DesktopServer 3.8 API

The DesktopServer 3.8 API allows you to extend DesktopServer's functionality in the following ways:

- Priority execution in PHP runtime (run code even before WordPress loads).
- Dynamic modification of the localhost developer "Sites" page.
- Run PHP in response to 3.8.X UI Events (i.e. site_created, site_imported, etc.)
- Load as "Global Plugin" in all WordPress virtual host instances.

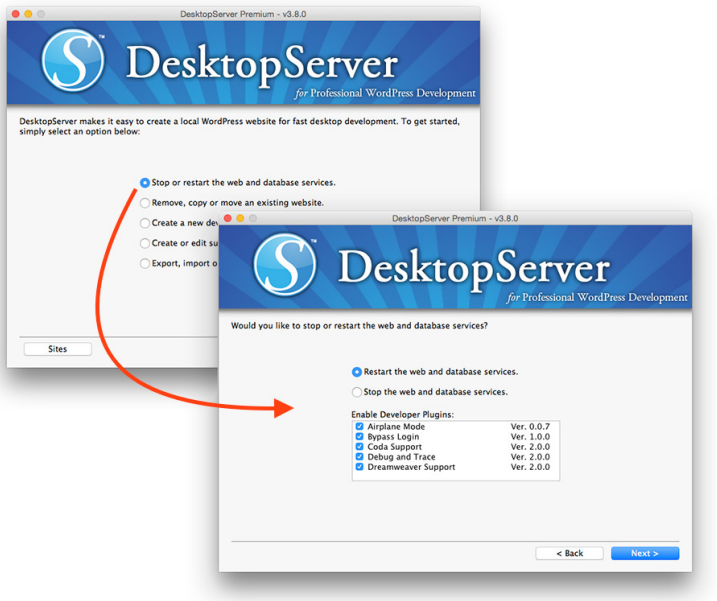
Getting Started

In order to extend the DesktopServer runtime, you will simply need to create a PHP file with a valid WordPress API plugin header within a subfolder of the ds-plugins folder. The ds-plugins folder can be found at the following locations:

On Macintosh at:
/Applications/XAMPP/ds-plugins

On Windows at:
C:\xampplite\ds-plugins

A plugin will not be active until it is "activated" via DesktopServer' first menu option to "Stop or restart the web and database services" followed by selecting the checkbox for the given plugin. The list also appears on Application startup.



The WordPress API plugin header format will be reflected within the DesktopServer 3.8 runtime in the following ways:

- Plugin Name – this will appear in the DesktopServer list box under the label "Enable Developer Plugins".
- Description – The description will appear as a tool top for the "Enable Developer Plugins" list box if the user highlights a given row in the list box.
- Version – The version number will appear to the right of the plugin name.

Global Plugin Behavior

Existing plugins that are activated in the ds-plugins folder will function just like regular WordPress plugins and will be active when any development WordPress website is visited. However, plugin behavior differs slightly in the following ways:

- No activation/deactivation actions. Like WordPress' mu-plugins, there are no activation or deactivation event hooks.

- Plugins must reside in a subfolder of the ds-plugins folder. Unlike regular WordPress plugins that can optionally reside in a subfolder or top level single file, or mu-plugins that cannot reside in a subfolder, ds-plugins must always reside in a uniquely named subfolder.

- A call to WordPress' plugins_url function will return the current virtually hosted development domain to the ds-plugins folder. I.e. http://www.example.dev/ds-plugins (as opposed to the WordPress plugins that would return http://www.example.dev/wp-content/plugins).

