Bitrix Site Manager 4.0

Integration Manual

Version 4.0.5 as of 04.27.2005

Introduction

This document is for use by web developers who are involved in developing web sites based on the Bitrix Site Manager 4.0. The document provides the detailed description of the system integration in a new or existing web site.

The document assumes that the web developer is closely acquainted with such web building technologies as HTML, CSS, PHP.

New Integration Technology

The Bitrix Site Manager 4.0 offers a wide range of innovations aimed to speed up the web site building process.

Design templates, the traditionally flexible facility, allow to integrate the software in the design within few hours, start developing the project structure and allocating the software components.

A design template can be uploaded in the form of a single file using the web interface and applied to one or more sites. The complete ready-to-go templates can be exported as a single file package and used in other projects.

Templates and sites

The Bitrix Site Manager 4.0 supports the multiple sites concept, which allows creating several sites using a single copy of the product. Each site has its own domain name, design, interface language and content.

The design of a site is the subject of a *design template*. Each site can be assigned an unlimited number of templates. The use of templates opens up vast possibilities in the site design customization depending on your needs and various conditions.

The software provides means for flexible design customization for different sites and site sections; for example, it allows using special holiday design for the specified period of time, assign certain design templates to different site visitor groups or depending on a parameter in the site URL etc (figure 1.1).

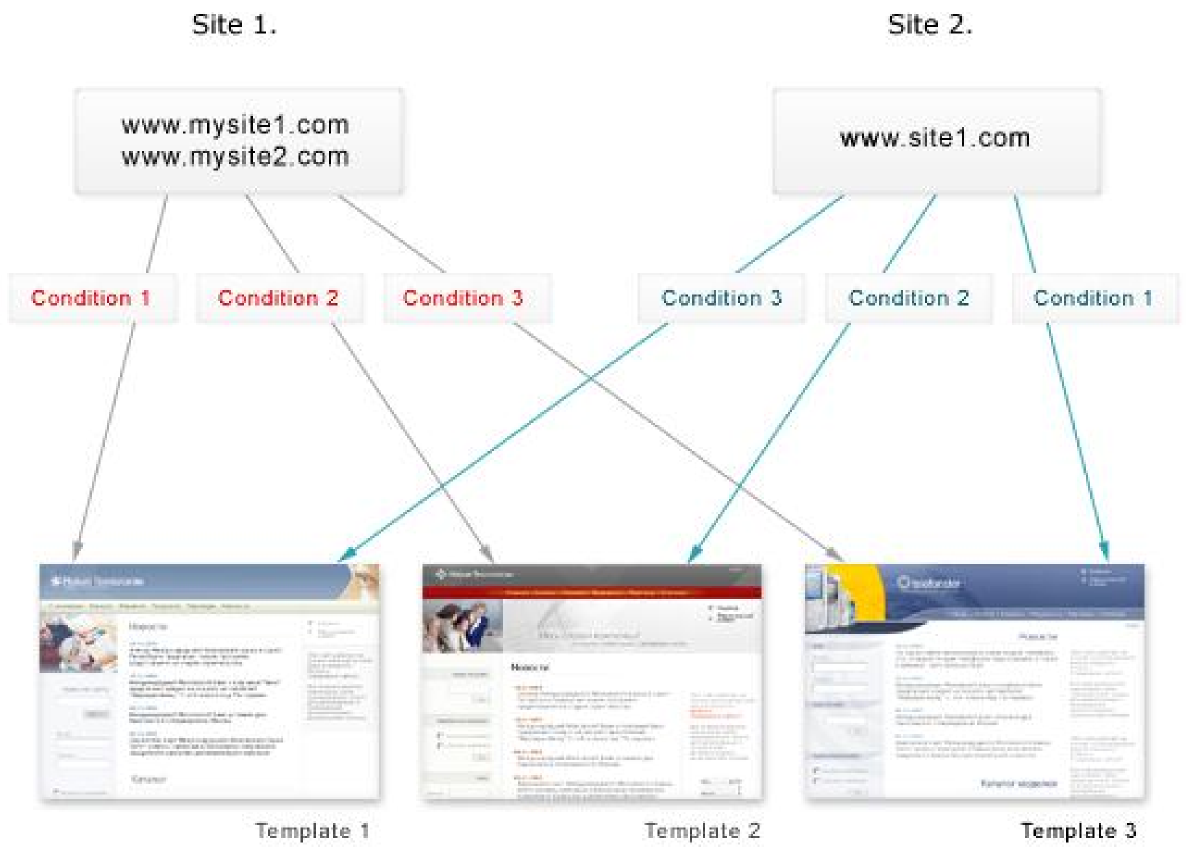
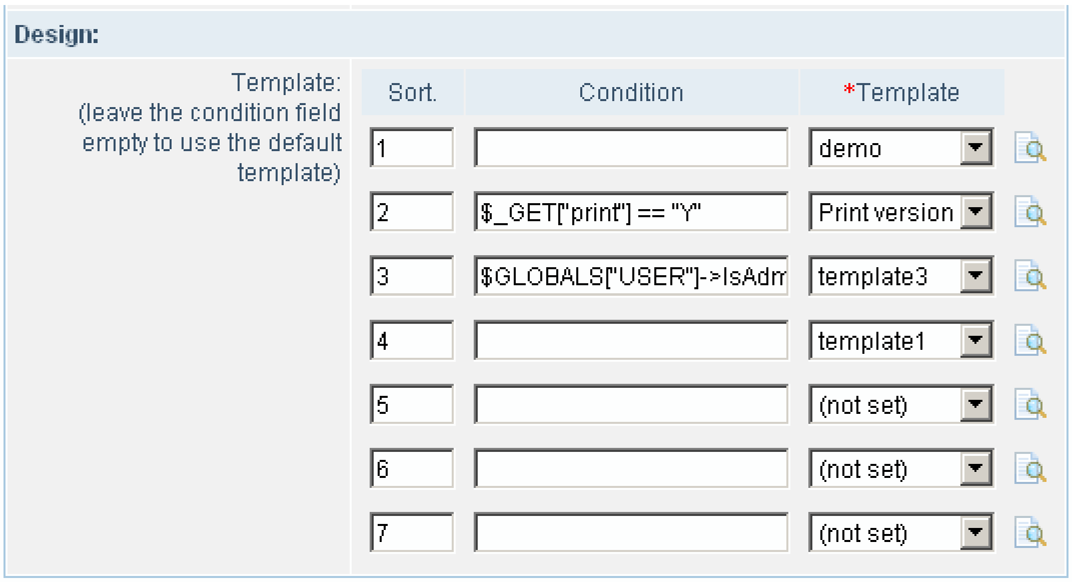


Fig.1.1. Templates and sites

You can select and assign a template to each site on the page *Settings->Sites* of the administrative section of the corresponding site.

You can define a set of conditions that will regulate the site design selection, in the site settings (figure 1.2). You may choose to not use conditions at all; in this case, the selected template is used as the default site template.


Design:
Template:
(leave the condition field empty to use the default template)

Sort.	Condition	*Template
1		demo
2	$_GET["print"] == "Y"	Print version
3	$GLOBALS["USER"]->IsAdm	template3
4		template1
5		(not set)
6		(not set)
7		(not set)

Fig. 1.2. Setting the site templates.

Settings on the above illustration tells the system to:

1. Display the site using the default template **demo**. This template will be used first because it is not assigned a condition and the sort weight is 1.
2. The **Print Version** template is used if the parameter **print=Y** is specified in the address URL. For example: if a visitor opens a page using a link like http://www.site2.com/?print=Y, a page will be displayed using the template that allows for the correct page printouts.
3. The **template3** is used if a user has been authorized and logged in as an administrator.

Note: a condition is allowed to contain an arbitrary PHP code including the Bitrix Site Manager SDK calls. For details, please see the SDK help section.

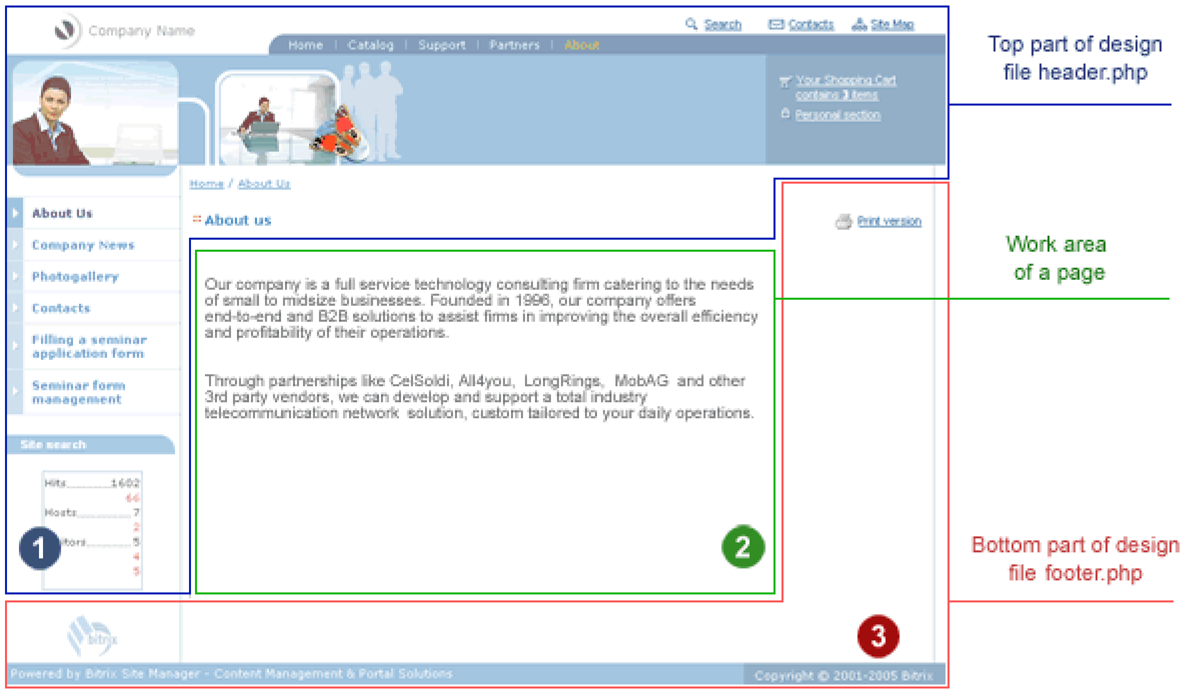
You can ensure that the template is configured correctly by clicking the icon near the templates drop-down list. This will open the *preview mode display* of a site with the selected template applied to it. (Site home page will be used for preview mode.) Clicking this button does not save your recent changes; this is for the preview purposes only.

An unlimited number of templates can be applied to a single site. A template can be reused for any other sites under different conditions.

Templates Management. Template Structure

A common site design usually includes three main parts (figure 2.1).

1. Topmost section (**header**).
2. Main section that is used to display the site presentation and information content, components or the code.
3. Bottom section (**footer**).


Company Name
Search
Contacts
Site Map
Home | Catalog | Support | Partners | About
Your Shopping Cart contains 3 items
Personal section
Home / About Us
About us
Print version
About Us
Company News
Photogallery
Contacts
Filling a seminar application form
Seminar form management
Site search
1
2
3
Top part of design file header.php
Work area of a page
Bottom part of design file footer.php
Powered by Bitrix Site Manager - Content Management & Portal Solutions
Copyright © 2001-2005 Bitrix

Our company is a full service technology consulting firm catering to the needs of small to midsize businesses. Founded in 1996, our company offers end-to-end and B2B solutions to assist firms in improving the overall efficiency and profitability of their operations.

Through partnerships like CelSoldi, All4you, LongRings, MobAG and other 3rd party vendors, we can develop and support a total industry telecommunication network solution, custom tailored to your daily operations.

Fig. 2.1. Main design sections.

Let us consider how the design sections can be implemented in a template. We shall elaborate on the template **demo**, included in the software installation package.

You manage your site templates in the administrative section *Settings->Site Templates*. There you can view and edit the existing templates and also add new templates. Click *Modify* to start editing the template **demo**.

The field **Site template design** contains the template code.

Note: the template contains a special separator #WORK_AREA# used to delimit the header and the footer sections.

This field displays a template as the joined header and footer parts of the site design in the HTML format. The code can contain components and functions written in the PHP language, which provides for the display of different information: metadata, page header, CSS administrative toolbar. You can add and edit components and functions using the visual editor tools.

The template is stored in files **header.php** and **footer.php**.

No limitations on the temlpate composition and site design.

The use of executable code in templates ensures the flexibility in implementing a web project, enables to develop both simple and complex templates with arbitrary logics and individual design. If you are not quite familiar with the PHP language, you can include software components by simple copying the required code from other templates.

Cascading style sheet (CSS) file used in the template is shown in the field **Style sheet**. The CSS is stored in file styles.css.

All templates are stored in directory /bitrix/templates/. Files that comprise the template are stored in subdirectory named by the template identifier. In this example, the template files are located in subdirectory /bitrix/templates/demo/.

