

SuperSmash Framework



Project: SuperSmash Framework
Author: Jeroen Saey
Startdate: 25-04-2012
Version: 1.0

Copyright

©SuperSmash.nl, SuperSmash

The SuperSmash Framework is copyrighted by SuperSmash.

This framework may not be used without acknowledgement of the creator.

You are not allowed to modify the SuperSmash Framework in any way.

The creator is not responsible for any damage this framework can cause while using irresponsible.









This framework may not be modified for personal use and may not be sold as your own.



Revision history

| Date | Version | Description | Author |
|------------|---------|-------------------------------|-------------|
| 25-04-2012 | 1.0 | SuperSmash Framework launched | Jeroen Saey |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Color information throughout the document.

| Color | Icon | Description |
|--------|---|-------------------------------|
| blue |  | Used for namespaces |
| orange |  | Used for classes |
| red |  | Used for critical information |
| purple |  | Used for warning information |
| brown |  | Used for constants |
| green |  | Used for public functions |
| green |  | Used for private functions |
| green |  | Used for protected functions |



Caution:

Settings that are changed inside the framework files can prevent your applications from running and can destroy the framework functionality.

It is advised that you don't change any more variables than named in the section:

'4. Settings that can be changed'

It is not allowed to change or modify the framework in any way.

It is NOT OK to take it, copyright under your name (or your group's name), and sell it as your own.

It is NOT OK to take it, revise it, copyright it and then sell it as your own.

Table of Contents

| | |
|--|----|
| 1. Preface..... | 6 |
| 1.1 Scope | 6 |
| 1.2 Audience..... | 6 |
| 1.3 Purpose..... | 6 |
| 2. Introduction..... | 7 |
| 3. System design..... | 8 |
| 3.1 Communication flow between components..... | 8 |
| 4. Settings that can be changed | 9 |
| 5. File information | 10 |
| 5.1 Root | 10 |
| 5.1.1 Index..... | 10 |
| 5.1.2 .htaccess..... | 10 |
| 5.2 System | 11 |
| 5.2.1 Applications | 11 |
| 5.2.2 Bootstrap..... | 12 |
| 5.3 SuperSmash..... | 12 |
| 5.3.1 Global | 12 |
| 5.3.2 Registry..... | 13 |
| 5.3.3 Router..... | 13 |
| 5.3.4 SuperSmash..... | 14 |
| 5.3.5 Benchmark..... | 14 |
| 5.3.6 Language | 14 |
| 5.3.7 Loader..... | 15 |
| 5.3.8 Input | 15 |
| 5.3.9 Configuration..... | 16 |
| 5.4 Cookie..... | 16 |
| 5.4.1 Database..... | 17 |
| 5.4.2 Debug | 18 |
| 5.4.3 Controller..... | 18 |
| 5.4.4 Session..... | 19 |
| 5.5 Library..... | 20 |



SuperSmash Framework { 5 of 21 } @ SuperSmash

PHP

<http://www.SuperSmash.nl>

| | |
|------------------------|----|
| 5.5.1 Cache | 20 |
| 5.5.2 Email | 20 |
| 5.5.3 Validation | 21 |



1. Preface

1.1 Scope

This document will be used as a reference to the SuperSmash Framework. This document will explain all the functions that can be used by the framework.

1.2 Audience

The target audiences for this document are the developers who need to create a PHP application.

1.3 Purpose

This document describes the technical design and contains information necessary for technical understanding of the SuperSmash framework.



2. Introduction

How to create PHP application without the overall picture?

This can easily be done by using frameworks. By using a framework you will have a clean directory structure and a clean way to maintain your application code. This framework uses the Model View Controller architecture to keep everything simple and clean.

The main aim of the MVC architecture is to separate the business logic and application data from the presentation data to the user.