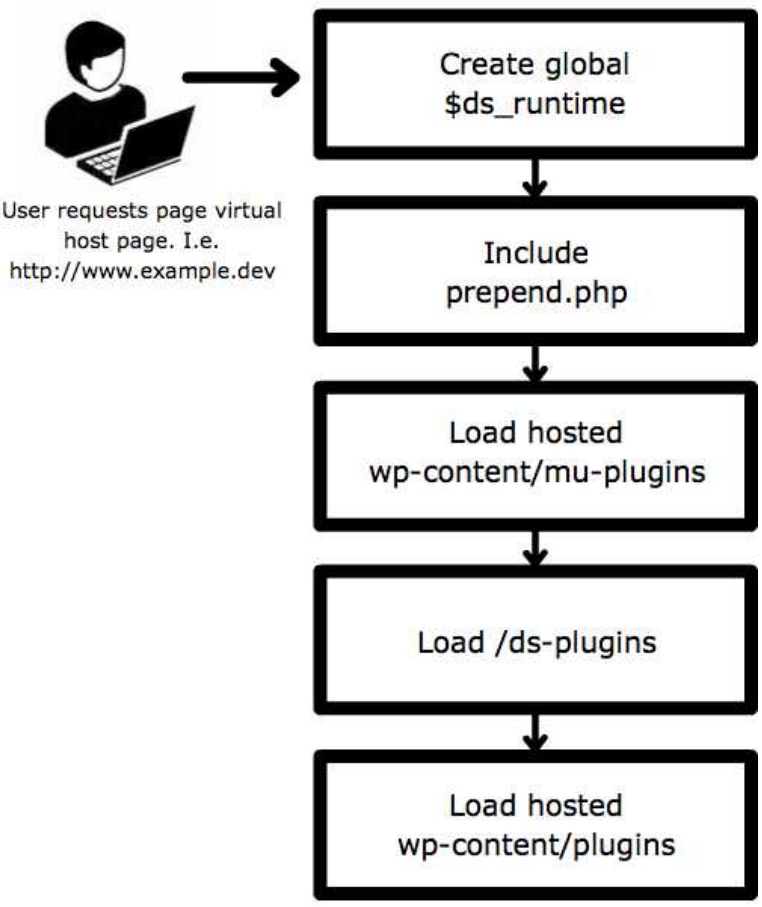
- As long as one ds-plugin is activated, every development website will receive an additional Apache directive to resolve the ds-plugins subfolder in the URL for the globally accessible ds-plugins folder (including the localhost page, i.e. http://localhost/ds-plugins).

- The global $ds_runtime object is created before loading any PHP scripts.

- An optional script can be included in the plugin subfolder with the title prepend.php or prepend_#.php (where # is a numeric value of 0 to 10). The prepend.php file provides early execution before the virtual hosted site is loaded.

For an example plugin that utilizes the prepend feature, please see the source for the debug-trace plugin.

The order of execution in PHP when a user requests the homepage of a hosted site (i.e. http://www.example.dev/index.php) is illustrated in the following diagram:



The global $ds_runtime object is created before any other PHP code is executed by the PHP interpreter. The runtime checks to see which ds-plugins have been marked as "active" and proceeds to look at each of the active ds-plugins' subfolder.

If an active plugin folder contains a file named prepend.php (or prepend_#.php whereby the # represents a numeric value 0 to 10), the file is included_once. Enumerated files are loaded in order 0 to 10 (i.e. all prepend_0.php load before all prepend_1.php, etc). The numeric versions ensure early or delayed execution across all defined plugins. When execution order is not required, it is recommend to simply name the file prepend.php and it will default to priority after 5 (prepend.php will execute after any prepend_5.php and before any prepend_6.php files).

After prepend files are processed, normal execution resumes at the virtual host. WordPress hosted websites will continue to load their respective mu-plugins, followed by any ds-plugins that were activated. Lastly, WordPress' active plugins will resume execution. DesktopServer plugins have a higher priority then native, activated WordPress plugins but occur after any local mu_plugins load.

