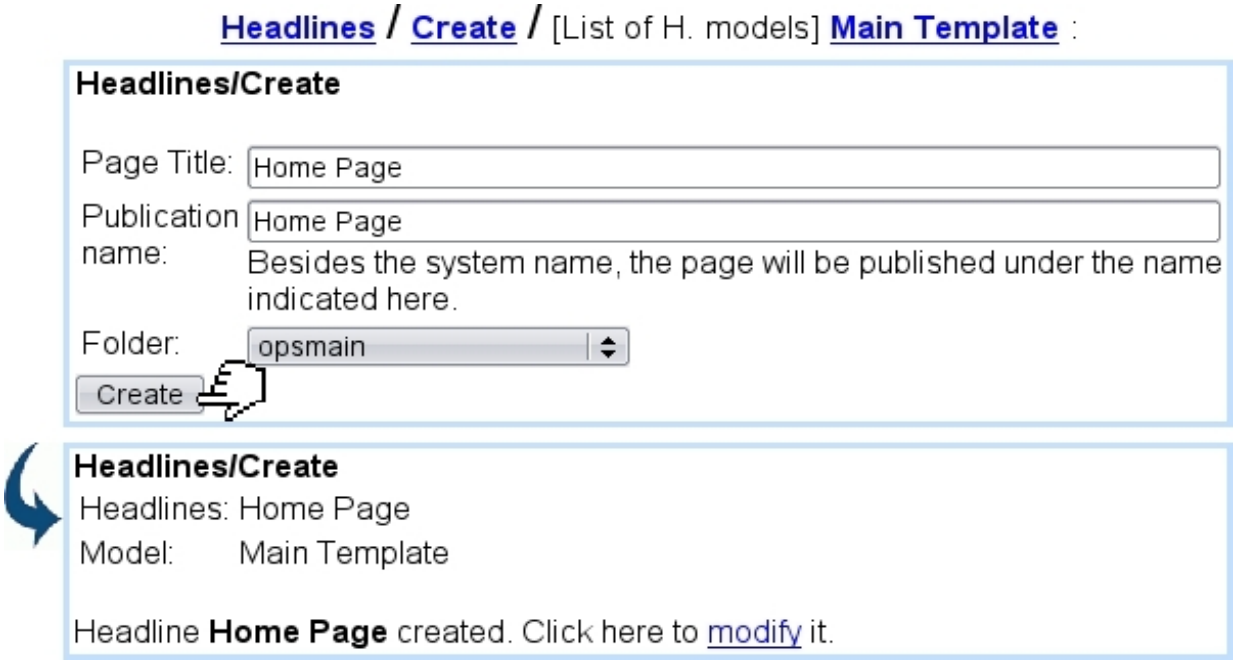


Figure 2.4: Creating a new headline.

After the selection of the headline model it will be based on, a headline is given a title (under which it will be referred to as when browsing and modifying headlines) and a name under which it will be published (2.7.1).

2.4.1 Editing a headline : a basic example

Through Headlines / Modify, the list of headlines in the current column is displayed for each to be edited.

When editing a headline for the first time, Blue Chameleon Content Management System displays what is featured at Fig.2.5 :

- `HTML` code as inputed in the headline model contents is interpreted : here, the display of image "Banner.gif" (cf. Code 1) ;
- frames that have to be displayed are shown yet unfilled ; then, clicking then on the  icon either fills the frame with the assigned frame template or shows the available frame templates for this frame (2.2.3.1.1).

