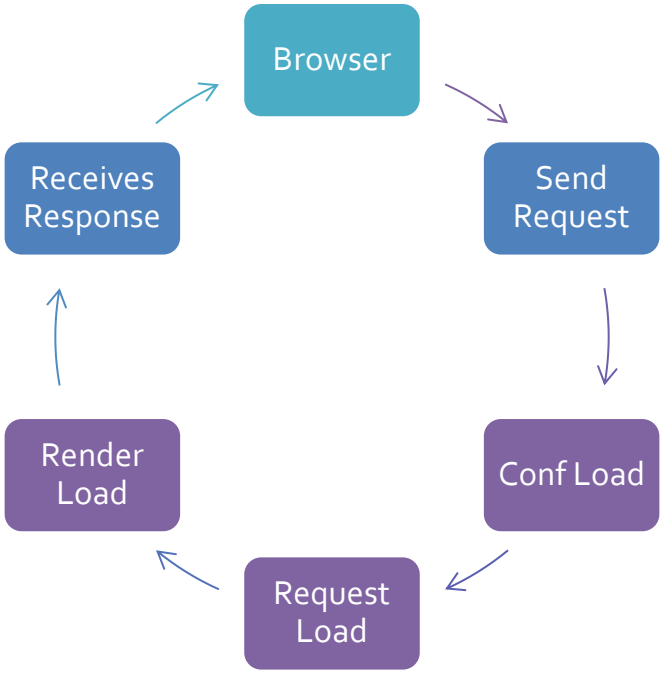
Lidiun Framework 6

Documentation

Developed by Dyon Enedi.

