

leaf framework Documentation



Contents

Package leaf Procedural Elements	2
index.php	2
Define LEAF APPS	2
Define LEAF BASE	2
Define LEAF REL STATUS	2
Define LEAF REL VERSION	2
Define LEAF VAR	2
Define LEAF WORKING DIR	2
Base.php	4
Benchmark.php	5
Config.php	6
Controller.php	7
Dispatcher.php	8
EndorsementManager.php	9
Exception.php	10
Hash.php	11
Input.php	12
Loader.php	13
Model.php	14
Registry.php	15
Request.php	16
Response.php	17
Router.php	18
View.php	19
Package leaf Classes	20
Class leaf Base	20
Class Constant LEAF CLASS_ID	20
Class Constant LEAF REG_KEY	21
Var \$Registry	21
Constructor construct	21
Method clone	21
Method get	22
Method toString	22
Class leaf Benchmark	22
Class Constant LEAF CLASS_ID	23
Class Constant LEAF REG_KEY	23
Var \$indexTable	23
Constructor construct	23
Method addIndex	23
Class leaf Config	24
Class Constant LEAF CLASS_ID	25
Class Constant LEAF REG_KEY	25
Var \$options	25

Var \$optionsTable	25
Constructor construct	25
Method getByHashKey	26
Method offsetExists	26
Method offsetGet	26
Method offsetSet	27
Method offsetUnset	27
Method toString	28
Class leaf Controller	28
Class Constant LEAF_CLASS_ID	28
Class Constant LEAF_REG_KEY	28
Constructor construct	28
Method toString	29
Class leaf Dispatcher	29
Class Constant LEAF_CLASS_ID	29
Class Constant LEAF_REG_KEY	30
Var \$controller	30
Var \$controllerName	30
Var \$target	30
Constructor construct	30
Method dispatchController	31
Method toString	31
Class leaf EndorsementManager	31
Class Constant LEAF_CLASS_ID	32
Class Constant LEAF_REG_KEY	32
Var \$allowEndorsement	32
Var \$currentClass	32
Var \$endorsed	32
Var \$registeredEndorsed	33
Constructor construct	33
Method getEndorsedClasses	33
Method introspectEndorsedClass	33
Method isEndorsed	34
Method loadEndorsedClass	34
Class leaf Exception	35
Constructor construct	35
Class leaf Hash	36
Var \$algorithms	36
Var \$defaultAlgorithm	36
Var \$useAltMethod	37
Constructor construct	37
Method digestFile	37
Method digestMsg	38
Method setAlgorithm	38
Class leaf Input	38
Class Constant LEAF_CLASS_ID	39
Class Constant LEAF_REG_KEY	39
Var \$input	39
Constructor construct	39

Method offsetExists	39
Method offsetGet	40
Method offsetSet	40
Method offsetUnset	40
Class leaf Loader	41
Class Constant LEAF_CLASS_ID	41
Class Constant LEAF_REG_KEY	41
Constructor construct	41
Method endorsed	41
Method extension	41
Method library	42
Method model	42
Method plugin	42
Method toString	43
Class leaf Model	43
Class Constant LEAF_CLASS_ID	43
Class Constant LEAF_REG_KEY	43
Constructor construct	43
Class leaf Registry	44
Var \$instance	44
Var \$registered	45
Constructor construct	45
Method getInstance	45
Method instance	45
Method isRegistered	46
Method register	46
Method toArray	47
Method unregister	47
Method get	47
Method set	48
Class leaf Request	48
Class Constant LEAF_CLASS_ID	49
Class Constant LEAF_REG_KEY	49
Var \$action	49
Var \$controller	49
Var \$controllerFile	49
Var \$queryElems	50
Var \$segments	50
Constructor construct	50
Method getApplicationName	50
Method getControllerFileName	51
Method getControllerName	51
Method getQueryString	51
Method getQueryStringAsString	52
Method reconstructUrl	52
Method redirect	52
Method segment	53
Method segmentsToArray	53
Method totalSegments	54

Method toString	54
Class leaf Response	54
Class Constant LEAF_CLASS_ID	55
Class Constant LEAF_REG_KEY	55
Var \$buffer	55
Var \$useGzip	55
Var \$useTidy	55
Constructor construct	55
Method clearHeaders	55
Method flushOutputBuffer	56
Method getOutputBuffer	56
Method outputBufferingStart	56
Method outputBufferingEnd	56
Method sendResponse	57
Method setHeader	57
Method toString	57
Class leaf Router	57
Class Constant LEAF_CLASS_ID	58
Class Constant LEAF_REG_KEY	58
Var \$queryString	58
Var \$queryStringElements	58
Var \$requestClass	59
Var \$requestMethod	59
Var \$requestUri	59
Var \$segments	60
Constructor construct	60
Method chopSegment	60
Method getClassName	61
Method getMethodName	61
Method queryStringElements	61
Method segments	61
Method segmentsSize	62
Method toString	62
Class leaf View	62
Class Constant LEAF_CLASS_ID	63
Class Constant LEAF_REG_KEY	63
Constructor construct	63
Method render	63
Method toString	63
Benchmark.php	65
Request.php	66
Hook.php	67
Hook_Conditional.php	68
Class leaf Hook	68
Class Constant LEAF_CLASS_ID	68
Class Constant LEAF_REG_KEY	69
Constructor construct	69
Method toString	69
Class leaf Hook_Conditional	69

Class Constant LEAF_CLASS_ID	69
Class Constant LEAF_REG_KEY	69
Constructor __construct	70
Method condition	70
Hooks.php	71
Function getHooks	71
Define HOOK_POST_CONTROLLER_DISPATCH	71
Define HOOK_POST_FRONT_CONTROLLER	71
Define HOOK_PRE_CONTROLLER_DISPATCH	72
Function introspectHooks	72
Function runHook	72
Function runHooks	73
Log.php	74
Logger.php	75
Class leaf_Log	75
Class leaf_Logger	75
Class Constant LEAF_CLASS_ID	76
Class Constant LEAF_REG_KEY	76
Var \$buffer	76
Var \$filename	76
Var \$fp	76
Var \$levels	77
Var \$stampPat	77
Var \$threshold	77
Constructor __construct	77
Method end_flush	78
Method flush	78
Method setLevel	78
Method setTimeStampPattern	79
Method __call	79
Debug.php	80
Front_Controller.php	81
Debug.php	83
Function inspect_var	83
Function i_var	83
Function inspect_var Array	84
Function inspect_var Boolean	84
Function inspect_var Generic	84
Function inspect_var Number	84
Function inspect_var Object	85
Function inspect_var Object_getSubclasses	85
Function inspect_var Resource	85
Function inspect_var String	86
Dependancies.php	87
Function dependsOn	87
Function dependsOnFunc	87
Function dependsOptionalOn	88
Error.php	89
Function showErrorPage404	89

Function showHtmlMessage	90
Handlers.php	91
Function errorHandler	91
Function exceptionHandler	91
Main.php	93
Function frameworkForceTerminate	93
Function frameworkSanityCheck	93
Function autoload	93
Appendices	95
Appendix A - Class Trees	96
leaf	96
Appendix B - README/CHANGELOG/INSTALL	98
TODO	99
LICENSE	99
README	99
CHANGELOG	100
INSTALL	100
Appendix C - Source Code	101
Package leaf	102
source code: Benchmark.php	103
source code: Request.php	104
source code: Hooks.php	105
source code: Debug.php	107
source code: Front Controller.php	111
source code: Debug.php	114
source code: Dependancies.php	119
source code: Error.php	121
source code: Handlers.php	123
source code: Main.php	125
Appendix D - Todo List	129

Package leaf Procedural Elements

index.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: index.php 48 2007-11-13 21:11:14Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

LEAF_APPS = LEAF_WORKING_DIR.'applications/' *[line 53]*

Subdirectory in which user`s applications are located.

LEAF_BASE = LEAF_WORKING_DIR.'leaf/' *[line 48]*

Subdirectory in which leaf`s files are located.

LEAF_REL_STATUS = 'DEV' *[line 37]*

leaf`s Status and Version information.

LEAF_REL_VERSION = '1.0' *[line 38]*

LEAF_VAR = LEAF_WORKING_DIR.'var/' *[line 58]*

Subdirectory in which cache and temporary files are stored.

LEAF_WORKING_DIR = dirname(realpath(__FILE__)).'/' *[line 43]*

Current directory where **this file is located.**

`require_once LEAF_BASE.'front/Front_Controller.php' [line 64]`

The [Front Controller](#) handles the startup sequence.

Base.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Benchmark.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Config.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Controller.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Dispatcher.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

EndorsementManager.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Exception.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Hash.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Input.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Loader.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Model.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Registry.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Request.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Response.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Router.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

View.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Package leaf Classes

Class leaf_Base

[line 34]

Assigns common behavior and properties in the internal classes.

Most of the leaf's internal classes inherit from this class.

This way, these classes, are provided with a unified base model, to communicate (by referencing) with the other objects, as if they were (private in this case) properties.

Example (**pure fictional, because the visibility is private**):

```
1  $router = new leaf_Router();
2  echo $router->config['base_url'];
```

We assume, that the "config" property has already been instantiated and automatically registered.

Also, we assume that the access of the properties is public.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Base.php 46 2007-11-12 07:41:49Z makism \$
- **Abstract Element**

leaf_Base::LEAF_CLASS_ID

= "LEAF_BASE-1_0_dev" [line 52]

A unique handle for each subclass.

leaf_Base::LEAF_REG_KEY

= "base" *[line 45]*

The name of the instance that the current subclass
will known as.

Think of it like a "key" and "value" relation.

"key" is a name, and "value" is the actual instance of a subclass.

leaf_Base::\$Registry

object leaf_Registry = NULL [line 63]

The one and only unique instance of the class leaf_Registry.

All subclasses are registered in this object and have the ability to refer to other objects stored in the registry.

- **Static**
- **Access** private

Constructor *void* function leaf_Base::__construct([\$descendant = NULL]) *[line 74]*

Function Parameters:

- *NULL|string* **\$descendant**

Each time a subclass calls the constructor, some checks are performed to find out if the subclass is already instantiated, and thus exists in the [\\$Registry](#).

- **Access** public

void function leaf_Base::__clone() *[line 100]*

Prevent object cloning.

- **Access** private

object function leaf_Base::__get(\$obj) [*line 90*]

Function Parameters:

- *string* \$obj

Returns the requested object from the [Registry](#)

- **Access** protected

string function leaf_Base::__toString() [*line 110*]

Returns a descriptive string about this class.

- **Abstract Element**
- **Access** public

Class leaf_Benchmark

[*line 25*]

Keeps statistics (execution time, memory usage) either for a specific class, or a code block.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Benchmark.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.

leaf_Benchmark::LEAF_CLASS_ID

= "LEAF_BENCHMARK-1_0_dev" *[line 29]*

leaf_Benchmark::LEAF_REG_KEY

= "benchmark" *[line 27]*

leaf_Benchmark::\$indexTable

array = array() [line 37]

All requested benchmark points are stored in this array, in a hashtable-like way.

- **Access** private

Constructor *void* function leaf_Benchmark::__construct() *[line 45]*

Declares dependencies.

- **Access** public

void function leaf_Benchmark::addIndex(\$name, \$type) *[line 59]*

Function Parameters:

- *string* **\$name**
- *integer* **\$type**

Add a begin/end benchmark point.

- **Access** public

Class leaf_Config

[line 52]

Provides access to the configuration files.

This class encapsulates all the configuration files.

The parameters that are read from the general.php file, are accessible as array offsets.

Example:

```
1  $conf = new leaf_Config();
2  echo $conf['base_uri'];
```

This behaviour is succeeded by implementing the [ArrayAccess](#) interface.

The other paremeters, are accessed by using the method `getByHashKey(configFileName)`. This method returns an array with all the related configuration parameters.

Example:

This design was selected because the "general" properties are the most commonly used, thus it is wise to provide a fast-access method.

Also, we could have used the "magic" methods "`__call`" and "`__set`", but they are declared as "final" in the parent class.

- **Package** leaf
- **Sub-Package** core

- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Config.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Maybe we should unset the global variable \$GLOBALS somewhere else.
 2. Possible implementation of a subclass that will read separate configuration files for each Controller, thus declaring different parameters and maybe unique for each Controller.

leaf_Config::LEAF_CLASS_ID

= "LEAF_CONFIG-1_0_dev" [*line 56*]

leaf_Config::LEAF_REG_KEY

= "config" [*line 54*]

leaf_Config::\$options

*array = array() [*line 63*]*

All configuration parameters will be stored in this array.

- **Access** private

leaf_Config::\$optionsTable

*array = array() [*line 71*]*

All configuration parameters are stored in this array, using their parent configuration file as index.

- **Access** private

Constructor *void* function leaf_Config::__construct() [*line 79*]

Encapsulates all the configuration files, and stores the parameters in an array.

- **Access** public

array|NULL function leaf_Config::getByHashKey(\$key, \$str) [*line 104*]

Function Parameters:

- *string* **\$str**
- **\$key**

Returns all parameters related with the specified key.

- **Access** public

boolean function leaf_Config::offsetExists(\$offset) [*line 118*]

Function Parameters:

- *string* **\$offset**

Checks if the request parameter-name exists.

- **Access** public

mixed function leaf_Config::offsetGet(\$offset) [*line 132*]

Function Parameters:

- *string* **\$offset**

Returns the value of the specified parameter.

- **Access** public

void function leaf_Config::offsetSet(\$offset, \$value) [line 144]

Function Parameters:

- *string* **\$offset**
- *mixed* **\$value**

Sets a value in the specific parameter.

- **Access** public

void function leaf_Config::offsetUnset(\$offset) [line 159]

Function Parameters:

- *string* **\$offset**

Unsets a parameter.

Unsets a parameter while the script is executed, that is, it doesn't mean it updates the configuration file itself.

