Fat-Free Framework 1.4.4 Quick Reference

Akismet

An API wrapper that you can use during input validation to determine if a blog comment, trackback, or pingback contains spam. This plug-in requires a key from akismet.com.

check(string **text,** string **author,** string **email,** string **url);**

Submit content (usually a blog comment) to akismet.com. Returns TRUE if the content is determined as spam.

ham(string **text,** string **author,** string **email,** string **url);**

Report content as a false positive.

spam(string **text,** string **author,** string **email,** string **url);**

Report content that was not marked as spam.

verify(string **key);**

Secure approval from akismet.com to use the public API for spam checking. A valid key is required to use the API. Returns TRUE if authentication succeeds.

Atom/RSS

Atom/RSS feed reader plugi-n.

read(string **url, [** integer **count], [** string **tags]);**

Grab Atom/RSS feed entries from URL and return as an array. Number of elements and allowed HTML tags may be specified.

Auth

A plug-in for various user authentication methods. The AUTH framework variable (array) must be set up prior to use of any of these methods.

basic(string **method, [** string **realm]);**

Basic HTTP authentication using the specified method (ldap, imap, sqldb, or nosqldb).

imap(string **user_id,** string **password);**

Authenticate user credentials against an IMAP server. AUTH should contain the following elements - server: IP address or host name of IMAP server, port: TCP port.

ldap(string **user_id,** string **password);**

Authenticate user credentials against an LDAP/ActiveDirectory server. AUTH should contain the following elements - dc: domain controller, rdn: connection DN, pw: connection password.

nosqldb(string **user_id,** string **password);**

Authenticate user credentials against a MongoDB server using the M2 mapper. AUTH should contain the following elements - db: database identifier, collection: collection name, id: user ID field, pw: password field. Returns FALSE if authentication fails, or the M2 instance representing the NoSQL object if successful.

sqldb(string **user_id,** string **password);**

Authenticate user credentials against a SQL database using the Axon mapper. AUTH should contain the following elements - db: database identifier, table: SQL table name, id: user ID field, pw: password field. Returns FALSE if authentication fails, or the Axon instance representing the SQL record if successful.

Data

Input data handlers and validators.

input(string **fields,** mixed **handler, [** string **tags],[** integer **filter], [** array **options]);**

Assign **handler** to HTML form fields for validation and manipulation. **handler** may be an anonymous or named function, a single or daisy-chained string of named functions similar to the route() method. This command passes two arguments to **handler** function: value and field name. HTML and PHP tags are stripped If the argument **tags** is not specified. PHP validation/sanitize filters, filter flags, and options may be passed as additional arguments. See the PHP filter_var() function for more details on filter types.

scrub(mixed **value, [** string **tags]);**

Remove all HTML tags to protect against XSS/SQL injection attacks. **tags**, if specified, will be preserved. If **value** is an array, HTML tags in all string elements will be scrubbed.

validCaptcha(string **hex);**

Return TRUE if SESSION.captcha is set and identical to the specified hex string.

validEmail(string **address);**

Return TRUE if e-mail address is valid, FALSE otherwise.

validURL(string **url);**

Return TRUE if url is valid, FALSE otherwise.

Log

A simple custom logger.

new Log(string **filename);**

Instantiate logger and create file.

write(string **text);**

Send text data to logger.

Expansion

A hodge-podge of tools that extend the framework's basic features.

binhex(string **binary_data);**
Convert binary-packed data to hexadecimal.

expect(mixed **condition,** string **true,** string **false);**
Test condition and append result to the framework's **TEST** variable.

hexbin(string **hex_data);**
Convert hexadecimal to binary-packed data.

http(string **pattern, [** string **content], [** string **headers], [** boolean **follow]);**
Send HTTP/S request to another host; forward headers received and return host's response. pattern consists of an HTTP GET/HEAD/POST/PUT/DELETE command, a space and an absolute URI. pattern should include port number after host name if not using a standard TCP port (80 for HTTP, 443 for HTTPS). content is optional URI query string. Respect HTTP 30x redirects if follow is TRUE (default).

isAjax(void **);**
Return TRUE if HTTP request origin is AJAX.