When you create a new template using the administrative interface, you provide its identifier, name and description for use in the list of templates. Also you need to provide the site design code, CSS and a set of included components and pictures. When you save the newly created template, the directory /bitrix/templates/<template_identifier> is created automatically.

The file and directory structure of each template is fairly identical. The simplest template may include only the following required files: header.php, footer.php, styles.css, as well as some more menu template files.

All images used in a template are stored in the subdirectory /bitrix/templates/<template_identifier>/images/.

Any other files and components may also be stored in the template directory. You can view the template structure by clicking the link near the site identifier. Template files are shown in the administrative section **File Manager**.

You can split the site template into any number of files representing the site design sections to facilitate access to them. Examples of such sections are: copyrights area, contact information area etc.

Include areas and components

To provide the visual dynamic control over a modern web site, some design parts are implemented as the software components.

Get back to the template **demo** and take a look at the basic components and include areas. The figure 3.1 spotlights components and include areas of this template.

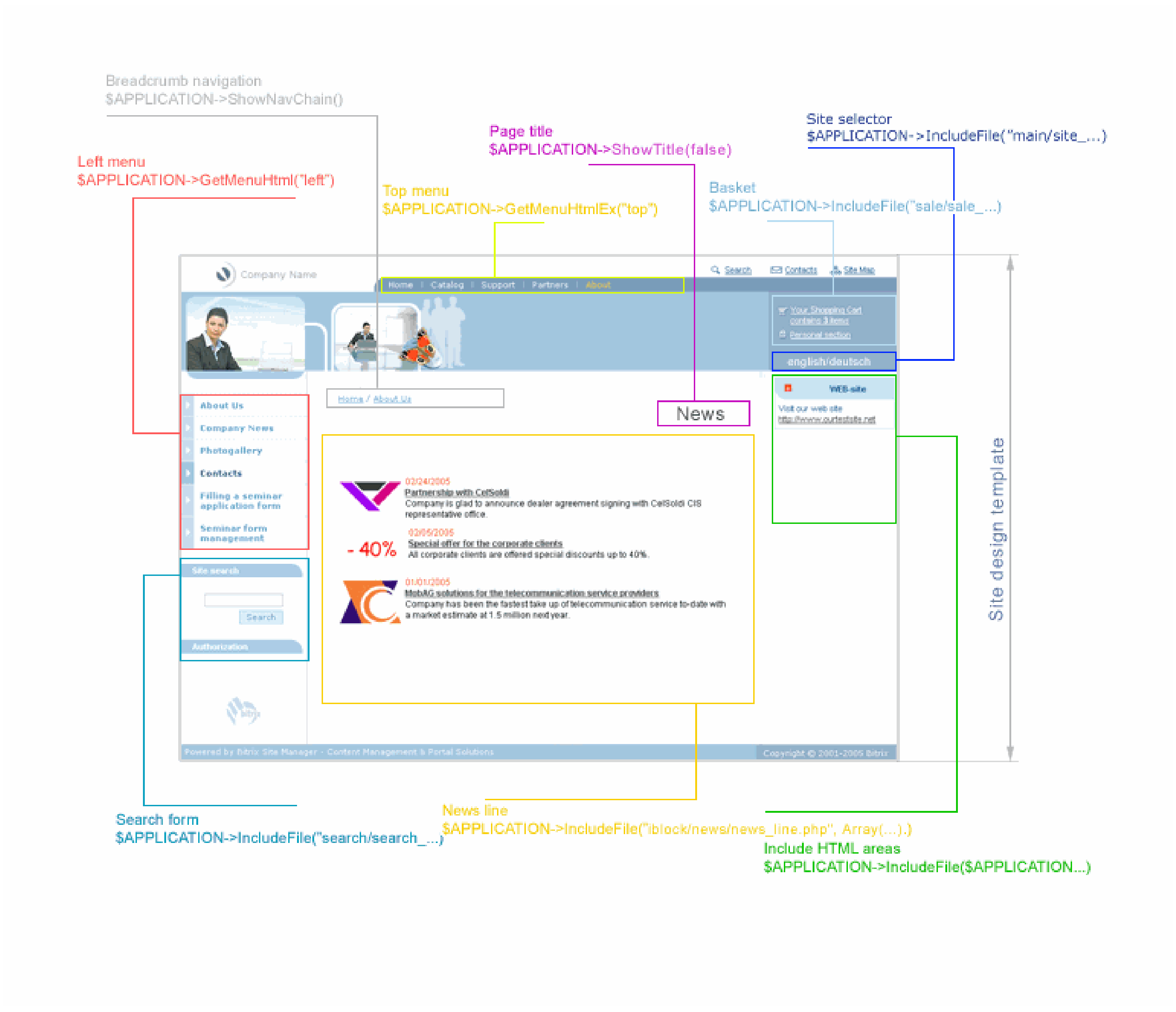

Breadcrumb navigation
$APPLICATION->ShowNavChain()
Page title
$APPLICATION->ShowTitle(false)
Site selector
$APPLICATION->IncludeFile("main/site_...)
Left menu
$APPLICATION->GetMenuHtml("left")
Top menu
$APPLICATION->GetMenuHtmlEx("top")
Basket
$APPLICATION->IncludeFile("sale/sale_...)
Company Name
Home | Catalog | Support | Partners | About
Search
Contacts
Site Map
Your Shopping Cart contains 3 items
Personal section
english/deutsch
Home / About Us
News
About Us
Company News
Photogallery
Contacts
Filling a seminar application form
Seminar form management
WEB-site
Visit our web site
Site search
Search
Authorization
Partnership with CelSoldi
Special offer for the corporate clients
- 40%
MobAG solutions for the telecommunication service providers
Powered by Bitrix Site Manager - Content Management & Portal Solutions
Copyright © 2001-2005 Bitrix
Site design template
Search form
$APPLICATION->IncludeFile("search/search_...)
News line
$APPLICATION->IncludeFile("iblock/news/news_line.php", Array(...).)
Include HTML areas
$APPLICATION->IncludeFile($APPLICATION...)

Fig. 3.1. Include areas in template.

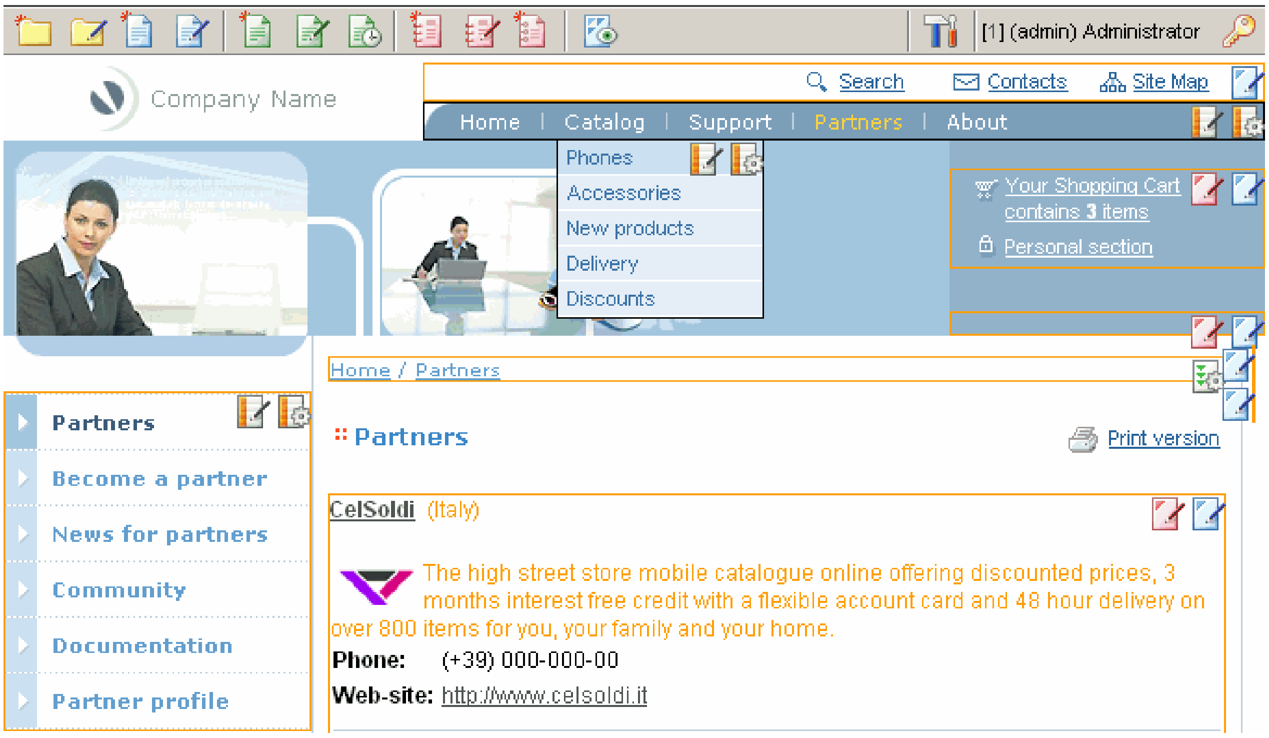
When generating the HTML code, the system replaces the spotlighted areas with inclusions of the corresponding components (figure 3.1).

The generated code includes functions that provide the display of different visual information: metadata, page header, CSS, administrative toolbar.

The following navigational templates are created separately: navigation chain (breadcrumb navigation), site menus.

After an HTML prototype has been created, function calls and components generated, you obtain the ready-to-go PHP template of the site design.

You can turn on the special mode that enables to view the include areas by clicking the button in the administrative toolbar (fig. 3.2). This provides quicker access to include areas and components and enables to modify their parameters visually.



CelSoldi (Italy)

The high street store mobile catalogue online offering discounted prices, 3 months interest free credit with a flexible account card and 48 hour delivery on over 800 items for you, your family and your home.

Phone: (+39) 000-000-00

Web-site: http://www.celsoldi.it

Fig. 3.2. Include areas and components view mode.

Include components may have various controls indicated by icons.

For example, the advertising banner controls provide access to the banner editing facilities. Additionally, it enables to view the list of banners filtered by the selected banner type, in the administrative section.

The menu controls enable to edit the menu items of the selected section or switch to the menu template editing.

You can define and use in your design various include HTML areas stored as individual files. You may want to set these areas to display under certain conditions only, for example in current site section or page. You can use the include area controls to switch to their editing immediately.

Integration Explained

File Structure

The Bitrix Site Manager framework is designed to separate the visual aspect from the software kernel of a site. The software kernel is located in the folder /bitrix/ relative to the site root.

Subdirectories of this folder contain the following files,

/bitrix/templates/ - site design templates; components used by the templates. Basically, the integration process affects files in this directory only;

/bitrix/admin/ - administrative interface. Provides forms for managing the system modules;

/bitrix/cache/ - cache files created while caching the dynamic information;

/bitrix/php_interface/ - auxiliary system files (database connection preferences, other information);

/bitrix/modules/ - class libraries of the system modules;

/bitrix/images/ - images for the system modules;

/bitrix/tools/ - auxiliary files;

/bitrix/updates/ - used for the updates downloads;

/bitrix/ - administrative files.

Typical Page

A typical page is assembled by attaching a header and footer to it. The below is the common structure of a site page.

```
<?
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetTitle("Bitrix Site Manager 4.0");
?>
Page body text…
<?
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");
?>
```

Properties of Pages and Folders

The Bitrix Site Manager features assigning properties to pages and sections. This enables to flexibly manage the information show. By setting properties to the site pages, you can assign the metadata values for different pages; change the visual aspect of the site design on the fly depending on the active section or page and so on.

You can set the property values by either using the administrative section or directly from the program script.

The product has several reserved names of properties used by some system functions.

Reserved properties include:

- **title** – used to set the additional page title (see *Setting the Page Title*);
- **adv_desired_target_keywords** – used to set the desirable keywords for the advertisement shows (see *Placing the Advertisement*)
- **not_show_nav_chain** – used to disable the navigational chain (breadcrumb navigation) for page or section (see *Customizing the Navigational Chain*).

Creating a Site Template

Process of converting an HTML template to a fully functional PHP template involves replacing the HTML code with the corresponding PHP functions calls, methods and software components.

Please note that the version 4.0 uses a new set of functions allowing to control the metadata, title, navigational chain and include areas of a template.

New functions ShowMeta(), ShowTitle(), ShowCSS() etc. allow to initialize certain elements directly from the script placed in a page or from a component. For example, you can add a page title after the script results has been written to a page. Thus, if in previous program release you had to initialize the page title before including the page design header, the new version allows to set the page title directly from the code located anywhere in the page body. The detailed description of functions used in the common template is provided below.

Meta Data Control

Design

An example of meta data control is the mechanism of assigning keywords and descriptions to pages and sections of a site. By default, the Bitrix Site Manager distributive has these two types of meta data configured. You can extend the list of the available meta data types by implementing the same algorithm.

