Stick 0.4 API

Modules: Stick 0.4 API

stick

The stick module provides the Application class which is the centerpiece of the Stick framework.

A Stick Application is a JSGI application which provides means to compose complex applications out of modular middleware components.

stick/helpers

A collection of helper functions that makes working with Stick middleware easier.

stick/middleware

Convenience module that provides access to all Stick middleware using a single require() call.

stick/middleware/accept

This module provides middleware to check if a HTTP request accepts the possible response and makes content negotiation more convenient.

stick/middleware/basicauth

Basic Authentication middleware.

To apply authentication to parts of your website configure this middleware and call the app's basicauth method with the URI path, user name, and SHA1 digest of the user's password as arguments for each path you want to protect:

stick/middleware/continuation

Provide support for JavaScript 1.7 generator actions.

This middleware supports two types of yield values from generators: Promises and JSGI response objects.

If a generator action yields a promise, this middleware adds a listener to that promise that will feed the value back to the

Stick 0.4 API

• Modules • Overview

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookie
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/metho
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

generator once the promise is resolved. If the promise resolves to an error, the error is thrown in the generator.

For example, if promise is a promise, the yield statement will interrupt execution of the action until the promise is resolved, at which point the generator is resumed with the value of the promise being assigned to the resolved variable.

```
var resolved = yield promise;
```

If a generator action yields a JSGI response, the response is sent to the client. To be able to yield more than one response from the same generator, the generator has to be associated with a continuation id and stored in the user's session. This is done by calling continuation.activate() before yielding the first response. The activate() method tells the middleware to store the generator in the user's session and returns a contination id.

For subsequent invocations of the generator, the continuation id has to be set as query string parameter with name _c. When suspended generator is resumed, the new request object is passed in as value for the last yield statement.

```
function continuation(request) {
    var c = app.continuation.activate();
    while(true) {
        request = yield response.html(linkTo(app, {_c: c}));
    }
}
```

See http://blog.ometer.com/2010/11/28/a-sequential-actor-like-api-for-server-side-javascript/ for background.

stick/middleware/cookies

This module provides middleware for reading cookies from the request.

stick/middleware/cors

Cross-site HTTP Request headers

Ability to configure all CORS headers. For a detailed explanation on what the different headers do see MDN on CORS

Even if allowOrigin is defined as "*" (allowOrigin: ['*']) this

middleware will always respond with an "Access-Control-Allow-Origin" header with the value of the "Origin" header.

stick/middleware/csrf

A middleware for CSRF mitigation, using either the synchronizer token or the double submission cookie pattern (see OWASP CSRF Prevention Cheat Sheet for a detailed explanation).

This middleware extends the request object by adding two methods:

- getCsrfToken() returns the CSRF token value for the current request, creating and storing it in the current session if necessary
- rotateCsrfToken() creates a new CSRF token and stores it in the current session

Use getCsrfToken() of the request object to add the token value as hidden input to all forms generated by the application, and make sure the name of the form input is configured accordingly (defaults to "csrftoken").

The token value returned by getCsrfToken() is additionally stored either in the current session (default) or in a cookie. The POST parameter and the token value stored are compared, and in case of a mismatch this middleware returns a 403 "Forbidden" response.

For non-POST requests this middleware accepts the token sent as query parameter or custom header field.

Configuration options:

app.csrf() accepts an object as parameter containing the following properties (default values in brackets):

- tokenLength: The length of the CSRF token (32)
- checkReferrer: Enable strict referrer checking for HTTPS requests (true)
- rotate: If true tokens are only used once and modified after each validation (false)
- getToken: By default the middleware expects the submitted

CSRF token in either a post or query parameter csrftoken or in a custom request header x-csrf-token (in this order). To customize this define a function that will receive the request object as single argument and is expected to return the CSRF token (or null).

- getFailureResponse: An optional function receiving the request as single argument. This method is called if the CSRF validation failed, and is expected to return a valid JSGI response (default: 403 forbidden).

