Quick Reference

System Variables

To use these variables, simply get them from the $f3 object, as in:

```
$current_url = $f3->get('PATH');
```

AGENT

Type: `string` , `Read-only`

A string containing the auto-detected HTTP user agent, e.g. `'Mozilla/5.0 (Linux; Android 4.2.2; Nexus 7) AppleWebKit/537.31'`

AJAX

Type: `bool` , `Read-only`

`TRUE` if an XML HTTP request is detected, `FALSE` otherwise. Default value: Result of the expression `$headers['X-Requested-With']=='XMLHttpRequest'`

ALIAS

Type: `string`

Contains the alias (name) of the current route. Empty if the current route is not named.

ALIASES

Type: `array`

This array contains all named routes (routing-engine#named-routes) which can be used to render the appropriate link urls in your templates.

AUTOLOAD

Type: `string|array` **Default:** `'./'`

Search path(**s**) for user-defined PHP classes that the framework will attempt to autoload at runtime. When specifying multiple paths, you can use a pipe (`|`), comma (`,`), or semi-colon (`;`) as path separator.

REMEMBER: paths must end with a slash. E.g: `$f3->set('AUTOLOAD', 'app/;inc/,./');`

See here (routing-engine#the-f3-autoloader) for more details.

BEWARE: if you define this variable in a config file, remember that commas have a special meaning (array separator).
So the following syntaxes are valid:

- `AUTOLOAD = foo/;bar/,Customer::casehandler`
- `AUTOLOAD = foo/;bar/`
- `AUTOLOAD = foo/|bar/`
- `AUTOLOAD = "foo/,bar/"`

while `AUTOLOAD = foo/,bar/` is not.

BASE

Type: `string` , `Read-only` **Default:** auto-detected

Path to the `index.php` main/front controller.

BODY

Type: `string` , `Read-only`

HTTP request body for ReSTful post-processing. Contains the `php://input` stream used by PUT requests, if RAW (quick-reference#raw) is `false` .

CACHE

Type: `bool|string` **Default:** `FALSE`

Cache backend. F3 can handle Memcache module, APC, WinCache, XCache and a filesystem-based cache.

For example: if you'd like to use the memcache module, a configuration string is required, e.g. `$f3->set('CACHE','memcache=localhost')` (port 11211 by default) or `$f3->set('CACHE','memcache=192.168.72.72:11212')` .

When set to `TRUE` , or when the connection with the specified memcached server above failed, F3 will auto-detect, in that order, the presence of APC, WinCache, XCache and use the first available of these PHP module. If none of these shared memory engine has been detected or is available, a filesystem-based backend is used as a fallback (default directory: `tmp/cache` or you can specify a folder outside the scope of the website, e.g. `$f3->set('CACHE','folder=/var/tmp/f3filescache/')` .

The framework doesn't use any cache engine when a `FALSE` value is assigned.

CASELESS

Type: `bool` **Default:** `TRUE`

Pattern matching of routes against incoming URIs is case-insensitive by default. Set to `FALSE` to make it case-sensitive.

CLI

Type: `bool` , `Read-only`

`TRUE` if the request originates from the command-line interface, `FALSE` if it comes from the web server.

See CLI mode (routing-engine#RoutinginCLImode) for more details on how to handle CLI requests.

CONTAINER

Type: `callable|Prefab|Psr\Container\ContainerInterface`

Defines the optional dependency injection container used by `Base->call()` (base#call) and the routing system. `CONTAINER` supports PSR-11 containers, callables (base#call) and classes extending `Prefab` . `Prefab` -based classes have to implement the `get(string $id)` method. Callables receive the requested `$id` (e.g. class name) as first argument.

API-incompatible third party containers can be made compatible with a tiny adapter.

```
$dice = … // Configure the API-incompatible Level-2/Dice container.

$f3->set('CONTAINER', function ($class) use ($dice) {
    return $dice->create($class);
});
```

NB: `CONTAINER` requires at least Fat-Free Framework `3.6.4` .

COOKIE, GET, POST, REQUEST, SESSION, FILES, SERVER, ENV

Type: `array`

Framework equivalents of PHP globals. For your convenience, F3 automatically synchronizes these variables with the underlying PHP globals. These variables may be used throughout an application. However, direct use in templates is not advised due to security risks.