- **Access** public

void function leaf_Config::__toString() [line 164]

- **Access public**

Class leaf_Controller

[line 30]

Assigns some common characteristics to all user`s Controllers.

All Controller, **must** inherit from this class, otherwise they will be **ignored**.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Controller.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Refactor.
 2. Possible implementation of "Dependencies Injection (DI)" method, so that each Controller, will load **only** the classes it depends on.
- **Abstract Element**

leaf_Controller::LEAF_CLASS_ID

= "LEAF_CONTROLLER-1_0_dev" *[line 34]*

leaf_Controller::LEAF_REG_KEY

= "controller" *[line 32]*

Constructor *void function leaf_Controller::__construct() [line 42]*

Calls the parent constructor.

- **Access** public

void function leaf_Controller::__toString() [line 47]

- **Access** public

Class leaf_Dispatcher

[line 29]

Prepares to dispatch the specifief Controller/Action.

Includes the file that the requested Controller is declared, and performs some basic checks in the Controller`s implementation and naming scheme.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Dispatcher.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Remove member properties.
 2. Replace the member properties with method calls to leaf_Request.

leaf_Dispatcher::LEAF_CLASS_ID

= "LEAF_DISPATCHER-1_0_dev" *[line 33]*

leaf_Dispatcher::LEAF_REG_KEY

= "dispatcher" [*line 31*]

leaf_Dispatcher::\$controller

object leaf_Controller = NULL [*line 59*]

An instance of the requested Controller.

- **Access** private

leaf_Dispatcher::\$controllerName

string = NULL [*line 52*]

The requested class name (Controller).

For more info take a look at the class leaf_Request.

- **Access** private

leaf_Dispatcher::\$target

string = NULL [*line 43*]

The file name in which the requested Controller is located.

For more info take a look at the class leaf_Request.

- **Access** private

Constructor *void* function leaf_Dispatcher::__construct() [*line 68*]

Tests the requested Controller and prepares for dispatching it, along with the Action.

- **Access** public

void function leaf_Dispatcher::dispatchController() [line 119]

Checks for the existence of the desired method and calls it.

- **Access** public

void function leaf_Dispatcher::__toString() [line 128]

- **Access** public

Class leaf_EndorsementManager

[line 30]

Handles the requests, to overlap the internal implementations of specific classes, with external, in userspace-like fashion.

This way, the framework can easily be updated, and hacked - allowing the users to provide their own implementations of specific classes.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: EndorsementManager.php 46 2007-11-12 07:41:49Z makism \$

- **TODO**

1. Implement.
2. Document.
3. Revamp.

leaf_EndorsementManager::LEAF_CLASS_ID

= "LEAF_ENDORSEMENTMANAGER-1_0_dev" *[line 34]*

leaf_EndorsementManager::LEAF_REG_KEY

= "endorse_man" *[line 32]*

leaf_EndorsementManager::\$allowEndorsement

array = array (
 "leaf_Locale", "leaf_Logger"
)
[line 57]

Array with the classes that are allowed to be overlapped.

- **Access** private

leaf_EndorsementManager::\$currentClass

string = NULL *[line 66]*

Current class that we will work on.

- **Access** private

leaf_EndorsementManager::\$endorsed

array = array() *[line 42]*

Currently classes that are overlapped.

- **Access** private

leaf_EndorsementManager::\$registeredEndorsed

array = array() [line 50]

Array with the classes that have been requested to be overlapped.

- **Access** private

Constructor *void* function leaf_EndorsementManager::__construct() *[line 74]*

Discovers the classes that are declared for overlapping.

- **Access** public

array function leaf_EndorsementManager::getEndorsedClasses() *[line 149]*

Returns all the classes, that are currently overlapping the internal classes.

- **Access** public

void function leaf_EndorsementManager::introspectEndorsedClass(\$className, \$fileName) *[line 138]*

Function Parameters:

- **\$className**
- **\$fileName**

Introspect the class` file.

- **TODO**
 1. Implement.
 2. Must check if the class, has declared the constants LEAF_REG_KEY and/or LEAF_CLASS_ID.
- **Access** private

boolean function leaf_EndorsementManager::isEndorsed(\$className) [*line 88*]
Function Parameters:

- **\$className**

Find out if the requested class is endorsed for overlapping.

- **Access** public

void function leaf_EndorsementManager::loadEndorsedClass(\$className) [*line 110*]
Function Parameters:

- **\$className**

Load the requested class.

- **TODO**
 1. Check again the registration function that the classes are using, as well as the "keys" that they use in the class

[leaf_Registry.](#)

- **Access** public

Class leaf_Exception

[line 25]

Custom exception class.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Exception.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://php.net/manual/en/language.exceptions.html>
- **TODO**
 1. Possible, internal logging function.

Constructor *void* function leaf_Exception::__construct(\$message, [\$code = 0]) [line 27]

Function Parameters:

- **\$message**
- **\$code**

- **Access** public

Class leaf_Hash

[line 25]

Handles internal hashing procedures, based on already existing native hash methods.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Hash.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Possible creation of an interface.

leaf_Hash::\$algorithms

array = array ("md5", "sha1") [line 39]

All supported hash algorithms.

- **Access** private

leaf_Hash::\$defaultAlgorithm

string = "md5" [line 32]

Default hash algorithm

- **Access** private

leaf_Hash::\$useAltMethod

boolean = FALSE [*line 50*]

Whether to use an alternative method or the classic hash functions.

The alternative method, is based on the "hash" function that is documented to be faster than the classic hash functions like "sha1" and "md5".

- **Access** private

Constructor *void* function leaf_Hash::__construct() [*line 58*]

Flags the method of hashing the we will be using.

- **Access** public

string function leaf_Hash::digestFile(\$filename, [\$raw = FALSE]) [*line 105*]

Function Parameters:

- *string* **\$filename**
- *boolean* **\$raw**

Returns (as text or as a byte sequence) the hash for the specific file`s contents.

- **Access** public

string function leaf_Hash::digestMsg(\$data, [\$raw = FALSE]) [*line 88*]

Function Parameters:

- *string* **\$data**
- *boolean* **\$raw**

Returns (as text or as a byte sequence) the hash for the specific give input.

- **Access** public

boolean function leaf_Hash::setAlgorithm(\$algorithm) [*line 70*]

Function Parameters:

- *string* **\$algorithm**

Sets the requested algorithm for usage.

- **Access** public

Class leaf_Input

[*line 24*]

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>

- **Version** \$Id: Input.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.

leaf_Input::LEAF_CLASS_ID

= "LEAF_INPUT-1_0_dev" [*line 28*]

leaf_Input::LEAF_REG_KEY

= "input" [*line 26*]

leaf_Input::\$input

array = array (
 "get" => array(), "post" => array()) [*line 37*]

Array with references to the super global variables "post" and "get".

- **Access** private

Constructor *void* function leaf_Input::__construct() [*line 42*]

- **Access** public

void function leaf_Input::offsetExists(\$offset) [*line 47*]

Function Parameters:

- **\$offset**

- **Access** public

void function leaf_Input::offsetGet(\$offset) [line 52]

Function Parameters:

- **\$offset**

- **Access** public

void function leaf_Input::offsetSet(\$offset, \$value) [line 57]

Function Parameters:

- **\$offset**

- **\$value**

- **Access** public

void function leaf_Input::offsetUnset(\$offset) [line 62]

Function Parameters:

- **\$offset**

- **Access** public

Class leaf_Loader

[line 23]

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Loader.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.

leaf_Loader::LEAF_CLASS_ID

= "LEAF_LOADER-1_0_dev" [line 27]

leaf_Loader::LEAF_REG_KEY

= "loader" [line 25]

Constructor *void* function leaf_Loader::__construct() [line 35]

- **Access** public

void function leaf_Loader::endorsed(\$class) [line 86]

Function Parameters:

- *string* **\$class**

- **Access** public

void function leaf_Loader::extension(\$ext, [\$namespace = NULL]) [line 75]

Function Parameters:

- *string* **\$ext**
- *string* **\$namespace**

- **Access public**

void function leaf_Loader::library(\$name) [*line 63*]

Function Parameters:

- *string* **\$name**

- **Access public**

void function leaf_Loader::model(\$modelName, [\$opts = NULL]) [*line 53*]

Function Parameters:

- *string* **\$modelName**
- *array* **\$opts**

- **Access public**

void function leaf_Loader::plugin(\$plugin) [*line 96*]

Function Parameters:

- *string* **\$plugin**

- **Access** public

void function leaf_Loader::__toString() [*line 41*]

- **Access** public

Class leaf_Model

[*line 23*]

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Model.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.
- **Abstract Element**

leaf_Model::LEAF_CLASS_ID

= "LEAF_MODEL-1_0_dev" [*line 27*]

leaf_Model::LEAF_REG_KEY

= "model" [*line 25*]

Constructor *void* function leaf_Model::__construct() [*line 35*]

- **Access** public

Class leaf_Registry

[line 32]

Registry class that lists most of the instantiated internal classes.

This class implements the [Registry design pattern](#).

When an internal class is instantiated, registers itself in this class.

This way, we provide a fast and easy way for the internal class to "communicate" each other.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Registry.php 46 2007-11-12 07:41:49Z makism \$
- **See** [leaf_Base](#)
- **TODO**
 1. Thinking about listing all possible entries with their types - so there are some sanity checks...?

leaf_Registry::\$instance

object leaf_Registry = NULL [line 49]

The unique instance of this class.

This class implements the [Singleton design pattern](#).

That is, there can be only one instance of this class.

- **Static**
- **Access** private

leaf_Registry::\$registered

array = array() [line 39]

All currently registered classes.

- **Access** private

Constructor *void* function leaf_Registry::__construct() *[line 57]*

Declared private in favor of the Singleton pattern.

- **Access** private

object leaf_Registry function leaf_Registry::getInstance() *[line 78]*

Returns the instance of this class.

- **Static**
- **Access** public

void function leaf_Registry::instance() *[line 67]*

If needed, instantiates an object of this class.

- **Static**
- **Access** public

boolean function leaf_Registry::isRegistered(\$key) [*line 156*]

Function Parameters:

- *string* **\$key**

Checks for the existence of the requested key.

- **Access** public

void function leaf_Registry::register(\$obj) [*line 127*]

Function Parameters:

- *object* **\$obj**

Registers an object.

When an object is passed to register, we suppose the constant LEAF_REG_KEY, exists and we use it as a reference key.

- **TODO**
 1. Perform some checks to find out if the object passed, has declared the constant LEAF_REG_KEY, and possible examine further it's behaviour.
- **Access** public

array function leaf_Registry::toArray() [line 170]

Returns an associative array with all the "general" configuration parameters.

- **Access** public

void function leaf_Registry::unregister(\$class) [line 145]

Function Parameters:

- *string* **\$class**

Removes a key from the registry.

- **TODO**
 1. Implement.
- **Access** public

mixed function leaf_Registry::__get(\$key) [line 90]

Function Parameters:

- *string* **\$key**

Return the request object, by refering to it`s instance name.

- **Access** public

`void function leaf_Registry::__set($key, $obj) [line 105]`

Function Parameters:

- *string* **\$key**
- *object* **\$obj**

Registers the requested instance using the designated key.

- **Access** private

Class leaf_Request

[line 31]

Provides access to all elements that compose the Uri.

This means, that we have request and refer to the file that the requested Controller is located in, the Action, the extra segments and finally the query string.

Also, this class implements some basic methods related with the Uri handling, like redirecting and Uri-reconstruction.

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Request.php 49 2007-11-13 21:15:18Z makism \$
- **See** [leaf_Router](#)
- **TODO**
 1. Possible implementation of [facade](#) functions.

leaf_Request::LEAF_CLASS_ID

= "LEAF_REQUEST-1_0_dev" [*line 35*]

leaf_Request::LEAF_REG_KEY

= "request" [*line 33*]

leaf_Request::\$action

string = NULL [*line 69*]

The requested method name (Action).

- **Access private**

leaf_Request::\$controller

string = NULL [*line 52*]

The requested class name (Controller), suffixed with "_Controller".

- **Access private**

leaf_Request::\$controllerFile

string = NULL [*line 62*]

The file name in which the requested Controller is located.

The complete file name will look something like this:

/var/www/http/applications/Blog/Blog_Controller.php

- **Access** private

leaf_Request::\$queryElems

array = NULL [line 78]

The query string found in the Uri.

For more info take a look at the class leaf_Router.

- **Access** private

leaf_Request::\$segments

array = NULL [line 45]

The extra segments found in the Uri.

For more info take a look at the class leaf_Router.

- **Access** private

Constructor *void* function leaf_Request::__construct() *[line 88]*

Internally uses the class [leaf_Router](#) in order to export information like, the Controller name, the Controller's name as is *must* be, the Action, etc.

- **Access** public

string function leaf_Request::getApplicationName() *[line 152]*

Returns the current Application's name.

- **Access** public

string function leaf_Request::getControllerFileName() [*line 142*]

Returns the file name that contains the current Controller.

- **Access** public

string function leaf_Request::getControllerName() [*line 132*]

ReturnstThe requested class name (Controller), suffixed with "_Controller".

- **Access** public

mixed function leaf_Request::getQueryString(\$offset) [*line 248*]

Function Parameters:

- *string* **\$offset**

Retrieves the value for the requested key.

- **TODO**
 1. Possible method refactor.
- **Access** public

string function leaf_Request::getQueryStringAsString() [*line 266*]

Return the current query string, as a string.

- **TODO**
 1. Possible method complete refactor.
- **Access** public

string function leaf_Request::reconstructUrl([\$className = NULL], [\$methodName = NULL], [\$segments = NULL], [\$queryString = NULL]) [*line 186*]

Function Parameters:

- *string* **\$className**
- *string* **\$methodName**
- *array* **\$segments**
- *array* **\$queryString**

Reconstructs a Uri, based on the data passed.