- safeMethods: An array containing request method names that are considered safe, so no token validation is done (["GET", "HEAD", "OPTIONS", "TRACE"])

Cookie mode

The following options switch the middleware into "double submission cookie" mode, and allow detailed configuration of the cookie:

- useCookie: If true the CSRF token is stored in a cookie

- cookieName: The name of the cookie to set ("csrftoken")

- cookieHttpOnly: If true the httpOnly flag of the cookie is set (true)

- cookieSecure: If true the secure flag of the cookie is set (false)

stick/middleware/error

Middleware to catch errors and generate simple error pages.

By default, resource stick/middleware/error.html is used as page template. This can be set through the app.error.template property. This is the complete list of properties that influence the behaviour of this middleware:

- **template** the error page template (*string*)

- **message** static error message to use instead of actual message (*string*)

- **location** whether to report any information about the code location of the error (*boolean*)
- **stack** whether to include a JavaScript stack trace (*boolean*)
- **javaStack** whether to include a Java stack trace (*boolean*)

stick/middleware/etag

Middleware for conditional HTTP GET request based on response body message digests.

The response body must implement a digest() method for this middleware to work.

stick/middleware/gzip

Middleware for on-the-fly GZip compression of response bodies.

By default only text content types are compressed. This can be controlled using the gzip.contentTypes property:

stick/middleware/method

JSGI middleware for HTTP method override.

Since older browsers are not able to send XMLHttpRequest with methods other than GET and POST, this middleware allows the method of POST requests to be overridden based on the value of a HTTP form parameter. The default name for the override parameter is _method. This can be configured through the method.key property.

stick/middleware/mount

This module provides middleware for mounting other applications on a specific URI path or virtual host.

Applying this middleware adds a mount method to the application. The mount method takes a path or virtual host specification and an application as arguments. If the spec is a string, it is interpreted as the URI path on which the app will be mounted. If it is an object, it may contain path or host properties that will be matched against the URI path and Host header of incoming requests. *Note that virtual host based mounting has not been tested so far.*

The mount method accepts an optional third boolean noRedirect argument. If set to true it will disable redirecting GET requests to the mount base URL without a trailing slash to the same URL with trailing slash. By default, mount middleware will send a redirect to the mount URL with trailing slash.

Mounting one application within another causes the scriptName and pathInfo properties in the request object to be adjusted so that the mounted application receives the same pathInfo as if it was the main application. This means that forward and reverse request routing will usually work as expected.

This middleware maintains an index mapping applications to mount points which can be accessed using the lookup function. The [stick/helpers][helpers] module provides higher level functions for this which include support for the route middleware.

stick/middleware/notfound

Middleware for simple Not-Found pages.

By default, resource stick/middleware/notfound.html is used as page template. This can be set through the app.notfound.template property.

- **template** the notfound page template (*string*)

stick/middleware/params

This module provides middleware for parsing HTTP parameters from the query string and request body.

It does not parse multipart MIME data such as file uploads which are handled by the [upload][middleware/upload] module.

stick/middleware/profiler

This module provides profiling middleware to measure the application's runtime behaviour.

Profiler data is written to the module's logger. You have to run the application in interpreted mode (passing -o -1 on the command line) to get meaningful results.

stick/middleware/render

This module provides middleware for rendering responses using ringo/mustache.

This middleware installs a render function in the application object which is used to render HTTP responses. The behaviour of app.render can be tweaked by setting the following properties:

- render.base – the base path or repository to look for templates
- render.helpers – helper functions that will be merged in the context
- render.master – master template which is applied on the top of processing the page with template given in the .render function.
- render.contentType – MIME type to use for HTTP response
- render.charset – charset name to use for HTTP response

stick/middleware/requestlog

Middleware for collecting log messages issued during execution of the current request.

This adds a requestlog property to the application object with an items property. During execution of a request items contains an array containing all the log messages issued for the request. Log messages are represented as arrays in the format [time, level, name, message].

During request execution, the requestlog property also defines a property called start containing the time the execution started.

By default, messages are appended to the response if its Content-Type is text/html. This can be controlled using the app.requestlog.append boolean flag.

stick/middleware/route

Middleware for HTTP method based local request routing.

This installs get, post, put, del and options methods in the

application object for routing requests with the corresponding HTTP methods. These methods take a path spec as first argument and a function as second argument.

