JShop Server

simple. effective. e-commerce

Developer Manual v2.2.0

Documentation Version: 2.2.0
(Revision Date: 26/03/2009)

Table of Contents

1 Introduction

1.1 Intended Audience

This document covers extending the PHP code for JShop Server by the use of hooks and functionality provided by the system. It is intended to be read by developers looking to extend some of the functionality of the system and requires good PHP knowledge and general programming experience. It is important to note that the extension facilities provided by JShop Server do not guarantee that your code is secure and we cannot provide support for your own custom additions to the system.

Where ever possible we recommend using the methods included in this document, in lieu of the plug-in architecture which we are currently developing, as the changes made in line with this document will not be lost when upgrades are applied to JShop Server. However, we cannot guarantee that changes in newer versions of JShop Server will not affect any code you have written and it is your responsibility to ensure compatibility with later versions as they are released.

1.2 Common Questions

Can I Edit The Core JShop Server PHP?

All the PHP code that JShop Server uses is available for viewing and, theoretically, you can edit this code should you wish to. However, editing the code is not supported by us and any problems that occur because of this cannot be dealt with by our support team. In addition to this editing the code may cause problems with future updates for JShop Server and we cannot give support for your store if it does not work correctly after an update patch has been applied if you have edited the code.

However, we are very keen to see JShop Server develop much further in the future and if you have some ideas on how we should extend the ability to bolt in new functionality please do let us know. We are currently developing a new plug-in architecture for JShop Server that should be available in 2009.

Can I Re-Use Parts Of JShop Server In Other Systems?

The short answer to this is no. All the JShop Server code is copyright Whorl Ltd. and may not be used outside of the JShop Server system, doing so is a breach of your license agreement and a breach of copyright. However, we can arrange for licenses to be granted for parts of the system's use outside of JShop Server and you should contact us for more details. There will be a charge for this and it will depend on the level and scope of the JShop Server code that is required. Agreement to issue a license for such use is at the sole discretion of Whorl Ltd.

When you purchase JShop Server you are purchasing a license to use the JShop Server system – you are not purchasing any ownership of the code itself and this remains, at all times, with Whorl Ltd.

Redistributing Your Own Changes

You are more than welcome to distribute your own changes for JShop Server, either for free or commercially. However, no core JShop Server code should be included in any code you release.

NOTE: You can redistribute code that includes calls to the default facilities in JShop Server as long as the code for those includes is not distributed. This means that you can include the use of the authentication system for any new admin pages you create in your distributed code.

2 Front-End Hooks

2.1 Extending cartOuputData.php

You can extend the processing loop in cartOutputData.php to allow you to create your own variables and attributes for use in the templates. To do this, you need to create a file in the routines directory called cartOutputExtra.php.

A basic example is below:

```
<?php
	switch ($requiredVars[$z]) {
		case "shops":
			$shops = array("test" => "hello","test2" => "world");
			$tpl->addVariable("shops",$shops);
			break;
	}
?>
```

The above example would pickup on a reference to a template variable called shop and, if required for the current template process your own code and create your own template variable to satisfy it. In the example above the template could reference `{shops.test}` and `{shops.test2}` to output the information "hello" and "world".

The important line is the line that injects the array into the template system:

```
$tpl->addVariable("shops",$shops);
```

This adds the information into the variable collection for the template.

NOTE: We strongly suggest using variable names that are particular to you, e.g. use myvarShops. This will ensure that no variables we may add in the future will have the same names and possibly cause issues.

2.2 Extending emailOutput.php

In a similar way to that you can use to extend the processing of cartOutputData.php, the same can also be done with emailOutput.php. Here you need to create a file called emailOutputExtra.php, again in the routines directory.

The format of this file is exactly the same:

```
<?php
	switch ($requiredVars[$z]) {
		case "shops":
			$shops = array("test" => "hello","test2" => "world");
			$tpl->addVariable("shops",$shops);
			break;
	}
?>
```

This example would add a template variable called shops to the template processing for emails so, again, you could reference `{shops.test}` and `{shops.test2}` in an email template.

NOTE: We strongly suggest using variable names that are particular to you, e.g. use myvarShops. This will ensure that no variables we may add in the future will have the same names and possibly cause issues.

2.3 PHP In Templates

You can also include PHP directly in the templates for JShop Server should you require such functionality. When doing this we strongly recommend that you don't include any database connection details or passwords in your templates for security reasons. In addition you should at least follow the basic template directory security guidelines found in the Installation documentation.

For more information on this please see the tSys section of the Templating documentation for the version of JShop Server you are using.

3 Administration System Hooks

3.1 Adding New Menu Options

In the admin/resources directory is the main section list file called sections.php – this is used to create the top menu buttons (in addition to sub-options and search panels). You can create a file called extrasections.php in the admin/resources directory that will also be included when menus are generated. A basic one, to add a menu called "New Menu" is below:

```
<?php
	$sectionsArray["myMenu"] = array("New Menu","myMainpage.php",10000);
?>
```

The format of these lines are as follows:

```
$sectionsArray[internalName] = array(visibleName,defaultPage,permissionID);
```

- **internalName** – This is an internal unique name that's also used to reference submenu options
- **visibleName** – This is the text that will appear on the menu button itself
- **defaultPage** – This is the default page that will be shown if somebody clicks the menu button
- **permissionID** – This is a unique ID used for this section for user group permissions.

You should choose permissionIDs much greater than the range we use to avoid any future conflicts. We would suggest starting at at least 10000 to be on the safe side. In addition, use an internal name that we'd never use, such as something prefixed by 'my' as in the example above.