- **TODO**
 1. Implement.
- **Access** public

void function leaf_Request::redirect(\$target, [\$isExternal = FALSE]) [*line 168*]

Function Parameters:

- *string* **\$target**
- *boolean* **\$isExternal**

Performs a redirect to the specified Uri.

- **TODO**
 1. Implement.
- **Access** public

string/NULL function `leaf_Request::segment($n)` [line 219]

Function Parameters:

- *integer* **\$n**

Retrieves the requested (numeric) offset from the segments.

- **TODO**
 1. Implement.
- **Access** public

array function `leaf_Request::segmentsToArray()` [line 233]

Returns the segments` array.

- **TODO**
 1. Implement.
- **Access** public

integer function leaf_Request::totalSegments() [*line 204*]

Returns the total number of segments.

- **TODO**
 1. Implement.
- **Access** public

void function leaf_Request::__toString() [*line 282*]

- **Access** public

Class leaf_Response

[*line 24*]

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Response.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.

leaf_Response::LEAF_CLASS_ID

= "LEAF_RESPONSE-1_0_dev" [*line 28*]

leaf_Response::LEAF_REG_KEY

= "response" [*line 26*]

leaf_Response::\$buffer

string = NULL [*line 50*]

- **Access** private

leaf_Response::\$useGzip

boolean = FALSE [*line 43*]

- **Access** private

leaf_Response::\$useTidy

boolean = FALSE [*line 36*]

- **Access** private

Constructor *void* function leaf_Response::__construct() [*line 59*]

Class constructor

- **Access** public

void function leaf_Response::clearHeaders() [*line 123*]

- **Access** public

void function leaf_Response::flushOutputBuffer() [*line 184*]

- **Access** public

string function leaf_Response::getOutputBuffer([\$flush = FALSE]) [*line 171*]

Function Parameters:

- **\$flush**

Gets output buffer.

- **Access** public

void function leaf_Response::outputBufferingStart() [*line 145*]

Start output buffering

- **Access** public

void/string function leaf_Response::outputBufferingEnd([\$returnBuffer = FALSE]) [*line 160*]

Function Parameters:

- *boolean* **\$returnBuffer**

End output buffering

- **Access** public

void function leaf_Response::sendResponse() [line 134]

- **Access** public

void function leaf_Response::setHeader(\$name, \$value) [line 119]

Function Parameters:

- *string* **\$name**
- *string* **\$value**

- **Access** public

void function leaf_Response::__toString() [line 107]

- **Access** public

Class leaf_Router

[line 31]

Processes and filters the current Uri.

Filters the Uri for erroneous characters, splits the query string (if enabled), matches the virtual file extension (if enabled), discovers the requested Controller(class) and the

Action(method).

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Router.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Maybe this class should completely overtake the class [leaf_Request](#), thus removing the later class.
 2. Provide an alternative class (which maybe will inherit from this one) that will use static routing rules.

leaf_Router::LEAF_CLASS_ID

= "LEAF_ROUTER-1_0_dev" *[line 35]*

leaf_Router::LEAF_REG_KEY

= "router" *[line 33]*

leaf_Router::\$queryString

string = NULL *[line 83]*

The query string found in the Uri.

If enabled in the general configuration file, and if found in the Uri, the query string is separated and processed so it produces the array \$queryStringElements.

- **Access** private

leaf_Router::\$queryStringElements

array = NULL *[line 90]*

The query string found in the Uri, as an associative array.

- **Access** private

leaf_Router::\$requestClass

string = NULL [*line 50*]

The requested class name (Controller).

- **Access** private

leaf_Router::\$requestMethod

string = NULL [*line 57*]

The requested method name (Action).

- **Access** private

leaf_Router::\$requestUri

string = NULL [*line 43*]

The current requested Uri string.

- **Access** private

leaf_Router::\$segments

```
array = NULL [line 72]
```

The extra segments found in the Uri.

Whatever follows after the method name, and is separated by "/", is considered to be a segment.

Example:

<http://localhost/Blog/view/2007/xx/xx/aTitle/>

The array `$segments`, will be populated by the strings:

```
"2007", "xx", "xx", "aTitle"
```

- **Access** private

Constructor *void* function leaf_Router::__construct() [line 105]

[illegible]

- See [leaf_Config](#)
- See [leaf_Request](#)
- **TODO**
 1. Possible break-down of the jobs that the constructor executes.
 2. Recheck the "virtual file extension" function.
- **Access** public

```
void function leaf_Router::chopSegment(&$seg, $seg) [line 312]
```

Function Parameters:

- *string* \$seg
- &\$seg

Removes the first string occurrence until the very first `"/"`, from the property `$this->requestUri`.

- **Access** private

string function leaf_Router::getClassName() [*line 328*]

Returns the class name (Controller) that is requested.

- See [leaf_Request](#)
- **Access** public

string function leaf_Router::getMethodName() [*line 339*]

Returns the method name (Action) that is requested.

- See [leaf_Request](#)
- **Access** public

array|NULL function leaf_Router::queryStringElements() [*line 277*]

Returns an associative array filled with the elements that found in the query string.

- See [leaf_Request](#)
- **Access** public

array|NULL function leaf_Router::segments() [*line 300*]

Returns an array with the extra segments that compose this Uri.

- See [leaf_Request](#)
- **Access** public

integer function leaf_Router::segmentsSize() [*line 288*]

Returns the total number of segments that found in the current Uri.

- **Access** public

void function leaf_Router::__toString() [*line 344*]

- **Access** public

Class leaf_View

[*line 23*]

- **Package** leaf
- **Sub-Package** core
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: View.php 49 2007-11-13 21:15:18Z makism \$
- **TODO**
 1. Implement.

leaf_View::LEAF_CLASS_ID

= "LEAF_VIEW-1_0_dev" [*line 27*]

leaf_View::LEAF_REG_KEY

= "view" [*line 25*]

Constructor *void* function leaf_View::__construct() [*line 35*]

- **Access** public

void function leaf_View::render(\$view, [\$data = NULL]) [*line 62*]

Function Parameters:

- *string* **\$view**
- *array* **\$data**

Includes and renders a view file.

View files, are common php script files.

- **TODO**

1. When this method is called, the Output Buffer, will be also flushed and terminating. This will spawn the "include" method which will be similar except the Ob thingie.
2. Maybe this method and "include", will be implemeneted inside the overload method "__call".
3. Create some "sanity" checks to run against the view name that is requested.
4. It is an obvious bug that is the user declares a variable with the "viewFile", this will result in overlapping the local variable that holds the file in which the view is located.</i>

- **Access** public

void function leaf_View::__toString() [*line 107*]

- **Access** public

Benchmark.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** core-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Benchmark.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

Request.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** core-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Request.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

Hook.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Hook_Conditional.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Class leaf_Hook

[line 24]

- **Package** leaf
- **Sub-Package** core-hook
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Hook.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.
- **Abstract Element**

leaf_Hook::LEAF_CLASS_ID

= "LEAF_HOOK-1_0_dev" *[line 28]*

leaf_Hook::LEAF_REG_KEY

= "hook" *[line 26]*

Constructor *void* function leaf_Hook::__construct() *[line 36]*

- **Access** public

string function leaf_Hook::__toString() *[line 46]*

- **Abstract Element**
- **Access** public

Class leaf_Hook_Conditional

[line 24]

- **Package** leaf
- **Sub-Package** core-hook
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Hook_Conditional.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.
- **Abstract Element**

leaf_Hook_Conditional::LEAF_CLASS_ID

= "LEAF_HOOK_CONDITIONAL-1_0_dev" *[line 28]*

leaf_Hook_Conditional::LEAF_REG_KEY

= "conditional_hook" *[line 26]*

Constructor *void* function leaf_Hook_Conditional::__construct() [*line 36*]

- **Access** public

boolean function leaf_Hook_Conditional::condition() [*line 46*]

- **Abstract Element**
- **Access** public

Hooks.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** core-hook-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Hooks.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

array function getHooks([\$level = NULL], [\$controller = NULL], [\$method = NULL]) [*line* [103](#)]

Function Parameters:

- *string|NULL* \$level
- *string|NULL* \$controller
- *string|NULL* \$method

Returns all registered hooks for one -or all- level. Also, the hooks can be filtered using the parameters \$controller and \$method.

HOOK_POST_CONTROLLER_DISPATCH = 'post_controller_dispatch' [*line* [31](#)]

Level, Post Controller Dispatch

That is, after the controller has been executed and returned.

HOOK_POST_FRONT_CONTROLLER = 'post_front_controller' [*line* [38](#)]

Level, Post Front Controller

Almost, before the end of execution of the leaf framework.

`HOOK_PRE_CONTROLLER_DISPATCH = 'pre_controller_dispatch'` [[line 24](#)]

Level, Pre Controller Dispatch

That is, before the controller is dispatched.

`array function introspectHooks([$level = NULL])` [[line 52](#)]

Function Parameters:

- *string* **\$level**

Returns the registered hooks for the requested level.

- **TODO**
 1. Refactor.
- **Access** private

`void function runHook($controller, $method)` [[line 89](#)]

Function Parameters:

- *string* **\$controller**
- *string* **\$method**

Executes a specific hook.

- **Access** private

`void function runHooks($level)` [[line 76](#)]

Function Parameters:

- *string* **\$level**

Runs all registered hooks in one level.

- **Access** private

Log.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Logger.php

This source file is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Link** <http://leaf-framework.sourceforge.net>
- **License** [New BSD License](#)

Class leaf_Log

[line 24]

- **Package** leaf
- **Sub-Package** core-log
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Log.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Implement.

Class leaf_Logger

[line 25]

- **Package** leaf
- **Sub-Package** core-log
- **Method** void log(): log(string message) log(string level, string message) log(string level, string message, string fileName) log(string level, string message, string fileName, string className)
Handles to log a message.
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Logger.php 46 2007-11-12 07:41:49Z makism \$
- **TODO**
 1. Possible implementation for logging in a database.

leaf_Logger::LEAF_CLASS_ID

leaf_Logger::LEAF_REG_KEY

leaf_Logger::\$buffer

- **Access** private

- **Access** private

- **Access** private

leaf_Logger::\$levels

array = array("All", "Debug", "Warning", "Info", "None") [*line 65*]

- **Access** private

leaf_Logger::\$stampPat

string = NULL [*line 44*]

- **Access** private

leaf_Logger::\$threshold

int = NULL [*line 58*]

- **Access** private

Constructor *void* function leaf_Logger::__construct([\$buffer = false]) [*line 81*]

Function Parameters:

- *boolean* **\$buffer**

Begins the logging procedures, like file locking and filename scheming.

- **Access** public

void function leaf_Logger::end_flush() [*line 139*]

Flushes the internal buffer, and releases all locks on the log files.

- **Access** public

array|NULL function leaf_Logger::flush([\$return = false]) [*line 120*]

Function Parameters:

- *boolean* **\$return**

Flushes (empties) the internal buffer.

If the first parameter is given and is set the boolean "true", then the method will return the buffer's contents.

- **Access** public

boolean function leaf_Logger::setLevel(\$filterLevel) [*line 164*]

Function Parameters:

- *string* **\$filterLevel**

Set the level of errors to capture.

- **Access** public

void function leaf_Logger::setTimeStampPattern(\$pat) [line 153]

Function Parameters:

- *string* **\$pat**

Set a timestamp pattern.

- **Access** public

void function leaf_Logger::__call(\$method, \$args) [line 182]

Function Parameters:

- *string* **\$method**
- *array* **\$args**

__call

- **Access** public

Debug.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** front
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Debug.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **TODO**
 1. Document.
 2. Recheck the information displayed under the tab "Config Settings".
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

Front_Controller.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** front
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Front_Controller.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **TODO**
 1. Add functionality for the 'hooks' subsystem.
 2. Add functionality for the benchmarking subsystem.
 3. Completion of the documentation.
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

`require_once LEAF_BASE.'core/hook/helpers/Hooks.php'` [*line 63*]

Helper functions handling hooks.

- See [leaf_Hook_Conditional](#)
- See [leaf_Hook](#)

`require_once LEAF_BASE.'front/Debug.php'` [*line 167*]

`require_once LEAF_BASE.'front/helpers/Debug.php'` [*line 54*]

Custom debug functions.

`require_once LEAF_BASE.'front/helpers/Error.php'` [[line 48](#)]

Functions used to present errors.

`require_once LEAF_BASE.'front/helpers/Dependancies.php'` [[line 36](#)]

Helper functions that are used to declare dependancies on specific extensions, functions or leaf's Classes.

`require_once LEAF_BASE.'front/helpers/Handlers.php'` [[line 42](#)]

Custom error and exception handling functions.

`require_once LEAF_BASE.'front/helpers/Main.php'` [[line 29](#)]

Helper functions needed to start-up the framework and to load its classes.