It could be possible that the PHP configuration is not populating all globals. If for instance the environment variables are missing, then you have to add `E` to the PHP configuration directive `variables_order` (https://www.php.net/manual/en/ini.core.php#ini.variables-order).

CORS

Type: `array`

Cross-Origin Resource Sharing (https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) configuration parameters. Consists of the following options:

- `headers` , string or array, default: `''` , allowed headers in the request
- `origin` , string or false, default: `false` , allowed origin host, i.e `*.mydomain.com`

- `credentials` bool, default: `false` , allow cookies
- `expose` , string or array, default: `false` , controls which headers from the response are exposed to the client browser
- `ttl` , int, detault: `0` , caching time of the preflight OPTIONS request

To enable basic CORS support, just set `CORS.origin` to `*` . For a more defined setup, you can use `$f3->copy('HEADERS.Origin','CORS.origin');` .

DEBUG

Type: `integer` **Default:** `0`

Verbosity level of the stack trace. Assign values between 0 to 3 for increasing verbosity levels as follow:

- 0 : suppresses logs of the stack trace.
- 1 : logs files & lines.
- 2 : logs classes & functions as well.
- 3 : logs detailed infos of the objects as well.

Notice: Only the default value of `0` should be used on production servers.

DIACRITICS

Type: `array` **Default:** `array()` , empty array

Additional key-value pairs for foreign-to-ASCII character translations, as used in web->slug (web#slug).

DNSBL

Type: `string` **Default:** `''` , empty string

Comma-separated list of DNS blacklist servers (http://whatismyipaddress.com/blacklist-check). Framework generates a `403 Forbidden` error if the user's IPv4 address is listed on the specified server(s).

EMOJI

Type: `array` **Default:** `array()` , empty array

Additional key-value pairs of emoji tokens to add to the basic set used when translating a string to Unicode font-supported symbols. (see `\UTF->emojify()` (utf-unicode-string-manager#emojify))

ENCODING

Type: `string` **Default:** `'UTF-8'`

Character set used for document encoding (views-and-templates#document-encoding).

ERROR

Type: `array` , `Read-Only`

Information about the last HTTP error that occurred:

- `ERROR.code` is the HTTP status code (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html). e.g. `307`
- `ERROR.status` is a brief description of the HTTP status code. e.g. `'Temporary Redirect'`
- `ERROR.text` contains a brief description of the error.
- `ERROR.trace` is used for HTTP 500 errors, to retrieve the stack trace. `string`
- `ERROR.level` - error reporting level (`E_WARNING` , `E_STRICT` , etc.)

ESCAPE

Type: `bool` **Default:** `TRUE`

Used to enable/disable auto-escaping @tokens (/quick-reference#token) used in templates.

EXEMPT

Type: `string` **Default:** `NULL`

Comma-separated list of IPv4 addresses to exempt from DNSBL lookups.

EXCEPTION

Type: `object` **Default:** `NULL`

Contains the exception object when unhandled exceptions occur.

FALLBACK

Type: `string` **Default:** `'en'`

Language (and dictionary) to use if no translation is available.

FORMATS

Type: `array`

Storage for custom format rules to add support for multiple localization formats or other cases. See code samples (https://github.com/F3Community/snippets/tree/master/formats).

```
$f3->set('FORMATS.polish','FormatHelper->polish');
```

FRAGMENT

Type: `string` **Default:** `NULL`

Portion of the URI after the optional hash (#) symbol `(http://www.example.org/foo.html#bar)` FRAGMENT = 'bar'.

HALT

Type: `bool` **Default:** `TRUE`

If `TRUE` , the framework, after having logged stack trace and errors, stops execution (`die` without any status) when a *non-fatal* error is detected.

HEADERS

Type: `array` , `Read-Only`

HTTP request headers received by the server. e.g. (simplified)

```
array (
    'Host' => 'fatfreeframework.com'
    'Accept-Encoding' => 'gzip,deflate,sdch',
    'Accept-Language' => 'en-US,en;q=0.8,ja;q=0.6'
)
```

HIGHLIGHT

Type: `bool` **Default:** `FALSE`

Enable/disable syntax highlighting of stack traces and Markdown (markdown) code blocks. When enabled, requires `code.css` stylesheet.

HOST

Type: `string` , `Read-Only`

Server host name.