How it works

You can control the meta data through the properties of pages and sections of the site. Page and section properties can be edited in the embedded visual HTML editor (button of the top toolbar). In the visual HTML editor, click the button and type the required values of these properties.

Also you can set the property values when editing a page in the *text* mode.

If you need to assign properties to the whole section, click the button in the administrative toolbar to open a dialog used to edit properties of a section.

Implementation

The following function calls result in output of the page keywords and description, respectively:

```
<head>
…
<?$APPLICATION->ShowMeta("keywords")?>
<?$APPLICATION->ShowMeta("description")?>
…
</head>
```

These function calls generate the following HTML code:

```
<meta name="keywords" content="Mobile phones,accessories,Alcatel,Siemens,
Motorolla">
<meta name="description" content="Mobile Store">
```

Page properties can be set from the script dynamically. For example: properties of pages that are used to display the catalog contents or the news (i.e. **information blocks**), can be set with respect to values of an information block elements. Thus, you can create properties *keywords* and *description* for each element of an information block and apply them to a page dynamically.

Setting the Page Encoding

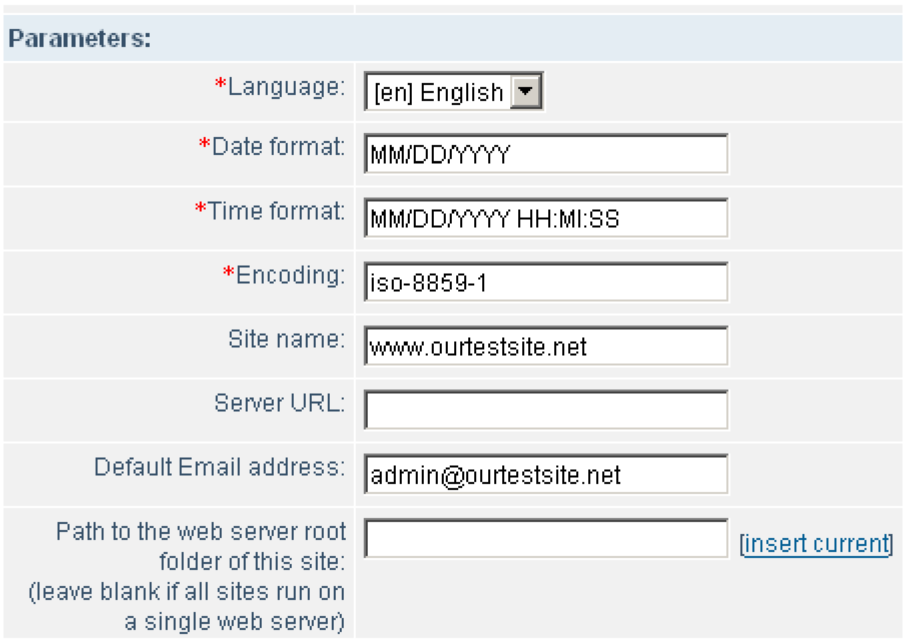
Design

Different interface languages require different symbol encoding. Browsers use encoding for the correct text display. Some languages cannot be rendered using only one single-byte charset, e.g. Greek and French.

How it works

You can control the public section languages in the administrative section **Sites**. You can assign a unique encoding, date and time format to a site.

You have to set the date and time format accurately so that this kind of information is displayed to visitor correctly.



Parameters:

*Language:	[en] English
*Date format:	MM/DD/YYYY
*Time format:	MM/DD/YYYY HH:MI:SS
*Encoding:	iso-8859-1
Site name:	www.ourtestsite.net
Server URL:	
Default Email address:	admin@ourtestsite.net
Path to the web server root folder of this site: (leave blank if all sites run on a single web server)	[insert current]

Fig. 4.1. Language encoding; date and time format.

Implementation

The encoding is rendered as the PHP constant, which is later replaced with a string from the site language settings:

```
<head>
…
<meta http-equiv="Content-Type" content="text/html; charset=<?echo
LANG_CHARSET?>">
…
</head>
```

Including the CSS

Design

Several stylesheets are applied to pages of a site built with the Bitrix Site Manager.

Stylesheets are uniquely customized for each site template in the system; each stylesheet set is stored in the folder of the corresponding template, as well as other files comprising the interface of the template.

It is important to add comments when building a stylesheet. You should add comments to the styles that you plan to utilize for visual editing of the pages in the WYSIWYG editor. Styles with the comments will be available in WYSIWYG editor in the styles drop-down list with the names specified in comments. (Note that there should not be any spaces between style definition and comment)

You need to add comments to your CSS files for each site template.

Please note that only comments after the style definition will be used for visual editor all the other commented lines will have no effect. This allows you to name sections of your CSS file for more flexible CSS control.

The following is an example of the site template CSS file.

Left menu styles:

```
.leftmenu, .leftmenuact {font-family: Arial, Helvetica, sans-serif; font-
size:12px; font-weight: bold; text-decoration: none;}
.leftmenu {color: #3F3F3F;}
.leftmenuact  {color:#FF5235;}
```

Top menu styles:

```
.topmenu  {font-family: Arial, Helvetica, sans-serif; font-size:12px; font-
weight: bold; color: #FFFFFF; text-decoration: none;}
.topmenuact  {font-family: Arial, Helvetica, sans-serif; font-size:12px;
font-weight: bold; color: #FAC535; text-decoration: none;}
```

Table styles:

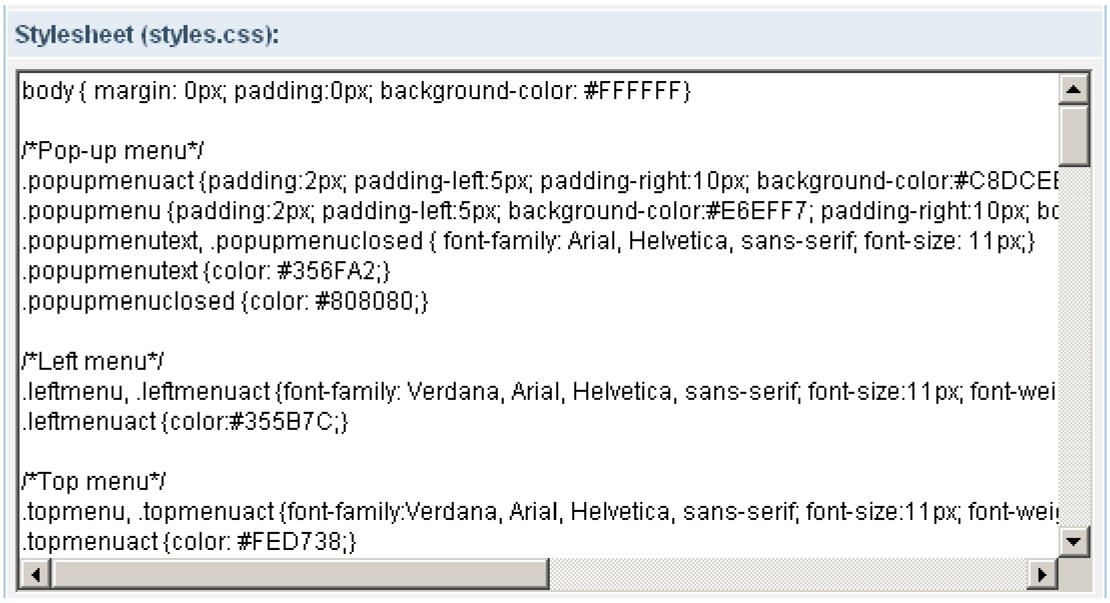
```
.tableborder  {background-color:#9C9A9C;}
.tablehead  {background-color:#D8D9DA;}
.tablebody  {background-color:#F8F8F8;}
.tablebodytext, .tableheadtext {font-family: Arial, Helvetica, sans-serif;
font-size:12px;}
.tablebodytext {color:#000000;}
.tableheadtext {color:#000066;}
```

Text formatting styles:

```
.text  {font-family: Arial, Helvetica, sans-serif; font-size:12px; font-
weight: normal;}/*Normal*/
.titletext  {font-family: Arial, Helvetica, sans-serif; color:#585858; font-
size:15px; font-weight: bold; line-height: 18px;}/*Title*/
.subtitletext  {font-family: Arial, Helvetica, sans-serif; color:#000000;
font-size:11pt; font-weight: bold;}/*Subtitle*/
```

How it works

Styles of each template can be customized in the **Site Templates** section.


Stylesheet (styles.css):

```
body { margin: 0px; padding:0px; background-color: #FFFFFF}

/*Pop-up menu*/
.popupmenuact {padding:2px; padding-left:5px; padding-right:10px; background-color:#C8DCEE
.popupmenu {padding:2px; padding-left:5px; background-color:#E6EFF7; padding-right:10px; bc
.popupmenutext, .popupmenuclosed { font-family: Arial, Helvetica, sans-serif; font-size: 11px;}
.popupmenutext {color: #356FA2;}
.popupmenuclosed {color: #808080;}

/*Left menu*/
.leftmenu, .leftmenuact {font-family: Verdana, Arial, Helvetica, sans-serif; font-size:11px; font-wei
.leftmenuact {color:#355B7C;}

/*Top menu*/
.topmenu, .topmenuact {font-family:Verdana, Arial, Helvetica, sans-serif; font-size:11px; font-wei
.topmenuact {color: #FED738;}
```

Fig. 4.2. Modifying a CSS table in the site template settings

Implementation

CSS tables should be linked within the <head> tag of a header (**header.php**) in the following way.

```
<?$APPLICATION->ShowCSS();?>
```

All the required HTML code will be inserted by the API function specified above.

This function includes a CSS file from the current template. Also it includes all additional styles defined for the page by calling the function SetAdditionalCSS() internally.

If called without arguments, the ShowCSS() method includes styles by adding the following code:

```
<LINK href="/bitrix/templates/demo/styles.css" type="text/css"
rel="STYLESHEET">
```

All styles added by calling SetAdditionalCSS() will be included in the page code as follows: require(…).

If called with the **false** argument:

```
 <?$APPLICATION->ShowCSS(false);?>
```

then the CSS file of the current template is also included by calling require().

Navigation chain (Breadcrumb navigation) setup

Design

A navigation chain helps to display the nesting level of the current page, site section or goods catalog, starting from the home page to the current document. Values indicated in the navigation chain can be specified for each section or document individually.

A separate PHP template should be created to show the navigation chain on a site page.

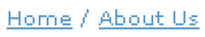


Fig. 4.3. Navigation chain

If needed, you can override the navigation chain item text for the specific page. To do so, just call the function AddChainItem() directly from the page.

Please note that some scripts and components included in the software distribution package may add specific elements to the navigation chain. For example, the Catalog component sequentially adds names of catalogs and product groups to the navigation chain according to the catalog nesting level. Forum and forum topic names are added to the navigation chain in the similar way.

How it works

A value that is specified for a site section is stored in the file .section.php of the corresponding section. The value can be set or modified using the administrative toolbar (button **Folder Properties**), or by directly editing the folder properties of a required section in the administrative interface (**Site Explorer** module).

Sample file.section.php:

```
<?
    $sSectionName = "Home";
?>
```

Implementation

A template that is used to display the navigation chain is defined in the file /bitrix/templates/<template name>/chain_template.php for each site template individually. The structure of a navigation chain template is similar to that of a menu template. The template includes a header, a body element and a footer parts. Body element defines a template used to display a single element. The whole chain is built in a loop.

Sample navigation chain template:

```
<?
//--- Chain header
$sChainProlog="";
//--- Body element
 $sChainBody = "";
//--- The variable $ITEM_INDEX contains the current element number.
if($ITEM_INDEX > 0)
$sChainBody = " &nbsp;&raquo;&nbsp;";
$sChainBody .= "<a href=\"".$LINK."\"
class=\"smalltext\">".htmlspecialchars($TITLE)."</a>";

//--- Footer
$sChainEpilog="";
?>
```

You can quickly switch to editing the navigation chain template by clicking the button while in the editable area display mode.

You can set the individual navigation chain template for a separate site section. To do so, define a variable $sChainTemplate in the file .section.php and assign it a string value containing the full path to the navigation chain template.

The navigation chain template also can be specified as one of parameters of the function ShowNavChain().

Trial version of the product contains the navigation chain template in the default site template folder: /bitrix/templates/.default/.

Sample code used to display the navigation chain in a common template:

```
<?
$APPLICATION->ShowNavChain();
?>
```

Navigation chain can be used on different pages of a site. For example, special navigation chain template is used in the Search component as one of its parameters (path to the navigation chain template).

Trial version contains the additional navigation chain template for use with the Search template. It can be found in the default site template folder /bitrix/templates/.default/chain_template_search.php.