Debug.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** front-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Debug.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **TODO**
 1. Document.
 2. Possible removal.
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

void function inspect_var(\$var, [\$depth = NULL]) [line 49]

Function Parameters:

- *mixed* **\$var**
- *boolean|integer* **\$depth**

void function i_var(\$var, [\$depth = NULL]) [line 37]

Function Parameters:

- *mixed* **\$var**
- *boolean|integer* **\$depth**

mixed function _inspect_var_Array(\$arr, [\$recursionLevel = 5]) [*line* [172](#)]

Function Parameters:

- *array* **\$arr**
- *integer* **\$recursionLevel**

- **Access** private

string function _inspect_var_Boolean(\$flag) [*line* [155](#)]

Function Parameters:

- *boolean* **\$flag**

- **Access** private

void function _inspect_var_Generic(\$var) [*line* [111](#)]

Function Parameters:

- *mixed* **\$var**

- **Access** private

void function _inspect_var_Number(\$num) [*line* [123](#)]

Function Parameters:

- *integer|double* **\$num**

- **Access** private

void function _inspect_var_Object(\$obj, [\$inDepth = FALSE]) [*line* [186](#)]

Function Parameters:

- *object* **\$obj**
- *boolean* **\$inDepth**

- **Access** private

string function _inspect_var_Object_getSubclasses(\$obj) [*line* [332](#)]

Function Parameters:

- *object* **\$obj**

- **Access** private

void function _inspect_var_Resource(\$var) [*line* [94](#)]

Function Parameters:

- *resource* **\$var**

- **Access** private

`void function _inspect_var_String($str) [line 138]`

Function Parameters:

- *string* **\$str**
- **Access** private

Dependancies.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** front-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Dependancies.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **TODO**
 1. Implement a function that will handle dependancies as optional.
 2. Recheck **all** functions.
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

boolean function dependsOn(\$deps) [*line* [31](#)]

Function Parameters:

- *string|array* **\$deps**

Handles dependacies on PHP extensions.

boolean function dependsOnFunc(\$func, \$funcs) [*line* [74](#)]

Function Parameters:

- *string* **\$funcs**
- **\$func**

Handles dependencies on specific functions.

boolean function dependsOptionalOn(\$deps) [[line 63](#)]

Function Parameters:

- *string* \$deps

Handles optional dependencies.

- **TODO**
 1. Implementd.

Error.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** front-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Error.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **TODO**
 1. Implement a function that will handle dependancies as optional.
 2. Recheck **all** functions.
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

void function showErrorPage404([\$str = NULL]) [*line* [35](#)]

Function Parameters:

- *string* \$str

Presents a "Page Not Found" message.

- **TODO**
 1. Implement.

void function showHtmlMessage(\$title, [\$str = NULL], [\$die = FALSE]) [*line* [48](#)]

Function Parameters:

- *string* **\$title**
- *string* **\$str**
- *boolean* **\$die**

Prints styled text. Used in printing debug messages and errors.

Handlers.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** front-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Handlers.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **TODO**
 1. Implement a function that will handle dependancies as optional.
 2. Recheck **all** functions.
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

void function errorHandler(\$errno, \$errstr, \$errfile, \$errline) [*line* [34](#)]

Function Parameters:

- *integer* **\$errno**
- *string* **\$errstr**
- *string* **\$errfile**
- *integer* **\$errline**

Custom error handler.

void function exceptionHandler(\$ex) [*line* [69](#)]

Function Parameters:

- *object Exception* **\$ex**

Custom exception handler.

Main.php

This source file is part of the leaf framework and is licensed under the New BSD license.

For the full copyright and license information, please view the LICENSE file that was distributed with this source code.

- **Package** leaf
- **Sub-Package** front-helpers
- **Author** Avraam Marimpis < makism@users.sf.net>
- **Version** \$Id: Main.php 46 2007-11-12 07:41:49Z makism \$
- **Link** <http://leaf-framework.sourceforge.net>
- **Filesource** [Source Code for this file](#)
- **License** [New BSD License](#)

void function frameworkForceTerminate() [*line 123*]

Terminates the execution of the framework.

void function frameworkSanityCheck([\$checkAll = false]) [*line 135*]

Function Parameters:

- *boolean* **\$checkAll**

Performs some basic tests to check whether the framework is usable or not.

void function __autoload(\$className) [*line 32*]

Function Parameters:

- *string* **\$className**

Includes all necessary files related to leaf's classes.

We use the "magic" function `__autoload` in order to have leaf's classes

loaded automatically.

Also, leaf_EndorsedManager is instantiated if needed.

- **See** leaf_EndorsedManager
- **Link** <http://www.php.net/manual/en/language.oop5.autoload.html>

Appendices

Appendix A - Class Trees

Package leaf

leaf_Base

- [leaf_Base](#)
 - [leaf_Benchmark](#)
 - [leaf_Config](#)
 - [leaf_Controller](#)
 - [leaf_Dispatcher](#)
 - [leaf_EndorsementManager](#)
 - [leaf_Input](#)
 - [leaf_Loader](#)
 - [leaf_Hook](#)
 - [leaf_Hook_Conditional](#)
- [leaf_Logger](#)
- [leaf_Model](#)
- [leaf_Request](#)
- [leaf_Response](#)
- [leaf_Router](#)
- [leaf_View](#)

leaf_Exception

- Exception
 - [leaf_Exception](#)

leaf_Hash

- [leaf_Hash](#)

leaf_Log

- [leaf_Log](#)

leaf_Registry

- [leaf_Registry](#)

Appendix B - README/CHANGELOG/INSTALL

TODO

LICENSE

Copyright (c) 2007, Avraam Marimpis
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of leaf nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
4. Products derived from this software may not be called "leaf" nor may "leaf" appear in their names without specific prior written permission from the author.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

README

leaf framework

- * Supported PHP versions, 5.1 - or newer.
- * This is a testing release, which means that.
 - 1) The API is not final, and maybe change.
 - &
 - 2) Since this is a test release, it is not wise to be used in production.

CHANGELOG

INSTALL

Appendix C - Source Code

Package leaf

File Source for Benchmark.php

Documentation for this file is available at [Benchmark.php](#)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license    http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link      http://leaf-framework.sourceforge.net
10  *
11  * @package    leaf
12  * @subpackage core.helpers
13  * @author     Avraam Marimpis <makism@users.sf.net>
14  * @version    $Id: Benchmark.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  */
17
18
19  ?>
```

File Source for Request.php

Documentation for this file is available at [Request.php](#)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    core.helpers
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Request.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  */
17
18
19  ?>
```

File Source for Hooks.php

Documentation for this file is available at [Hooks.php](http://leaf-framework.sourceforge.net/LICENCE/)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENCE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    core.hook.helpers
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Hooks.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  */
17
18  /**
19   * Level, Pre Controller Dispatch
20   *
21   * That is, before the controller is dispatched.
22   */
23  define('HOOK_PRE_CONTROLLER_DISPATCH', 'pre_controller_dispatch');
24
25  /**
26   * Level, Post Controller Dispatch
27   *
28   * That is, after the controller has been executed and returned.
29   */
30  define('HOOK_POST_CONTROLLER_DISPATCH', 'post_controller_dispatch');
31
32  /**
33   * Level, Post Front Controller
34   *
35   * Almost, before the end of execution of the leaf framework.
36   */
37  define('HOOK_POST_FRONT_CONTROLLER', 'post_front_controller');
38
39  /**
40   * Returns the registered hooks for the requested level.
41   *
42   * @access private
43   * @param string $level
44   * @return array
45   * @todo
46   * <ol>
47   * <li>Refactor.</li>
48   * </ol>
49   */
50  function introspectHooks($level=NULL)
51  {
52      if ($level!=NULL)
53      {
54          if (!defined('HOOK_' . strtoupper($level)))
55              throw new leaf_Exception("Unknown Hook Level!");
56
57          static $allHooks;
58
59          if ($allHooks==NULL)
60              $allHooks = leaf_Registry::getInstance()-> config-> hooks;
61
62          if ($level!=NULL)
63              return $allHooks[$level];
64          else
65              return $allHooks;
66      }
67  }
```



```

68
69 /**
70  * Runs all registered hooks in one level.
71  *
72  * @access private
73  * @param string $level
74  * @return void
75  */
76 function runHooks($level)
77 {
78     $hooks = introspectHooks($level);
79 }
80
81 /**
82  * Executes a specific hook.
83  *
84  * @access private
85  * @param string $controller
86  * @param string $method
87  * @return void
88  */
89 function runHook($controller, $method)
90 {
91 }
92
93
94 /**
95  * Returns all registered hooks for one -or all- level. Also, the hooks
96  * can be filtered using the parameters $controller and $method.
97  *
98  * @param string|NULL $level
99  * @param string|NULL $controller
100  * @param string|NULL $method
101  * @return array
102  */
103 function getHooks($level=NULL, $controller=NULL, $method=NULL)
104 {
105 }
106 }
107
108 ?>

```

File Source for Debug.php

Documentation for this file is available at [Debug.php](#)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    front
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Debug.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  * @todo
17  * <ol>
18  *   <li>Document.</li>
19  *   <li>Recheck the information displayed under the tab
20  *     "Config Settings".</li>
21  * </ol>
22  */
23
24
25  /*
26  * The statistics will be produced only if leaf's version
27  * is tagged as "dev".
28  * This is most surely that will change.
29  */
30  if (LEAF_REL_STATUS=='DEV') {
31      dependsOn('xdebug');
32      dependsOnFunc('memory_get_usage');
33
34      $parseTime      = xdebug_time_index();
35      $memoryUsage     = memory_get_usage();
36      $memoryUsageKbs = sprintf("%.2fkbs", $memoryUsage/1024);
37      $baseUrl         = $reg->config['base_url'];
38
39      echo "<br /><br /><br /><br /><br />";
40
41      echo <<<DEBUG_STYLES
42      < link rel="stylesheet" type="text/css"
43      href="{ $baseUrl}yui/tabview.css">
44      < link rel="stylesheet" type="text/css"
45      href="{ $baseUrl}yui/border_tabs.css">
46      < script type="text/javascript"
47      src="{ $baseUrl}yui/yahoo.js"></script>
48      < script type="text/javascript"
49      src="{ $baseUrl}yui/event.js"></script>
50      < script type="text/javascript" src="{ $baseUrl}yui/dom.js"></script>
51      < script type="text/javascript" src="{ $baseUrl}yui/element-
52      beta.js"></script>
53      < script type="text/javascript"
54      src="{ $baseUrl}yui/tabview.js"></script>
55      < style type="text/css">
56      #leaf_debugDiv .yui-content { padding:1em; }
57      .yui-nav li a:visited { color: #0000FF; }
58      </style>
59      < script type="text/javascript">
60      YAHOO.example.init = function() {
61      var leafDebug = new YAHOO.widget.TabView('leaf_debugDiv');
```

```

62      * Start Main Div
63      */
64      echo
65      "<div style=\"width: 600px;\">"
66      "<div id=\"leaf_debugDiv\" class=\"yui-navset yui-navset-
left\">"
67      "<ul class=\"yui-nav\">"
68      ;
69      // tabs
70      echo
71      "<li title=\"active\" class=\"selected\"><a
href=\"#leaf_Debug_leaf_Statistics\"><em>leaf
Statistics</em></a></li>"
72      "<li><a href=\"#leaf_Debug_Global_Settings\"><em>Global
Settings</em></a></li>"
73      "<li><a
href=\"#leaf_Degug_Registry\"><em>Registry</em></a></li>"
74      "<li><a href=\"#leaf_Debug_Request\"><em>Request
Information</em></a></li>"
75      ;
76      if ($reg-> config['allow_hooks']=="Yes" ) {
77          echo
78          "<li><a
href=\"#leaf_Debug_Hooks\"><em>Hooks</em></a></li>"
79          ;
80      }
81      if ($reg-> config['log_level']!="None" ) {
82          echo
83          "<li><a href=\"#leaf_Debug_Log_Buffer\"><em>Log
Buffer</em></a></li>"
84          ;
85      }
86      if ($reg-> config['allow_endorsed']=="Yes" ) {
87          echo
88          "<li><a href=\"#leaf_Debug_Endorsed\"><em>Endorsed
Classes</em></a></li>"
89          ;
90      }
91      // tabs (end)
92      echo "</ul>"
93      ;
94
95      /*
96      * Data
97      */
98      echo
99      "<div class=\"yui-content\">"
100      ;
101      //
102      // Status
103      //
104      echo
105      "<div id=\"leaf_Debug_leaf_Statistics\" style=\"display:
block;\">"
106      . " <pre style=\"font-size: 12px; font-family: Verdana, Arial, helvetica,
sans-serif;\">"
107      . " <span style=\"color: #4e9a06;\"><b>leaf
Version</b></span>\n"
108      . " <span style=\"color: #ff0000;\"> " . LEAF_REL_VERSION .
" " . LEAF_REL_STATUS . "\n"
109      . " <span style=\"color: #4e9a06;\"><b>Script memory
usage</b></span>\n"
110      . " <span style=\"color: #ff0000;\"> { $memoryUsageKbs}
< small>< i>({ $memoryUsage} bytes)</ i></ small></ span>\n "
111      . " <span style=\"color: #4e9a06;\"><b>Total parsing
time</b></span>\n"
112      . " <span style=\"color: #ff0000;\"> { $parseTime}
seconds</ span> "
113      . " </pre>"
114      . "</div>"
115      ;
116      //
117      // Settings
118      //
119      echo
120      "<div id=\"leaf_Debug_Global_Settings\" style=\"display:
none;\">"
121      . " <pre style=\"font-size: 12px; font-family: Verdana, Arial, helvetica,
sans-serif;\">"
122      ;