Paths and Placeholders

The path spec can consist of static parts and placeholders. Named placeholders are prefixed by : (colon) and match all characters except for / (slash) and . (dot). A named placeholder can be marked as optional by appending ? (question mark). Unnamed placeholders are denoted by the asterisk character * and match all characters including slashes and dots.

In the following example, ":id" is a named placeholder:

```
"/post/:id"
```

All placeholders are passed to the action function as positional arguments following the request object in the order in which they appear in the path spec. Unmatched optional placeholders will be undefined.

```
app.get("/post/:id", function(req, id) {...});
```

Reverse Routing

The route middleware supports generating URLs from route names and parameters required by the route.

Routes names can either be defined explicitly by passing the route name as third argument, or are derived from the route's path spec by stripping out all placeholders and removing a leading slash. For example, a path spec /post/:id.html results in route name "post.html". If a path spec does not contain any static part, its route name is "index".

Passing a valid route name and the parameters required by the route to the route.reverse method will return the URI path for the corresponding action. For example, with a route spec /post/:id.html, calling app.route.reverse({action: "post.html", id: 5}) will return the string "/post/5.html".

The [stick/helpers][helpers] module provides higher level helpers for reverse routing including support for mounted applications.

stick/middleware/session

This module provides middleware for HTTP sessions.

It adds a session property to the request object that allows to store arbitrary data on on a per-visitor basis.

The default session implementation is based on Java Servlet sessions. This can be overridden by setting the app.session.impl property to an alternative session constructor.

```
app.session.impl = MySession;
```

The session constructor will be called with the request object as only argument when the session is first accessed.

stick/middleware/static

Middleware for serving static resources.

This installs a static() method in the application that accepts the following arguments:

- base: the base resource directory (required)
- index: the name of a file to serve if the path matches a directory (e.g. "index.html")
- baseURI: a common prefix for a resource URI (e.g. "/static")

You can call static() multiple times to register multiple resources to be served.

stick/middleware/upload

This module provides support for parsing multipart MIME messages used for file uploads.

This module behaves analogous and can be used in combination with the [params][middleware/params] middleware.

Index: Stick 0.4 API

Module stick

Class Application(function)

- configure ()
- env (string)
- base
- request

Module stick/helpers

- linkTo (object|string, object, string)
- redirectTo (string|object, bindings)
- resolveApp ()
- urlFor (object|string, object)

Module stick/middleware

- accept
- basicauth
- continuation
- cookies
- csrf
- error
- etag
- gzip
- method
- mount
- notfound
- params
- profiler
- render
- requestlog
- route
- session
- static
- upload

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/metho
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

Module stick/middleware/accept

middleware (Function, Object)

Module stick/middleware/basicauth

middleware (Function, Object)

Module stick/middleware/continuation

middleware (,)

Module stick/middleware/cookies

middleware (Function, Object)

request.cookies

Module stick/middleware/cors

middleware (,)

Module stick/middleware/csrf

middleware (Function, Object)

Module stick/middleware/error

middleware (Function, Object)

Module stick/middleware/etag

middleware (Function, Object)

Module stick/middleware/gzip

middleware (Function, Object)

Module stick/middleware/method

middleware (Function, Object)

Module stick/middleware/mount

lookup ()

middleware (Function, Object)

Module stick/middleware/notfound

middleware (,)

Module stick/middleware/params

middleware (Function, Object)

request.params

request.postParams

request.queryParams

Module stick/middleware/profiler

middleware (Function, Object)

Module stick/middleware/render

middleware (Function, Object)

Module stick/middleware/requestlog

middleware (Function, Object)

Module stick/middleware/route

middleware (Function, Object)

Module stick/middleware/session

middleware (Function, Object)

request.session

Class ServletSession()

invalidate ()

creationTime

data

isNew

lastAccessedTime

maxInactiveInterval

volatile

Module stick/middleware/static

middleware (Function, Object)

Module stick/middleware/upload

middleware (Function, Object)

request.postParams

Module stick

The stick module provides the Application class which is the centerpiece of the Stick framework.

A Stick Application is a JSGI application which provides means to compose complex applications out of modular middleware components.

Class Application

Instance Methods

configure (middleware...)

env (name)

Instance Properties

base

request

Application (nested)

The application object is a JSGI application that wraps a middleware chain.