3.2 Adding Menu Sub-Options

You can extend extrasections.php to also include new menu sub-options. For instance, taking the myMenu example in the previous section, let's add a few pages:

```
<?php
	$menuOptions["myMenu"] = array(
		array("NEWMENU","General","",0),
		array("New Option 1","newoption1.php","&myvar=hello",10001),
		array("New Option 2","newoption2.php","",10002),
		);
?>
```

The format of the individual lines is as follows:

```
array(visibleName,optionPage,getFields,permissionID),
```

- **visibleName** – This is the text that will appear as a sub-option.
- **optionPage** – This is the page that will be shown when the sub-option is selected.
- **getFields** – This is where you can add in any extra get fields you want to pass to the page.
- **permissionID** - This is a unique ID used for this sub-option for user group permissions.

You'll notice that the first option in our example is different to the others:

```
array("NEWMENU","General","",0),
```

This creates a menu separator on both the top menu button drop-downs and on the left menu once you're in a section in the administration system. The above example would create a separator with the title "General" and you can add in multiple such lines in amongst your sub-options to split them up into more logical groups.

The getFields option allows you to pass variables. We use this within the system to configure the same php page for adding or editing, for example. It's important to include a & at the start of the options for it to be added to the get variables correctly for the page call. You don't have to worry about login information as this is handled automatically by JShop Server (please see Creating Pages With Authentication).

As with the main menu options, the permissionID allows you to set a unique ID that will then be used in the user group permissions to allow you to set access on or off for individual options. Anything with a permissionID of 0 will bypass permissions and allow access to anybody that's logged in.

3.3 Adding Search Panels

You can also add extra search panels via the extrasections.php file, like this:

```
$searchArray[]  =  array("Orders","searches/orders.php",1508);
```

The format of these lines is as follows:

```
$searchArray[] = array(title,panelPage,permissionID);
```

- **title** – This is the text that will appear on the tab on the search panel.
- **panelPage** – This is the location of the PHP page to use to create the panel.
- **permissionID** - This is a unique ID used for this search panel for user group permissions.

In addition to creating the entry in extrasections.php you'll also need to create the search panel itself. By default these are contained within the admin/searches directory. An important inclusion to put at the top of your search panel code is the following:

```
<?php
      if  (!is_object($dbA))  {
            exit;
      }
?>
```

This stops the PHP being executed if it hasn't been run from within JShop Server – trying to execute it directly will simply produce a blank page.

All the search panels that you'll find in the admin/searches/ directory include examples of how to link to pages themselves to execute the searches (or any other processing you wish to do). The basic format is as follows:

```
<script  language="JavaScript">
//<![CDATA[
      function  goSearchCustomers(goreturn)  {
            self.jssMain.location='section.php?xAdminSection=customers&<?php  print  userSess
            if  (goreturn  ==  1)  {
                  return  false;
            }
      }
//]]>
</script>
```

This may look rather complex on first look but it's actually pretty straight forward. First of all you'll see the `xAdminSection` GET variable. This is the internal name of the main menu option that you want displayed (specifically this will show that menu's sub-options on the left hand menu.) Next you'll see the following:

```
&<?php  print  userSessionGET();  ?>
```

This transfers the current user's login information for authorisation on the loaded page. Without it, the user will automatically be logged out.

Then you'll see the `xShow` option which is set with a URL value – this is the actual page you would like to show and here you can include any other GET variables from any form you've created on your panel. It's important that the value given to `xShow` is escaped using `escape()` as this ensures that it's transferred correctly to the main section.php page, that organises what to display.

In addition the `xShow` value should also include the user's login information using the `userSessionGET()` call (as above), otherwise the user will be logged out.

3.4 Putting It All Together

The example below combines all the previous examples into a single extrasections.php file:

```
<?php
	$sectionsArray["myMenu"] = array("New Menu","myMainpage.php",10000);
	$menuOptions["myMenu"] = array(
		array("NEWMENU","General","",0),
		array("New Option 1","newoption1.php","&myvar=hello",10001),
		array("New Option 2","newoption2.php","",10002),
	);

	$searchArray[] = array("My Search","searches/mysearch.php",10003);
?>
```

You can add any number of new menu options, sub-options and search panels by using these methods.

3.5 Creating Pages With Authentication

If you're creating your own PHP pages for the administration system you'll also want to protect them using the authentication system that JShop Server uses for it's own pages and possibly use the default includes for common actions, such as database access etc. Here's a good default section to put at the top of your own pages:

```
<?php
	include("resources/includeBase.php");
	include("routines/checkaccess.php");
	checkUserPermission(10000);
	checkUserPermission(10001);
	dbConnect($dbA);
?>
```

The first line includes the basic files for JShop Server, including the config file, database access class, loading up store options and a host of other processing.

The second line includes the basic authentication that ensures that the user is logged in – this will be executed automatically and the user would be taken straight to the login screen if the credentials provided are not valid.

Underneath this you'll see 2 `checkUserPermission()` lines. These check that the user actually has the required permissions to use this page. We have two here for completeness – the first checks the permissionID for the main menu, and the second checks the permissionID for the particular sub-option that's called this page. You would obviously use your own permissionIDs here and you do not have to include both lines – if you don't want to check the

main menu button permission, for instance.

Finally there's an example of opening a connection to the database. This creates the connection and creates the object `$dbA` that can be used to access the database.

That's the basics of creating your own page and leveraging the structure already setup for JShop Server. Investigating the source code for other pages will outline other default classes and code that's used that will be available to you. However, that's outside the scope of this document.

Index