
LRsoft Docs Documentation

Rilis release

LRsoft

03 Mar, 2018

1	Web Assets	3
2	LRS Engine	5
2.1	Core	5
2.1.1	Configuration	5
2.1.2	Defines	6
2.1.3	Functions	8
2.1.4	Models	9
2.1.5	Controllers	10
2.2	Modules	11
2.3	Skins	11
3	GIT Updater	13
3.1	How to use	13
3.2	Easy setup	13
3.3	Screenshot	14
3.4	References	16

Halaman ini berisi dokumentasi dari project-project yang dikemabangkan oleh tim development [LRsoft](#).

Web Assets

Beberapa media sebagai resources yang dapat diakses secara langsung, baik dengan SSL maupun protokol HTTP biasa.

<https://assets.lrsoft.org>

css

img

favicon.png

lrsoft.png

lombokreisen.png

indotravelonline.png

payment

paypal-pack.png

bank-pack.png

social

48x48

twitter.png

facebook.png

googleplus.png

tripadvisor.png

js

LRS Engine

LRS Engine adalah web based CMS yang dikembangkan untuk fokus pada kinerja dan skalabilitas, dibangun dengan konsep **MVC** sehingga sangat mendukung agile development.

Secara umum **LRS Engine** terbagi menjadi 3 bagian yakni *Core*, *Modules*, dan *Skins*. Masing-masing bagian bertujuan untuk membatasi ruang kerja pengembangan, sehingga tidak akan mengganggu bagian program lainnya.

2.1 Core

Core merupakan inti dari LRS Engine, dimana memuat class untuk semua kebutuhan web. Pada dasarnya LRS Engine merupakan platform untuk blogging. Fitur dasar seperti Posts, Pages, Media, dan lain sebagainya secara native telah tersedia.

2.1.1 Configuration

Buat script `config.php` pada *root* yang berisi konfigurasi utama untuk menentukan beberapa variable yang digunakan dalam Engine.

```
1  /** =====
2   * konfigurasi path dan url
3   * ===== */
4
5  /** app url path */
6  define( 'LRS_URL_PATH', 'http://localhost/lrsoft/lrs-engine' );
7
8
9  /** =====
10   * Informasi umum aplikasi
11   * ===== */
12
13  /** app name */
14  define( 'LRS_APP_NAME', 'Local Development' );
```

```
15
16 /** app author */
17 define( 'LRS_APP_AUTHOR', 'LRsoft Corp.' );
18
19 /** app author url */
20 define( 'LRS_APP_AUTHOR_URL', 'http://lrsoft.co.id' );
21
22
23 /** =====
24  * konfigurasi database
25  * ===== */
26
27 /** host */
28 define( 'LRS_DB_HOST', 'localhost' );
29
30 /** username */
31 define( 'LRS_DB_USERNAME', 'user' );
32
33 /** password */
34 define( 'LRS_DB_PASSWORD', 'pass' );
35
36 /** database name */
37 define( 'LRS_DB_NAME', 'dbname' );
38
39 /** prefix tabel */
40 define( 'LRS_DB_PREFIX', 'lrs_' );
41
42 /** installer enable */
43 define( 'LRS_INSTALLER_ENABLE', true );
44
45
46 /** load fungsi inti lainnya */
47 require_once( 'function.php' );
```

2.1.2 Defines

LRS Engine memerlukan beberapa variabel untuk menjaga konsistensi nilai maupun path.

define.abs.path.php

Berisi variabel path yang dapat diakses untuk keperluan akses file sistem.

```
1 /** absolute path untuk asset */
2 define( 'LRS_ABS_ASSET_PATH', LRS_ABS_PATH . DS . 'asset' );
3
4 /** absolute path untuk css */
5 define( 'LRS_ABS_CSS_PATH', LRS_ABS_PATH . DS . 'asset' . DS . 'css' );
6
7 /** absolute path untuk js */
8 define( 'LRS_ABS_JS_PATH', LRS_ABS_PATH . DS . 'asset' . DS . 'js' );
9
10 /** absolute path untuk js */
11 define( 'LRS_ABS_IMG_PATH', LRS_ABS_PATH . DS . 'asset' . DS . 'img' );
12
13 /** absolute path untuk content upload */
```

```

14 define( 'LRS_ABS_UPLOAD_PATH', LRS_ABS_PATH . DS . 'asset' . DS . 'upload' );
15
16 /** absolute path untuk model */
17 define( 'LRS_ABS_MODEL_PATH', LRS_ABS_PATH . DS . 'model' );
18
19 /** absolute path untuk view */
20 define( 'LRS_ABS_VIEW_PATH', LRS_ABS_PATH . DS . 'view' );
21
22 /** absolute path untuk controller */
23 define( 'LRS_ABS_CONTROLLER_PATH', LRS_ABS_PATH . DS . 'controller' );
24
25 /** absolute path untuk module */
26 define( 'LRS_ABS_MODULE_PATH', LRS_ABS_PATH . DS . 'module' );
27
28 /** absolute path untuk skin */
29 define( 'LRS_ABS_SKIN_PATH', LRS_ABS_PATH . DS . 'skin' );
30
31 /** absolute path untuk version */
32 define( 'LRS_ABS_VERSION_PATH', LRS_ABS_PATH . DS . 'VERSION.md' );
33
34 /** absolute path untuk config */
35 define( 'LRS_ABS_CONFIG_PATH', LRS_ABS_PATH . DS . 'config.php' );

