Perl Dancer 1.X cheat sheet v1.0

ROUTES

any '/route' => sub {}	Defines a route for multiple HTTP methods at once
get '/route' => sub {}	Defines a route for HTTP GET requests to the given path
patch '/route' => sub {}	Defines a route for HTTP PATCH requests to the given URL
post '/route' => sub {}	Defines a route for HTTP POST requests to the given URL
del '/route' => sub {}	Defines a route for HTTP DELETE requests to the given URL
options '/route' => sub {}	Defines a route for HTTP OPTIONS requests to the given URL
put '/route' => sub {}	Defines a route for HTTP PUT requests to the given URL

HOOKS/FILTERS

after	Add a hook at the after position
before sub {}	Defines a before filter
before_template sub {}	Defines a before_template filter
hook 'name' => sub {}	Adds a hook at some position
before_deserializer	This hook receives no arguments
before_file_render	This hook receives as argument the path of the file to render
before_error_init	This hook receives as argument a Dancer::Error object
before_error_render	This hook receives as argument a Dancer::Error object
before	This is an alias to **before**. This hook receives no arguments
before_template_render	This is an alias to **before_template**.This hook receives as argument a HashRef, containing the tokens that will be passed to the template
before_layout_render	This hook receives two arguments. The first one is a HashRef containing the tokens. The second is a ScalarRef representing the content of the template
before_serializer	This hook receives as argument a Dancer::Response object
after_deserializer	This hook receives no arguments
after_file_render	This hook receives as argument a Dancer::Response object
after_template_render	This hook receives as argument a ScalarRef representing the content generated by the template
after_layout_render	This hook receives as argument a ScalarRef representing the content generated by the layout
after	This is an alias for after. This hook runs after a request has been processed, but before the response is sent. It receives a Dancer::Response object
after_error_render	This hook receives as argument a Dancer::Response object
on_handler_exception	This hook is called when an exception has been caught, at the handler level, just before creating and rendering Dancer::Error. This hook receives as argument a Dancer::Exception object
on_route_exception	This hook is called when an exception has been caught, at the route level, just before rethrowing it higher. This hook receives the exception as argument. It can be a Dancer::Exception, or a string, or whatever was used to die

UTILS

	ret type	
dirname 'path'	*scalar*	Returns the dirname of the path given
path 'path', 'path', …	scalar	Concatenates multiple paths together, without worrying about the underlying operating system
false	*scalar*	Constant that returns a false value (0)
true	*scalar*	Constant that returns a true value (1)
from_dumper 'struct'	*hashref*	Deserializes a Data::Dumper structure
from_json 'struct' => {%opts}	*hashref*	Deserializes a JSON structure
from_yaml 'struct'	*hashref*	Deserializes a YAML structure
from_xml 'struct'	*hashref*	Deserializes a XML structure
to_dumper $struct	*scalar*	Serializes a structure with Data::Dumper
to_json $struct => {%opts}	*scalar*	Serializes a structure to JSON
to_yaml $struct	*scalar*	Serializes a structure to YAML
to_xml $struct => {%opts}	*scalar*	Serializes a structure to XML

ACCESSORS/SETTERS

	ret type	
captures	*hashref*	Returns a reference to a copy of %+, if there are named captures in the route Regexp
cookie 'name' => 'value'	*scalar/none*	Get/set cookie value
cookies	*hashref*	Accesses to cookies keys/values
config	*hashref*	Accesses the configuration of the application
content-type 'content/type'	*none*	Sets the content-type rendered, for the current route handler
engine 'name'	*objectref*	Given a namespace, returns the current engine object
headers 'name' => 'value', …	*none*	Adds custom headers to response
header 'name => 'value'	*none*	Adds custom header to response
push_header 'name' => 'value'	*none*	Do the same as header, but allow for multiple headers with the same name
mime	*objectref*	Shortcut to access the instance object of Dancer::MIME
params	*hashref*	Returns an hash reference to all defined parameters
param 'name'	*scalar*	This method is an accessor to the parameters hash table
prefix '/route''	*none*	Defines a prefix for each route handler
request	*objectref*	Returns a Dancer::Request object representing the current request
set 'name' => 'value', …	*none*	Defines a configuration setting(s)
setting 'name'	*any*	Returns the value of a given setting
set_cookie 'name' => 'value', %opts	*none*	Creates or updates cookie values
session 'name' => 'value'	*hashref/none*	Provides access to all data stored in the user's session (if any). It can also be used as a setter to store data in the session
splat	*array*	Returns the list of captures made from a route handler with a route pattern which includes wildcards
upload 'input'	*objectref*	Provides access to file uploads
uri_for '/route'	*scalar*	Returns a fully-qualified URI for the given path
var 'name' => 'value'	*any/none*	Provides an accessor for variables shared between filters and route handlers. Given a key/value pair, it sets a variable
vars	*hashref*	Returns the HashRef of all shared variables set during the filter/route chain with the var keyword

ACTIONS

dance	Alias for the **start** keyword
debug 'message'	Logs a message of **debug** level
error 'message'	Logs a message of **error** level
forward '/route' => {%params} => {method => 'METHOD'}	Runs an internal redirect of the current request to another request
halt 'message'	Sets a response object with the content given
load 'script'	Loads one or more perl scripts in the current application's namespace. Syntactic sugar around Perl's require
load_app 'app_name'	Loads a Dancer package. This method sets the libdir to the current ./lib directory
pass	Tells Dancer to pass the processing of the request to the next matching route
redirect '/route' => CODE	Generates a HTTP redirect (302)
send_error 'message' => CODE	Returns a HTTP error. By default the HTTP code returned is 500
send_file '/path' => %opts	Lets the current route handler send a file to the client
start	Starts the application or the standalone server (depending on the deployment choices)
status 'status'	Changes the status code provided by an action. By default, an action will produce an HTTP 200 OK status code, meaning everything is OK
template 'name' {%tokens}, {%opts}	Returns the response of processing the given template with the given parameters (and optional settings), wrapping it in the default or specified layout too, if layouts are in use
warning 'message'	Logs a warning message through the current logger engine