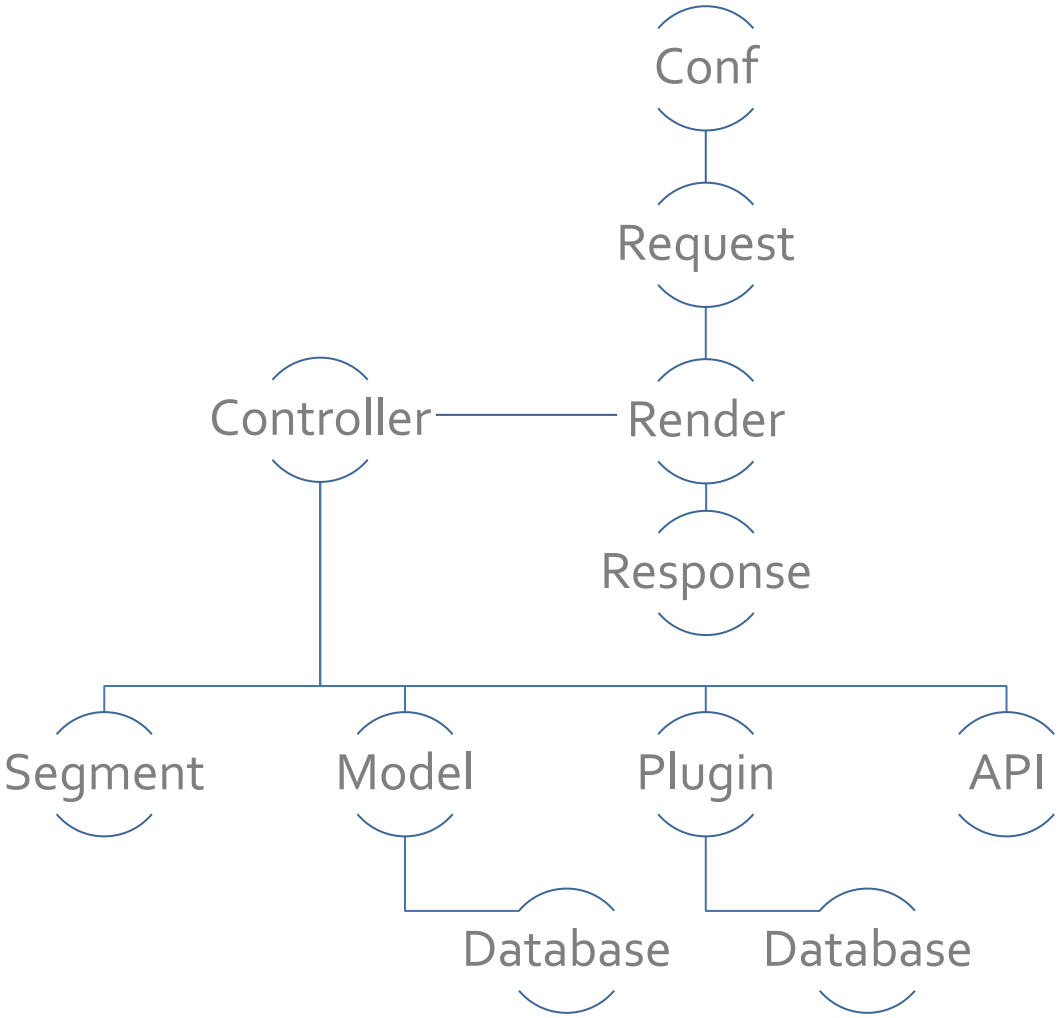
Request Cycle Flowchart



Request Cycle Explanation

1. Browser send request to framework
2. Framework receives request
3. Framework loads configurations
    a. Set application status (On/Off)
    b. Set environment ("dev/test/stat/prod")
    c. Set debug mode
    d. Set time zone
    e. Start Session
4. Framework loads request
    a. Set Paths
    b. Set Links
    c. Set Url
    d. Set Language (Translate)
    e. Set Parameters (Parameters, GET, POST, FILES)
    f. Set Controller
    g. Set default header
5. Framework loads render
    a. Run Global Render
    b. Run Controller
    c. Destroy Database Connection (If exists)
    d. Delivery response (Layout html if WEB, Json array if AJAX)
6. Framework send response to browser