```

```

123         echo
124         "<fieldset style=\"border: 0px solid
#ffffff;\"><legend><small><b>General</b></small></legend>&quot;
;
125         foreach($reg->    config->    getByHashKey("general"                ) as $Var =>    $Val) {
126             echo
127             "        < span style=\"        color:
#4e9a06;\"><
b>{    $Var}</ b></ span>\n        "
128             . "        < span style=\"        color:
#ff0000;\"><
small>{    $Val}</ small></ span>\n        "        ;
129         }
130         echo "</fieldset>"
131         ;
132         echo "<br/>"
133         ;
134         echo
135         "<fieldset style=\"border: 0px solid
#ffffff;\"><legend><small><b>Autoload</b></small></legend>&quot;
t;
136         foreach($reg->    config->    getByHashKey("autoload"                ) as $Var =>    $Val) {
137             echo
138             "        < span style=\"        color:
#4e9a06;\"><
b>{    $Var}</ b></ span>\n        "
139             . "        < span style=\"        color:
#ff0000;\"><
small>{    $Val}</ small></ span>\n        "        ;
140         }
141         echo "</fieldset>"
142         ;
143         echo
144         "<fieldset style=\"border: 0px solid
#ffffff;\"><legend><small><b>Endorsed</b></small></legend>&quot;
t;
145         echo "</fieldset>"
146         ;
147         echo "<br/>"
148         ;
149         echo
150         "<fieldset style=\"border: 0px solid
#ffffff;\"><legend><small><b>Hooks</b></small></legend>"
151         echo "</fieldset>"
152         ;
153         echo
154         " </pre>"
155         . "</div>"
156         ;
157         //
158         // Registry
159         //
160         echo
161         "<div id=\"leaf_Degug_Registry\" style=\"display:
none;\">"
162         . " <pre style=\"font-size: 12px; font-family: Verdana, Arial, helvetica,
sans-serif;\">"
163         ;
164         foreach($reg->    toArray() as $InstanceName =>    $ClassType)
165             echo
166             "        key: < span style=\"        color:
#4e9a06;\"><
b>{    $InstanceName}</ b></ span>\n        "
167             . "        type: < span style=\"        color:
#ff0000;\"><
i>{    $ClassType}</ i></ span>\n\n        "        ;
168
169         echo
170         " </pre>"
171         . "</div>"
172         ;
173         //
174         // Request
175         //
176         $className = $reg->    router->    getClassName();
177         $methodName = $reg->    router->    getMethodName();
178         $segments = implode(" " , (array)$reg->    router->    segments());
179         $queryString= $reg->    request->    getQueryStringAsString();
180
181         echo
182         "<div id=\"leaf_Debug_Request\" style=\"display:
none;\">"
183         . " <pre style=\"font-size: 12px; font-family: Verdana, Arial, helvetica,
sans-serif;\">"
184         . " <span style=\"color: #4e9a06;\"><b>Controller
(Class)</b></span>\n"

```

```

185         . "          < span style=\"          color: #ff0000;\">
186         { $className }</ span>\n          " <span style=\"color: #4e9a06;\"><b>Action
187         (Method)</b></span>\n
188         . "          < span style=\"          color: #ff0000;\">
189         { $methodName }</ span>\n          " <span style=\"color: #4e9a06;\"><b>Extra
190         segments</b></span>\n
191         . "          < span style=\"          color: #ff0000;\">
192         { $segments }</ span>\n          " <span style=\"color: #4e9a06;\"><b>Query
193         String</b></span>\n
194         . "          < span style=\"          color: #ff0000;\">
195         { $queryString }</ span>
196         . " </pre>"
197         . "</div>"
198         ;
199
200         //
201         // Hooks
202         //
203         if ( $reg-> config['allow_hooks']=="Yes" ) {
204             echo
205             "<div id=\"leaf_Debug_Hooks\" style=\"display:
206             none;\">"
207             . " </div>"
208             ;
209         }
210
211         //
212         // Log Buffer
213         //
214         if ( $reg-> config['log_level']!="None" ) {
215             echo
216             "<div id=\"leaf_Debug_Log_Buffer\" style=\"display:
217             none;\">"
218             . " </div>"
219             ;
220         }
221
222         //
223         // Endorsed
224         //
225         if ( $reg-> config['allow_endorsed']=="Yes" ) {
226             echo
227             "<div id=\"leaf_Debug_Endorsed\" style=\"display:
228             block;\">"
229             . " <pre style=\"font-size: 14px; font-family: Verdana, Arial,
230             helvetica, sans-serif;\">"
231             ;
232             foreach( $reg-> endorse_man-> getEndorsedClasses() as $Key => $Value )
233             {
234                 $endorsedClass = key($Value);
235                 $classObject = $Value[$endorsedClass]['reg_key'];
236                 echo
237                 "<span style=\"color: #4e9a06;\">"
238                 . "          < b>{ $endorsedClass }</ b>\n
239                 . "          < small style=\"          color: #b91f49;\">{
240                 > $classObject }</ small>
241                 . "          </span>\n\n
242                 "
243                 ;
244             }
245             echo
246             " </pre>"
247             . " </div>"
248             ;
249         }
250
251         /*
252         * End Main Div
253         */
254         echo
255         "</div>"
256         . " </div>"
257         ;
258     }
259 }
260 ?>

```

File Source for Front_Controller.php

Documentation for this file is available at [Front_Controller.php](#)

```
1  i»¿<?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    front
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Front_Controller.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  * @todo
17  * <ol>
18  * <li>Add functionality for the 'hooks' subsystem.</li>
19  * <li>Add functionality for the benchmarking subsystem.</li>
20  * <li>Completion of the documentation.</li>
21  * </ol>
22  */
23
24
25  /**
26   * Helper functions needed to start-up the framework
27   * and to load its classes.
28   */
29  require_once LEAF_BASE . 'front/helpers/Main.php';
30
31
32  /**
33   * Helper functions that are used to declare dependancies
34   * on specific extensions, functions or leaf`s Classes.
35   */
36  require_once LEAF_BASE . 'front/helpers/Dependancies.php';
37
38
39  /**
40   * Custom error and exception handling functions.
41   */
42  require_once LEAF_BASE . 'front/helpers/Handlers.php';
43
44
45  /**
46   * Functions used to present errors.
47   */
48  require_once LEAF_BASE . 'front/helpers/Error.php';
49
50
51  /**
52   * Custom debug functions.
53   */
54  require_once LEAF_BASE . 'front/helpers/Debug.php';
55
56
57  /**
58   * Helper functions handling hooks.
59   *
60   * @see leaf_Hook
61   * @see leaf_Hook_Conditional
62   */
63  require_once LEAF_BASE . 'core/hook/helpers/Hooks.php';
64
65
66  /**
67   * Perform some basic checks to determine whether or not
```

```

68      * the system's sanity (and thus, all will be run normal).
69      */
70      #frameworkSanityCheck();
71
72
73      /*
74      * Startup leaf's Registry and register some basic objects.
75      */
76      $reg = leaf_Registry::getInstance();
77      $reg-> register(new leaf_Config());
78      $reg-> register(new leaf_Locale());
79      #$reg->register(new leaf_Benchmark());
80
81
82      /*
83      * Configure timezone.
84      */
85      $timezoneSetting = (empty($reg-> config['timezone']))
86          ? @date_default_timezone_get()
87          : $reg-> config['timezone'];
88
89      date_default_timezone_set($timezoneSetting);
90
91
92      /*
93      * Handle errors and exceptions.
94      */
95      #set_error_handler("errorHandler");
96      #set_exception_handler("exceptionHandler");
97
98
99      /*
100     * Register the logger if it has been enabled.
101     */
102     #if ($reg->config['log_level']!="None" ||
103     #     empty($reg->config['log_level'])) {
104     #     $reg->register(new leaf_Logger());
105     #}
106
107
108     /*
109     * Register the "base" classes that are needed
110     * for leaf to work properly.
111     */
112     $reg-> register(new leaf_Router());
113     $reg-> register(new leaf_Request());
114     $reg-> register(new leaf_Loader());
115     $reg-> register(new leaf_Dispatcher());
116     $reg-> register(new leaf_Response());
117
118
119     /*
120     * Begin output buffering.
121     */
122     $reg-> response-> outputBufferingStart();
123
124
125     /*
126     * Run all hooks for level: Pre-Controller-Dispatch
127     */
128     #runHooks(HOOK_PRE_CONTROLLER_DISPATCH);
129
130
131     /*
132     * Dispatch controller.
133     */
134     $reg-> dispatcher-> dispatchController();
135
136
137     /*
138     * Run all hooks for level: Post-Controller-Dispatch
139     */
140     #runHooks(HOOK_POST_CONTROLLER_DISPATCH);
141
142
143     /*
144     * Flush buffer, and turn off.
145     */
146     $reg-> response-> outputBufferingEnd();
147

```

```

148
149  /*
150   * Close the logger if it has been
151   * requested and registered.
152   */
153  #if ($reg->isRegistered("logger"))
154  #   $reg->logger->end_flush();
155
156
157  /*
158   * Run all hooks for level: Post-Front-Controller
159   */
160  #runHooks(HOOK_POST_FRONT_CONTROLLER);
161
162
163  /*
164   * General statistics like memory usage, parsing time and other.
165   */
166  if ($reg->    config['enable_debug_stats']=="Yes"          )
167      require_once LEAF_BASE . 'front/Debug.php';
168
169
170  ?>

```


File Source for Debug.php

Documentation for this file is available at [Debug.php](#)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    front.helpers
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Debug.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  * @todo
17  * <ol>
18  *   <li>Document.</li>
19  *   <li>Possible removal.</li>
20  * </ol>
21  */
22
23
24  /*****
25   * It is suggested to used XDebug`s functions since they are *
26   * natively implemented thus much faster. *
27   *****/
28
29
30  /**
31   *
32   *
33   * @param mixed $var
34   * @param boolean|integer $depth
35   * @return void
36   */
37  function i_var($var, $depth=NULL)
38  {
39      inspect_var($var, $depth);
40  }
41
42  /**
43   *
44   *
45   * @param mixed $var
46   * @param boolean|integer $depth
47   * @return void
48   */
49  function inspect_var($var, $depth=NULL)
50  {
51      $varType = gettype($var);
52
53      echo "<div>"
54          . "<pre style=\"font-family: Arial, sans; font-size: 14px;\">"
55
56      switch ($varType) {
57          case "integer" :
58          case "int" :
59          case "double" :
60          case "float" :
61              inspect_var_Number($var);
62              break;
63          case "string" :
64              inspect_var_String($var);
65              break;
66          case "object" :
67              inspect_var_Object($var, (boolean)$depth);
```

```

68         break;
69         case "array" :
70             _inspect_var_Array($var, (integer)$depth);
71         break;
72         case "resource" :
73             _inspect_var_Resource($var);
74         break;
75         case "boolean" :
76             _inspect_var_Boolean($var);
77         break;
78         default:
79             _inspect_var_Generic($var);
80         break;
81     }
82
83     echo "</pre>"
84     . "</div>"
85 }
86
87 /**
88  *
89  *
90  * @access private
91  * @param resource $var
92  * @return void
93  */
94 function _inspect_var_Resource($var)
95 {
96     $resourceName = get_resource_type($var);
97
98     echo
99     "<span style=\"font-size: 11px; color:
100 #56a0ee\">(resource)</span> "
101     . "<span style=\"color: #f06f00;\">{ $resourceName}</span> "
102     . "<span style=\"color: #3f6b5b; font-style: italic; font-size:
103 12px;\">({ $var}</span> "
104     . ";
105 }
106
107 /**
108  *
109  *
110  * @access private
111  * @param mixed
112  * @return void
113  */
114 function _inspect_var_Generic($var)
115 {
116 }
117
118 /**
119  *
120  *
121  * @access private
122  * @param integer/double
123  * @return void
124  */
125 function _inspect_var_Number($num)
126 {
127     $varType = gettype($num);
128     echo
129     "<span style=\"font-size: 11px; color:
130 #000000\">({ $varType}</span> "
131     . "<span style=\"color: #ff0000;\">{ $num}</span> "
132     . ";
133 }
134
135 /**
136  *
137  *
138  * @access private
139  * @param string
140  * @return void
141  */
142 function _inspect_var_String($str)
143 {
144     $len = strlen($str);
145
146     echo
147     "<span style=\"font-size: 11px; color: #4e9a06\">(string)</span>

```

```

144         "        < span style=\"          color: #0000ff;\">{'$str}'</ span>      "
145         "        < span style=\"          font-style: italic; font-size:
12px;\">(<          length={ $len }</ span>      " ;
146     }
147
148 /**
149  *
150  *
151  * @access private
152  * @param boolean $flag
153  * @return string
154  */
155 function _inspect_var_Boolean($flag)
156 {
157     $str = ($flag===TRUE) ? "TRUE"          : "FALSE"          ;
158
159     echo
160         "<span style=\"font-size: 11px; color:
161 #999999\">(boolean)</span> "
162         "        < span style=\"          color: #75507b; font-weight:
163 bold;\">{'$str}'</ span>      " ;
164 }
165
166 /**
167  *
168  *
169  * @access private
170  * @param array $arr
171  * @param integer $recursionLevel
172  * @return mixed
173  */
174 function _inspect_var_Array(array $arr, $recursionLevel=5)
175 {
176 }
177
178 /**
179  *
180  *
181  * @access private
182  * @param object $obj
183  * @param boolean $inDepth
184  * @return void
185  */
186 function _inspect_var_Object($obj, $inDepth=FALSE)
187 {
188     dependsOn('Reflection');
189
190     $reflect = new ReflectionClass($obj);
191
192     /*
193     *
194     */
195
196     $className = get_class($obj);
197     $parentName = get_parent_class($obj);
198     $subClasses = NULL;
199     $interfaces = NULL;
200     $interfacesStr = NULL;
201     $isUserDefined = $reflect-> isUserDefined();
202     $properties = $reflect-> getProperties();
203     $constants = $reflect-> getConstants();
204     $statics = $reflect-> getStaticProperties();
205
206
207     if ($parentName)
208         $subClasses = _inspect_var_Object_getSubclasses($parentName);
209
210     $interfaces = $reflect-> getInterfaces();
211     if (sizeof($interfaces)) {
212         foreach ($interfaces as $interface)
213             $interfacesStr .= $interface-> getName() . ", " ;
214
215         $interfacesStr = substr($interfacesStr, 0, strlen($interfacesStr)-2);
216     }
217

```