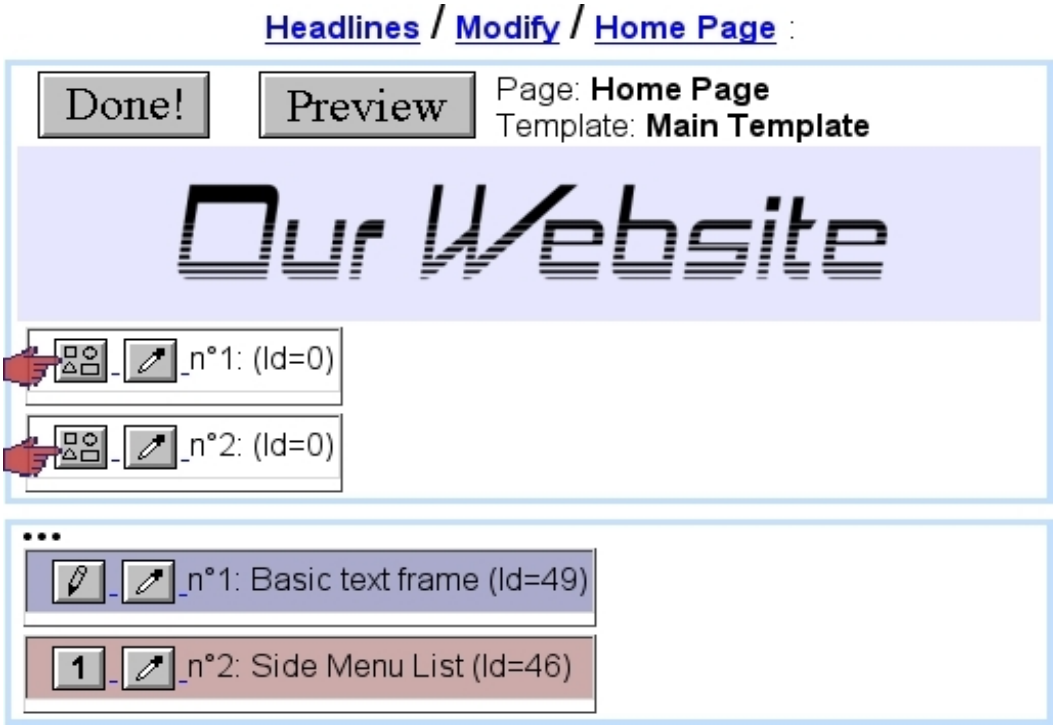


Figure 2.5: First edit ; as both frames use only one template ("Basic text frame" and "Side Menu List"), so clicking on the icon shows them immediately.

The buttons Done! and Preview as seen at the top of the page allow respectively to validate any modification on the headline and see how page would be rendered in real (without the frame menus and icons).

It is to note that these buttons have been created thanks to the inclusion of the `design.phs` script (Code 1), hence the importance of always including it for a headline model.

Outside the headline editing context, the preview of a headline can also be done through Headlines / Preview. .

Any editing to a frame is then performed through its own icon.

2.4.2 Editing a leaf frame

The output for the example Code 3 (a simple leaf frame template contents and variables) is featured at Fig.2.6. There, user fills the different variables that had been defined such as text fields, text zone (as using the Document Editor) and uploaded images (3.1).