```

define.uri.path.php

Berisi variabel path yang dapat diakses untuk keperluan akses publik.

```

1 /** path untuk asset */
2 define( 'LRS_ASSET_PATH', LRS_URL_PATH . DS . 'asset' );
3
4 /** path untuk css */
5 define( 'LRS_CSS_PATH', LRS_URL_PATH . DS . 'asset' . DS . 'css' );
6
7 /** path untuk js */
8 define( 'LRS_JS_PATH', LRS_URL_PATH . DS . 'asset' . DS . 'js' );
9
10 /** path untuk img */
11 define( 'LRS_IMG_PATH', LRS_URL_PATH . DS . 'asset' . DS . 'img' );
12
13 /** path untuk content upload */
14 define( 'LRS_UPLOAD_PATH', LRS_URL_PATH . DS . 'asset' . DS . 'upload' );
15
16 /** path untuk module */
17 define( 'LRS_MODULE_PATH', LRS_URL_PATH . DS . 'module' );
18
19 /** path untuk skin */
20 define( 'LRS_SKIN_PATH', LRS_URL_PATH . DS . 'skin' );

```

define.version.php

Berisi variabel path yang dapat diakses untuk keperluan akses publik.

```

1 /** versi dari LRS Engine */
2 define( 'LRS_ENGINE_VERSION', $current_version );

```

2.1.3 Functions

Berisi fungsi-fungsi pendukung yang banyak digunakan dalam Core, Modules, dan Skins.

fn.engine.php

Daftar fungsi untuk keperluan engine.

check_required_engine_version (*\$required_version*, *\$exit*, *\$messages*)

Memeriksa kebutuhan versi engine untuk verifikasi module dan skin.

Parameter

- **\$required_version** (*string*) – Versi engine yang ingin dicek.
- **\$exit** (*string*) – Jika tidak sesuai, apakah langsung exit atau tidak (default true).
- **\$messages** (*string*) – Pesan ketika exit (default Insufficient Version of Engine).

is_admin_page ()

Cek apakah suatu halaman merupakan halaman admin atau tidak

Kembali yes | no

sync_default_params (*\$default*, *\$destination*)

Melakukan sinkronisasi terhadap 2 array

Parameter

- **\$default** (*array*) – Array default sebagai sumber.
- **\$destination** (*array*) – Array tujuan.

in_array_r (*\$needle*, *\$haystack*)

Melakukan pencarian secara rekursif didalam array multidimensi

Parameter

- **\$needle** (*string*) – Nilai yang akan dicari.
- **\$haystack** (*array*) – Array tujuan.

lrs_exit (*\$messages* = "", *\$type* = 'info')

Menampilkan format pesan error pada saat exit (menghentikan script dengan paksa)

Parameter

- **\$messages** (*string*) – Nilai yang akan dicari (default string kosong, tapi script tetap `exit()`).
- **\$type** (*string*) – Jenis pesan (info | alert).

lrs_redirect (*\$target* = *LRS_URL_PATH*)

Melakukan redirect (mengarahkan) ke sebuah halaman tertentu sesuai parameter

Parameter

- **\$target** (*string*) – Halaman tujuan.

get_menus (*\$array*)

Mendapatkan built in menu, dari daftar *Pages*

- **post_type**: jenis *post* (*post* | *page*)
- **post_parent**: ID untuk *page* induk
- **post_status**: *publish* | *draft*

- `orderby`: order berdasarkan suatu field
- `order`: ASC | DESC
- `number`: jumlah output maksimum item

Parameter

- `$array` (*array*) – Array yang berisi key.

`validate_url()`

Memastikan konsistensi URL antara *requests* dan `config.php` dengan melakukan redirect.