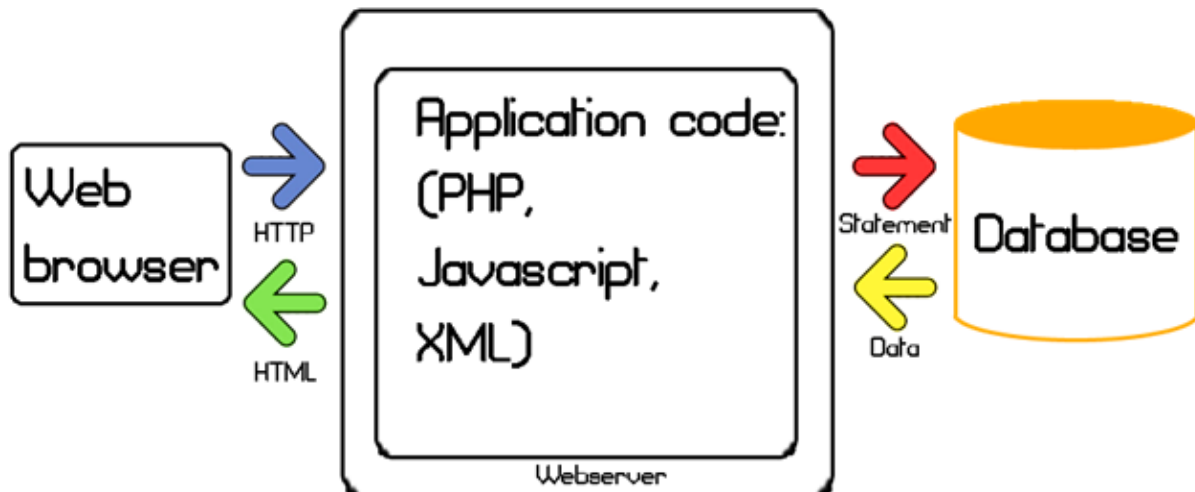
The main reason this framework was created is because we needed a simple and effective way to manage the code. Nowadays allot of frameworks are created. But the problem with most frameworks is that they are too big for a simple application. This framework tries to be small and effective for everyone.

Because coding needs to be simple!



3. System design

3.1 Communication flow between components



When a user connects to the website using a browser, the user builds a data request by navigating through PHP pages . The requests are send to the webservice and are processed by the framework . If needed the framework will request some data from the database by using specific statements.

4. Settings that can be changed



5. File information

5.1 Root

5.1.1 Index

The index file will create the following constants:

| Constant | Usage | Type |
|---------------------|--|---------|
| APPLICATIONSCHOOSER | Will be used to show a start page of your applications | Boolean |

When all the constant are loaded the index file will load the applications file.

When the applications file is loaded the framework will set the specified application that the framework will use. Afterwards the framework will load the bootstrap file and starts the SuperSmash Framework.

5.1.2 .htaccess


The .htaccess file (can) be present in the root folder of the system folder. Below you will find a default sample of the content of the file:

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  # Tell PHP that the mod_rewrite module is ENABLED.
  SetEnv HTTP_MOD_REWRITE On
  # Dont redirect direct links to files or directories to the index.php
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteCond %{REQUEST_FILENAME} !-d
  # Rewrite all other URLs to index.php/URL
  RewriteRule ^(.*)$ index.php?url=$1 [PT,L]
</IfModule>
```




5.2 System

5.2.1 Applications


The applications file will be set in the namespace: `settings` .


The file will create the following constants:

| Constant | Usage | Type |
|---------------------|--|---------|
| DS | Will be used as the directory separator for files and folders | System |
| ROOT | Will be used to specify the root path of the framework | String |
| SYSTEM | Will be used to specify the system path of the framework | String |
| SUPERSMASHFRAMEWORK | Will be used to specify if the user has access to the page requested | Boolean |

When all the constants are loaded the applications file will load the `settings`  class. This class will be used to set some global settings for the framework. The class will hold:

- The actively loaded application
- An array with all the applications in the framework
- And the applications path for the SuperSmash Framework and user applications.

The `Cookie`  class will be loaded and initialized.

The user application is loaded by using a cookie or session variable. (This is only the case when the `APPLICATIONSCHOOSER`  constant is set to true.




5.2.2 Bootstrap

The bootstrap file will load the global file to load all the global framework settings. After the file has been loaded the bootstrap file will load the registry file.


The bootstrap file will set the error handler to the SuperSmash Framework Error Handling System. The benchmark class gets loaded and the benchmark class will start measuring time of the System files. Afterwards the SuperSmash Framework will be loaded (Core files).



5.3 SuperSmash


5.3.1 Global


The global file be set in the namespace: `settings\settings` .


This file creates several functions that can be accessed globally throughout the framework.


The `autoloader`  function will automatically load classes that are not yet included/loaded in the SuperSmash framework.


The `errorhandler`  function will automatically handle all the errors that where given by PHP. (By giving error information to the debugging class )


The `show404`  function will render a 404 error page to the browser.



The `logmessage` function  will log all the errors and warnings to a specified (debug.log) filename.



The `configuration` function  will create an array of all the settings in the configuration file.

The `configurationset` function  will save a configuration setting to the configuration array.


The `configurationsave` function  will save a configuration array to the configuration file.

The `configurationload` function  will load a specified configuration from the configuration file.

The `getinstance` function  will get an active instance of the controller class .

The `geturlinformation` function  will return the website URL along with the extra URL information. (Loaded through the Router class .


The `loadclass` function  will load a specified classname given by a parameter.


The `redirect` function  will redirect the user to a specified location after an amount of time.


5.3.2 Registry


The registry file will hold all global variables that can be used for multiple objects. This class is used to easily pass data along objects.


This file creates several functions.

The static `singleton`  function will be used to create a singleton registry object so it can be accessed from anywhere in the framework.

The `get` function  will get a specified key from the registry and return it to the requested class.

The `set` function  will set a specified key in the registry so it can be accessed from anywhere in the framework.

The `load` function  will load a specified key and returns the singleton of it to the requested class.


The `store` function  will store an object as a singleton object.


5.3.3 Router


The router file is used to match specific URL statements so it can load and execute the specific requested controller. All the URL requests need to come from one central location namely the index file.

The router file be set in the namespace: `System\SuperSmash` .



This file creates several functions for routing URLs:


The `checkroutingurl`  function will check how the URL should be loaded. If there is no controller named by the URL the default controller will be loaded defined in the configuration file.


The `geturlinformation`  function will return all the URL information as an array.

The `getursegment`  function will return the specified URI segment.


5.3.4 SuperSmash

The SuperSmash file is the core of the framework. This file will start the framework itself. While starting the framework initiates the router class  and will create the specific controller(s) classes  that needs to be loaded with the user application.



The SuperSmash file be set in the namespace: `System\SuperSmash` .

The SuperSmash file will also use the namespace: `settings\settings` .

This file creates several functions :


The `start`  function will start the SuperSmash framework, by loading specific controllers and classes.

The `performaction`  function will perform an action on the specified controller.


The `database`  function will return the database class .


5.3.5 Benchmark

The benchmark file can be used as a timer class.


The benchmark file be set in the namespace: `System\SuperSmash` .

This file creates several functions :


The `start`  function will start a specified timer.