minify(string **path,** array **filenames);**
Compress specified files by removing all whitespaces and comments, then send as a single block of CSS or Javascript.

pick(string **variable,** mixed **column);**
Retrieve the specified **column** of a framework array **variable**.

send(string **file, [** string **kbps], [** boolean **partial]);**
Transmit file to HTTP client. Limit bandwidth usage, if speed in kilobytes per second is specified. Support partial downloads if third argument is present (default is TRUE).

sign(mixed **number);**
Returns -1 if the specified number is negative, 0 if zero, or 1 if the number is positive.

sitemap(string **root_url);**
Generate XML sitemap starting at specified url.

slug(string **text, [** integer **max_length]);**
Returns an RFC 1738-compliant URL-friendly version of specified string. Result is truncated if max_length characters is present.

transpose(string **variable,** boolean **replace);**
Rotate a two-dimensional framework array variable, so rows become columns, and vice versa. Content of variable is replaced if second argument is TRUE (default).

Geo

World-related reference tables.

countries(void **);**
Return array of countries indexed by 2-letter country code.

location([string **ip_address]);**
Return geolocation information related to the IP address. The framework auto-detects the IP address if not provided.

timezones(void **);**
Return zoneinfo array indexed by Unix time zone. Each time zone is defined as an array containing UTC offset, 2-letter country code, latitude, longitude, and daylight-saving time (dst) usage.

weather(float **latitude,** float **longitude);**
Return an array of current weather conditions at a specific location defined by latitude and longitude.

Google

A nice collection of Google AJAX API adaptors for Google's language translator, Web search, Google Maps and Atom/RSS feeds.

feed(string **url, [** boolean **is_xml]);**
Retrieve Atom/RSS feed using Google AJAX Feed API. If is_xml is TRUE (default), XML string is returned, otherwise, a PHP array.

map(string **center, [** integer **zoom], [** string **size], [** string **type], [** string **format], [** string **language], [** array **markers]);**
Generate a static map using Google Maps API. center defines the center of the map (a string address, e.g. 'city hall, new york, ny' or a comma-separated latitude-longitude pair, e.g. '40.714728, -73.998672'. zoom determines magnification level (default is 15). size defines rectangular dimensions (default is 400x400). type should be one of these values: roadmap, satellite, or hybrid (roadmap default). format should be png, jpeg or gif (png default), and language is used for the display of labels on the map (en default). markers define one or more markers to attach to the image at specified locations (see http://code.google.com/apis/maps/documentation/staticmaps/#Markers for details).

search(string **text, [** integer **page]);**
Perform a Google search on specified text (starting at zero-indexed page). Result is an array with a maximum of eight (8) search hits.

translate(string **text,** string **from,** string **to);**
Translate text from one language to another. from and to must be valid ISO 639-1 alpha-2 language codes supported by the Google language translator.

Graphics

Graphics generators dependent on PHP's GD extension.

captcha(integer **width,** integer **height,** integer **length, [** string **font]);**
Generate CAPTCHA image of specified width and height (in pixels). length represents the number of hex characters to generate. The default TrueType font is cube.ttf. The string equivalent of the CAPTCHA image is stored in $_SESSION ['captcha'] and the framework variable SESSION['captcha'].

fakeimage(integer **width,** integer **height,** integer **bgcolor);**
Generate a blank image for use as a placeholder.

identicon(string **hash, [** integer **size]);**
Generate icon from a hex-string hash value. Default size is 64 pixels.

invert(string **filename);**
Invert colors of specified JPG/GIF/PNG image.

thumb(string **filename,** integer **width,** integer **height);**
Generate thumbnail image of specified maximum width and maximum height. Resulting image has the same aspect ratio (proportions) as original.

Network

Network utilities (requires PHP sockets extension).

ping(string **address,** boolean **dnsresolve,** integer **packets,** integer **wait);**
Send ICMP echo requests to specified address; Returns an array containing: min, avg, and max round-trip time (milliseconds), and number of packets received, or FALSE if host is not reachable. Resolves IP address to host name if dnsresolve is TRUE. Default settings - packets: 3 and wait time: 3 seconds.

privateIP(string **address);**
Returns TRUE if IP address is local or within a private IPv4 range.

realIP(void **);**
Sniff headers for real IP address and returns the detected IPv4 address.

spam(string **address);**
Returns TRUE if IPv4 address is listed in spam database.

traceroute(string **address,** boolean **dnsresolve,** integer **wait,** integer **hops);**
Returns an array representing the path taken by packets to a specified network destination.. Default wait time is 3 seconds per ICMP echo request and 30 hops.