`lrs_paging_nav($base_url, $total_data, $current_page, $data_per_page = 10, $range_data = 3)`

Membantu membuat navigasi halaman berdasarkan query database.

Parameter

- `$base_url` (*string*) – URL dasar pembentukan navigasi
- `$total_data` (*int*) – Total keseluruhan data
- `$current_page` (*int*) – Halaman saat ini
- `$data_per_page` (*int*) – Jumlah data tiap page
- `$range_data` (*int*) – Range data pada list navigasi, misal: 1 ... 23 24 25 26 27 ... 100

`is_SSL_URI()`

Cek apakah URI yang ada pada `config.php` adalah *HTTPS*

`is_SSL_request()`

Cek apakah URI yang di-submit merupakan *HTTPS*

`minify_HTML_output()`

Melakukan kompresi out HTML dengan menghilangkan whitespace yang tidak diperlukan. Masukkan `ob_start('minify_HTML_output')` pada bagian awal dan `ob_end_flush()` pada bagian akhir script.

2.1.4 Models

Post

Model *Post* berfungsi sebagai objek untuk menyimpan data-data sebuah post, diantaranya:

- `$post_id` ID unik dari sebuah post.
- `$post_status` (active | inactive)
- `$post_type` (Post | Page)
- `$post_title` Judul utama dari sebuah post.
- `$post_name` Judul post dalam format URL, biasanya digunakan untuk *permalink*.
- `$post_content` Materi dari post.
- `$post_thumbnail_image` Image thumb, biasanya berukuran kecil.
- `$post_featured_image` Image cover, biasanya berukuran cukup besar.

Model *Post* berada pada `/model/Post.php`, dan berisi beberapa method pendukung berikut:

Post.php

Daftar method objek post untuk menyimpan data.

init (*\$post*)

Init *value* sekaligus dalam waktu bersamaan.

Kembali Object *post*.

get_{property_name}()

Mendapatkan nilai dari masing-masing *property name*.

Kembali *property value*

set_{property_name} (\$value)

Memberikan nilai ke masing-masing *property name*.

Parameter

- **\$value** (*int / string*) – Nilai untuk masing-masing *property*.

set_{property_name} (\$value)

Memberikan nilai ke masing-masing *property name*.

Parameter

- **\$value** (*int / string*) – Nilai untuk masing-masing *property*.

get_permalink (*\$structured = false*)

Mendapatkan *URL* ke masing-masing post, format *\$structured* adalah */category/post-name*, dan default */post-name*.

Kembali String dalam format *URL*.

is_empty (*\$field = 'post_id'*)

Memeriksa apakah objek *Post* NULL (kosong), beberapa field yang bisa dicek:

- *post_name*
- *post_title*
- *post_content*
- *post_date*

Kembali Boolean (*true/false*).

2.1.5 Controllers

Kelas-kelas yang berisi fungsi inti dari setiap bagian layanan.

Posts

Controller *Posts* berfungsi untuk mendapatkan data *post* dari database, kelas ini berada dalam direktori */controller/* dan terdiri dari beberapa method.

Posts.php

Daftar method untuk keperluan management post dari database.

get_instance ()

Method static yang dapat dipanggil tanpa melakukan instansiasi, berguna untuk memanggil secara cepat kontroller *Posts*, caranya *Posts::get_instance() -> {method}*.

Kembali Objek controller `Posts`.

get_post (*\$post_id*, *\$by* = 'post_id', *\$meta_key* = "", *\$meta_value* = "")

Mendapatkan 1 objek post dari database dengan mapping dari *Model Post*.

Kembali Objek *Model Post*.

get_posts (*\$params* = array())

Mendapatkan beberapa objek post *Model Post* sekaligus dengan beberapa parameter berikut:

- `post_title` judul dari post, *default empty*
- `post_content` isi dari post, *default empty*
- `post_type` jenis post, *default Post (Post | Page | -1)*
- `post_parent` ID dari post induk, *default -1*
- `post_status` status post, *default Publish (Publish | Draft | -1)*
- `exclude` array dari ID *Model Post* yang tidak ingin dimunculkan dalam hasil, *default array()*
- `category_id` ID dari kategori, *default -1*
- `category_exclude` ID dari kategori yang tidak ingin dimunculkan dalam hasil, *default array()*
- `post_month` bulan pembuatan post, *default -1*
- -tbd-

Kembali Array dari objek *Model Post*.