The below is given an example of adding elements to the navigation chain using the function AddChainItem():

```
<?
//--- The first parameter of the function AddChainItem() is the name
//--- to be shown in the navigation chain;
//--- the second parameter is the link URL.
//--- Parameter values can be both static and dynamic.
//--- In this example, section name is a static value, while
//--- the link is generated dynamically.
$APPLICATION->AddChainItem("Product details",
"catalog.php?BID=".$arIBlock["ID"]."&ID=".$arSection["ID"]);

//--- The next example shows how to generate both parameters dynamically.
//--- Current name of the catalog section is used as the name.
$APPLICATION->AddChainItem($arSection["NAME"],
"catalog.php?BID=".$arIBlock["ID"]."&ID=".$arSection["ID"]);
?>
```

To display the title of a current page in the navigation chain, call the function AddChainItem() in file **footer.php**; that is included after the main content is generated.

```
<?$APPLICATION->AddChainItem($APPLICATION->GetTitle());?>
```

You can set some of the navigation chain elements to be displayed with no link, as a common text (for example, display the current page title without link):

```
if (strlen($LINK)>0)
    $sChainBody .= "<a href=\"".$LINK."\"
class='".$strclass."'>".$TITLE."</a>";
else
    $sChainBody .= "<font class='".$strclass."'>".$TITLE."</font>";
```

You can turn off the navigation chain show for the desired pages or sections. To do so, specify a page or section property with the name not_show_nav_chain and set it to Y.

Page title setup

Design

A page title is displayed in the browser window caption and in the main area of the page body.

About us - Microsoft Internet Explorer

Fig. 4.4. Browser window caption with the page title

Home / About Us

:: About us

Fig. 4.5. Page title in the main area of the page body

How it works

You can change the page title while editing a document in the WYSWYG editor in the HTML mode (button on the top toolbar). In the visual mode, click the button in the editor toolbar and enter the desired text. The required PHP code will be generated automatically (see below).

You can change the page title in the plain text editing mode.

Also, you can edit the title directly by editing the HTML code of your page.

The title displayed in the browser window caption may differ from the main title of a page. You can take advantage of the page property **title** to define it.

Implementation

You can define the page title in any place of a page. Usually, page title is added in the following way:

```
<?
//--- Static title
$APPLICATION-> SetTitle("About us");?>

//--- Dynamic title. An information block name for the given $ID
//--- is used as the page title.
$arIBlock = GetIBlock($ID, "news")
$APPLICATION->SetTitle($arIBlock["NAME"]);
…
?>
```

The browser window caption text (title) can be defined using different techniques. By default, the title is set using the special page property **title**. If the property value is void, the browser window caption is the same as the current page title.

You can set the browser window caption text by calling the following code in a template:

`<?$APPLICATION->ShowTitle()?>`

The above code is to be placed within the <head> tag of the main prolog:

```
<head><title><?$APPLICATION->ShowTitle()?></title></head>
```

and within the document <body> tag exactly where the page title is to be displayed.

```
<?$APPLICATION->ShowTitle(false)?>
```

In the latter case, the parameter **false** tells to not check the value of the property **title**. A common title previously set by calling `SetTitle()` is used instead.

Showing the control toolbar

Design

The administrator can access the control toolbar on top of a page after the successful authorization. It can be used for setting the current section parameters, editing of the currently displayed page and/or editable areas, adding or editing menu of the current section. The toolbar can also be used to quickly switch to the administrative section of the site or log off the administrative session on the site.

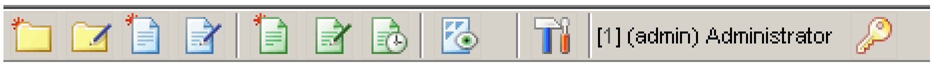


Fig. 4.6. Administrative toolbar

How it works

A certain administrative function is assigned to each toolbar button.

The below is the short review of the functions available on the administrative toolbar (the functionality of the Workflow and Information Blocks modules are omitted):

Create New Folder. Shows the dialog box used to create a new folder using the Site Explorer. You can define both the folder name and the section name to be used for building the navigation chain. This button allows to automatically create the index page for the section using a predefined template and edit it without hassle.

Current Folder Properties. Shows the dialog box used to manage the folder properties using the Site Explorer. You can define or edit the section name which is used for building the navigation chain.

Create a new document. Switches to the default editor (can be specified in Site Explorer module settings) using the predefined page template.

Edit current page. Switches to the default editor to create or edit the current page in the current section.

Button for switching to the Administrative section.

Log off button – for logging off the administrative session.

Edit the include area of the section. Switches to the default editor to create or edit the include area for the current section.

Edit the include area of the current page. Switches to the default editor to create or edit the include area for the current page.

Note: Buttons “Edit the include area of the section” and “Edit the include area of the current page” are provided for compatibility with the product versions 3.x. They are obsolete and no longer in use in 4th product version. Can be turned on/off on the panel in the **Site Explorer** module settings. (option *Show additional buttons in toolbar*). Buttons were replaced with the ones available in the page work area.

Implementation

The code that includes the administrative toolbar should be added immediately after the <body> tag before any tables.

```
<?
$APPLICATION->ShowPanel();
?>
```

Certain modules may add buttons to the toolbar. Examples of such modules are the Workflow and Information Blocks modules.

A user can also add buttons to the toolbar. The following two methods can be used to add buttons:

- by editing the file: "/bitrix/php_interface/include/add_top_panel.php". The function $APPLICATION->ShowPanel() checks whether this file exists and adds buttons described in this file to the toolbar.

- by using the function CIblock::ShowPanel() in the display script, in case of displaying additional buttons for the Information Blocks module. The number of buttons may vary; for example whether the list of elements or the element details are displayed.

Placing the Advertisement

Design

A site may have several different advertisement types and advertising areas. Such advertisement may include both common image-based and text-based banners. Advertisement can be shown on continuing basis as well as its shows can be regulated by the administrator-assigned probability. Advertisement shows can be restricted to certain sections or pages of the site. To get the detailed information on the advertisement, please see the corresponding topics of the help section.

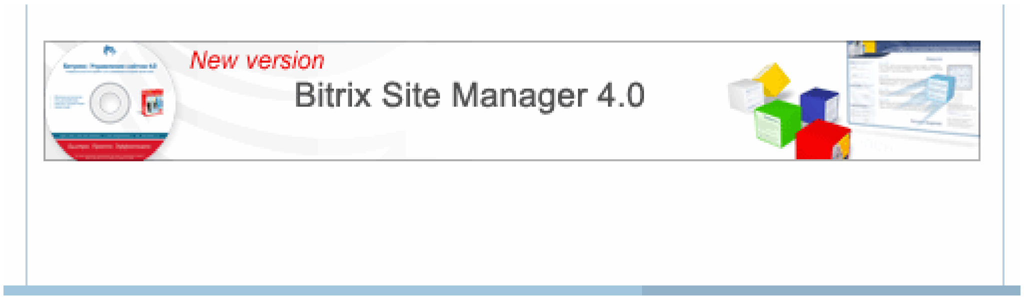
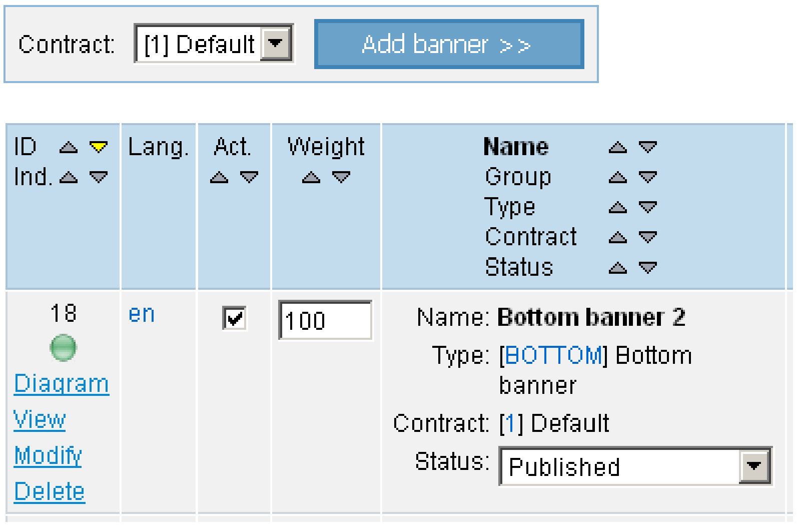


Fig. 4.7. Sample advertisement

How it works

You can add new types of advertisement and advertising media through the use of the **Advertising and Banners** section of the administrative interface. The administrator can assign an arbitrary name to an advertising block type. Such names will be used as the reference symbol to select the advertisement for the show. For example, advertisement of the top area can be assigned a type name of TOP, TOP1 etc.

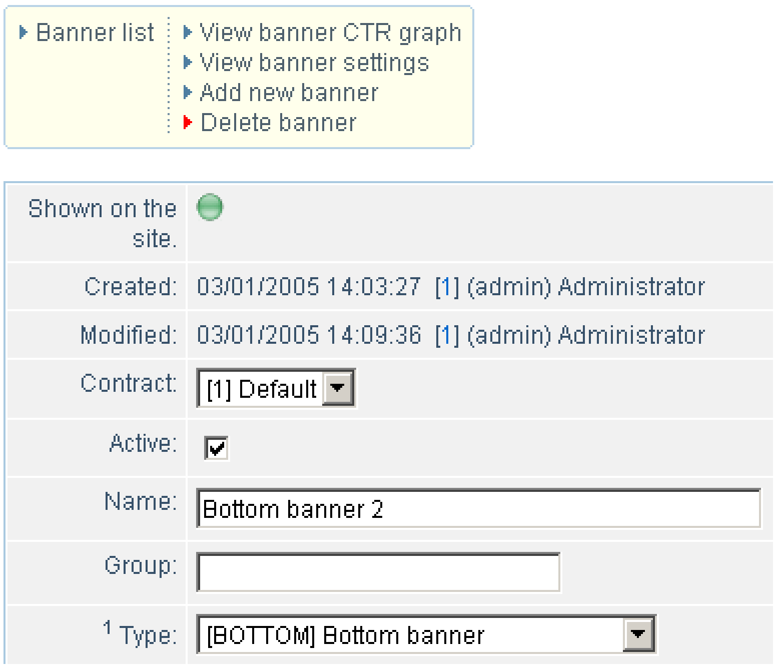
You can quickly access the desired banner (or a set of banners of the selected advertising area) by clicking the button (*Edit this banner*) or (*Filter the list of banners*) near the banner when in the editable areas display mode.



Contract: [1] Default | Add banner >>

ID / Ind.	Lang.	Act.	Weight	Name / Group / Type / Contract / Status
18 ● Diagram View Modify Delete	en	✓	100	Name: **Bottom banner 2** Type: [BOTTOM] Bottom banner Contract: [1] Default Status: Published

Fig. 4.8. Managing the advertisement types of a site



Shown on the site.	●
Created:	03/01/2005 14:03:27 [1] (admin) Administrator
Modified:	03/01/2005 14:09:36 [1] (admin) Administrator
Contract:	[1] Default
Active:	✓
Name:	Bottom banner 2
Group:	
1 Type:	[BOTTOM] Bottom banner

Fig. 4.9. The form used to add an advertisement

Implementation

Advertising areas are to be located in the spots that had been preliminarily defined during the template design process.

```
<?
//--- Example of placing an advetising area in the left part of the design.
//--- Any other type can be selected similarly:
//--- TOP, BOTTOM, COUNTER,… (first parameter of the function)
//--- Both predefined and user-defined types can be used.
//--- Other two optional parameters can be used to specify the HTML code
//--- that is to wrap the advertising area.

$APPLICATION->ShowBanner("LEFT", '<div align="center">', '<br></div><br>');
?>
```

Advertisement shows are controlled by the set of keyword that can be specified for either individual banners of the whole contract. There are two kinds of keywords used in the system: required and desired. See the help section for details on using keywords (functions **CAdvBanner::GetKeywords**, **CAdvBanner::SetRequiredKeywords**, **CAdvBanner::SetDesiredKeywords**).

Please note that if the page code does not provide any keywords to control the advertisement shows, the function **CAdvBanner::SetDesiredKeywords** uses the value of a page property **adv_desired_target_keywords** by default. If the value is still void, the value of property **keywords** is used instead as the function parameter.

The function **CAdvBanner::SetDesiredKeywords** is called automatically at the page construction time with the above described parameters. You should not call it again in the file *header.php* if you do not need to redefine the advertisement keywords.

Menu customization

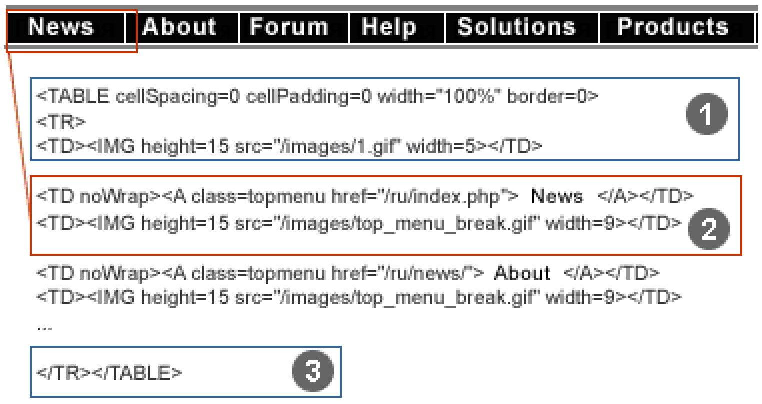
Design

Any menu of a site is built based on the two components:

- array of data provided in the administrative section (the product ships with the demo menu);
- menu design template.