Framework Flow



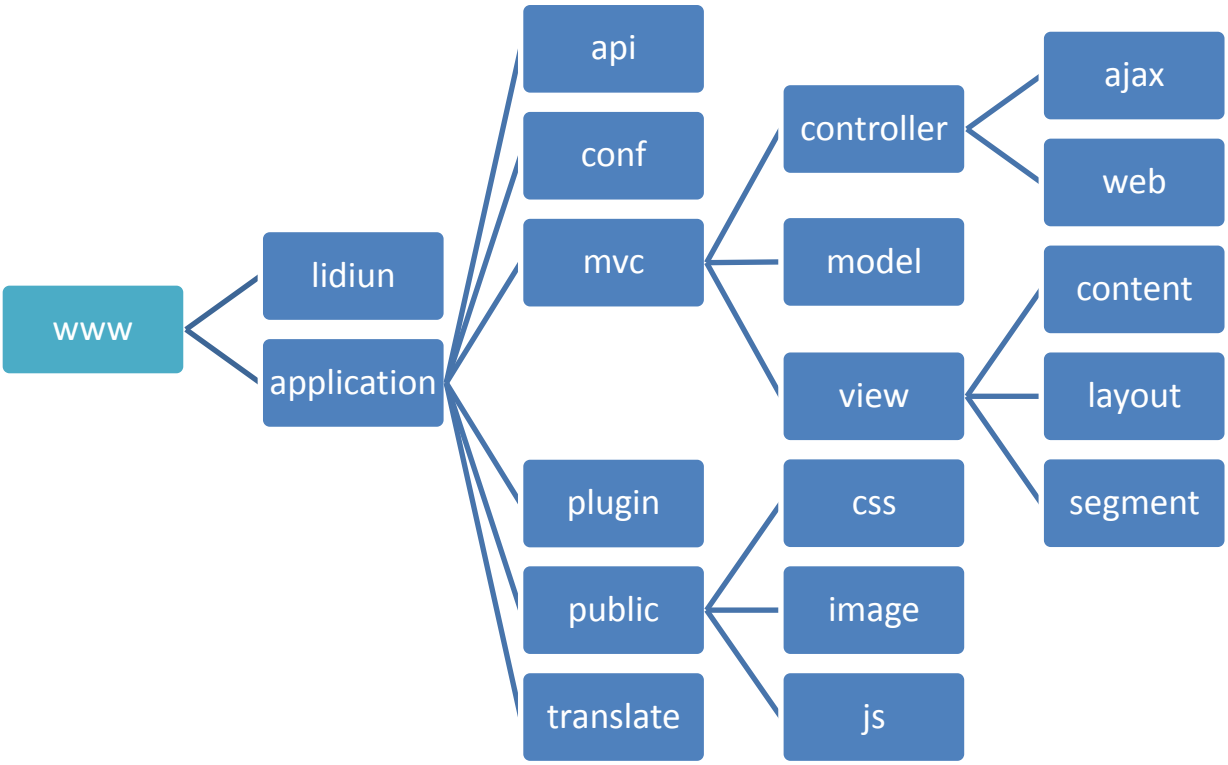
Framework Flow Explanation

1) Framework loads configurations
2) Framework loads and treat request
3) Framework loads Render
    a) Render calls Controller class
        i) Controller can call Segments
        ii) Controller can call Model classes
            (1) Models can use Database class
            (2) Model can use Plugins
        iii) Controller can call Plugin classes
            (1) Plugins can use Database class
        iv) Controller can call API classes
    b) Render gets Controller response (Content HTML or JSON Array)
    c) Render prepare content, menu, footer and header, replace tags and translate.
4) Framework send response request

Folders Application

The structure of the folders is divided in Framework and Application. This is good because it allows versioning Lidiun direct from Github and at the same time versioned or not the application. Also allows you to use the same framework for more than one application. In this case you have just to rename the folder's application name.

Below you can see the folders structure.



How to Get Lidiun

You have two ways to use Lidiun: Versioned or not versioned.

Versioned way (recommended):

1. Download a clear application structure to your "www". http://www.lidiun.com/download/application_clear
2. Rename application folder to your application name
3. Use "git clone" to clone Lidiun repository to your "www": git clone https://github.com/dyonenedi/lidiun-framework and enjoy.

Not versioned way:

1. Download package as an optional way to use Lidiun. In this package have Lidiun Framework and a clear application structure: http://www.lidiun.com/download/packege
2. Rename application folder to your application name and enjoy.

Example Application:

1. You can download an example application to learn more about how developer with Lidiun and use it to start your project just becoming the layout.

Get Start: Configuring Lidiun

1. Configure the config file in application/config/
    a. Configure your environment.
    b. Configure your database.
    c. Configure your public paths.
    d. Configure your common files.
    e. Configure your preset settings.
2. *Congratulations, you are ready to start a project using Lidiun Framework! In the next page you can find a list of all Lidiun Classes and Methods to help you with your application.*

Static Property and Methods of Lidiun

Autoload::

------ Methods ------

includePath(string $path);

Include $path to "include_path". It's necessary to call classes like an API or Plugin in Controller if you won't use namespace. Keep in mind that the path base to "include_path" is the App Path. So to include plugins that be placed in App the right syntax is includePath("plugin/some_class.php");
Return null.

Database::

------ Methods ------

connect(array $con);

Start a connection with database using database conf setted on settings or using the $con array passed by parameter, where $con must have follow keys:
$con["host_name"], $con["db_name"], $con["user_name"], $con["password"].
Return true or false.

query(string $sql, string $return);

Execute query using sql passed by parameter.
Return defined by $return and can be:
"boolean", "array", "object", "num_rows";

autocommit(boolean $autocommit);

Set auto commit true or false in database connection.
Return null.

commit();

Execute commit.
Return null.

rollback();

Execute rollback.
Return null.

getInsertId();

Return the last insert id from this connection.

getErrorMessage();

Return message error if exist.

close();

Close connection with database.

Return null.

------ Properties ------

boolean $_error;

True if happened an error or false if not.

Language::

------ Methods ------

setLanguage(string $language);

Set language to be used in translator system into Lidiun. The language must follow this syntax: "pt-br" or "en-us" or ...

Return null.

getLanguage();

Return language setted.

translation(string $content);

Translate string content using language setted and files in translation path.

Return string translated.

Link::

------ Properties ------

array $_link;

Array with links setted by conf in path to use in controller. $_link["public"] is setted by default.