```

218      /*
219      *
220      *
221      */
222      if ($inDepth==TRUE)
223      ;
224
225
226      /*
227      * Class name.
228      */
229      echo
230      "<fieldset style=\"border: 1px solid #c5c5c5;\">"
231      "      < legend> < i> object</ i> < b>{ $className}</ b>
232      </ legend> "
233      "<div style=\"padding: 10px;\">"
234      ;
235
236      /*
237      * Show information.
238      */
239      echo
240      "<div style=\"color: #3f6b5b; float: left; width: 150px; overflow:
241      hidden;\">Defined by</div>"
242      ;
243      echo
244      "<span style=\"font-weight: bold; font-style: italic; color:
245      #f06f00;\">"
246      ((($isUserDefined==TRUE) ? "User" : "System" ) .
247      "</span>"
248      ;
249      echo "<hr />"
250      ;
251
252      echo
253      "<div style=\"color: #3f6b5b; float: left; width: 150px; overflow:
254      hidden;\">Is abstract</div>"
255      ;
256      echo
257      "<span style=\"font-weight: bold; font-style: italic; color:
258      #f06f00;\">"
259      ((($reflect-> isAbstract()==TRUE) ? "Yes" : "No" ) .
260      "</span>"
261      ;
262      echo "<hr />"
263      ;
264
265      echo
266      "<div style=\"color: #3f6b5b; float: left; width: 150px; overflow:
267      hidden;\">Is interface</div>"
268      ;
269      echo
270      "<span style=\"font-weight: bold; font-style: italic; color:
271      #f06f00;\">"
272      ((($reflect-> isInterface()==TRUE) ? "Yes" : "No" ) .
273      "</span>"
274      ;
275      echo "<hr />"
276      ;
277
278      /*
279      * Class filename.
280      */
281      echo
282      "<div style=\"color: #3f6b5b; float: left; width: 150px; overflow:
283      hidden;\">Filename</div>"
284      ;
285      echo
286      "<span style=\"font-weight: bold; font-style: italic; color:
287      #f06f00;\">"
288      $reflect-> getFileName() .
289      "</span>"
290      ;
291      echo "<hr />"
292      ;
293
294      /*
295      * The name of the classes this object extends.
296      */
297      echo
298      "<div style=\"color: #3f6b5b; float: left; width: 150px; overflow:
299      hidden;\">Extends</div>"
300      ;
301      echo
302      "<span style=\"font-weight: bold; font-style: italic; color:
303      #f06f00;\">"
304      (($parentName!=NULL) ? $subClasses : "none" ) .
305      "</span>"
306      ;
307      echo "<hr />"
308      ;
309
310      /*
311      * The list of the interfaces of class implements or inherits.
312      */

```

```

287     echo
288     "<div style=\"color: #3f6b5b; float: left; width: 150px; overflow:
hidden;\">>Implements</div>"
289     echo
290     "<span style=\"font-weight: bold; font-style: italic; color:
#f06f00;\">>"
291     ((($interfacesStr!=NULL) ? $interfacesStr : "none"
292     "</span>"
293
294
295     echo "<br /><br /><br />"
296
297     /*
298     * Properties dumping.
299     */
300     echo "<ul><b>Properties</b>"
301     foreach ($properties as $prop) {
302         $status = ($prop-> isPublic())
303         ? "public"
304         : (($prop-> isPrivate()) ? "private" : "protected"
305         );
306
307         $value = "
308         if ($status=="public"
309         $value = " (value={$prop-> getValue($obj)})"
310
311     echo
312         "<li>"
313         " <span style=\" color: #4e9a06; font-size:
12px;\">{ $status}</span> "
314         " <span style=\" color: #56a0ee; font-style: italic;\">>{ $prop-
> getName()}</span> "
315         "{ $value}"
316         "</li>"
317     }
318     echo "</ul>"
319
320     echo
321     "</div>"
322     "</legend>"
323 }
324
325 /**
326 *
327 *
328 * @access private
329 * @param object $obj
330 * @return string
331 */
332 function _inspect_var_Object_getSubclasses($obj)
333 {
334     $parent = get_parent_class($obj);
335
336     if ($parent!=NULL)
337         return $obj . ", " . _inspect_var_Object_getSubClasses($parent);
338     else
339         return $obj;
340 }
341
342 ?>

```

File Source for Dependancies.php

Documentation for this file is available at [Dependancies.php](http://www.phpdoc.org)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    front.helpers
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Dependancies.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  * @todo
17  * <ol>
18  * <li>Implement a function that will handle dependancies as
19  * optional.</li>
20  * <li>Recheck <b>all</b> functions.</li>
21  * </ol>
22  */
23
24
25 /**
26  * Handles dependencies on PHP extensions.
27  *
28  * @param  string/array  $deps
29  * @return boolean
30  */
31 function dependsOn($deps)
32 {
33     if (is_array($deps)) {
34         foreach ($deps as $dependency)
35             if (!extension_loaded($dependency))
36                 showHtmlMessage(
37                     "Dependency Failure"
38                     "      Extension \"{$dependency}\"      not found."
39                 );
40         die();
41     } else {
42         if (!extension_loaded($deps))
43             showHtmlMessage(
44                 "Dependency Failure"
45                 "      Extension \"{$deps}\"      not found."
46                 TRUE
47             );
48     }
49     return TRUE;
50 }
51
52
53 /**
54  * Handles optional dependencies.
55  *
56  * @param  string  $deps
57  * @return boolean
58  * @todo
59  * <ol>
60  * <li>Implementd.</li>
61  * </ol>
62  */
63 function dependsOptionalOn($deps)
64 {
65     return TRUE;
66 }
67
```

```
68  /**
69   * Handles dependacies on specific functions.
70   *
71   * @param string $funcs
72   * @return boolean
73   */
74  function dependsOnFunc($func)
75  {
76      return function_exists($func);
77  }
78
79  ?>
```

File Source for Error.php

Documentation for this file is available at [Error.php](#)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    front.helpers
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Error.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  * @todo
17  * <ol>
18  * <li>Implement a function that will handle dependancies as
19  * optional.</li>
20  * <li>Recheck <b>all</b> functions.</li>
21  * </ol>
22  */
23
24
25 /**
26  * Presents a "Page Not Found" message.
27  *
28  * @param      string      $str
29  * @return      void
30  * @todo
31  * <ol>
32  * <li>Implement.</li>
33  * </ol>
34  */
35 function showErrorPage404($str=NULL)
36 {
37     die();
38 }
39
40 /**
41  * Prints styled text. Used in printing debug messages and errors.
42  *
43  * @param      string      $title
44  * @param      string      $str
45  * @param      boolean     $die
46  * @return      void
47  */
48 function showHtmlMessage($title, $str=NULL, $die=FALSE)
49 {
50     static $hasOutputedHtml;
51
52     $errorMsg = (is_array($str))
53                 ? implode("<br/>" , $str)
54                 : $str;
55
56     if ($hasOutputedHtml==FALSE)
57     echo "
58         <style>
59         body {
60             padding: 10px;
61             font-family: Tahoma, sans-serif;
62             font-size: 12pt;
63         }
64         .topic {
65             width: 80px;
66             overflow: hidden;
67             float: left;
```



```

68         text-align: right;
69         padding: 0px 10px 0px 0px;
70     }
71     #msg {
72         color: red;
73         font-weight: bold;
74     }
75     #file {
76         color: green;
77         font: italic;
78     }
79     #line {
80         color: blue;
81         text-decoration: underline;
82     }
83     </style>"          ;
84
85
86     echo "<div style=\"border: 1px solid #c5c5c5; padding: 10px;\">"
87         ."    < h2> $title</ h2> "
88         ."    $errorMsg"
89         ."</div><br />"          ;
89
90
91     $hasOutputedHtml = TRUE;
92
93     if ($die)
94         die();
95 }
96
97 ?>

```

File Source for Handlers.php

Documentation for this file is available at [Handlers.php](#)

```
1  <?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    front.helpers
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Handlers.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  * @todo
17  * <ol>
18  * <li>Implement a function that will handle dependancies as
19  *   optional.</li>
20  * <li>Recheck <b>all</b> functions.</li>
21  * </ol>
22  */
23
24  /**
25   * Custom error handler.
26   *
27   * @param integer $errno
28   * @param string $errstr
29   * @param string $errfile
30   * @param integer $errline
31   * @return void
32   */
33  function errorHandler($errno, $errstr, $errfile, $errline)
34  {
35      static $errorTypes;
36
37      $errorTypes = array (
38          2 => "Warning"
39          ,
40          8 => "Notice"
41          ,
42          256 => "User Error"
43          ,
44          512 => "User Warning"
45          ,
46          1024 => "User Notice"
47          ,
48          4096 => "Recoverable Error"
49      );
50
51      $dieStatus = ($errno==256) ? TRUE : FALSE;
52
53      showHtmlMessage(
54          $errorTypes[$errno],
55          "<div class=\"topic\">Message:</div> <span
56          id=\"msg\">" . $errstr . "</span><br/>\n"
57          ,
58          "<div class=\"topic\">In file:</div> <span
59          id=\"file\">" . $errfile . "</span><br/>\n"
60          ,
61          "<div class=\"topic\">On line:</div> <span
62          id=\"line\">#" . $errline . "</span>\n"
63          ,
64          $dieStatus
65      );
66
67      return true;
68  }
69
70  /**
71   * Custom exception handler.
```

```
65  *
66  * @param    object    Exception    $ex
67  * @return    void
68  */
69  function exceptionHandler(Exception $ex)
70  {
71      showHtmlMessage(
72          "Uncaught Exception"
73          $ex->    getMessage(),
74          true
75      );
76
77
78  }
79
80  ?>
```

File Source for Main.php

Documentation for this file is available at [Main.php](#)

```
1  i»¿<?php
2  /**
3   * This source file is part of the leaf framework and
4   * is licensed under the New BSD license.
5   * For the full copyright and license information, please view the LICENSE
6   * file that was distributed with this source code.
7   *
8   * @license      http://leaf-framework.sourceforge.net/LICENSE/  New BSD License
9   * @link         http://leaf-framework.sourceforge.net
10  *
11  * @package      leaf
12  * @subpackage    front.helpers
13  * @author        Avraam Marimpis <makism@users.sf.net>
14  * @version       $Id: Main.php 46 2007-11-12 07:41:49Z makism $
15  * @filesource
16  */
17
18
19 /**
20  * Includes all necesarry files related to leaf`s classes.
21  *
22  * We use the "magic" function __autoload in order
23  * to have leaf`s classes loaded automatically.<br>
24  * Also, {@link leaf_EndorsedManager Endorsed Manager} is
25  * instantiated if needed.
26  *
27  * @link          http://www.php.net/manual/en/language.oop5.autoload.html
28  * @see           leaf_EndorsedManager
29  * @param         string $className
30  * @return        void
31  */
32 function __autoload($className)
33 {
34     static $baseClasses;
35     static $hasInitd;
36     static $enableEndorsementManager;
37     static $endorsementManager;
38
39     /*
40      * Determine if EndorsedManager is enabled.
41      */
42     if ($hasInitd) {
43         if (leaf_Registry::getInstance()->    config['allow_endorsed']=="Yes"    )
44             $enableEndorsementManager = TRUE;
45         $hasInitd=NULL;
46     }
47
48     /*
49      * If, EndorsedManager has been enabled, we
50      * instantiate it.
51      */
52     if ($endorsementManager==NULL &&    $enableEndorsementManager==TRUE) {
53         require_once LEAF_BASE . "core/EndorsementManager.php"    ;
54         leaf_Registry::getInstance()->    register(new leaf_EndorsementManager());
55         $endorsementManager = leaf_Registry::getInstance()->    endorse_man;
56     }
57
58     /*
59      * Most of the leaf`s classes.
60      */
61     if ($baseClasses==NULL)
62         $baseClasses = array (
63             'leaf_Base' =>    'Base.php',
64             'leaf_Benchmark' =>    'Benchmark.php',
65             'leaf_Config' =>    'Config.php',
66             'leaf_Controller'=>    'Controller.php',
67             'leaf_Debug' =>    'Debug.php',
```

```

68         'leaf_Dispatcher'=>    'Dispatcher.php',
69         'leaf_Exception' =>    'Exception.php',
70         'leaf_Hash'      =>    'Hash.php',
71         'leaf_Hook'      =>    'hook/Hook.php',
72         'leaf_Hook_Conditional'=> 'hook/Hook_Conditional.php',
73         'leaf_Loader'    =>    'Loader.php',
74         'leaf_Locale'    =>    'Locale.php',
75         'leaf_Log'       =>    'log/Log.php',
76         'leaf_Logger'    =>    'log/Logger.php',
77         'leaf_Model'     =>    'Model.php',
78         'leaf_Registry'  =>    'Registry.php',
79         'leaf_Request'   =>    'Request.php',
80         'leaf_Response'  =>    'Response.php',
81         'leaf_Router'    =>    'Router.php',
82         'leaf_View'      =>    'View.php'
83     );
84
85     /*
86     * Check to dermine whether or not, the requested class
87     * is part of leaf`s core.
88     */
89     if (array_key_exists($className, $baseClasses)) {
90
91         /*
92         * We let the EndorsedManager to handle the external
93         * classes that will overlap the internal ones.
94         */
95         if ($enableEndorsementManager) {
96             if ($endorsementManager->isEndorsed($className))
97                 $endorsementManager->loadEndorsedClass($className);
98             else
99                 require_once LEAF_BASE
100                     . 'core/'
101                     . $baseClasses[$className];
102         } else {
103             require_once LEAF_BASE
104                 . 'core/'
105                 . $baseClasses[$className];
106         }
107     }
108
109     /*
110     * If the class {@link leaf_Config} has been called,
111     * we can access the configuration parameters and
112     * determine if EndorsedManager is enabled.
113     */
114     if ($className=="leaf_Config" )
115         $hasInitied = TRUE;
116 }
117
118 /**
119  * Terminates the execution of the framework.
120  *
121  * @return void
122  */
123 function frameworkForceTerminate()
124 {
125     showHtmlMessage("Forced Termination" , NULL, TRUE);
126 }
127
128 /**
129  * Performs some basic tests to check whether
130  * the framework is usable or not.
131  *
132  * @param    boolean    $checkAll
133  * @return    void
134  */
135 function frameworkSanityCheck($checkAll=false)
136 {
137     $msg = array();
138
139     if (extension_loaded('spl')==FALSE) {
140         $msg[] = "SPL functions are disabled." ;
141     }
142
143     /*
144     * LEAF_REL_STATUS
145     */
146     if (!defined('LEAF_REL_STATUS'))
147         $msg[] = "No releasing status information found -"

```