Array of data defines the menu composition, text and links of all the menu items. A menu template is the PHP code that describes the menu design. Menu template processes the data array and generates the HTML code.

The first thing you need to do to customize the template is to choose the repeating parts of a menu (fig. 4.10). For example, the horizontal menu is comprised of table cells, while the vertical menu consists of rows. After that, move the repeating portion of a menu to your template. The whole menu is built in the header or footer using the PHP function responsible for building a menu.



```
<TABLE cellSpacing=0 cellPadding=0 width="100%" border=0>
<TR>
<TD><IMG height=15 src="/images/1.gif" width=5></TD>

<TD noWrap><A class=topmenu href="/ru/index.php"> News </A></TD>
<TD><IMG height=15 src="/images/top_menu_break.gif" width=9></TD>

<TD noWrap><A class=topmenu href="/ru/news/"> About </A></TD>
<TD><IMG height=15 src="/images/top_menu_break.gif" width=9></TD>
...

</TR></TABLE>
```

Fig. 4.10. Defining the code of a menu item.

The process of constructing a menu includes: choosing the HTML elements used to build a menu, creating the menu templates, calling the menu function in a common template (header and footer), filling the menu according to a site structure.

You may want to create additional styles in the CSS file. For example, a text-based menu can be described by the color of both menu item text and the active menu item text. An individual design may be required for section titles (e.g. “Products”); additionally, some graphics or text elements can be used for displaying the existence of the nested menu levels etc.

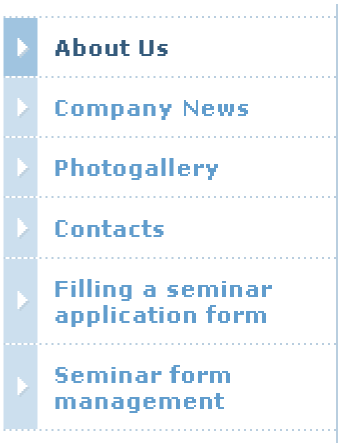


Fig. 4.11 Menu items.

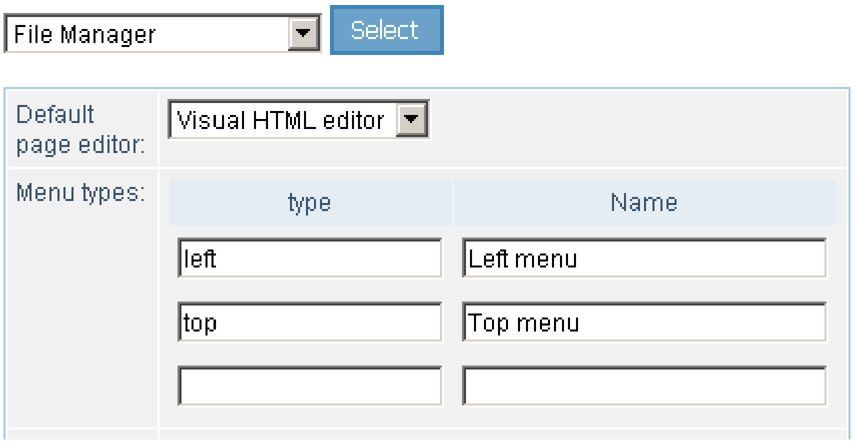
How it works

The menu of a section is customized via the system administrative interface. To edit a section menu, you can either find a corresponding menu file in the site structure using the Site Explorer, or click the button (*Edit menu*) near the corresponding menu while in the include areas display mode.

A section can be assigned menus of different types: upper, left, bottom etc. In the general case, a site contains a single upper menu that corresponds to the topmost level and is connected to all sections. A system may also contain a "left" (or any other) menu of the second level with items and commands pertaining to documents of a given section (folder).

Types of menus can be assigned in the Site Settings section of the administrative interface. Let a system have two types of menus:

- Left menu: type "left";
- Upper menu: type "top".



Default page editor: Visual HTML editor

Menu types:

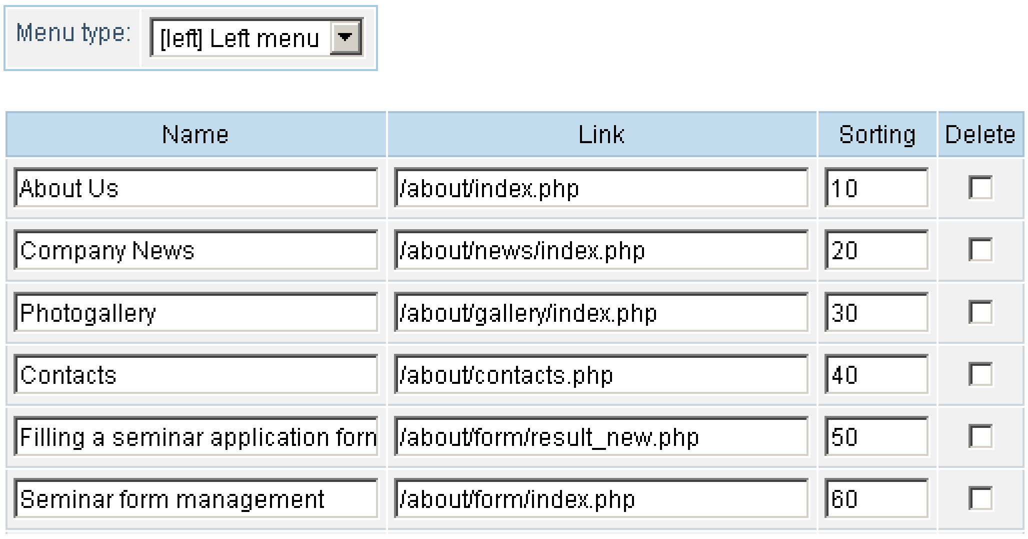
type	Name
left	Left menu
top	Top menu

Fig. 4.12. Setting types of menus.

Typename specified in these settings will be used as a prefix for the name of the menu template file. This prefix is also used to identify files containing the section menu items.

An individual file is used to store information on each type of menu. For example, left menu file is named .left.menu.php, while the upper menu file is named .top.menu.php.

Two menu editing modes are provided: simple and extended. A simple mode allows specifying menu item text, link URL and sort weight only.



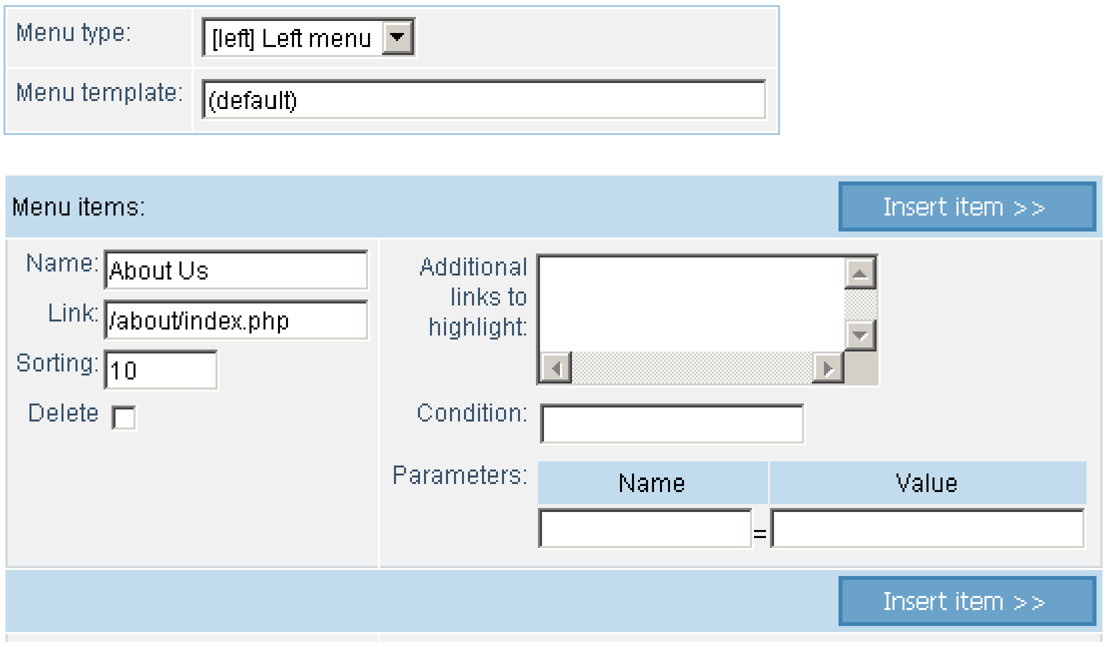
Menu type: [left] Left menu

Name	Link	Sorting	Delete
About Us	/about/index.php	10	☐
Company News	/about/news/index.php	20	☐
Photogallery	/about/gallery/index.php	30	☐
Contacts	/about/contacts.php	40	☐
Filling a seminar application forn	/about/form/result_new.php	50	☐
Seminar form management	/about/form/index.php	60	☐

Fig. 4.13 Simple editing mode.

An extended mode is used to alter the following parameters:

- menu item text;
- link URL;
- set of additional links that correspond to this menu item. For example, active menu item *News* can match two pages: *Newsfeed* and *News Details*;
- sorting defines the menu item position in a list. Lesser values move an item higher;
- display conditions. For example, you can restrict shows of this item to users with certain permissions only;
- additional parameters which can be processed by the menu template and displayed as desired. For example, if a menu item is the section title, it can be highlighted by setting the parameter SEPARATOR to a value of "Y". You can check this parameter in your template and highlight an item.



Menu type: [left] Left menu
Menu template: (default)

Menu items:

Name: About Us
Link: /about/index.php
Sorting: 10
Delete ☐
Additional links to highlight:
Condition:

Parameters:

Name	Value

Fig. 4.14 Extended editing mode.

All information provided via the administrative interface is stored in files .left.menu.php and .top.menu.php for the left and upper menu, respectively. These files contain arrays of values for each menu item.

Files are located in those sections (directories) where they are to be used for the menu display. If a given section does not contain a menu description file, the system will search a section of upper level for such files.

Menu is built in the following way. A common design template calls the function that generates the code responsible for the menu display. When a page is being loaded, this function checks for presence of a menu description file in the given section; calls the template of the corresponding type to build the menu; generates the HTML code of the menu.

Implementation

Template creation process commences with allocating the HTML areas of the menu in the site template. Then, constant upper and bottom parts are selected as well as the repeating parts.

Templates for upper and left menu are stored in folder "/bitrix/templates/<site identifiers>/" in files top.menu_template.php and left.menu_template.php respectively.

You can quickly edit the desired template of any type by clicking the button (*Edit menu template*) near the corresponding menu while in the include areas display mode.

All menu templates share the common structure:

- menu template header part;
- description of substitutions for different processing conditions;
- menu template body;
- menu template footer part.