The `stop`  function will stop a specified timer.

The `elapsed`  function will return the elapsed time between the start and stop of a timer.


The `usage`  function will return the memory the page used while loading.


5.3.6 Language


The language file be set in the namespace: `System\SuperSmash` .


The language file will also use the namespace: `settings\settings` .


This file creates several functions :

The `setlanguage`  function is used to set the specified application / framework language.

The `load`  function is used to load the specified application / framework language in the array.


The `get`  function is used to get the specified variable of the configuration array.


The `getlanguages`  function is used to get an array of all the languages in the framework / application folder.

The `scanlanguagefolders`  function is used to scan and find all the installed languages.


5.3.7 Loader

The loader file is used to load the models, helpers, databases, libraries and views.


The loader file be set in the namespace: `System\SuperSmash` .


The loader file will also use the namespace: `settings\settings` .


This file creates several functions :

The `model`  function will load the specified model.

The `view`  function will load the specified view and displays it to the browser.

The `library`  function will be used to load library classes from the user application library or the system library folders.

The `database`  function will be used to setup a new database connection.


The `helper`  function will be used to load a specified helper module from the user application helpers or the system helpers folders.


5.3.8 Input


The input file is used to filter specific input from the user.


The input file be set in the namespace: `System\SuperSmash` .


This file creates several functions :


The `post`  function will return clean post variables.


The `get`  function will return clean get variables.

The `cleanelement`  function will clean the specified variable.


The `cookie`  function will return the cookie variable.


The `setcookie`  function will set the cookie variable.


The `useragent`  function will get the browser from the user.


The `ipaddress`  function will get the IP Address from the user.

The `setrules`  function will create and set some cleaning rules.

The `clean`  function will clean the specified input

The `remove`  function will remove unwanted tags.


The `filtertags`  function will strip certain tags of strings.


The `filterattribute`  function will strip certain tags of attributes.

The `decode`  function will decode the source to a clean string.

5.3.9 Configuration


The configuration file is used to get and set settings from the user configuration file.


The configuration file be set in the namespace: `System\SuperSmash` .

The configuration file will also use the namespace: `settings\settings` .


This file creates several functions :

The `get`  function will get a specified variable from the configuration file.

The `getall`  function will get all the variables that where set in the data array.

The `set`  function will set a variable in the data array.

The `load`  function will load a specified configuration file and will set all the variables in the array.

The `save`  function will write all the configuration variables to the configuration file and will make a backup of the old configuration file.

The `revert`  function will revert the configuration file to the last saved configuration file.


5.4 Cookie


The cookie file is used to get and set cookies throughout the framework / application.


The cookie file be set in the namespace: `System\SuperSmash` .


This file creates several functions :


The static `init`  function will initialize the cookie.

The static `getdomain`  function will get the current domain of the specified URL.

The static `set`  function will set the cookie with the given specified variables.

The static `exists`  function will check if the specified cookie exists.


The static `get`  function will get a specified cookie.


The static `remove`  function will remove a specified cookie.


5.4.1 Database


The database file is used to create a connection to the database.

This file creates several functions :


The `open`  function will open a connection to the specified database.


The `log`  function will log an error message to the log file.


The `showdebuginformation`  function will show the debug information to the screen (if debugging is enabled).


The `drivers`  function will print all the available PDO drivers to the screen.


The `query`  function will execute a query to the database.


The `query_secure`  function will execute a query to the database with anti SQL injection.


The `query_first`  function gets the first row of a query in the database.


The `query_single`  function gets the first table cell of a query in the database.


The `rowcount`  function will return the row count of a query in the database.


The `columns`  function will return all the column names as an array from the database.


The `insert`  function will insert a query and returns the ID of the inserted query.


The `update`  function updates a specified table in the database.

The `delete`  function deletes a specified record from the database.


The `execute`  function executes a specified stored procedure.


The `getlatestid`  function gets the latest id of the specified table from the database.

The `showtables`  function gets all the tables from a specified database.

The `showdatabases`  function will get all the database from the server (and where you got permission to use them).

The `geterror`  function will show the latest error that occurred in the connection.