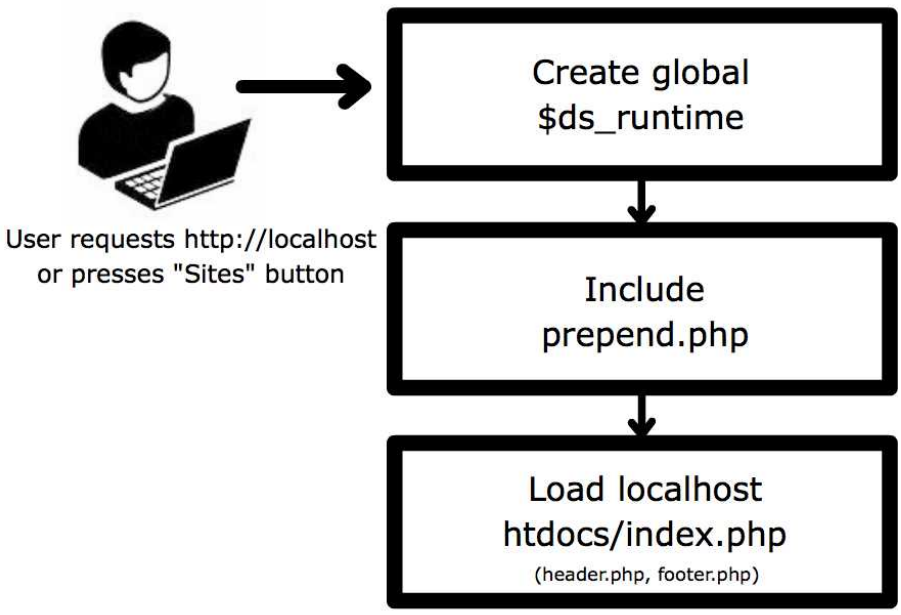
Plugin Authoring Tips

Use WordPress' `plugin_url()` function to obtain the current domain's URL and ds-plugin path to load resources from your plugin's folder. For example, to enqueue javascript from your plugin, http://www.example.dev/ds-plugins/notes/js/script-name.js use:

```php
<?php
  wp_enqueue_script( 'script-name', plugins_url() . '/notes/js/example.js', array(),
'1.0.0', true );
?>
```

Developer Sites (localhost) Page

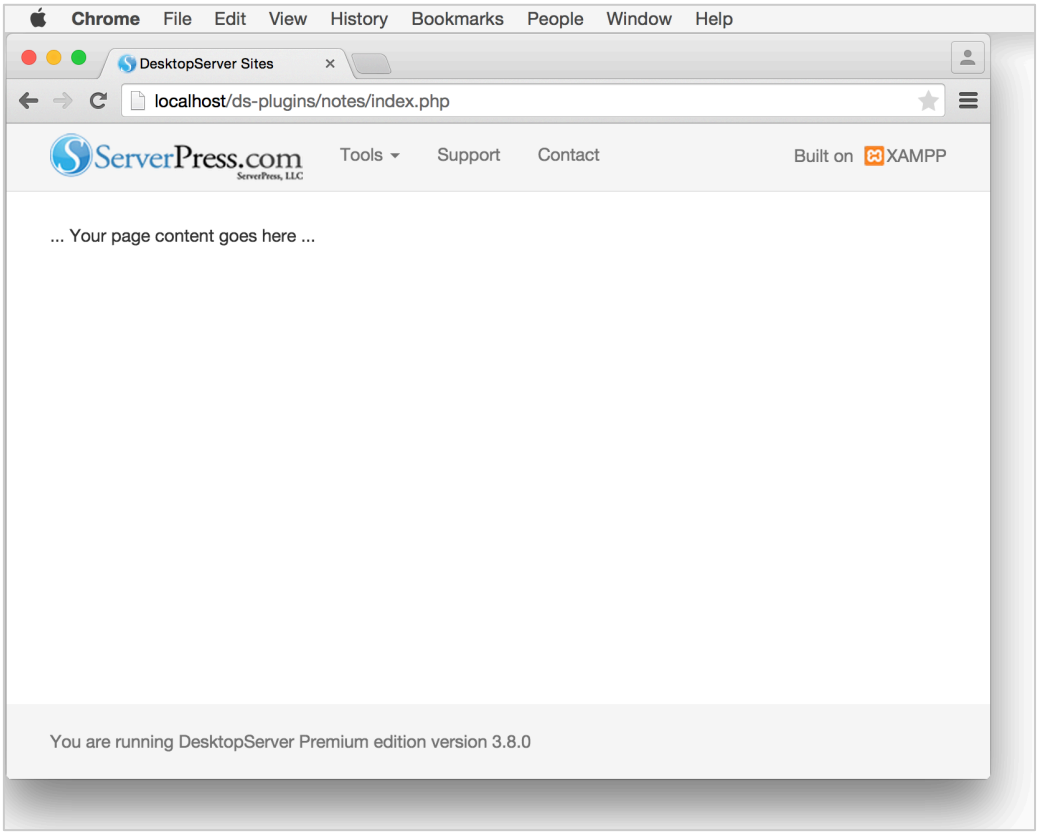
The localhost page appears when a user visits http://localhost in their web browser (via typing the address or clicking the "Sites" button in the lower left corner of the DesktopServer application's user interface). The localhost page uses the PHP interpreter and can be leveraged as another virtually hosted site. When the home page is loaded, the global $ds_runtime object is created that loads all subsequent activated plugins' prepend files before loading the localhost's index.php file (located in the htdocs folder). The localhost page does not include execution of the WordPress Plugin header PHP file because localhost itself is not a WordPress installation. The order of PHP execution when a user requests a localhost page is:



Creating Localhost Pages

To create a new localhost page, DesktopServer plugin developers should note that the localhost index.php page includes the header.php file and footer.php file. It is recommended that additional pages to be accessed from the localhost interface should also include the respective header.php and footer.php files to maintain access to the menu system and client side runtime scripts. For instance, creating an additional localhost page within a ds-plugin called “notes” would require the following code in a file located at ds-plugins/notes/index.php. It could contain:

```
<?php include_once('../../htdocs/header.php'); ?>
   <div class="container">
      ...
      Your page content goes here
      ...
   </div>
<?php include_once('../../htdocs/footer.php'); ?>
```



The DesktopServer plugin developer can alter the existing menu of the localhost’s default index.php page’s Developer Sites list via the global $ds_runtime object. See the next section about how to add custom actions and act on additional DesktopServer UI events by accessing the global $ds_runtime object.

DesktopServer Runtime Object

All scripts running under DesktopServer have access to the global $ds_runtime object. This object provides access to runtime information for the hosted sites within DesktopServer as well as utility functions to extend the DesktopServer localhost (aka "developer sites") page. Additional properties allow the developer to sense simple DesktopServer user interface events.

Prepends Processing

It should be noted that the prepend system in DesktopServer should be used with care, as it can be detrimental to overall system performance. This is because all PHP pages will invoke the prepend system (including calls to AJAX, etc.). Therefore, it is important to return control to the calling script if the desired conditions do not pertain to your plugin's behavior. For instance, prepend files that add behavior to the localhost page, or wish to take effect in response to DesktopServer User Interface events (last_ui_event), should first check to see if the $ds_runtime->is_localhost property is true, otherwise return:

```
<?php
   // Exit if we're not http://localhost (i.e. http://www.example.dev )
   global $ds_runtime;
   if ( ! $ds_runtime->is_localhost ) return;
?>
```

Additional methods and properties of interest to the developer are documented in the following pages.

Object:

`$ds_runtime`

The global $ds_runtime object is accessible by all scripts.

Properties:

`$ds_runtime->is_localhost`

Boolean that determines if domain is localhost. Plugin scripts intended to manipulate localhost or last_ui_events should return immediately if this value is false.

`$ds_runtime->preferences`

Contains the current DesktopServer preferences file; the preferences file is a JSON object containing a list of virtual host configurations, domain names, site paths (document_root), aliases, shares, and database. In PHP the JSON object is decoded and represented as a named array (key pair). See Opening the DesktopServer Preferences File for viewing of the current file format.

`$ds_runtime->last_ui_event`

Contains false if no current event exists or a JSON object that has been decoded into a PHP named array (key pair) about the event that has occurred. The DesktopServer application sets this value when a user-initiated event has completed. The object contains the following keys:

`action`

Contains a string describing the user-initiated event from the DesktopServer application. Action values appear in the table below.

`info`

A linear array of strings that contains additional information about the event. The following additional information will be present based on the given action:

action	info
alias_edited, shares_edited, start_services	*none*
site_created	*.dev domain name*
site_deployed	*.dev domain name, deployed domain name*
site_copied, site_moved	*original .dev domain name, destination .dev domain name*
site_removed	*original .dev domain name*
site_imported	*.dev domain name, path to archive filename*
site_exported	*original .dev domain name, exported domain name, path to archive filename*

Methods:

`$ds_runtime->add_action`

Allow plugin authors to add a function hook and extend the localhost pages.

`$ds_runtime->do_action`

Allow plugin authors to create new action hook definitions.

Customizing Localhost

The existing localhost page utilizes the $ds_runtime object with a number of do_action method calls to allow developers to inject runtime scripts, append to the "Tools" pull down menu, or alter the "Developer Websites" list. A complete list of actions follows:

action	description
ds_head	Occurs within the localhost <head></head> tags. Used to inject custom scripts, css, etc. *Receiving Arguments: none*
append_tools_menu	Allows the insertion of additional list items on the tools pull down menu. *Receiving Arguments: none*
list_domain *$domain*	Occurs after a domain is listed in the Developer Websites table. *Receiving Arguments: the domain name*
domain_button_group *$domain*	Occurs after the domain button group is output in the Developer Websites table. *Receiving Arguments: the domain name*
list_subdomain *$subdomain*	Occurs after a subdomain is listed in the Developer Websites table. *Receiving Arguments: the subdomain name*
subdomain_button_group *$subdomain*	Occurs after the subdomain button group is output in the Developer Websites table. *Receiving Arguments: the subdomain name*
ds_footer	Occurs within the <footer></footer> tags.