Let us consider how a template can be constructed by the example of left menu:

```
<?
//--- The following variables can be used when creating a menu template:
//--- $TEXT – menu item text;
//--- $LINK – link URL of the menu item;
//--- $ITEM_TYPE – menu item type. Can have the following values:
//---   "D" – directory, "P" - page, "U" – parametrized page.
//--- $ITEM_INDEX – menu item index.
//--- $SELECTED – set to true if the menu item is selected.
//--- $PERMISSION – access permissions for the current page.
//--- Can have the following values: "D" – access denied, "R" – read,
//--- "W" – write, "X" – write and permission modification.
//--- $PARAMS["<parameter>"] – array of parameter values
//--- set in the extended editing mode.

//------------- Menu Header -------------------------------------
//--- Upper constant part of the table.

$sMenuProlog="<table width='100%' border='0' cellspacing='0' cellpadding='0'
background=''>\n";

//--- Description of substitutions for different processing conditions
//--- Template elements are altered according to the display conditions.
//--- If any menu item is set to display as the separator,
//--- a corresponding parameter  value is checked:
//---     $PARAMS["SEPARATOR"]=="Y".
//--- Parameter "SEPARATOR" can be assigned
//--- when editing the menu in the extended mode.
//--- Other parameters are optional and can be set as desired.
//--- Please note that a parameter named "SEPARATOR"
//--- is also checked when building a site map.
//--- Certain alterations are made to template elements
//--- according to conditions.

if ($PARAMS["SEPARATOR"]=="Y")
{
    $clrbg = " class='menuborder'";
    $clrtext = "leftmenub";
}
else
{
    $clrbg = "";
    $clrtext = "leftmenu";
}
if($SELECTED)
{
    $clrtext = "leftmenuact";
}
else
{
    $clrtext = "leftmenu";
}
if ($ITEM_TYPE=="D")
{
    $clrimg = "&nbsp;<img src='/images/arrows_r.gif' width='11' height='9'
border='0'>";
}
else
{
    $clrimg = "";
}

//------------- Menu template body ------------------------------------
//--- This section describes a single element of the common structure,
//--- i.e. a single menu item. All menu items are generated in a loop.
//--- Template elements that can vary depending on conditions,
//--- are represented by variables, e.g. $clrbg, $clrimg, $clrtext.
//--- The mandatory parameters (menu item text and link)
```

```
//--- are described by the variables $TEXT and $LINK, respectively.
//--- Vriable $PARAMS["ALT"] contains the value of the "ALT" attribute
//--- specified in the extended editing mode.
//--- User permissions are checked before generating a menu item.
if ($PERMISSION>"D")
{
$sMenuBody =
  "<tr>\n".
  "<td width='0%' ". $clrbg ." nowrap valign=\"top\" ><img
src='/images/1.gif' width='2' height='8'><img src='/images/1.gif' width='30'
height='15'></td>\n".
  "<td width='100%' ". $clrbg ."><a href='".$LINK."' class='".$clrtext."'
title='".$PARAMS["ALT"]."'>".$TEXT."". $clrimg ."</a></td>\n".
  "<td width='0%' ". $clrbg ."><img src='/images/1.gif' width='5'
height='1'></td>\n".
  "</tr>\n";
}
else
{
$sMenuBody =
  "<tr>\n".
  "<td width='0%' ". $clrbg ." nowrap valign=\"top\" ><img
src='/images/1.gif' width='2' height='8'><img src='/images/1.gif' width='30'
height='15'></td>\n".
  "<td width='100%' ". $clrbg .">".$TEXT."". $clrimg ."</td>\n".
  "<td width='0%' ". $clrbg ."><img src='/images/1.gif' width='5'
height='1'></td>\n".
  "</tr>\n";
}
//------------- Menu template footer ------------------------------------
//--- Table closing tag. Bottom constant part.

$sMenuEpilog="</table>";
?>
```

Resulting menus can be incorporated in the design template by calling the following functions:

```
<?
//--- Upper menu inclusion code
echo $APPLICATION->GetMenuHtml("top");
?>
…
<?
//--- Left menu inclusion code
echo $APPLICATION->GetMenuHtml("left");
?>
```

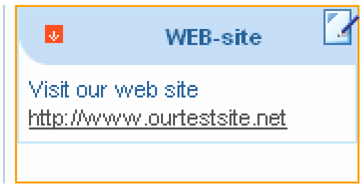
Inclusion of auxiliary editable areas

Design

Templates of documents and include areas are common HTML documents containing the predefined formatting elements: tables, images, styles etc. Created templates are copied to folders "page_templates", either globally for all templates (/bitrix/templates/.default/page_templates/) or for each template individually (/bitrix/templates/<template identifier>/page_templates/).

The system provides means to edit various types of include areas, for example:

- include area of the certain file – displayed only when viewing the current document;
- include area of the certain section – displayed for all documents of the given section;
- other types of include areas.


WEB-site

Visit our web site
http://www.ourtestsite.net

Fig. 4.15. Sample include area.

How it works

Include areas are stored in separate files with a certain suffix in the file name, e.g. "_inc". For example, a name of the section include file could be "sect_inc.php".

Number of editable areas for each file or section can be increased. In this case, a template should be slightly modified. The include areas can be created directly from the page while in the include areas display mode. In this mode, spots that are intended to display the auxiliary include areas will have special buttons which can be clicked to quickly edit the area.

Include areas allow using specially prepared templates. List of such templates is stored in the file .content.php. These templates will be available in the drop-down list of the “New document” button when editing the page in the WYSIWYG editor and in the administrative interface.

Please note that the template name can be passed as the parameter when attaching the area in the site template (in the example below, the template name is highlighted with blue).

Implementation

The code used to include the auxiliary editable areas:

```
<?
$APPLICATION->IncludeFile(substr($APPLICATION->GetCurPage(), 0,
strlen($APPLICATION->GetCurPage())-4)."_inc.php", Array(),
Array("MODE"=>"html", "NAME"=>GetMessage("PAGE_INC"),
"TEMPLATE"=>"page_inc.php"));
?>
<?
$APPLICATION->IncludeFile($APPLICATION->GetCurDir()."sect_inc.php", Array(),
Array("MODE"=>"html", "NAME"=>GetMessage("SECT_INC"),
"TEMPLATE"=>"sect_inc.php"));
?>
```

Using components

It is a good practice to arrange frequently used areas and code as the software components. Virtually any script can be shaped as a component.

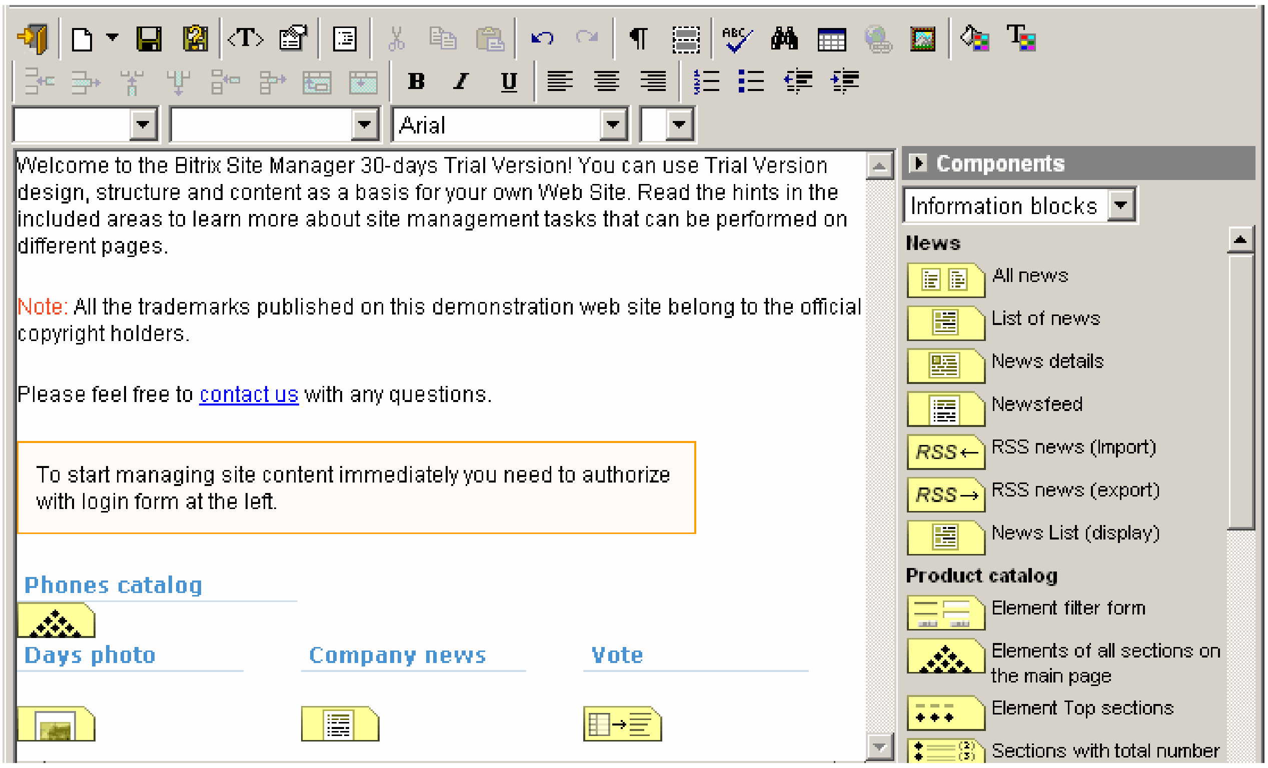
A distinctive feature of a component is the ability not only to be controlled on the source code level, but also through the use of parameters that define the component behaviour.

A user can quickly edit the component source code or just change its parameters. For example, the component *Newsfeed* can be assigned a number of news per page or the sort order of elements etc.

Each component is the software script that “exports” special variables by the use of which it can be controlled via the visual interface.

To describe the component parameters, special files are used. These files have a fixed name .description.php and are located in the same or parent directory of the component file.

You can quickly add components to pages using the visual editor (fig. 7.1). To insert a component into a page, simply select it in the list and drag and drop it to the desired position within the page.



Welcome to the Bitrix Site Manager 30-days Trial Version! You can use Trial Version design, structure and content as a basis for your own Web Site. Read the hints in the included areas to learn more about site management tasks that can be performed on different pages.

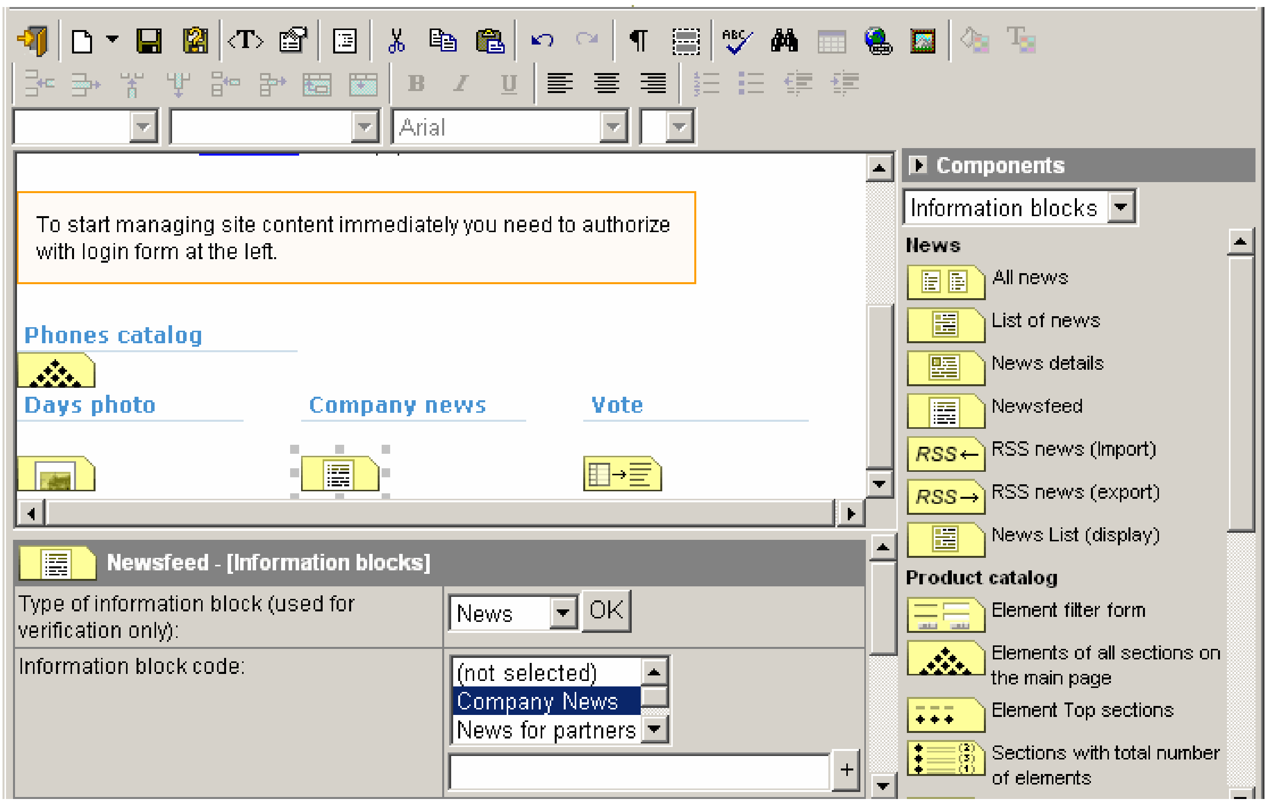
Note: All the trademarks published on this demonstration web site belong to the official copyright holders.

Please feel free to contact us with any questions.

To start managing site content immediately you need to authorize with login form at the left.

Fig. 7.1. Visual editor.

After you have added a component, you can customize its parameters. To do so, select an existing component in the page; the list of available component parameters will appear in the bottom of the screen (fig 7.2).



To start managing site content immediately you need to authorize with login form at the left.