SQLdb

A stand-alone API wrapper for SQL databases.

sql(mixed **statement, [** string **db], [** integer **timeout]);**
Execute SQL **statement**. If an array of **statements** is specified, it is treated as a single SQL transaction - in which case, commit and rollback are dependent on the successful completion of the transaction. If **timeout** is specified, previously-generated output associated with the SQL statement is retrieved from cache (if timer has not expired). Default **db** identifier is 'DB'.

sqlbind(string **statement,** array **kv_pairs, [** string **db], [** integer **timeout]);**
Similar to the sql() method but allows late binding of parameters/placeholders in the SQL statement to corresponding values (represented as key-value pairs in the 2nd argument), thereby allowing the underlying database driver to apply the appropriate escaping rules for various data types.. Values in the associative array **kv_pairs** may be strings or an array consisting of a value and a PDO class data type.

type(mixed **value);**
Returns PDO class constant corresponding to the argument's data type.

Twitter

A Twitter API wrapper. Search for tweets matching specified text, get information about a user, friends and followers.

friends(string **user_id);**
Retrieve information about a user's friends.

followers(string **user_id);**
Retrieve information about a user's followers.

search(string **text, [** integer **page], [** string **since], [** string **type], [** string **language]);**
Find all tweets that match specified text. page index starts at 1 (default). since follows YYYY-MM-DD format (current date default). type should be mixed, recent, or popular (mixed default). language must be valid ISO 639-1 alpha-2 language code. Returns a maximum of 10 tweets.

show(string **user_id);**
Get information about specified user_id.

Yahoo

API wrapper for Yahoo! Web services.

inlinks(string **appid,** string **path, [** integer **count], [** integer **start], [** boolean **omit]);**
Retrieve info about inbound links to a particular Web page. Yahoo! appid (application ID) and path (query URL) are required. This method returns FALSE if an error occurred or an array containing maximum count links (default 100, also the maximum per query). start is used for pagination within the result set (starting at 1). omit may be an empty string (default), 'domain' or 'subdomain' - determines whether links from the same domain/subdomain should be omitted.

online(string **id);**
Return TRUE if specified Yahoo! user ID is online.

ping(string **url);**
Notify Yahoo! of changes to the Web page pointed to by the url.

Zip

A general-purpose utility class for handling ZIP archives.

new Zip(string **file);**
Create/open a ZIP archive.

clear(string **path);**
Delete specified file from ZIP archive.

get(string **path);**
Return content of specified file from ZIP archive.

set(string **path, [** string **content]);**
Add or replace file in ZIP archive using specified content. Create folder if content is NULL or unspecified.

Axon ORM

An easy-to-use object-relational mapper for representing SQL data as objects. Depends on the SQLdb plug-in.

new Axon(string **table, [** string **db]);**
Instantiate Axon object and map to specified **table**. Default cache timeout is 60 seconds (see SYNC variable). Adjust if necessary. Default **db** identifier is 'DB'.

copyfrom(string **name);**
Hydrate Axon with elements from framework array variable, keys of which must be identical to field names in the record.

copyto(string **name, [** string **fields]);**
Populate framework array variable with Axon properties, keys of which will have names identical to fields in the record. Limit record replication to specified **fields**, if present.

def(string **name,** string **expression);**
Create a virtual field and assign SQL **expression** for use in queries.

dry(void **);**
Returns TRUE if Axon is devoid of values in its properties.

erase(void **);**
Delete record and reset Axon.

find([string **criteria], [** string **order], [** string **limit], [** integer **timeout]);**
Returns an array of database records matching specified **criteria**. If **criteria** is omitted or NULL, all records in table are returned. Result is sorted according to the sequence of fields specified in **order** and short-listed by **limit**.

findone([string **criteria], [** string **order], [** string **limit], [** integer **timeout]);**
Returns the first record that matches the specified **criteria**.