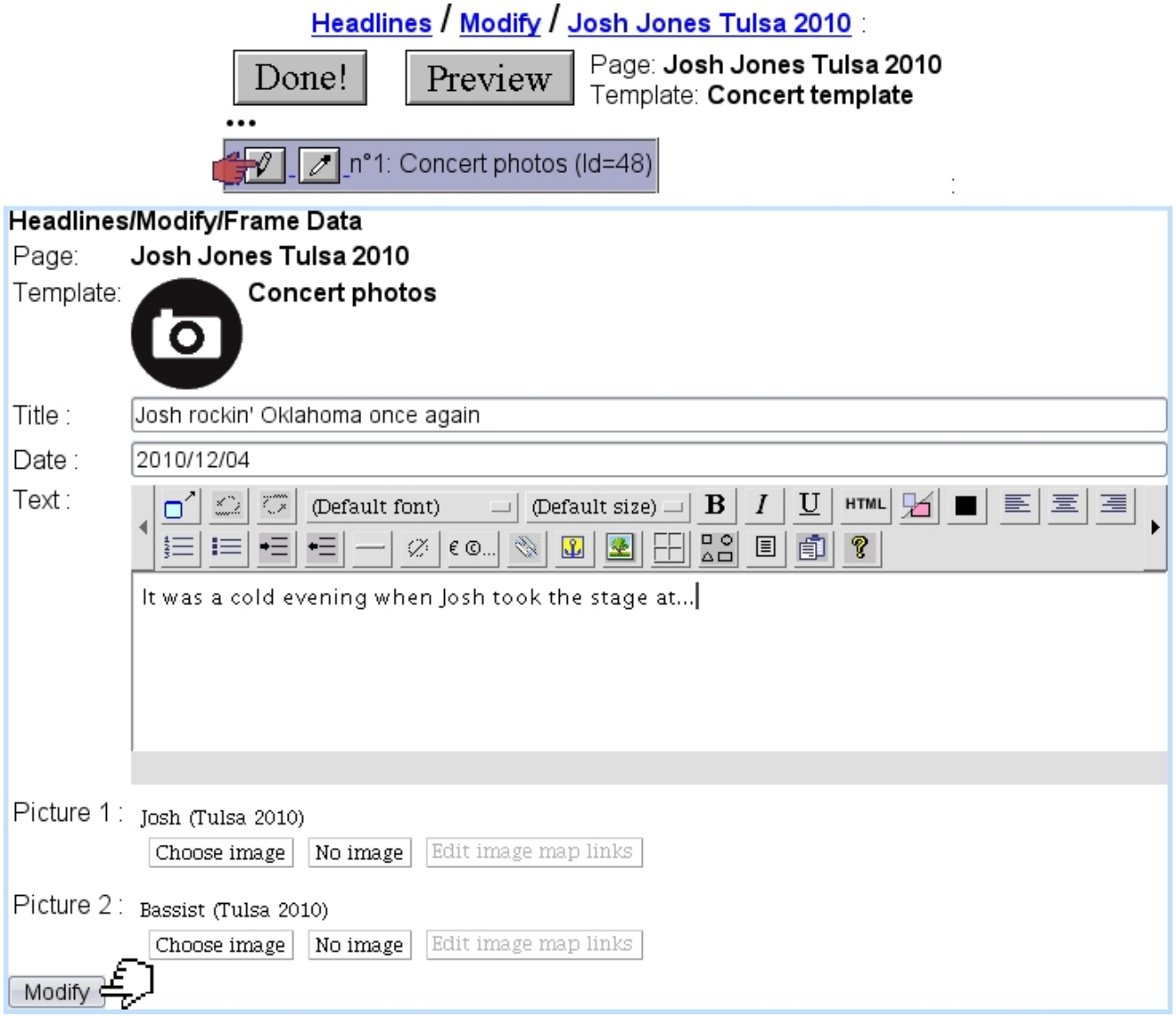


Figure 2.6: Filling in with information the leaf's various variables.

When everything is done, page is firstly validated via Modify ; it can then be Preview'ed (Fig.2.7), and, if modifications done are fine to finally validated, Done! is hit.

2.4.3 Editing a list frame

Lists frames are slightly more complicated, as they can yield an indefinite number of frames. In the following example, a list frame template "Side Menu List" has been created as in Code 2 and is using (2.2.3.1.1) leaf model "Simple List Element" (see 2.4.3.1 for more details about it).

As featured on Fig.2.8 :

Figure 2.7: A preview of the page.

- first, the 1 button of the list frame is clicked, which adds a new sub-frame, yet undefined ;
- the of this sub-frame is clicked, which already fills it with a "Simple List Element" leaf (as it is the only frame Side Menu List uses) ;
- the now-appearing Simple List Element frame then offers several icons, amongst which and : clicking on the latter adds another Simple List Element on the bottom (the former would have added it on the top) ;
- and so on, until all needed list elements are here.

Each list element features of course the icon, which allow to edit it ; also, the , allow to place the list element before the previous one (if not on the top of the list) or after the next one (if not on the bottom of the list).

A list element can be removed through .

As usual, the icon allows to edit list element's frame style (2.3.4).

2.4.3.1 Example for a leaf frame as used by a list

The leaf model "Simple List Element" as used above has the following contents :

```
<A HREF='«_Link »'>«_Text »</A>
```

and the corresponding **Variable** are defined as :

```
_Link;Link;4;
_Text;Text;1;
```