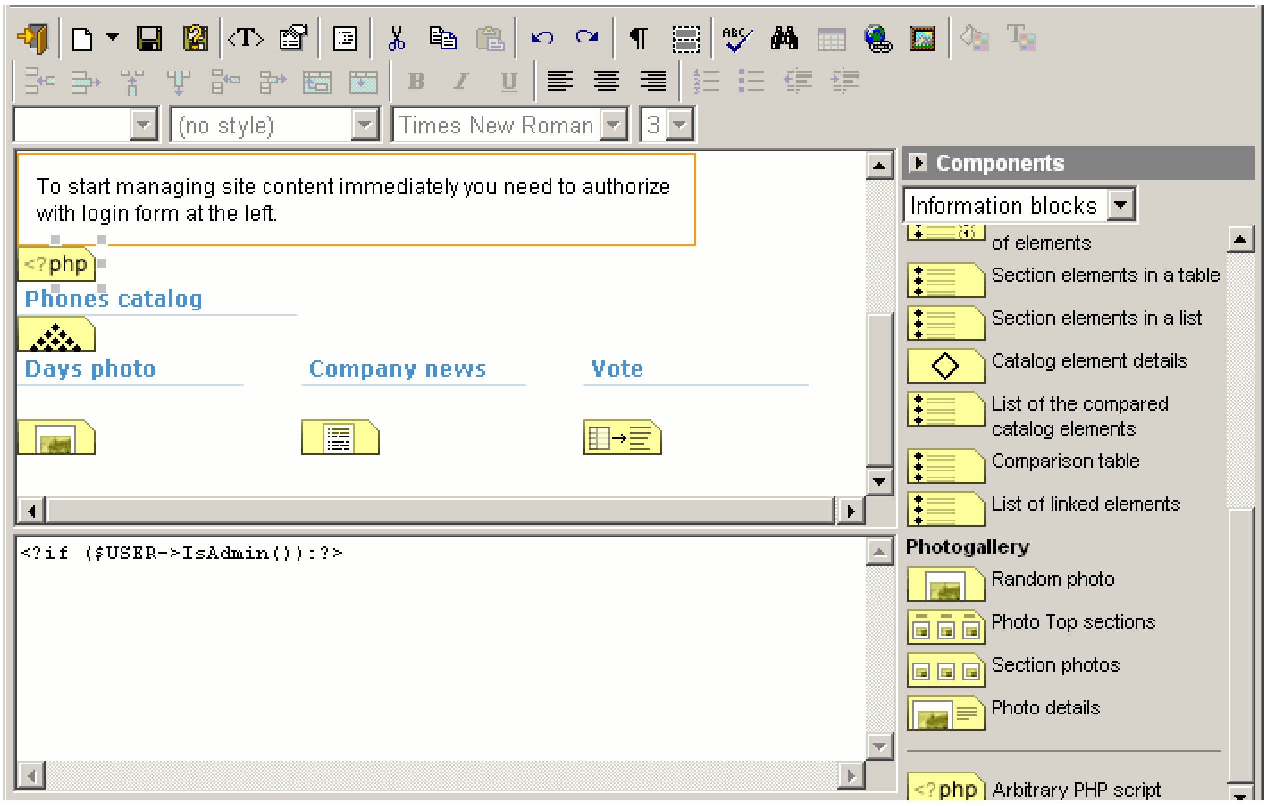
Newsfeed - [Information blocks]

Type of information block (used for verification only):	News
Information block code:	(not selected) / Company News / News for partners

Fig. 7.2. Parameters of component.

You can add an arbitrary PHP code to your page in the similar manner. In this case, you will modify it in the text editor in the bottom of the screen (fig. 7.3).

The *PHP Script* component allows to insert almost any PHP code in a page while leaving the page design undisturbed in the visual editing mode.



```
<?if ($USER->IsAdmin()):?>
```

Fig. 7.3 Adding PHP code in the visual mode.

The generated code includes the components using the function IncludeFile(). The function parameters include the path to the file with the component code and the component parameters assigned to it in the visual editing mode.

Files pertaining to a component can be stored in the folder of a module to which this component relates.

For example, components used with the *Information Blocks* module are located in the following directories:

/bitrix/modules/iblock/install/templates/iblock/news/

and

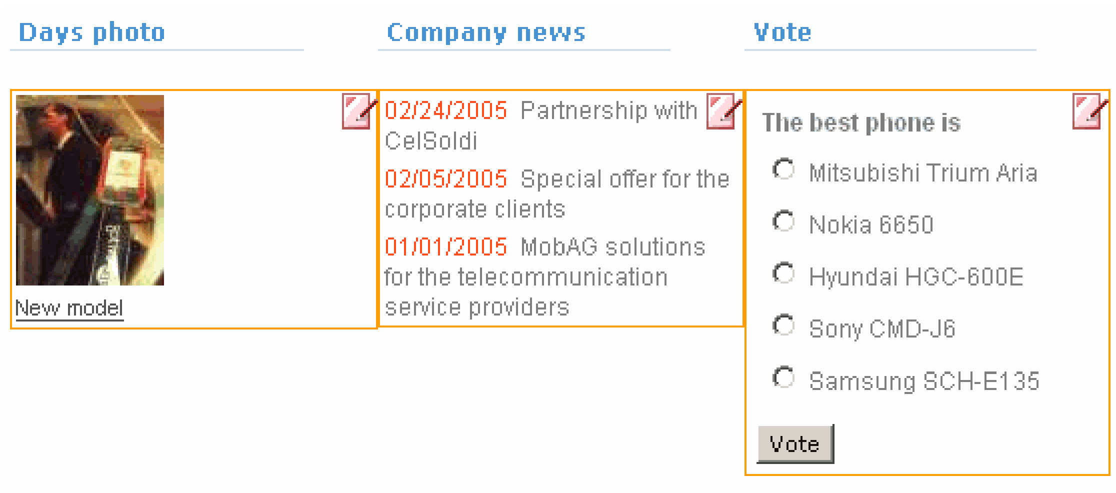
/bitrix/modules/iblock/install/templates/iblock/catalog/.

Alternatively, they can be installed in the default templates folder: /bitrix/templates/.default/iblock/news/ and /bitrix/templates/.default/iblock/catalog/ respectively.

Components included in the product installation package have been developed using language resource files containing text messages required by a component. Description files description.php are also included. For example, descriptions for all components of the *Information Blocks* module can be found in the file /bitrix/modules/iblock/install/templates/iblock/.description.php

If you need a special customization of the component (for example: specific design of a search or authorization form), you can copy the component to directory of the corresponding template.

You can also edit the component code directly in the default template folder. This will affect all templates that do not have a copy of the original component code.



Days photo

New model

Company news

02/24/2005 Partnership with CelSoldi

02/05/2005 Special offer for the corporate clients

01/01/2005 MobAG solutions for the telecommunication service providers

Vote

The best phone is

- Mitsubishi Trium Aria
- Nokia 6650
- Hyundai HGC-600E
- Sony CMD-J6
- Samsung SCH-E135

Vote

Fig. 7.4 Example of the default template components

Button switches to editing the component source code directly in the default template directory (/bitrix/templates/.default/).

Button copies the selected component to the current template folder and opens the editor with the component code.

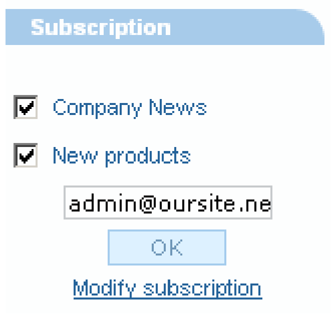
Chapters below contain the explanation of the basic set of components shipped with the product. In addition, a brief description of creating pages for main system modules is provided.

Subscription module

Subscription Form component

Design

A standard subscription form includes: the dynamically generated list of subscription themes; subscriber's e-mail address entry field; submit button and the link used to manage subscriptions. Upon filling a form, a site visitor will be redirected to the simple registration form and then added to the subscribers list. Later, a visitor can change the subscription parameters or discontinue the subscription. The simplified registration page is only available if the option *Allow users to self-register* is switched on in the site settings.



Subscription

- ✓ Company News
- ✓ New products

admin@oursite.ne

OK

Modify subscription

Fig. 8.1. Subscription form.

How it works

Subscription themes and the list of subscribers can be managed in the *Subscription* section of the administrative interface.

Subscription themes are created and made available to site visitors by the site administrator. Site users who are granted appropriate permissions create subscription messages and send them to subscribers.

Implementation

A template used to display the subscription form is stored in the file:

/bitrix/templates/<template name>/subscribe/subscr_form.php

Generally, if the template does not require additional customization, it can be included from

/bitrix/templates/.default/subscribe/subscr_form.php

The form can be included as follows:

```
<?
    $APPLICATION->IncludeFile("subscribe/subscr_form.php",
                              Array("PAGE" =>
                                    SITE_DIR."personal/subscr_edit.php"));
?>
```

Subscription Page component

The component allows creating a standard subscription page.

The component parameters include path to the subscription management page and an option indicating whether to display the number of subscribers for each subscription theme or not.

Subscription Management component

The component is used to create a page that a site visitor opens to alter the subscription parameters. The component parameters include options to allow anonymous subscriptions and whether to display the authorization link for anonymous subscribers or not.

Search module

Search form

Design

A standard search form is displayed to site visitors:

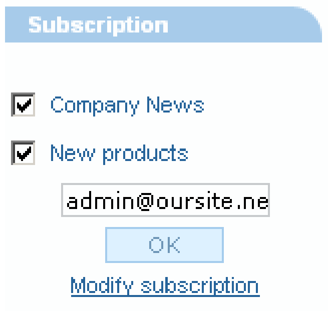


Fig. 8.2. Standard search form.

Upon submit, a visitor will be redirected to the search results page.

How it works

The search covers static information, information blocks (news, catalogues, vacancies etc.) and the forum.

Implementation

The form can be included in the main template as follows:

```
<?
    $APPLICATION->IncludeFile("search/search_form.php",
                              Array("SEARCH_PAGE" =>
                                    SITE_DIR."search/index.php"));
?>
```

The component parameter can be the path to the search results page.

Search Page component

The component parameters include:

- values of the drop-down list *Search Where*;
- number of search results on the page;
- path to the additional navigation chain template used for displaying search results;
- time to store the cached values of the drop-down list *Search Where*.

Main module

Authorization form

Design

A standard authorization form provides a neat interface allowing site visitors to enter their login information and access the secured site sections upon the successful authorization.

How it works

The authorization form allows to check the user's permissions to access certain sections and pages of a site. Access permissions can be assigned to the site pages, sections and information blocks.

Implementation

The form can be included in the main template as follows:

```
<?
    $APPLICATION->IncludeFile("main/auth/auth_form.php");
?>
```

Site switch panel

Displays a panel that a user can take advantage of to switch between sites. Can be included in the main template as follows:

```
<?
    $APPLICATION->IncludeFile("main/site_panel.php", "php");
?>
```

Site Map component

The component displays the site map in the public section. The component has no parameters.

User Profile component

The component can be used to display and edit the profile of a current user.

Statistics module

Statistics Table

Design

Site visitors may be given a chance to view the table displaying the site attendance statistics. The table shows information on the total site visitors and for the current date. The site administrator views the information as links that can be clicked to view the detailed statistics.

How it works

The statistical system collects and processes information on site visitors, advertising campaigns, search engine hits, referrer sites, sessions etc.

Implementation

The statistics table can be included in the main template as follows:

```
<?
$APPLICATION->IncludeFile("statistic/stat_table/default.php");
?>
```

Information Blocks module

Design

The Bitrix Site Manager features managing both static and dynamic information. The dynamic information includes documents renewed on the regular basis: news, press releases, photos, vacancies, product catalogue etc.

The system provides a special facility to manage the dynamic information – the Information Blocks. The blocks consist of a definite number of elements. For example, the news section *Company News* forms the information block, while each news is an element of the information block.

How it works

Information blocks are controlled in the administrative interface, *Information Blocks* section.

The system provides different types of information blocks: news, photo gallery, vacancies, and catalogue. Each information block can be assigned properties that will be inherited by all child elements of this information block. For example, products can be assigned such specific properties as weight, dimensions, colour etc.

All properties can have the preset default value, which applies to all elements of a block. The important settings of an information block are the path to a page with the list of elements of this block and the path to a page with the detailed view of an element.

Access to elements of a block is the subject of ID and the mnemonic name of the block.

Some information blocks may have two element nesting levels: information block – groups – elements.

Implementation

The new version of the product uses special components to display the information block contents.

Individual components are used to display lists, news details, newsfeed, and imported RSS data. Special component is used to export the information block data to RSS.

The product is equipped with a wide range of components for displaying the product catalogue information.

A group of components enables creating a photo gallery.

How to implement the news display

First step in implementing pages for the news display is to create the appropriate information block in the administrative interface and populate it with elements; that is, the news.

When creating a new information block, you have to specify the path to a page with the list of elements of this block (news) and the path to a page with the detailed view of an element (a single news).