IP

Type: `string` , `Read-Only`

Remote IP address. The framework derives the address from headers if HTTP client is behind a proxy server. Default value: First match of `Client-IP` then `X-Forwarded-For` then `$_SERVER['REMOTE_ADDR']` , otherwise set to `''`

JAR

Type: `array`

Default cookie parameters. Consists of the following options:

- `expire` Unix timestamp, when the cookie should expire. Default: `0`
- `path` The path on the server in which the cookie will be available. Default: `'/'`
- `domain` The domain that the cookie is available to. Default: `$_SERVER['SERVER_NAME']` if available, else `''`
- `secure` Set the cookie when a secure HTTPS connection exists. Default: `$_SERVER['HTTPS']=='on'`

- `httponly` Make the cookie accessible only through the HTTP protocol. Default: `TRUE`

You can refer to session_set_cookie_params() in the PHP Manual (http://www.php.net/manual/en/function.session-set-cookie-params.php) for more information.

You can also watch a video (https://youtu.be/a-zTNeDeqU0) that goes over using cookies in the Fat-Free Framework.

LANGUAGE

Type: `string` **Default:** auto-detected

Current active language(s). Value is used to load the appropriate language(s) translation file(s) in the folder pointed to by `LOCALES` . Default: auto-detected from the HTTP `Accept-Language` request header, e.g. `'en-US,en,es'` .

NB: The system locale is loaded accordingly to this variable. E.g:

```
$f3->set('ENCODING','UTF-8');
$f3->set('LANGUAGE','it-IT');// the locale it_IT.UTF-8 will be automatically loaded u
sing setlocale
```

See the Localisation section (base#language) in `Base` for more details and an example.

LOCALES

Type: `string` **Default:** `'./'`

Location of the language(s) dictionaries.

To enable caching for dictionaries from a config file, you need to write it this way:

```
LOCALES=/path/to/lexicons | 3600
```

LOGGABLE

Type: `string|array` **Default:** `'*'`

You can supply this with either an array or a comma/semi-colon separated list of HTTP status codes to allow to be passed into the `error_log()` function when an error occurs. This is particularly useful when you are building a CLI app with FatFree routes and need to intercept a 404 not found error and display a custom message or action.

```
LOGGABLE='403;500;'
```

LOGS

Type: `string` **Default:** `'./'`

Location of custom logs.

ONERROR

Type: `mixed` **Default:** `NULL`

Callback function to use as custom error handler, or `NULL` .

> **Notice:** If no callback function is specified, a default error page is generated (HTML5 for synchronous requests, JSON string for AJAX requests).

ONREROUTE

Type: `mixed` **Default:** `NULL`

Callback function that is called before redirect headers are send. The default behavior (301/302 redirection) will be bypassed unless `FALSE` is returned.

```
$f3->set('ONREROUTE',function($url,$permanent){
  // do something
});
$f3->reroute('/foo?bar=baz');
```

PACKAGE

Type: `string|null` **Default:** `'Fat-Free Framework'`

A string containing the `X-Powered-By` header.

If empty, the header is not sent.

PARAMS

Type: `array` **Default:** `array()`

Captured values of tokens defined in a `route()` pattern. `PARAMS[0]` contains the captured URL relative to the Web root.

PATH

Type: `string` , `Read-Only`

The URL relative to BASE. Default value: `parse_url($_SERVER['REQUEST_URI'],PHP_URL_PATH)`

PATTERN

Type: `string` , `Read-Only`

Contains the routing pattern that matches the current request URI.

PLUGINS

Type: `string` **Default:** `__DIR__.'/'`

Location of F3 plugins. The default value is the folder where the framework code resides, i.e. the path to `base.php`.

PORT

Type: `integer`, `Read-Only`

TCP/IP listening port used by the Web server. Default value: `$_SERVER['SERVER_PORT']` or `NULL` if not available.

PREFIX

Type: `string` **Default:** `NULL`

Prefix to use with LANGUAGE and LOCALES.

For example, if your dictionary file contains `hello = Hello World`, the term will be accessible via:

- `$f3->get('hello')` without prefix
- `$f3->get('DICT.hello')` if `PREFIX=DICT.` (Notice the `.`, it's intentional)

IMPORTANT: this variable should be set *before* `LANGUAGE` and `LOCALES`.