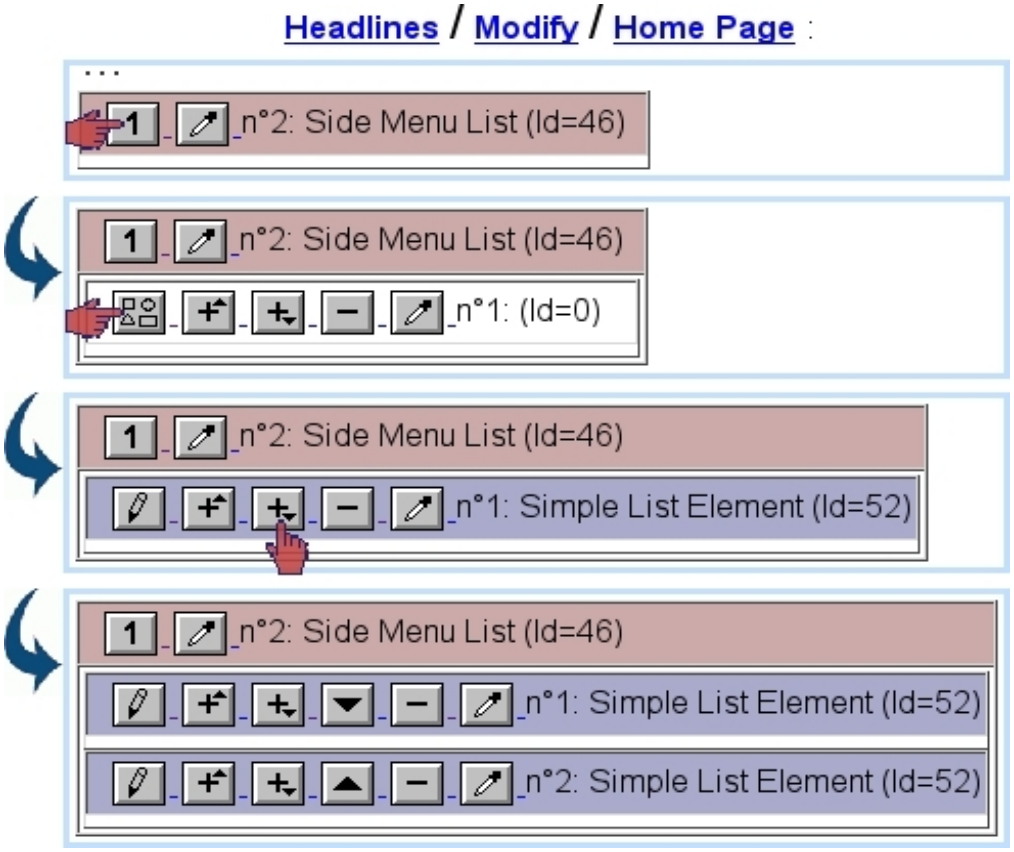
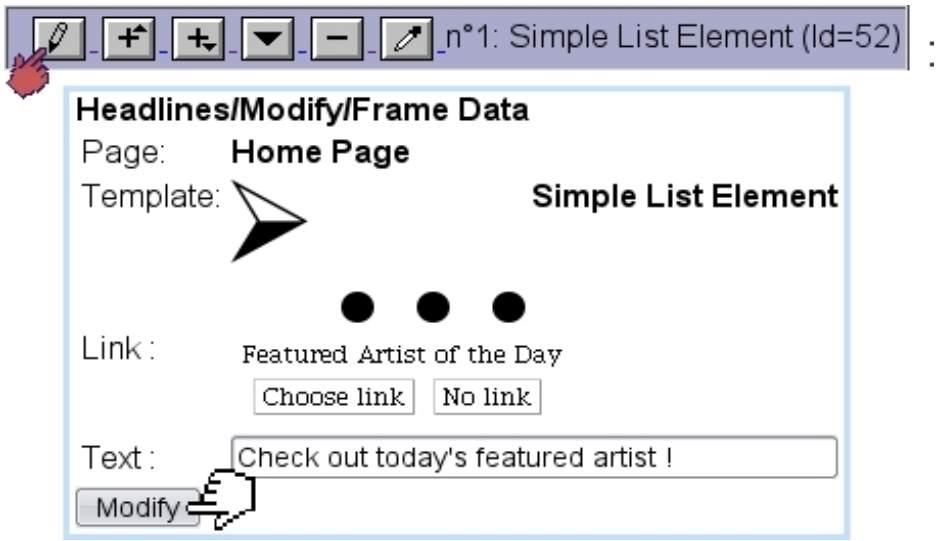


Figure 2.8: Populating a list frame.