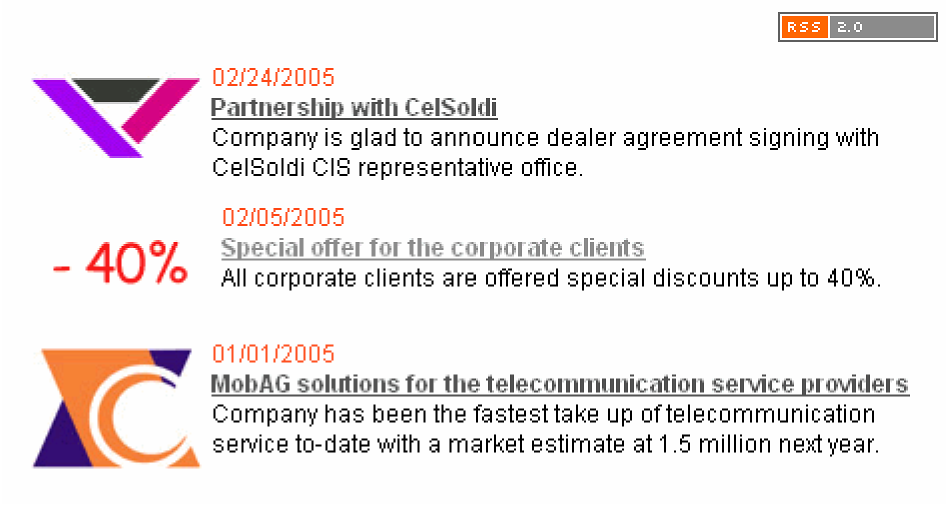
The above-mentioned pages must be present in the site structure. For example, /about/news/index.php for the list of news and /about/news/detail.php for the news details.

You can create pages in any way you like: using either the Site Explorer module or WYSIWYG editor.

After you have created a new page, place the required components in it. If you are working with the news list page, drag and drop the *News List* component from the palette on the page. Drag and drop the *News Details* component on the page intended to display the news details.

If you wish to arrange the news export to RSS, place a descriptive image in the page (RSS 2.0) and apply a link to it. The link should point to an existing page (e.g. /about/news/rss.php) with the RSS Export component placed in it.

Please note that you should make sure that the RSS export is allowed for the given information block.



02/24/2005
Partnership with CelSoldi
Company is glad to announce dealer agreement signing with CelSoldi CIS representative office.

02/05/2005
Special offer for the corporate clients
All corporate clients are offered special discounts up to 40%.

01/01/2005
MobAG solutions for the telecommunication service providers
Company has been the fastest take up of telecommunication service to-date with a market estimate at 1.5 million next year.

Fig. 10.1 News list with the RSS link

Web Forms module

The common algorithm of arranging a web form in the public section of a site is as follows.

- create a web form in the administrative section of a site: *Web Forms - Forms*;
- in the public section, add the component *Web Form Entry (default template)* to the required page;
- provide the form name as well as the titles of pages used to display the list of results and editing of the result of a web form;
- create the above stated pages using the product facilities and place the appropriate components in them;
- customize the component preferences according to your needs.

Polls and Surveys module

Polls can be allocated in the similar manner. The component should be assigned pages that will display the poll results.

The system contains a component allowing to include a poll form on any page of a site with the subsequent poll result display. If a visitor had not voted before, a poll form is displayed and the poll results upon voting. If a visitor had once voted, the poll results are displayed.

Currently, components exist for all major modules of the system. The range of components is continually extended by both Bitrix team and its partners. The help section is now being enriched with the detailed information on the existing components and the component development.

Using message files for localization

If you develop a template that should be common to more that one language, you should pay attention to text elements of the design: titles, ALT attributes, controls text etc.

The site display template should be set up in such a way that the HTML code is common to all languages while all text elements are replaced with text symbols stored separately in the language folders, for example /bitrix/templates/<template identifier>/lang/en/ for English.

For example: language messages for the copyright editable area (the file /bitrix/templates/<template identifier>/copy.php) are stored in the file /bitrix/templates/<template identifier>/lang/en/copy.php for the English interface.

Sample English message file:

```
<?
$MESS['SEARCH_TITLE'] = "Search";
?>
```

The file /bitrix/templates/<template identifier>/lang/en/copy.php includes language-dependent messages in the very beginning of the file:

```
<?
IncludeTemplateLangFile(__FILE__);
?>
```

In a document, language-dependent text can be displayed by the following code:

```
<font class="copy"><?echo GetMessage("COPY");?></font>
```

How to set up a partner technical support

Partners of the Bitrix corporation can alter the administrative part of the product if they want to provide technical support to their clients themselves. To do so, follow these steps.

Change the site address and the techsupport link in the right bottom corner of the administrative screen.

Create a file /bitrix/php_interface/this_site_support.php containing the required text. In this file exists, it will be used instead of the standard text.

Sample file /bitrix/php_interface/this_site_support.php:

```
<a href="http://www.flowersweb.info"
class="pagebottomtext">www.flowersweb.info</a>&nbsp;&nbsp;|&nbsp;&nbsp;<a
class="pagebottomtext"
href="http://www.flowersweb.info/support/">Techsupport</a>
```

Embed a logo or text in the bottom corner of the administrative screen.

Create a file /bitrix/php_interface/this_site_logo.php containing the required design. In this file exists, it will be used instead of the standard text. The content of the file is arbitrary.

Change the e-mail address for the support requests from the Site Checker (/bitrix/admin/site_checker.php)

Define the constants:

```
define("THIS_SITE_SUPPORT_EMAIL", "Support e-mail");

define("THIS_SITE_SUPPORT_CHARSET", "Encoding");
```

If you do not override encoding, the default value of iso-8859-1 is used. These constants can be defined in the following files:

/bitrix/php_interface/dbconn.php (connected in the beginning)

or

/bitrix/php_interface/after_connect.php (included after the successful connection to the database)

or

/bitrix/php_interface/<site code>/init.php (included after the site has been defined)

Format of the component template description files

The files are located in the same folder as the template. For example:

(/bitrix/modules/iblock/install/templates/iblock/news/.description.php),

Alternatively, they may be stored in the root folder of the module templates:

(/bitrix/modules/iblock/install/templates/iblock/.description.php).

Anyway, a file .description.php must exist in the module templates root folder. This file must contain the name of the section (module) in variable $sSectionName.

Templates are described in the variable $arTemplateDescription of the following format:

```
"array index" –  path to the template
                 relative to folder with file .description.php.
"array entry" –  array of the following format:
"NAME" –         name of the template (required),
"DESCRIPTION" –  template description,
"ICON" –         path to the template icon,
"PARAMS" –       array of the template parameters:
    "NAME" –     prameter name.
    "TYPE" -     type (STRING or LIST)
    "MULTIPLE" – plurality. If set to "Y", parameter will be passed
                 to the template as an array.
    "VALUES" –   values for the LIST type. Array "param"=>"value".
    "DEFAULT" –  default value. Array if MULTIPLE is “Y”.
    "SIZE" –     size of the field “select” for the LIST type.
    "CNT" –      number of fields for plural entries.
    "COLS" –     number of columns in the entry fields.
    "ROWS" -     number of rows in the entry fields.
    "ADDITIONAL_VALUES"=>"N" – used to disable the manual entry field
                               for the LIST type.
```

The input fields interpret text as a string.

If you want to include a PHP expression, enclose all the value in "={...}". For example:

```
={$_REQUEST["BID"]}
```

Or:

```
={(isset($_REQUEST["BID"])?$_REQUEST["BID"]:"15")}
```

Example of the file .description.php at the module templates root (please note that the text is usually replaced with GetMessage() calls):

```
<?
IncludeTemplateLangFile(__FILE__);
$sSectionName = "Information Blocks";
$arTemplateDescription =
  Array
  (
    "iblock/news/index.php" =>
       Array(
         "NAME"           => "List of News",
         "DESCRIPTION"    => "Show the list of news
                              split by information blocks",
         "ICON"           => "/bitrix/images/iblock/components.gif",
         "PARAMS"         =>
                             Array(
              "IBLOCK_TYPE" => Array("NAME"=>"Block type code",
                                     "TYPE"=>"LIST",
                                     "MULTIPLE"=>"N",
                                     "VALUES"=>Array("news"=>"News"),
                                     "DEFAULT"=>""),
              "IBLOCK_ID"   => Array("NAME"=>"Block code",
                                     "TYPE"=>"LIST",
                                     "MULTIPLE"=>"Y",
                                     "VALUES"=>Array("32"=>"Company news",
                                                     "12"=>"Product news"),
                                     "DEFAULT"=>"12"),
       "COUNT_FOR_COLUMN"   => Array("NAME"=>"Rows per column",
                                     "TYPE"=>"STRING",
                                     "DEFAULT"=>"10"),
           "COLUMN_COUNT"   => Array("NAME"=>"Columns",
                                     "TYPE"=>"STRING",
                                     "DEFAULT"=>"1")
                             )
       )
  );
?>
```

Appendix A. Recommendations on preparing the HTML template

Additional requirements imposed by the Bitrix Site Manager should be taken into consideration when preparing the graphic design of a site.

- Imagine a demarcation line between the prologue (**header.php**) and the epilogue (**footer.php**) and follow this guideline during the design process.
- Mark out basic design elements for further insertion in the CSS file: fonts, colors, backgrounds etc.
- Single out the repeating elements of a menu to facilitate the menu template creation and management.
- To facilitate the maintenance of different language versions, avoid graphic elements if possible; use text instead.
- When slicing the ready design and preparing the HTML template, you should provide a place for the site controls, menus, advertising areas, auxiliary forms.
- Templates should respect the table formatting. Layers are also allowed.
- When slicing, please keep in mind that the whole-coloured areas can be displayed by table cells of the solid colour.

Appendix B. Customizing auxiliary elements design

Customizing the error messages

An error message is also a point to apply the design efforts to make it accurate and unison with the site design.

Database connection errors can be customized by editing the file:

/bitrix/php_interface/dbconn_error.php

Database query errors can be customized by editing the file:

/bitrix/php_interface/dbquery_error.php

Customizing the "site temporarily closed" page

If you want to customize the design the page displayed when temporarily closing the public section, copy the file:

/bitrix/modules/main/include/site_closed.php

to:

/bitrix/php_interface/<language>/

or to:

/bitrix/php_interface/include/

Customizing the page-wise navigation

The page-wise display allow using the following parameters to alter the desing:

```
NavPrint($title, $show_allways=false, $StyleText="text", $template_path)
```

$template_path – path to the display template,

$StyleText – font style.

The default page-wise navigation template is generated as shown below. You can make it a separate file, modify it and pass the path to this file to the function:

```
<?
echo('<font class="'.$StyleText.'">'.$title.' ');
echo(($this->NavPageNumber-1)*$this->NavPageSize+1);
echo(' - ');
if($this->NavPageNumber != $this->NavPageCount)
    echo($this->NavPageNumber * $this->NavPageSize);
else
    echo($this->NavRecordCount);
echo(' '.GetMessage("nav_of").' ');
echo($this->NavRecordCount);
echo("\n<br>\n</font>");

echo('<font class="'.$StyleText.'">');

if($this->NavPageNumber > 1)
    echo('<a href="'.$sUrlPath.'?PAGEN_'.$this-
>NavNum.'=1'.$strNavQueryString.'#na
v_start'.$add_anchor.'">'.$sBegin.'</a>&nbsp;|&nbsp;<a
href="'.$sUrlPath.'?PAGEN_'.$this->NavNum.'='.($this->NavPageNumber-1).
$strNavQueryString.'#nav_start'.$add_anchor.'">'.$sPrev.'</a>');
else
    echo($sBegin.'&nbsp;|&nbsp;'.$sPrev);

echo('&nbsp;|&nbsp;');

$NavRecordGroup = $nStartPage;
while($NavRecordGroup <= $nEndPage)
{
    if($NavRecordGroup == $this->NavPageNumber)
        echo(''.$NavRecordGroup.'&nbsp');
    else
        echo('<a href="'.$sUrlPath.'?PAGEN_'.$this-
>NavNum.'='.$NavRecordGroup.$strNavQ
ueryString.'#nav_start'.$add_anchor.'">'.$NavRecordGroup.'</a>&nbsp;') ;
    $NavRecordGroup++;
}

echo('|&nbsp;');
if($this->NavPageNumber < $this->NavPageCount)
    echo ('<a href="'.$sUrlPath.'?PAGEN_'.$this->NavNum.'='.($this-
>NavPageNumber+1).
$strNavQueryString.'#nav_start'.$add_anchor.'">'.$sNext.'</a>&nbsp;|&n bsp;<a
href="'.$sUrlPath.'?PAGEN_'.$this->NavNum.'='.$this->NavPageCount.$str
NavQueryString.'#nav_start'.$add_anchor.'">'.$sEnd.'</a>&nbsp;');
else
    echo ($sNext.'&nbsp;|&nbsp;'.$sEnd.'&nbsp;');

//"All" section is being show with the code below

if($this->bShowAll)
    echo ($this->NavShowAll? '|&nbsp;<a href="'.$sUrlPath.'?SHOWALL_'.$this-
>NavNum.'=0'.$strNavQueryString.'#
nav_start'.$add_anchor.'">'.$sPaged.'</a>&nbsp;' : '|&nbsp;<a
href="'.$sUrlPath.'?SHOWALL_'.$this->NavNum.'=1'.$strNavQueryString.'#
nav_start'.$add_anchor.'">'.$sAll.'</a>&nbsp;');

echo('</font>');
?>
```