```

148         . "Missing \"LEAF_REL_STATUS\" declaration.\n"
149
150     /*
151     * LEAF_WORKING_DIR
152     */
153     if (!defined('LEAF_WORKING_DIR')) {
154         $msg[] = "No Working directory is defined - "
155             . "Missing \"LEAF_WORKING_DIR\" declaration.\n"
156     } else {
157         if (!is_readable(LEAF_WORKING_DIR)) {
158             $msg[] = "Working directory is NOT readable.\n"
159         } else {
160             if (is_writeable(LEAF_WORKING_DIR)) {
161                 $msg[] = "Working directory IS writeable."
162                 . "Security issues arise.\n"
163             }
164         }
165     }
166
167     /*
168     * LEAF_BASE
169     */
170     if (!defined('LEAF_BASE')) {
171         $msg[] = "No leaf Base directory is defined - "
172             . "Missing \"LEAF_BASE\" declaration.\n"
173     } else {
174         if (!file_exists(LEAF_BASE)) {
175             $msg[] = "leaf Base directory NOT found.\n"
176         } else {
177             if (!is_readable(LEAF_BASE)) {
178                 $msg[] = "leaf Base directory is NOT readable.\n"
179             } else {
180                 if (is_writeable(LEAF_BASE)) {
181                     $msg[] = "leaf Base directory IS writeable."
182                     . "Security issues arise.\n"
183                 }
184             }
185         }
186     }
187
188     /*
189     * LEAF_APPS
190     */
191     if (!defined('LEAF_APPS')) {
192         $msg[] = "No leaf Application directory is defined - "
193             . "Missing \"LEAF_APPS\" declaration.\n"
194     } else {
195         if (!file_exists(LEAF_APPS)) {
196             $msg[] = "leaf Application directory NOT found.\n"
197         } else {
198             if (!is_readable(LEAF_APPS)) {
199                 $msg[] = "leaf Application directory is NOT readable.\n"
200             }
201         }
202     }
203
204     /*
205     * LEAF_VAR
206     */
207     if (!defined('LEAF_VAR')) {
208         $msg[] = "No Variable directory is defined - "
209             . "Missing \"LEAF_VAR\" declaration.\n"
210     } else {
211         if (!is_readable(LEAF_VAR)) {
212             $msg[] = "Variable directory is NOT readable.\n"
213         } else {
214             if (is_writable(LEAF_VAR)) {
215                 $msg[] = "Variable directory IS writable."
216                 . "Security issues arise.\n"
217             } else {
218                 if (!is_readable(LEAF_VAR . 'cache/')) {
219                     $msg[] = "Cache directory is NOT readable.\n"
220                 } else {
221                     if (!is_writeable(LEAF_VAR . 'cache/')) {
222                         $msg[] = "Cache directory is NOT writable.\n"
223                     }
224                 }
225             }
226         }
227     }

```