When a list element will be edited, the following output will be shown :



2.5 Layouts

2.6 Articles and article models

User group rights for Articles and Article Models to be handled are respectively detailed at 1.3.2 and 1.3.14.

Articles and article models are in fact a deprecated version of Headlines, and therefore are currently not used anymore.

2.7 Managing pages

How headlines are managed through time is described next.

2.7.1 Publishing pages

User group rights for publishing pages are detailed at 1.3.8.

As a matter of fact, while most objects in Blue Chameleon Content Management System can be published at the time of their creation/modification, a page that is edited is not available right now on the website : it needs to be explicitely published first.

Publishing is done through **Headlines** / **Publish**, where a list of headlines in the current column - including its virtual columns (1.4.4.1) - is shown, with their publish status, related `.htm` file in the current folder, and the name under which it is published. For instance, the following :

Headlines / **Publish** :

would show that headline "Home Page", which is published under `Home_Page.htm` using file `20110323143529_16608.htm` (see 'Properties of headlines' (2.7.2) for more details about publication name and related file) has been edited but not published yet : simply clicking on **Home Page** would then publish it, and a (Published) status would replace (Modify).

As a column could contain a quite sizable number of pages, it is possible to publish multiple pages with a single click.

2.7.1.1 Multiple publish

On the **Headlines** / **Publish** page, a link **Click here for multiple headline publish** (also available for articles) redirects to a page as featured in Fig.2.9.