found(string **criteria]);**
Returns number of records that match specified **criteria**.

isdef(string **name);**
Returns TRUE if specified virtual field exists, FALSE otherwise.

load(string **criteria, [** string **order], [** integer **offset]);**
Retrieve first record that satisfies **criteria**, sort according to comma-separated list of fields specified in **order**, and starting at the relative **offset.**

lookup(string **fields, [** string **criteria], [** string **grouping], [** string **order], [** string **limit], [** integer **timeout]);**
Similar to Axon find() method but provides fine-grained control over specific **fields** and **grouping** of results. The select() method is an alias of this method.

reset(void **);**
Dehydrate Axon so instantiation of new Axon is not necessary.

save(void **);**
Insert or update record based on contents of Axon properties.

skip([integer **offset]);**
Retrieve the record at the offset relative to current position following the same criteria used to hydrate the Axon. Positive **offset** moves forward, negative backward. Zero re-syncs Axon. Triggers error if skipping beyond bounds of hydration criteria.

sync(string **table, [** string **db]);**
Synchronize new Axon and **table** structure. Default timeout is 60 seconds (see SYNC variable). Called automatically when instantiating new Axon. Default **db** identifier is 'DB'.

undef(string **name);**
Destroy the specified virtual field.

Event Listeners

beforesync(void **); aftersync(** void **);**
Invoked before/after Axon sync method is executed.

beforeload(void **); afterload(** void **);**
Invoked before/after Axon load method is executed.

beforesave(void **); aftersave(** void **);**
Invoked before/after Axon save method is executed.

beforeerase(void **); aftererase(** void **);**
Invoked before/after Axon erase method is executed.

M2: MongoDB Mapper

Fat-Free's data mapper for MongoDB objects.

new M2(string **collection, [** string **db]);**
Instantiate M2 object and map to specified collection. Default **db** identifier is 'DB'.

copyfrom(string **name);**
Hydrate M2 with elements from framework array variable, keys of which must be identical to field names in the collection.

copyto(string **name, [** string **fields]);**
Populate framework array variable with M2 properties, keys of which will have names identical to fields in the record. Limit record replication to specified **fields**, if present.

dry(void **);**
Returns TRUE if M2 is devoid of values in its properties.

erase(void **);**
Delete record and reset M2.

find([array **criteria], [** array **order], [** integer **limit], [** integer **offset], [** integer **timeout]);**
Returns an array of collection objects matching specified criteria. If criteria is omitted or NULL, all objects in the collection are returned. Result is sorted according to the sequence of fields specified in order and short-listed by limit. Default offset is zero.

findone([array **criteria], [** array **order], [** integer **limit], [** integer **offset], [** integer **timeout]);**
Returns the first collection object that matches the specified criteria.

found(array **criteria]);**
Returns number of collection objects that match specified criteria.

load(array **criteria, [** array **order], [** integer **offset]);**
Retrieve first object that satisfies **criteria**, sort according to comma-separated list of fields specified in **order**, and starting at the relative **offset**.

lookup(array **fields, [** array **criteria], [** array **mapreduce], [** array **order], [** string **limit], [** integer **offset], [** integer **timeout]);**
Similar to M2 find method but provides fine-grained control over specific fields and map-reduction of results.

reset(void **);**
Dehydrate M2 so instantiation of new M2 is unnecessary.

save(void **);**
Insert or update collection object based on contents of M2 properties.

skip([integer **offset]);**
Retrieve the object at the offset relative to current position in the collection following the same criteria used to hydrate M2. Positive offset moves forward, negative backward. Zero re-syncs M2. Triggers error if skipping beyond bounds of hydration criteria.

sync(string **collection, [** string **db]);**
Synchronize new M2 and **collection**. Called automatically when instantiating new M2. Default **db** identifier is 'DB'.

Event Listeners

beforesync(void **); aftersync(** void **);**
Invoked before/after M2 sync method is executed.

beforeload(void **); afterload(** void **);**
Invoked before/after M2 load method is executed.

beforesave(void **); aftersave(** void **);**
Invoked before/after M2 save method is executed.

beforeerase(void **); aftererase(** void **);**
Invoked before/after M2 erase method is executed.