PLESK7

How To: Create Applications for the Application Vault

The **Application Vault** functionality allows for installing and configuring various web applications, such as counters, guest books, forums, photo galleries, etc. The Administrator of Plesk can add web applications in the form of *application packages* to the server enabling users deploy and configure the available *applications* on their domains (as well as subdomains).

In order for an application to be suitable for installation in Plesk it must meet certain requirements. For example, it must have a specific directory structure, it must follow a prescribed order of passing parameters between specific forms, etc. This document describes the different aspects of creating such an application.

1. Directory structure

Selected application should contain the following subdirectories:

- `scripts/` - directory where the installation, configuration and uninstall scripts are stored;
- `forms/` - directory where the main configuration and installation forms for configuring applications within virtual hosts are stored;
- `apps/` - tarballs which will be extracted into virtual host;
- `uninstall/` - subdirectory for the script, which deletes the application from repository (for non-rpm packages);
- `docs/` - online documentation directory for the end-users;
- `info/` - directory with application description xml-file. `info.xml` contains the system requirements and disk space usage. The main variables are also described there.

The sample of the file structure is given below:

```
/supercounter-1.2-5.6
     /scripts
          preinstall
          postinstall
          preuninstall
          postuninstall
          preupgrade
          postupgrade
     /forms
          installer-form-1.php
          installer-handler-1.php
```

SWsoft

```
        installer-form-2.php
        installer-handler-2.php
        installer-form-3.php
        installer-handler-3.php
    /apps
        cgi-bin-files.tar
        httpdocs-files.tar
    /uninstall
        uninstall
    /docs
        index.de.html
        index.en.html
    /info
        info.xml
```

2. Application directories content

2.1 The scripts/ directory

This directory contains following scripts:

- `postinstall` - is executed after tarballs' extraction. It is used for applying the necessary changes to the configuration to make the application fully functional after installing into the virtual host directory. The script can generate not only the configuration files, but also script files. It should apply necessary changes to dir and file attributes, rename directories if necessary. `.htaccess` files should also be created with this script. If the script returns non-zero value, the warning will be displayed, but the application will be installed. In case of the problems the script should output the warning to `stdout` and exit with 1;
- `reconfigure` - is executed after the new parameters were given by the user. It adds required changes to the script configuration.

Parameters are passed to `stdin` as a set of strings in the following format:

`< parameter name >=< parameter value <`

We strongly recommend that the `postinstall` script backs up the original config if it exists. The functions described below are used in our `postinstall` scripts.

```
#!/bin/sh

# gallery postinstall script
# required parameters: app_path
# *Put nesessary variables here
```

```
#
# there are also some standard parameters, that must be specified:
# vhost_path - full path to vhost root directory
# domain_name - name of domain
# install_prefix - path of application inside vhost directory
# ssl_target_directory - true, if application is in httpsdocs

read_parameters()

var=`cat | awk '
        eqpos=index($0, "=");
        if (eqpos>1)
                var=substr($0, 1, eqpos-1);
                val=substr($0, eqpos+1);

                tmp="[cc]";
                tmp2="cccc";
                gsub(tmp,tmp2,val);


                tmp2="ccc";
                gsub("",tmp2,val);
                print var "=" val "";
        ;
'`

eval $var




check_standard_parameters()

        if [ "X$vhost_path" = "X" ]; then
                echo "postinstall: no vhost_path parameter specified for
application"
                exit 1
        fi
        if [ "X$domain_name" = "X" ]; then
                echo "postinstall: no domain_name parameter specified for
application"
                exit 1
        fi
        if [ "X$install_prefix" = "X" ]; then
                echo "postinstall: no install_prefix parameter specified for
application"
                exit 1
        fi
        if [ "X$ssl_target_directory" = "X" ]; then
                echo "postinstall: no ssl_target_directory parameter specified
for application"
                exit 1
        fi
;

parse_standard_parameters()

        if [ "X$ssl_target_directory" = "Xtrue" ]; then
                proto="https"
                documents_directory="httpsdocs"
        else
                # don't forget to fix it
                proto="http"
```

```
                documents_directory="httpdocs"
        fi
;

create_dirs()

        if [ ! -d "$app_path/tmp" ] ; then
                mkdir "$app_path/tmp"
        fi


backup_original_config()

        if [ -f "$app_config" ] ; then
                if [ ! -f "$app_config.orig" ] ; then
                        cp $app_config $gallery_config.orig
                fi
        fi
```

2.2 The forms/ directory

This directory contains forms in PHP or HTML format which are being used for receiving the script configuration parameters.

- `installer-handler-<step number>.php` - input form for the data processing during the application installation;
- `installer-form-<step number>.php` - input form for application configuration values;
- `reconfigure-handler-<step number>.php` - reconfiguration handler for the data; processing during the upgrade;
- `reconfigure-form-<step number>.php` - reconfiguration form.

The absence of both installer files means that only http/https part of the virtual host and subdirectory of the application are being requested when the application is selected for the installation.

The absence of the first step reconfiguration files means that there is no possibility to upgrade the package.

NOTE

Application upgrade is not supported in current Plesk version.