2.2 Modules

- db exporter
- mailer
- nationality
- currency
- fastboat
- tour
- flight

2.3 Skins

- structure

GIT Updater

GIT Source Updater merupakan script yang dapat membantu dalam melakukan update sebuah repo dalam deployment secara otomatis. Update dilakukan dengan membaca setiap commit yang di-push, selanjutnya melakukan fetch raw source dari GIT server. GIT server yang didukung saat ini hanya [Bitbucket](#) dan [Github](#), silakan kunjungi halaman [GitHub Project](#) untuk source lengkap.

3.1 How to use

Buat script php dengan nama tertentu (misalnya `update.php`), tempatkan script tersebut di ROOT project web. Untuk deployment project [LRsoft](#), kami gunakan script ini untuk melakukan update ke semua project yang menggunakan LRS Engine.

3.2 Easy setup

```
1  set_time_limit( 0 );                // set jadi unlimited
2
3  require_once( 'cURL.php' );          // class pembantu untuk keperluan URL
4  require_once( 'GitHub.php' );        // untuk github
5  require_once( 'Bitbucket.php' );     // untuk bitbucket
```

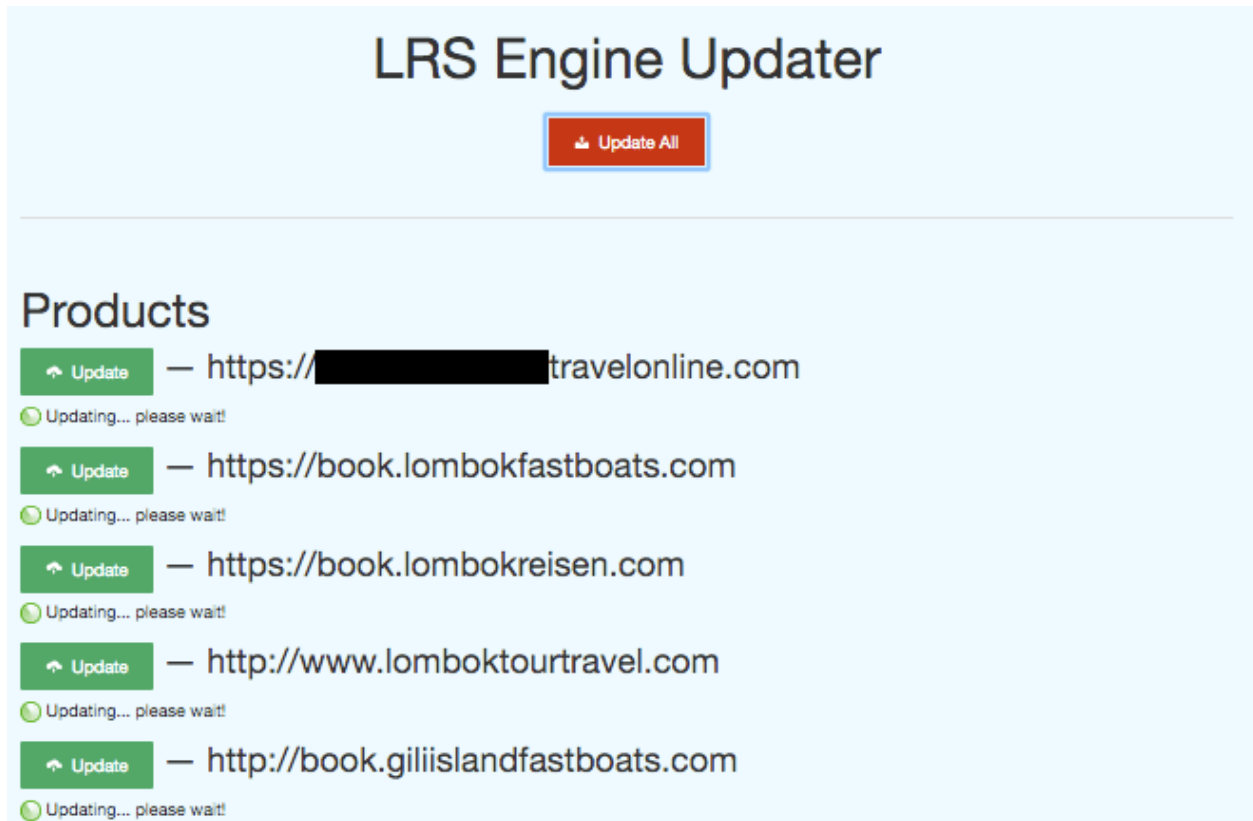
Isi credentials.

```
1  /** Bitbucket credentials */
2  $username    = 'User';               // username
3  $password    = 'password';           // password
4  $account_slug = 'user';              // user slug, biasanya terdapat pada URL repo
5  $repo_slug   = 'repo';               // repo slug, biasanya juga terdapat pada URL
    ↪ repo
```