The `close`  function will disconnect the database.


The `stmntcount`  function will build a query that counts a specified query result.




5.4.2 Debug


The debug file is used to debug errors.

The debug file be set in the namespace: `System\SuperSmash` .


The debug file will also use the namespace: `settings\settings` .


This file creates several functions :


The `triggererror`  function will trigger an error.


The `showerror`  function will show a specified error page.

The `logerror`  function will log the error to the error log file.

The `log`  function will log the message to the debugging log file.


The `errorreporting`  function will enable or disable the error reporting.

The `createerrorpage`  function will create an error page that will be shown in the browser.


The `vardump`  function will create a dump of the specified message in a clean way so the user can easily see what an why the dump was created.


5.4.3 Controller


The controller file is used as a base controller. (a controller extends from this one)

The controller file be set in the namespace: `System\SuperSmash` .

This file creates several functions :


The static `getinstance`  function will get the current instance of the controller.


The `_beforeaction`  function will be called before an action is taken.

The `_afteraction`  function will be called after an action is taken.


5.4.4 Session

The session file is used to debug errors.


The session file be set in the namespace: `System\SuperSmash` .

The session file will also use the namespace: `settings\settings` .


This file creates several functions :


The `regenerateid`  function will regenerate a session id for the current session.


The `setdata`  function will set a specified item in the session data array.


The `unsetdata`  function will unset a specified item from the session data array.


The `getdata`  function will get a specified item from the session data array.

The `getalldata`  function will return all the data specified in the session data array.

The `destroy`  function will destroy the current session and all its data in the database.


The `read`  function will read a session from the database.


The `create`  function will create a new session and puts it in the database.


The `_update`  function will update the current session in the database.


The `_write`  function will write specified data to the session in the database.


The `_setcookie`  function will set a session cookie.

The `_cleanexpired`  function will remove all the expired sessions from the database.

The `_generateid`  function will generate a unique session id.

The `_checkidrenewal`  function will check if the session id needs to be renewed (and does so if necessary).

The `_flagforupdate`  function will flag the session so the session id gets updated at the next subsequent request.


The `_checkforupdateflag`  function will check if the session has requested to update (and does so if necessary).


The `_setconfig`  function set the configuration variables to the session class.

5.5 Library


5.5.1 Cache


The cache file is used to cache contents.


The cache file be set in the namespace: `System\Library` .


The cache file will also use the namespace: `settings\settings` .


This file creates several functions :

The static `set_path`  function will set the cache path.

The `get`  function will read and return the content of a cached file.


The `save`  function will save the contents into a give cache file.

The `delete`  function will delete a specified cache file.


The `clear`  function will delete all the cached files.


5.5.2 Email


The email file is used to send email messages.


The email file be set in the namespace: `System\Library` .


This file creates several functions :


The `send`  function will send the email.


The `to`  function will add a recipient to the email message.


The `from`  function will add a sender to the email message.


The `reply_to`  function will add a reply to to the email message.


The `cc`  function will add a cc to the email message.


The `bcc`  function will add a bcc to the email message.


The `subject`  function will add a subject to the email message.


The `message`  function will add the message to the email headers.

The `attachment`  function will add an attachment to the email message.

The `build_header`  function will build the email headers.


The `validate`  function will validate if the specified email address is a valid email address.

The `mime_types`  function will get the mime type of an attachment file.

The `clear`  function will clear the current email.


5.5.3 Validation


The validation file is used to cache contents.


The validation file be set in the namespace: `System\Library` .


This file creates several functions :


The `set`  function is used to set specified rules to certain `$_POST` variables.

The `validate`  function will validate all the `$_POST` variables that have rules attached to them.


The `get_errors`  function will return an array with all the errors.

The `set_error`  function sets an error for the field.


The `required`  function determines if the string passed has any values.


The `email`  function determines if the string is a valid email address.


The `number`  function determines if the string passed is numeric.

The `url`  function determines if the string passed is valid URL.

The `float`  function determines if the string passed is a float.

The `min`  function determines if the string passed has a minimum value of a given value.

The `max`  function determines if the string passed has a maximum value of a given value.

The `pattern`  function determines if the string passed contains the specified pattern.