2.3 The apps/ directory

- `httpdocs-files.tar`
- `cgi-bin-files.tar`

Both files are optional. If there is none of them, then the application must be installed by `postinstall` script.

2.4 The uninstall/ directory.

For the non-RPM packages only! It contains the uninstall script which should be equivalent to `preun` scripts in RPM packages.

2.5 The docs/ directory

This directory contains the documentation for the web-application.

2.6 The info/ directory

This directory contains the file `info.xml` which describes the basic application information like disc usage, application version, application size, used variables, necessary for the application to work within a virtual host.

The structure of the `info.xml` file is following:

```
<WEBAPP name="Application name" version="1.1" release="1">
<VERSIONHISTORY>
       <VER value="1.1-1"/>
</VERSIONHISTORY>

<ATTRIBUTES>
       <DESCRIPTION> Application description </DESCRIPTION>
       <LICENSE accept_required="no"> GPL </LICENSE> #Licence type, GPL, Freeware, etc.
       <ATTRIBUTE name="disc_space" value="111222" /> # disc usage in bytes
</ATTRIBUTES>

#The virtual host requirements for the package work
<REQUIREMENTS>
       <APACHE_VHOST name="PHP" value="on" />
       <APACHE_VHOST name="SSI" value="on" />
</REQUIREMENTS>

</WEBAPP>
```

3. Functions

Functions used in form and handler scripts:

sapp_create_database($db_name, $db_type, $user_name, $user_passwd); - checks the database limits, database name existence and creates database. Returns true in case of success. $db_type can be mysql, postgresql, mssql;

sapp_set_install_prefix($prefix); - checks if the folder name is already being used by another application or already exists. If the folder exists then the function returns false;

void sapp_set_param($name, $value); - adds the parameter and it's name to database;

string sapp_get_param($name); - access to web_params(); function;
void sapp_set_errormsg($msg); - returns the error text to the form or handler script;
bool sapp_is_ssl_enabled(); - checks if SSL is enabled;
sapp_get_locale(); - returns current locale, like en, jp, de
sapp_get_domain_name(); - returns the domain name;

check_sys_login($login); - checks whether the login name is valid;
check_sys_passwd($login, $passwd);- validates the password and login name, including the case when the password contains the login name;

check_pg_login($login); - check, if postgresql login name is valid;
check_pg_passwd($login, $passwd); - check if postgresql password is valid;
check_mailname($mail_name); - check if mailname is valid;
check_mail_passwd($login, $passwd); - check if mailname password is valid;
check_domain($dom_name); - checks if domain name is valid;
check_url($url); - checks if URL is valid;
check_shortUrl($url); - checks that URL is like http://domain.com (i.e. only protocol and hostname without slash at the end);
check_dbName($db); - checks if database name is valid;
check_dbUserName($usr); - checks if database user name is valid;
check_phone($phone); - checks if phone number is valid;
check_email($email); - check if e-mail address is valid;
check_ip($ip); - checks if IP address is valid;
check_mask($mask); - checks if network mask is valid;
check_filename($filename); - checks if filename is valid;
check_int($num); - checks if argument is an integer number.

4. Creating the Application

To create an application for uploading through Plesk control panel you should build an RPM from the package.

First of all you should find out the list of necessary configuration variables. They should be described in `info.xml` file along with the build number, version number and application disk usage.

NOTE

We strongly recommend that you use the patch syntax to apply the necessary changes to the applications like correcting the paths, etc.

You can find more information at the official RPM site: http://www.rpm.org/RPM-HOWTO/build.html

4.1 The spec file

The spec file should include the patch info, correct build number, version number and program name. The patches must be also listed there. The directory structure for the applications is described above.

The following variables are required in the spec file:

Summary: # Short description
Name: %name
Version: #version number
Release: #release number
Copyright: # License type, for example GPL, Freeware
Vendor: %vendor
Group: Applications/WWW
Packager: SWsoft Inc <info@plesk.com> # Set your or your company name there
BuildRoot: %prodbui,ldroot/../cgitory
Prefix: %product_root_d/var/cgitory
Requires: php # set any
Provides: Plesk-application-vault

If no comment is given for the variable, then the value should be exactly the same as given.

The application should have the attributes like below, it must be owned by user root:

```
%attr(-, root,root)%product_root_d/var/cgitory/%namesrc-%version-%release
```

4.2. Scripts

You should write the necessary forms and handlers using the functions described in section 3 to complete the list of application configuration variables.

4.3 Building the RPM package

Make sure that all the necessary files and subdirectories are all in place. Create tarballs for `httpdocs` and `cgi-bin`.

Write a `Makefile`, including all the necessary correction patches.

Copy the spec file to `/usr/src/redhat/SPECS`

NOTE

You must be logged in as `root` to build the RPM package.

Execute the following command:

```
# rpmbuild --bb /usr/src/redhat/SPECS/APPLICATION_NAME.spec
```

Here `APPLICATION_NAME.spec` should be replaced with the name of your application spec file described in the section 4.1.

Once the process of building the RPM package is complete you will find the package in the following directory: `/usr/src/redhat/RPMS/noarch`.

5. Uploading the Application

Log on the Plesk control panel as `admin`. In section *Server -> Application Vault* click the button [Browse], then select the application rpm and click [Send File].