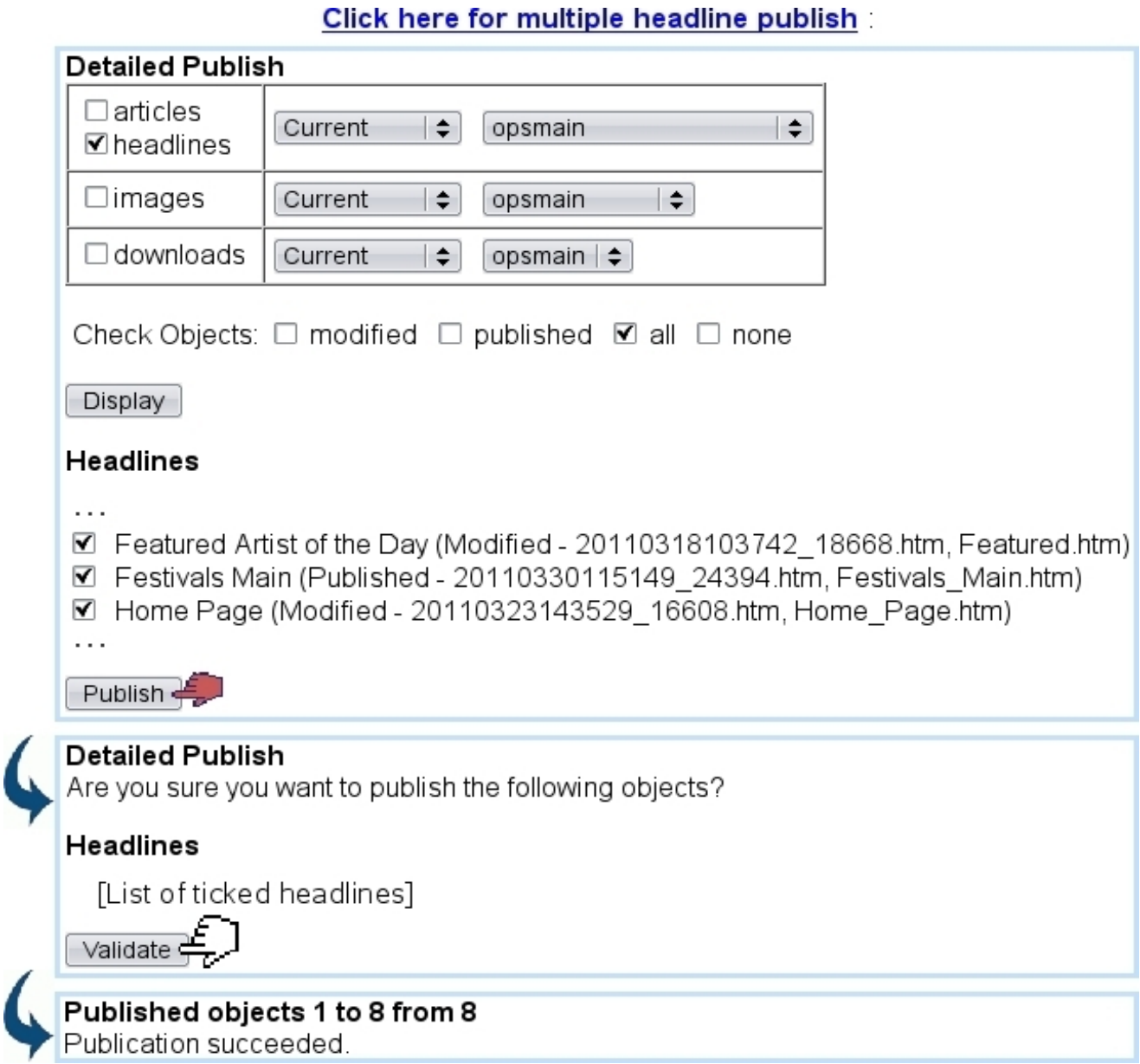


Figure 2.9: Publishing several headlines in one single time.

There :

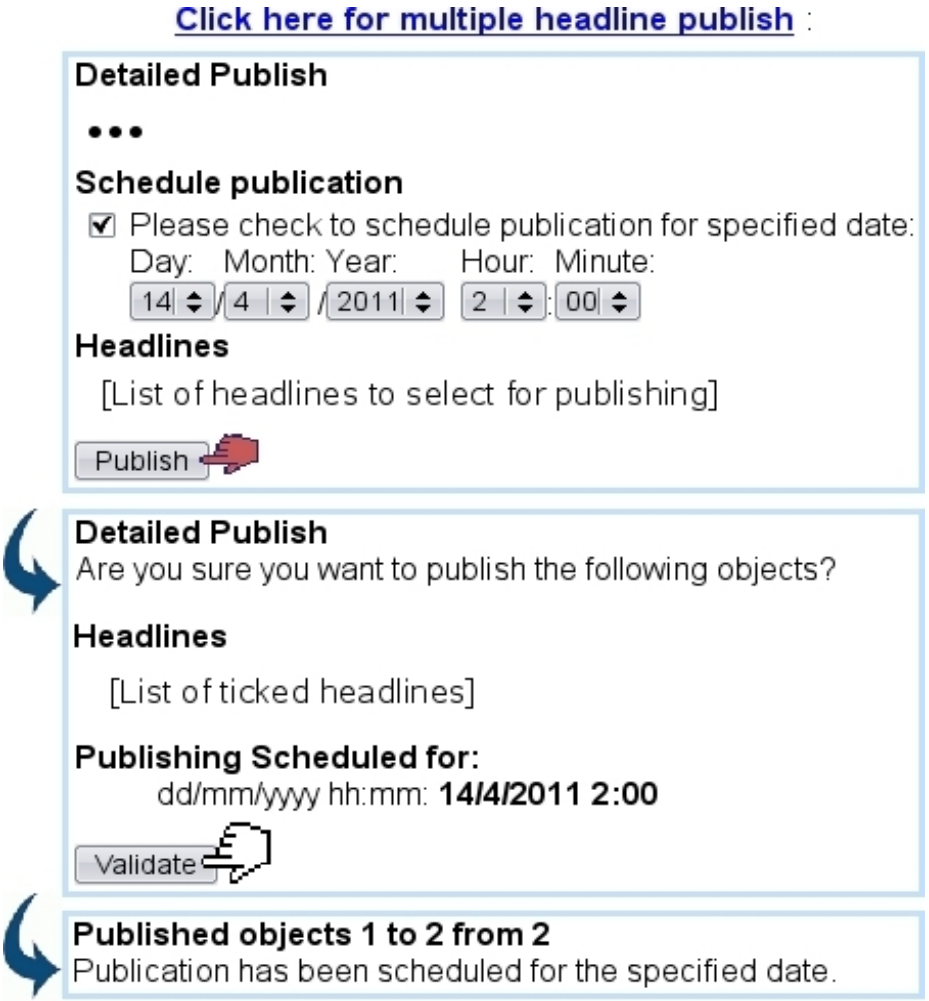
- a framed table with checkboxes (articles, headlines,...) stands as to pick objects of interest ; menus with options 'Current', '+Subfolder', 'Section' let then choose where the publishing task takes place, to where specified by the right right-hand menus, detailing the current column and its virtual columns ;
- a possibility to check objects, whatever their publication status is ;

- ...for the above choices, a Display button used to refresh the...
- list of headlines, each accompanied with a checkbox deciding if the publishing applies on it ;
- upon click on the Publish, a confirmation screen then appears, featuring the number of ticked pages to be published ;
- finally, upon validation, confirmation screen shows the result of the mass-publication.