Path::

------ Properties ------

array $_path;

Array with paths setted by conf in path to use in controller. The follow paths are setted by default: $_link["app"], $_link["conf"], $_link["mvc"], $_link["plugin"], $_link["translation"], $_link["public"], $_link["model"], $_link["view"], $_link["layout"], $_link["content"], $_link["segment"], $_link["controller"], $_link["ajax"], $_link["web"].

Url::

------ Properties ------

array $_url;

Array with url parts from application generate automatic: $_url["protocol"],
$_url["host"],
$_url["port"],
$_url["base"] = $_url["protocol"] + $_url["host"] + $_url["port"],

$_url["uri"],
$_url["full_url"] = $_url["protocol"] + $_url["host"] + $_url["uri"].

Render::

------ Methods ------

setReply(array $reply);

Set array $_reply to response with json_encode when Ajax Request.
Return null.

Request::

------ Methods ------

setAjaxResponseHeader(array $header);

unset header default from ajax and reset headers to values in array.
Ex: $header[0] = "Content-Type: text/html; charset=UTF-8"; $header[1] = "Access-Control-Allow-Methods: GET, POST";
Return null.

setWebResponseHeader(array $header);

unset header default from web and reset headers to values in array.
Ex: $header[0] = "Content-Type: text/html; charset=UTF-8"; $header[1] = "Access-Control-Allow-Methods: GET, POST";
Return null.

setParameter(* $parameter);

Set some value to $_parameter array.
Return null.

unsetParameter(int $key);

Unset some key from $_parameter array.
Return null.

getParameter();

Return $_parameter array.

getPost();

Return $_post array.

getGet();

Return $_getr array.

getFiles();

Return $_files array.

getEntity();

Return $_entity if exists. It's used often in social network when the entity is the first parameter in url and is user name or id.

getController();

Return $_controller.

redirect(string $content);

Change immediate the controller to Lidiun/Redirect. When Render load Redirect controller, it will use $content passed by parameter to find an html content in segments to load. If it doesn't exists, will use $content like a message in content of page or ajax reply.

redirectTo(string $url);

Redirect to Url::$_url['base'] + url passed by $url, ex: 'home/index'

Layout::

------ Methods ------

replaceLayout(string $ search, string $replace);

Search for $search (in tag '<%$search %>') in layout html and replace for $replace.
Return null.

replaceContent(string $ search, string $replace);

Search for $search (in tag '<%$search %>') in content html and replace for $replace.
Return null.

replaceMenu(string $ search, string $replace);

Search for $search (in tag '<%$search %>') in menu html and replace for $replace.
Return null.

replaceFooter(string $ search, string $replace);

Search for $search (in tag '<%$search %>') in footer html and replace for $replace.
Return null.

replaceView(string $ search, string $replace, string $view);

Search for $search in $segment and replace for $replace.
Return $view replaced.

getView(string $view);

Find for view html in mvc/view/content/.
Return view content html.

putContent(string $html);

Put $html in $_content.
Return null.

getSegment(string $segment);

Find for segment html in mvc/view/segment/.
Return segment html.

renderMenu(boolean $ render);

Set if menu will show or not.
Return null.

renderFooter(boolean $ render);

Set if footer will show or not.
Return null.

addCss(string $ file);

Find for $file in css path, if exist add for call to current controller.
Return null.

addJs(string $ file);

Find for $file in js path, if exist add to for call to current controller.
Return null.

removeCss(string $ file);

Find for $file in css array, if exist remove for don't call to current controller.
Return null.

removeJs(string $ file);

Find for $file in js array, if exist remove for don't call to current controller.
Return null.

setTitle(string $tilte);

Set a title to current controller. If not setted, the controller name will be setted by default.
Return null.

setDescription(string $text);

Set a description to current controller. If not setted, the description setted in "conf/preset" will be setted by default.
Return null.

mountSegment (string $segment, array $data);

This method will pass to array $data, and for each $row, it will replace <%$row->keys)%> to $row->values in $segment concatenating row by row.
Return html segment filled with $data values.

Lidiun References

Lidiun about: http://www.lidiun.com
Lidiun repository: https://github.com/dyonenedi/lidiun-framework-6
Lidiun social: https://facebook.com/lidiunframework
Lidiun creator contact: dyonenedi@hotmail.com
Lidiun creator portfolio: www.dyonenedi.com