```

228         if (!is_readable(LEAF_VAR . 'logs/')) {
229             $msg[] = "Logs directory is NOT readable.\n" ;
230         } else {
231             if (!is_writable(LEAF_VAR . 'logs/')) {
232                 $msg[] = "Logs directory is NOT writable.\n" ;
233             }
234         }
235
236         if (!is_readable(LEAF_VAR . 'sigs/')) {
237             $msg[] = "Sigs directory is NOT readable.\n" ;
238         } else {
239             if (!is_writable(LEAF_VAR . 'sigs/')) {
240                 $msg[] = "Sigs directory is NOT writable.\n" ;
241             }
242         }
243
244         if (!is_readable(LEAF_VAR . 'tmp/')) {
245             $msg[] = "Tmp directory is NOT readable.\n" ;
246         } else {
247             if (!is_writable(LEAF_VAR . 'tmp/')) {
248                 $msg[] = "Tmp directory is NOT writable.\n" ;
249             }
250         }
251     }
252 }
253
254 if ($checkAll) {
255     /*
256     * ETC
257     */
258     if (!file_exists(LEAF_BASE . 'Etc/')) {
259         $msg[] = "Etc directory NOT found.\n" ;
260     } else {
261         if (!is_readable(LEAF_BASE . 'Etc/')) {
262             $msg[] = "Etc directory is NOT readable.\n" ;
263         }
264     }
265 }
266
267 /*
268 * CONFIGURATION FILES
269 */
270 $configFiles = array (
271     "autoload.php" , "config.php" , "database.php" ,
272     "hooks.php" , "routes.php"
273 );
274 foreach ($configFiles as $file) {
275     if (!file_exists(LEAF_BASE . 'Etc/' . $file)) {
276         $msg[] = "Configuration file \"{ $file }\", does not exist.\n" ;
277     } else {
278         if (!is_readable(LEAF_BASE . 'Etc/' . $file)) {
279             $msg[] = "Configuration file \"{ $file }\", cannot be
read.\n" ;
280         }
281     }
282 }
283
284 unset($configFiles);
285 }
286
287 if (sizeof($msg) > 0)
288     showHtmlMessage("Sanity Check Failure" , $msg, true);
289 }
290
291 ?>

```

Appendix D - Todo List

In Package leaf

In [Debug.php](#)

- 1. Document.
 2. Recheck the information displayed under the tab "Config Settings".

In [Debug.php](#)

- 1. Document.
 2. Possible removal.

In [Dependancies.php](#)

- 1. Implement a function that will handle dependancies as optional.
 2. Recheck **all** functions.

In [dependsOptionalOn\(\)](#)

- 1. Implementd.

In [Error.php](#)

- 1. Implement a function that will handle dependancies as optional.
 2. Recheck **all** functions.

In [Front_Controller.php](#)

- 1. Add functionality for the 'hooks' subsystem.
 2. Add functionality for the benchmarking subsystem.
 3. Completion of the documentation.

In [leaf_Request::getQueryString\(\)](#)

- 1. Possible method refactor.

In [leaf_Request::getQueryStringAsString\(\)](#)

- 1. Possible method complete refactor.

In [Handlers.php](#)

-

1. Implement a function that will handle dependancies as optional.
2. Recheck **all** functions.

In [leaf_EndorsementManager::introspectEndorsedClass\(\)](#)

- 1. Implement.
 2. Must check if the class, has declared the constants LEAF_REG_KEY and/or LEAF_CLASS_ID.

In [introspectHooks\(\)](#)

- 1. Refactor.

In [leaf_Benchmark](#)

- 1. Implement.

In [leaf_Config](#)

- 1. Maybe we should unset the global variable \$GLOBALS somewhere else.
 2. Possible implementation of a subclass that will read seperate configuration files for each Controller, thus declaring different parameters and maybe unique for each Controller.

In [leaf_Controller](#)

- 1. Refactor.
 2. Possible implementation of "Dependencies Injection (DI)" method, so that each Controller, will load **only** the classes it depends on.

In [leaf Dispatcher](#)

- 1. Remove member properties.
 2. Replace the member properties with method calls to leaf_Request.

In [leaf EndorsementManager](#)

- 1. Implement.
 2. Document.
 3. Revamp.

In [leaf Exception](#)

- 1. Possible, internal logging function.

In [leaf Hash](#)

- 1. Possible creation of an interface.

In [leaf Hook](#)

- 1. Implement.

In [leaf Hook Conditional](#)

- 1. Implement.

In [leaf Input](#)

- 1. Implement.

In [leaf Loader](#)

- 1. Implement.

In [leaf Log](#)

- 1. Implement.

In [leaf_Logger](#)

- 1. Possible implementation for logging in a database.

In [leaf_Model](#)

- 1. Implement.

In [leaf_Registry](#)

- 1. Thinking about listing all possible entries with their types - so there are some sanity checks...?

In [leaf_Request](#)

- 1. Possible implementation of [facade](#) functions.

In [leaf_Response](#)

- 1. Implement.

In [leaf_Router](#)

-

1. Maybe this class should completely overtake the class [leaf_Request](#), thus removing the later class.
2. Provide an alternative class (which maybe will inherit from this one) that will use static routing rules.

In [leaf_View](#)

-

1. Implement.

In [leaf_EndorsementManager::loadEndorsedClass\(\)](#)

-

1. Check again the registration function that the classes are using, as well as the "keys" that they use in the class [leaf_Registry](#).

In [leaf_Request::recostructUrl\(\)](#)

-

1. Implement.

In [leaf_Request::redirect\(\)](#)

-

1. Implement.

In [leaf_Registry::register\(\)](#)

- 1. Perform some checks to find out if the object passed, has declared the constant LEAF_REG_KEY, and possible examine further it`s behaviour.

In [leaf_View::render\(\)](#)

- 1. When this method is called, the Output Buffer, will be also flushed and terminating. This will spawn the "include" method which will be similar except the Ob thingie.
 2. Maybe this method and "include", will be implemeneted inside the overload method "__call".
 3. Create some "sanity" checks to run against the view name that is requested.
 4. It is an obvious bug that is the user declares a variable with the "viewFile", this will result in overlapping the local variable that holds the file in which the view is located.</i>

In [leaf_Request::segment\(\)](#)

- 1. Implement.

In [leaf_Request::segmentsAsArray\(\)](#)

- 1. Implement.

In [showErrorPage404\(\)](#)

-

1. Implement.

In [leaf_Request::totalSegments\(\)](#)

-

1. Implement.

In [leaf_Registry::unregister\(\)](#)

-

1. Implement.

In [leaf_Router::construct\(\)](#)

-

1. Possible break-down of the jobs that the constructor executes.
2. Recheck the "virtual file extension" function.

Index

B

Benchmark.php	103
Source code	
Benchmark.php	65
This source file is part of the leaf framework and is licensed under the New BSD license.	
Benchmark.php	5
This source file is licensed under the New BSD license.	
Base.php	4
This source file is licensed under the New BSD license.	

C

constructor leaf Response:: construct()	55
<i>Class constructor</i>	
constructor leaf Request:: construct()	50
<i>Internally uses the class leaf Router in order to export information like, the Controller name, the Controller's name as is <i>must</i> be, the Action, etc.</i>	
constructor leaf Registry:: construct()	45
<i>Declared private in favor of the Singleton pattern.</i>	
constructor leaf Model:: construct()	43
constructor leaf Router:: construct()	60
<i>•!»!»!½!°!® !¼!µ!,!-!+!±!f!· !€!-!µ!' !µ!'İŽ...</i>	
constructor leaf View:: construct()	63
CHANGELOG	100
constructor leaf Logger:: construct()	77
<i>Begins the logging procedures, like file locking and filename scheming.</i>	
constructor leaf Hook Conditional:: construct()	70
constructor leaf Hook:: construct()	69
constructor leaf Loader:: construct()	41
constructor leaf Input:: construct()	39
constructor leaf Config:: construct()	25
<i>Encapsulates all the configuration files, and stores the paremeters in an array.</i>	
constructor leaf Benchmark:: construct()	23
<i>Declares dependencies.</i>	
constructor leaf Base:: construct()	21
<i>Each time a subclass calls the constructor, some checks are performed to find out if the subclass is already instantiated, and thus exists in the \$Registry.</i>	
Controller.php	7
<i>This source file is licensed under the New BSD license.</i>	
constructor leaf Controller:: construct()	28

<i>Calls the parent constructor.</i>	
constructor leaf Dispatcher::__construct()	30
<i>Tests the requested Controller and prepares for dispatching it, along with the Action.</i>	
constructor leaf Hash::__construct()	37
<i>Flags the method of hashing the we will be using.</i>	
constructor leaf Exception::__construct()	35
constructor leaf EndorsementManager::__construct()	33
<i>Discovers the classes that are declared for overlapping.</i>	
Config.php	6
<i>This source file is licensed under the New BSD license.</i>	

D

dependsOptionalOn()	88
<i>Handles optional dependencies.</i>	
Debug.php	107
<i>Source code</i>	
Debug.php	114
<i>Source code</i>	
Dependencies.php	119
<i>Source code</i>	
dependsOnFunc()	87
<i>Handles dependencies on specific functions.</i>	
dependsOn()	87
<i>Handles dependencies on PHP extensions.</i>	
Debug.php	80
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Debug.php	83
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Dependencies.php	87
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Dispatcher.php	8
<i>This source file is licensed under the New BSD license.</i>	

E

exceptionHandler()	91
<i>Custom exception handler.</i>	
Error.php	121
<i>Source code</i>	
errorHandler()	91
<i>Custom error handler.</i>	
Error.php	89
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Exception.php	10
<i>This source file is licensed under the New BSD license.</i>	

EndorsementManager.php	9
<i>This source file is licensed under the New BSD license.</i>	

F

Front_Controller.php	111
<i>Source code</i>	
frameworkSanityCheck()	93
<i>Performs some basic tests to check whether the framework is usable or not.</i>	
frameworkForceTerminate()	93
<i>Terminates the execution of the framework.</i>	
Front_Controller.php	81
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	

G

getHooks()	71
<i>Returns all registered hooks for one -or all- level. Also, the hooks can be filtered using the parameters \$controller and \$method.</i>	

H

HOOK_PRE_CONTROLLER_DISPATCH	72
<i>Level, Pre Controller Dispatch</i>	
Handlers.php	91
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Hooks.php	105
<i>Source code</i>	
Handlers.php	123
<i>Source code</i>	
HOOK_POST_FRONT_CONTROLLER	71
<i>Level, Post Front Controller</i>	
HOOK_POST_CONTROLLER_DISPATCH	71
<i>Level, Post Controller Dispatch</i>	
Hook.php	67
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Hook_Conditional.php	68
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Hooks.php	71
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Hash.php	11
<i>This source file is licensed under the New BSD license.</i>	

I	
i_var()	83
INSTALL	100
inspect_var()	83
introspectHooks()	72
	<i>Returns the registered hooks for the requested level.</i>
Input.php	12
	<i>This source file is licensed under the New BSD license.</i>
index.php	2
	<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>

L	
leaf_Response::\$buffer	55
leaf_Response::\$useGzip	55
leaf_Response::\$useTidy	55
leaf_Response::clearHeaders()	55
leaf_Response::LEAF_REG_KEY	55
leaf_Response::LEAF_CLASS_ID	55
leaf_Request::totalSegments()	54
	<i>Returns the total number of segments.</i>
leaf_Request::toString()	54
leaf_Response	54
leaf_Response::flushOutputBuffer()	56
leaf_Response::getOutputBuffer()	56
	<i>Gets output buffer.</i>
leaf_Router	57
	<i>Processes and filters the current Uri.</i>
leaf_Router::LEAF_CLASS_ID	58
leaf_Router::LEAF_REG_KEY	58
leaf_Router::\$queryString	58
	<i>The query string found in the Uri.</i>
leaf_Response::toString()	57
leaf_Response::setHeader()	57
leaf_Response::outputBufferingStart()	56
	<i>Start output buffering</i>
leaf_Response::outputBufferingEnd()	56
	<i>End output buffering</i>
leaf_Response::sendResponse()	57
leaf_Request::segmentsToArray()	53
	<i>Returns the segments` array.</i>
leaf_Request::segment()	53
	<i>Retrieves the requested (numeric) offset from the segments.</i>
leaf_Request	48
	<i>Provides access to all elements that compose the Uri.</i>
leaf_Request::LEAF_CLASS_ID	49
leaf_Request::LEAF_REG_KEY	49
leaf_Request::\$action	49
	<i>The requested method name (Action).</i>
leaf_Registry::set()	48

<i>Registers the requested instance using the designated key.</i>	
leaf_Registry::get()	47
<i>Return the request object, by refering to it's instance name.</i>	
leaf_Registry::register()	46
<i>Registers an object.</i>	
leaf_Registry::toArray()	47
<i>Returns an associative array with all the "general" configuration parameters.</i>	
leaf_Registry::unregister()	47
<i>Removes a key from the registry.</i>	
leaf_Request::\$controller	49
<i>The requested class name (Controller), suffixed with "_Controller".</i>	
leaf_Request::\$controllerFile	49
<i>The file name in which the requested Controller is located.</i>	
leaf_Request::getQueryString()	51
<i>Retrieves the value for the requested key.</i>	
leaf_Request::getQueryStringAsString()	52
<i>Return the current query string, as a string.</i>	
leaf_Request::reconstructUri()	52
<i>Reconstructs a Uri, based on the data passed.</i>	
leaf_Request::redirect()	52
<i>Performs a redirect to the speficed Uri.</i>	
leaf_Request::getControllerName()	51
<i>Returnsthe requested class name (Controller), suffixed with "_Controller".</i>	
leaf_Request::getControllerFileName()	51
<i>Returns the file name that contains the current Controller.</i>	
leaf_Request::\$queryElems	50
<i>The query string found in the Uri.</i>	
leaf_Request::\$segments	50
<i>The extra segments found in the Uri.</i>	
leaf_Request::getApplicationName()	50
<i>Returns the current Application`s name.</i>	
leaf_Router::\$queryStringElements	58
<i>The query string found in the Uri, as an associative array.</i>	
leaf_Router::\$requestClass	59
<i>The requested class name (Controller).</i>	
leaf_Logger	75
leaf_Logger::LEAF_CLASS_ID	76
leaf_Logger::LEAF_REG_KEY	76
leaf_Logger::\$buffer	76
leaf_Log	75
Logger.php	75
<i>This source file is licensed under the New BSD license.</i>	
leaf_Hook_Conditional::LEAF_REG_KEY	69
leaf_Hook_Conditional::condition()	70
Log.php	74
<i>This source file is licensed under the New BSD license.</i>	
leaf_Logger::\$filename	76
leaf_Logger::\$fp	76
leaf_Logger::setLevel()	78
<i>Set the level of errors to capture.</i>	
leaf_Logger::setTimeStampPattern()	79

Set a timestamp pattern.	
leaf_Logger::__call()	79
__call	
LICENSE	99
leaf_Logger::flush()	78
Flushes (empties) the internal buffer.	
leaf_Logger::end_flush()	78
Flushes the internal buffer, and releases all locks on the log files.	
leaf_Logger::\$levels	77
leaf_Logger::\$stampPat	77
leaf_Logger::\$threshold	77
leaf_Hook_Conditional::LEAF_CLASS_ID	69
leaf_Hook_Conditional	69
leaf_Router::getMethodName()	61
Returns the method name (Action) that is requested.	
leaf_Router::queryStringElements()	61
Returns an associative array filled with the elements that found in the query string.	
leaf_Router::segments()	61
Returns an array with the extra segments that compose this Uri.	
leaf_Router::segmentsSize()	62
Returns the total number of segments that found in the current Uri.	
leaf_Router::getClassName()	61
Returns the class name (Controller) that is requested.	
leaf_Router::chopSegment()	60
Removes the first string occurrence until the very first "/", from the property \$this->requestUri.	
leaf_Router::\$requestMethod	59
The requested method name (Action).	
leaf_Router::\$requestUri	59
The current requested Uri string.	
leaf_Router::\$segments	60
The extra segments found in the Uri.	
leaf_Router::__toString()	62
leaf_View	62
leaf_Hook::LEAF_CLASS_ID	68
leaf_Hook::LEAF_REG_KEY	69
leaf_Hook::__toString()	69
leaf_Hook	68
leaf_View::__toString()	63
leaf_View::LEAF_CLASS_ID	63
leaf_View::LEAF_REG_KEY	63
leaf_View::render()	63
Includes and renders a view file.	
leaf_Registry::isRegistered()	46
Checks for the existence of the requested key.	
leaf_Registry::instance()	45
If needed, instantiates an object of this class.	
leaf_Config::offsetSet()	27
Sets a value in the specific parameter.	
leaf_Config::offsetUnset()	27

<i>Unsets a parameter.</i>	
leaf_Config::toString()	28
leaf_Controller	28
<i>Assigns some common characteristics to all user's Controllers.</i>	
leaf_Config::offsetGet()	26
<i>Returns the value of the specified parameter.</i>	
leaf_Config::offsetExists()	26
<i>Checks if the request parameter-name exists.</i>	
leaf_Config::\$options	25
<i>All configuration parameters will be stored in this array.</i>	
leaf_Config::\$optionsTable	25
<i>All configuration parameters are stored in this array, using their parent configuration file as index.</i>	
leaf_Config::getByHashKey()	26
<i>Returns all parameters related with the specified key.</i>	
leaf_Controller::LEAF_CLASS_ID	28
leaf_Controller::LEAF_REG_KEY	28
leaf_Dispatcher::\$controllerName	30
<i>The requested class name (Controller).</i>	
leaf_Dispatcher::\$target	30
<i>The file name in which the requested Controller is located.</i>	
leaf_Dispatcher::dispatchController()	31
<i>Checks for the existence of the desired method and calls it.</i>	
leaf_Dispatcher::toString()	31
leaf_Dispatcher::\$controller	30
<i>An instance of the requested Controller.</i>	
leaf_Dispatcher::LEAF_REG_KEY	30
leaf_Controller::toString()	29
leaf_Dispatcher	29
<i>Prepares to dispatch the specifief Controller/Action.</i>	
leaf_Dispatcher::LEAF_CLASS_ID	29
leaf_Config::LEAF_REG_KEY	25
leaf_Config::LEAF_CLASS_ID	25
Loader.php	13
<i>This source file is licensed under the New BSD license.</i>	
leaf_Base	20
<i>Assigns common behavior and properties in the internal classes.</i>	
leaf_Base::LEAF_CLASS_ID	20
<i>A unique handle for each subclass.</i>	
leaf_Base::LEAF_REG_KEY	21
<i>The name of the instance that the current subclass</i>	
LEAF_WORKING_DIR	2
<i>Current directory where *this* file is located.</i>	
LEAF_VAR	2
<i>Subdirectory in which cache and temporary files are stored.</i>	
LEAF_BASE	2
<i>Subdirectory in which leaf's files are located.</i>	
LEAF_REL_STATUS	2
<i>leaf's Status and Version information.</i>	
LEAF_REL_VERSION	2
leaf_Base::\$Registry	21
<i>The one and only unique instance of the class leaf_Registry.</i>	
leaf_Base::clone()	21

<i>Prevent object cloning.</i>	
leaf_Benchmark::\$indexTable	23
<i>All requested benchmark points are stored in this array, in a hashtable-like way.</i>	
leaf_Benchmark::addIndex()	23
<i>Add a begin/end benchmark point.</i>	
leaf_Config	24
<i>Provides access to the configuration files.</i>	
leaf_Benchmark::LEAF_REG_KEY	23
leaf_Benchmark::LEAF_CLASS_ID	23
leaf_Base::get()	22
<i>Returns the requested object from the \$Registry.</i>	
leaf_Base::toString()	22
<i>Returns a descriptive string about this class.</i>	
leaf_Benchmark	22
<i>Keeps statistics (execution time, memory usage) either for a specific class, or a code block.</i>	
leaf_EndorsementManager	31
<i>Handles the requests, to overlap the internal implementations of specific classes, with external, in userspace-like fashion.</i>	
leaf_EndorsementManager::LEAF_CLASS_ID	32
leaf_Loader::LEAF_CLASS_ID	41
leaf_Loader::LEAF_REG_KEY	41
leaf_Loader::endorsed()	41
leaf_Loader::extension()	41
leaf_Loader	41
leaf_Input::offsetUnset()	40
leaf_Input::offsetExists()	39
leaf_Input::offsetGet()	40
leaf_Input::offsetSet()	40
leaf_Loader::library()	42
leaf_Loader::model()	42
leaf_Registry	44
<i>Registry class that lists most of the instantiated internal classes.</i>	
leaf_Registry::\$instance	44
<i>The unique instance of this class.</i>	
leaf_Registry::\$registered	45
<i>All currently registered classes.</i>	
leaf_Registry::getInstance()	45
<i>Returns the instance of this class.</i>	
leaf_Model::LEAF_REG_KEY	43
leaf_Model::LEAF_CLASS_ID	43
leaf_Loader::plugin()	42
leaf_Loader::toString()	43
leaf_Model	43
leaf_Input::\$input	39
<i>Array with references to the super global variables "post" and "get".</i>	
leaf_Input::LEAF_REG_KEY	39
leaf_EndorsementManager::getEndorsedClasses()	33
<i>Returns all the classes, that are currently overlapping the internal classes.</i>	
leaf_EndorsementManager::introspectEndorsedClass()	33

<i>Introspect the class` file.</i>	
leaf_EndorsementManager::isEndorsed()	34
<i>Find out if the requested class is endorsed for overlapping.</i>	
leaf_EndorsementManager::loadEndorsedClass()	34
<i>Load the requested class.</i>	
leaf_EndorsementManager::\$registeredEndorsed	33
<i>Array with the classes that have been requested to be overlapped.</i>	
leaf_EndorsementManager::\$endorsed	32
<i>Currently classes that are overlapped.</i>	
leaf_EndorsementManager::LEAF_REG_KEY	32
leaf_EndorsementManager::\$allowEndorsement	32
<i>Array with the classes that are allowed to be overlapped.</i>	
leaf_EndorsementManager::\$currentClass	32
<i>Current class that we will work on.</i>	
leaf_Exception	35
<i>Custom exception class.</i>	
leaf_Hash	36
<i>Handles internal hashing procedures, based on already existing native hash methods.</i>	
leaf_Hash::setAlgorithm()	38
<i>Sets the requested algorithm for usage.</i>	
leaf_Input	38
leaf_Input::LEAF_CLASS_ID	39
leaf_Hash::digestMsg()	38
<i>Returns (as text or as a byte sequence) the hash for the specific give input.</i>	
leaf_Hash::digestFile()	37
<i>Returns (as text or as a byte sequence) the hash for the specific file`s contents.</i>	
leaf_Hash::\$algorithms	36
<i>All supported hash algorithms.</i>	
leaf_Hash::\$defaultAlgorithm	36
<i>Default hash algorithm</i>	
leaf_Hash::\$useAltMethod	37
<i>Whether to use an alternative method or the classic hash functions.</i>	
LEAF_APPS	2
<i>Subdirectory in which user`s applications are located.</i>	

M

Main.php	125
<i>Source code</i>	
Main.php	93
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Model.php	14
<i>This source file is licensed under the New BSD license.</i>	

R

runHooks()	73
<i>Runs all registered hooks in one level.</i>	
README	99
Request.php	104
<i>Source code</i>	
runHook()	72
<i>Executes a specific hook.</i>	
Request.php	66
<i>This source file is part of the leaf framework and is licensed under the New BSD license.</i>	
Request.php	16
<i>This source file is licensed under the New BSD license.</i>	
Response.php	17
<i>This source file is licensed under the New BSD license.</i>	
Router.php	18
<i>This source file is licensed under the New BSD license.</i>	
Registry.php	15
<i>This source file is licensed under the New BSD license.</i>	

S

showHtmlMessage()	90
<i>Prints styled text. Used in printing debug messages and errors.</i>	
showErrorPage404()	89
<i>Presents a "Page Not Found" message.</i>	

T

TODO	99
----------------------	----

V

View.php	19
<i>This source file is licensed under the New BSD license.</i>	

_inspect_var_Resource()	85
_inspect_var_String()	86
_autoload()	93
<i>Includes all necessary files related to leaf's classes.</i>	
_inspect_var_Object_getSubclasses()	85
_inspect_var_Object()	85
_inspect_var_Boolean()	84
_inspect_var_Generic()	84
_inspect_var_Number()	84
_inspect_var_Array()	84