Headlines that belong to edition-enabled columns (Fig.1.6) have dedicated publishing functionalities (4.5.2).

2.7.1.2 Scheduled publishing

Scheduled publication allows - in the multiple publish context - to program the automatic publication a selection of headlines on a specified date and time ; for it to be proposed, the Publisher settings (1.5) must have a user selected for 'Scheduled Publishing User'. Then, menus allowing to choose date and time will be provided :



2.7.2 Properties of headlines

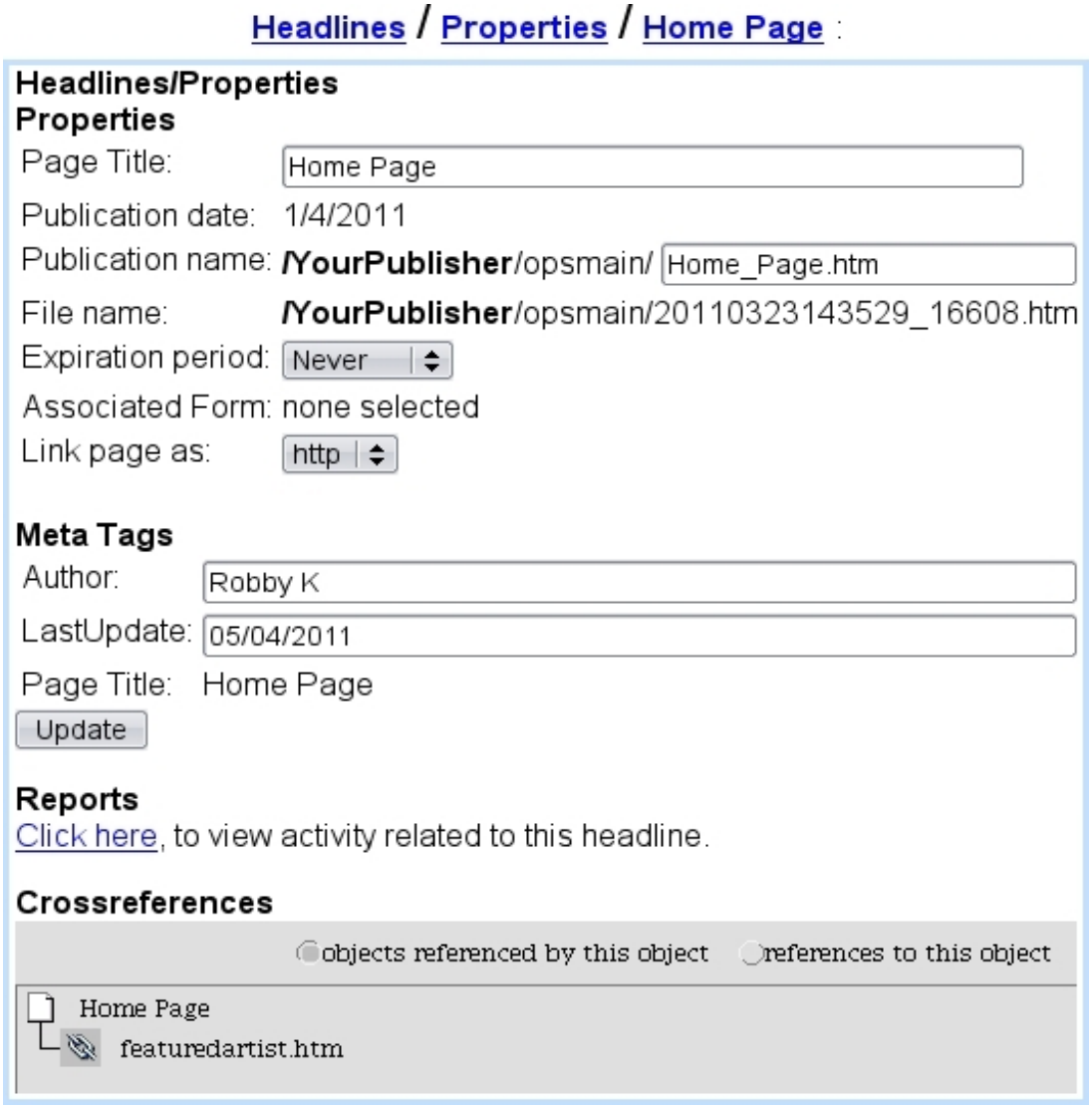


Figure 2.10: Configuring and viewing data related to headline "Home Page".

This page is useful to check any activity and useful data related to a headline. As featured on Fig.2.10, information and data configuring is multifarious :

- the headline's name, which can be modified ;
- when the last publication was done ;
- under which name this page is published.

After having created a headline, it is necessary, for linking purposes, to change this name to something URL-compliant, e.g. `Home_Page.htm`.
The page can then be added as a link, or accessed, via the path displayed on the screen (here, `YourPublisher/opsmain/Home_Page.htm`)

- by which physical file of the server (e.g. `20110323143529_16608.htm`) it is represented (name is mainly composed of year, month, day of when it was first created) ;