PREMAP

Type: `string` **Default:** `''`, empty string

This variable allows mapped (routing-engine#ReST:RepresentationalStateTransfer) route handlers to be prefixed. For example, defining:

```
$f3->PREMAP = 'action_';
$f3->map('/item','Item');
```

is the same as defining:

```
$f3->route('GET /item','Item->action_get');
$f3->route('POST /item','Item->action_post');
$f3->route('PATCH /item','Item->action_patch');
$f3->route('PUT /item','Item->action_put');
$f3->route('DELETE /item','Item->action_delete');
```

QUERY

Type: `string`, `Read-Only`

Contains the request URI query string (all after the question mark `?`).

QUIET

Type: `bool` **Default:** `FALSE`

Toggle switch for suppressing or enabling standard output and error messages. Particularly useful in unit testing (unit-testing).

RAW

Type: `bool` **Default:** `FALSE`

RAW should be TRUE when processing large data coming from `php://input` which will not fit in memory (cf. BODY (quick-reference#body)).

REALM

Type: `string` , `Read-Only`

Full canonical URL. Default value: Result of `'http(s)://'.$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI']`

RESPONSE

Type: `string` , `Read-Only`

The body of the last HTTP response. F3 populates this variable regardless of the `QUIET` setting.

ROOT

Type: `string` , `Read-Only`

Absolute path to document root folder.

ROUTES

Type: `array` **Default:** `array()`

Contains the defined application routes.

> **Notice:** A route is more than just a URL. It's an HTTP verb (or verbs) and an URL.

SCHEME

Type: `string` , `Read-Only`

Server protocol. Default: `'http'` or `'https'`

SEED

Type: `string`

The SEED string is used as prefix name for cache entries and temp filenames to avoid cache key collisions. In case you use multiple domains with your application, the auto-generated SEED value would be different by default. If you want to share a common cache and temp file storage across both domains, set a custom SEED before initializing the CACHE:

```
$f3->set('SEED', $f3->hash('myDomainSEED'));
$f3->set('CACHE', TRUE);
```

NB: The SEED key is also used for CSRF token generation within the Session handlers.

SERIALIZER

Type: `string` **Default:** auto-detected

The default serializer used by the Base->serialize() (base#serialize) method. Default value: `igbinary` if available, otherwise `php`.

TEMP

Type: `string` **Default:** `'tmp/'`

Temporary folder for cache, filesystem locks, compiled F3 templates, etc. The default value is the `'tmp/'` folder inside the Web root. *Adjust accordingly to conform to your site's security policies*.

When you're using Google App Engine (GAE) to deploy your application, it's recommended to set it to a cloud storage dir.

TIME

Type: `float` **Default:** auto-detected

Starting time of the framework. Default value: The current Unix time in seconds accurate to the nearest microsecond as per the PHP function microtime(**TRUE**) (http://php.net/microtime).

TZ

Type: `string` **Default:** auto-detected

Timezone to use. Changing this value automatically calls the underlying PHP function `date_default_timezone_set()`. See the list of supported timezones (http://php.net/manual/en/timezones.php) to get a possible value to use here. Falls back to `'UTC'` if auto-detection fails.

UI

Type: `string` **Default:** `'./'`

Search path for user interface files used by the `View` and `Template` classes' `render()` method. Accepts a pipe (`|`), comma (`,`), or semi-colon (`;`) as separator for multiple paths.

UNLOAD

Type: `callback` **Default:** `NULL`

Defines the shutdown handler the framework will execute on application shutdown (base#unload).

UPLOADS

Type: `string` **Default:** `'./'`

Directory where file uploads are saved.

URI

Type: `string` **Default:** auto-detected

A reference to the current HTTP request URI.

VERB

Type: `string` **Default:** auto-detected

A reference to the current HTTP request method.

VERSION

Type: `string` **Default:** e.g. `'3.2.1-Release'`

A string containing the version of the Framework.

XFRAME

Type: `string|NULL` **Default:** e.g. `'SAMEORIGIN'`

A string containing the `X-Frame-Options` header.

If empty, the header is not sent.

Template Directives

Token

- `@token`

  Replace `@token` with the value of the equivalent F3 variable.