Ekekusi!

```
1 $github = new GitHub( $username, $password, $account_slug, $repo_slug );  
2 $status = $github->_init()->get_status();           // untuk keperluan debugging
```

3.3 Screenshot



LRS Engine Updater

[Update All](#)

Products

[Update](#) — [https://\[REDACTED\]travelonline.com](https://[REDACTED]travelonline.com)

✓ Results

/home/[REDACTED]module/fastboat/controller/FastboatReservations.php updated.
/home/[REDACTED]module/fastboat/follow-up.php updated.
VERSION.md updated.

[Update](#) — [https://\[REDACTED\]fastboats.com](https://[REDACTED]fastboats.com)

✓ Results

/home/[REDACTED]module/fastboat/controller/FastboatReservations.php updated.
/home/[REDACTED]module/fastboat/follow-up.php updated.
VERSION.md updated.

[Update](#) — [https://\[REDACTED\]reisen.com](https://[REDACTED]reisen.com)

✓ Results

/home/[REDACTED]module/fastboat/controller/FastboatReservations.php updated.
/home/[REDACTED]module/fastboat/follow-up.php updated.
VERSION.md updated.

Others

 Update — <http://www.geometrin.com>

✓ Results

VERSION.md updated.
Engine's up to date!

 Update — <http://kpu-mataramkota.go.id>

✓ Results

VERSION.md updated.
Engine's up to date!

 Update —  bokutara.go.id

✓ Results

VERSION.md updated.
Engine's up to date!

Demo

 Update — <http://projects.lrsoft.org/demo/scootcruise>

✓ Results

/home/ module/fastboat/controller/FastboatReservations.php updated.
/home/ module/fastboat/follow-up.php updated.
VERSION.md updated.



3.4 References

class GitHub

GIT server dari GitHub

__construct (\$username, \$password, \$account_slug)

Basic init.

Parameter

- **\$username** (*string*) – Username akun GitHub.
- **\$password** (*string*) – Password akun GitHub.
- **\$account_slug** (*string*) – Slug akun, biasanya terdapat pada URL.

Kembali Class object

__init ()

Jalankan proses update.

Kembali Class object

set_branch (\$branch)

Set branch tertentu.

Parameter

- **\$branch** (*string*) – Nama branch.

Kembali Class object

append_ignored_file (*\$ignored_file*)

Tambahan pengecualian untuk file-file tertentu yang tidak diinginkan.

Parameter

- **\$ignored_file** (*string*) – Nama (dan path) file.

Kembali Class object

get_status ()

Mendapatkan status perubahan dari tiap file.

Kembali Array perubahan file

class Bitbucket

GIT server dari Bitbucket

__construct (*\$username*, *\$password*, *\$account_slug*)

Basic init.

Parameter

- **\$username** (*string*) – Username akun Bitbucket.
- **\$password** (*string*) – Password akun Bitbucket.
- **\$account_slug** (*string*) – Slug akun, biasanya terdapat pada URL.

Kembali Class object

__init ()

Jalankan proses update.

Kembali Class object

set_branch (*\$branch*)

Set branch tertentu.

Parameter

- **\$branch** (*string*) – Nama branch.

Kembali Class object

set_limit (*\$limit*)

Set limit selama fetching list commit dari suatu branch.

Parameter

- **\$limit** (*int*) – Batas jumlah commit.

Kembali Class object

append_ignored_file (*\$ignored_file*)

Tambahan pengecualian untuk file-file tertentu yang tidak diinginkan.

Parameter

- **\$ignored_file** (*string*) – Nama (dan path) file.

Kembali Class object

get_status ()

Mendapatkan status perubahan dari tiap file.

Kembali Array perubahan file

Symbols

() (method), [8–11](#)
__construct() (method Bitbucket), [17](#)
__construct() (method GitHub), [16](#)
_init() (method Bitbucket), [17](#)
_init() (method GitHub), [16](#)

A

append_ignored_file() (method Bitbucket), [17](#)
append_ignored_file() (method GitHub), [17](#)

B

Bitbucket (class), [17](#)

G

get_status() (method Bitbucket), [17](#)
get_status() (method GitHub), [17](#)
GitHub (class), [16](#)

S

set_branch() (method Bitbucket), [17](#)
set_branch() (method GitHub), [16](#)
set_limit() (method Bitbucket), [17](#)