- whether this page will expire (by default set as in 1.4.3.2) ;
- whether it has an associated form (3.6) ;
- whether it is linked as `http` or `https` ;
- the Meta Tags, where the information configured in 2.2.3.2 is finally filled ; as a result, the `Home_Page.htm` page will contain the following :

```
...
<HEAD>
<meta name="Author" content="Robby K" />
<meta name="LastUpdate" content="05/04/2011" />
<meta name="Page Title" content="Home Page" />
</HEAD>
...
```

- a direct click-link to check the reports (4.6.2) related to this headline ;
- where it refers to, as well as where it is referenced from, as a tree structure that can be browsed.

Any modification in the fields and/or drop-down menus must be validated through the Update button.

2.7.3 Other management options for pages

A tree-structure view aimed at managing headlines directory-wise is accessible via **Headlines** / **Manage**, as illustrated in Fig.2.11.

Through a right-click on a headline's name, it is possible to copy a headline in the same column or to another and also - if headline is in the current column, to rename it. Previewing and refreshing are also available.

When right-click is done on a folder's name, the only options are refreshing and creating a new folder inside it (if clicked folder corresponds to the current column).

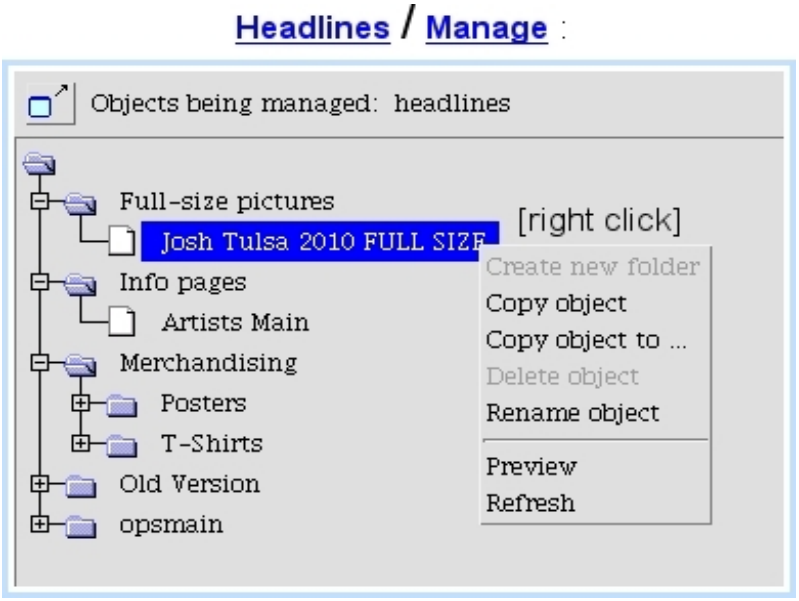


Figure 2.11: Managing headlines in a file system way.