- `{{ mixed expr }}`

Evaluate expression `expr`. Expression can include template tokens, constants, operators (unary, arithmetic, ternary and relational), parentheses, data type converters, and functions. *If not an attribute of a template directive, the result is echo'ed*.

- `{{ string expr | esc }}`

  Render expression `expr` *escaped*. This is the default framework behavior. The `| esc` suffix is only necessary if the ESCAPE (/quick-reference#error) global variable has been set to `FALSE`.

- `{{ string expr | raw }}`

  Render expression `expr` *unescaped*. As F3 auto-escapes string tokens by default, you can use this suffix to by-pass the escaping of a particular token.

- `{{ string expr, arg0, …, argN | format }}`

  Render the expression `expr` in ICU-format and pass the comma-separated arguments, where `arg0, …, argN` is used in `expr` as reference, each with an optional formatter that can be one of: `'date'`, `'time'`, `'number'` or `'plural'` (additional formatter options possible). Have a look at the format (base#format) method for more usage examples. More information about ICU formatting of Numbers, Currencies, Dates and Times (http://userguide.icu-project.org/formatparse/). Sample:
  `{{'date: {0,date} - time: {0,time} - price: {1,number,currency}',time(),@price | format}}`

- `{{ string name, args | alias }}`

  builds an URL of a named route, i.e: `{{ @name, 'a=5,b='.@id | alias }}`

- `{~ string expr ~}`

  Evaluate expression `expr`, similar to `{{expr}}` but does not echo the result.

- `{* text-block *}`

  Exclude a segment of your template. Alias to `<exclude>`

- `{- {{@BASE}} -}`

  Ignore all tokens within `{- -}` expression and print them as they are.

Include

```
<include [ if="{{ bool condition }}" ] href="{{ string subtemplate }}" [ with="{{ str
ing additional_variables }}" ] />
```

Get contents of `subtemplate` and insert at current position in template [if optional `condition` is true].

The current data hive is passed to the subtemplate, enriched with `additional_variables` if provided (see here (views-and-templates#templates-within-templates) for examples).

Exclude

```
<exclude>text-block</exclude>
```

Exclude `text-block` at runtime. Used for embedding comments in templates. An Alias for this is:

```
{* text-block *}
```

Ignore

```
<ignore>text-block</ignore>
```

Display `text-block` as it is, without any interpretation/modification by the template engine.

Check

```
<check if="{{ bool condition }}">
    <true>true-block</true>
    <false>false-block</false>
</check>
```

Evaluate `condition`. If `TRUE`, the `true-block` is rendered; else the `false-block` is rendered.

Short form: If you don't need and don't specify a false block, then, for your convenience, F3 makes the opening and closing tags for true optional:

```
<check if="{{ @debugmode && @showtrace }}"><code>{{ @showtrace }}</code></check>
```

Loop

```
<loop from="{{ statement }}" to="{{ bool expr }}" [ step="{{ statement }}" ]>
    text-block
</loop>
```

Evaluate `from` statement once. Check if the expression in the `to` attribute is `TRUE`, render `text-block` and evaluate `step` statement. Repeat iteration until `to` expression is `FALSE`.

Example:

```
<loop from="{{ @i=0 }}" to="{{ @i < count(@bar) }}" step="{{ @i++ }}">

</loop>
```

Repeat

```
<repeat group="{{ array @group|expr }}" [ key="{{ scalar @key }}" ] value="{{ mixed @
value }}" [ counter="{{ scalar @key }}" ]>
    text-block
</repeat>
```

Repeat `text-block` as many times as there are elements in the array variable `@group` or the expression `expr`. `@key` and `@value` function in the same manner as the key-value pair in the equivalent PHP `foreach()` statement. Variable represented by `key` in `counter` attribute increments by `1` with every iteration.

Switch

```
<switch expr="{{ scalar expr }}">
    <case value="{{ scalar @value|expr }}" break="{{ bool TRUE|FALSE }}">
        text-block
    </case>
    .
    .
    .
    <default>
        message
    </default>
</switch>
```

Equivalent of the PHP switch-case jump table structure.

Set

```
<!-- set some variables -->
<set foo="{{ 1+2 }}" bar="{{ @foo+3 }}" baz="xyz" />
<!-- set an array -->
<set myarray="{{ array('a','b','c') }}" />
```

Used to set some variables dynamically within the template.

← 8. Unit Testing (unit-testing)