When invoked without arguments, the Application constructor returns an application that wraps an unhandled middleware that throws an Error when called. Use configure to add middleware modules to the application.

When invoked with an argument, it is used as initial value for the application's middleware chain.

Parameters

function **nested** the nested application (optional)

Application.prototype.base

This application's base URI path as used by the current request, or null if unknown.

Stick 0.4 API

• Modules • Overview

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/methoc
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

Application.prototype.configure (middleware...)

Apply one or more middleware factories to this Application. Middleware will be wrapped around the existing middleware chain, starting with the rightmost argument. For example, the following invocation of configure on a newly created Application object:

```
app.configure("error", "route");
```

will result in error(route(unhandled())) as middleware chain.

Parameters

middleware... one or middleware factories. These can be defined as module names, imported modules, or middleware factories. For the middleware that is part of Stick it is sufficient to use the last part of the module id. e.g. "error" or "route".

Application.prototype.env (name)

Returns an Application object that shares the middleware chain of the original application, but can be configured to contain additional middleware that is not shared with the parent application.

Repeated calls of the env method with the same argument return the same application object.

Parameters

string **name** the environment name

Application.prototype.request

The request object associated with the current thread, or null.

Module stick/helpers

A collection of helper functions that makes working with Stick middleware easier.

Functions

linkTo (app, bindings, text)

redirectTo (app, bindings)

resolveApp (app)

urlFor (app, bindings)

linkTo (app, bindings, text)

Return a link to an action configured using the route middleware.

The link's URL is generated from the bindings argument as described for the urlFor helper.

Parameters

object|string	**app**	the application to link to
object	**bindings**	an object containing the bindings for the target URL.
string	**text**	the link text.

---

redirectTo (app, bindings)

Create a response that redirects the client to a different URL.

Parameters

string|object	**app**	either the URL as string or an app to be passed to `urlFor`
bindings	**bindings**	to pass to `urlFor` if first argument is an app.

Returns

object	a JSGI response that will redirect the client to the specified target

---

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/methoc
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

resolveApp (app)

Resolve a module name or module object to a JSGI application.

Parameters

app

---

urlFor (app, bindings)

Return a URL for an action configured using the mount and route middlewares.

The app argument specifies the Stick application to link to, either as reference to the app itself or as module id of a module exporting the app as app. The bindings argument contains information needed to determine the target action within that application. Properties in the bindings argument are interpreted as follows:

- The action property, if present, identifies the name of the action to link to. Action names are determined from the path arguments provided to the route middleware by removing all placeholders and the leading slash. For example, the name for an action routed with /edit/:id is "edit".
- Properties in the bindings object that have a placeholder with the same name in the target route are used to provide the value for this placeholder. For example, to URI path to an action in application app routed with "/edit/:id" with id 5 can be generated as follows:

```
urlFor(app, {action: "edit", id: 5})
```

- All other properties in the bindings object are set as query parameters in the generated URL. For example, if "index" is a route with no placeholders then calling urlFor("index", {do: "search"}) will generate the URL "/?do=search".

Note that the values for the current request are used as default values for action and route placehoders, so if you want to use to a different action or placeholder value you need to make that explicit.

Parameters

object|string	**app**	the application to link to
object	**bindings**	an object containing the bindings for the target URL.

Module stick/middleware

Convenience module that provides access to all Stick middleware using a single require() call.

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/methoc
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

Properties

accept
basicauth
continuation
cookies
csrf
error
etag
gzip
method
mount
notfound
params
profiler
render
requestlog
route
session
static
upload

accept

Middleware for HTTP content negotiation.

---

basicauth

Middleware for Basic HTTP Authentication.

---

continuation

Middleware for generator based continuations.

---

cookies

Middleware for cookie support.

---

csrf

Middleware for CSRF support.

---

error

Middleware for generating error pages.

---

etag

Middleware for ETag based conditional GET.

---

gzip

Middleware for GZip compression of response bodies.

---

method

Middleware for HTTP method overriding.

---

mount

Middleware for mounting other applications.

---

notfound

Middleware for generating 404 pages.

params

Middleware for request parameter parsing.

profiler

Middleware for JavaScript code profiling.

render

Middleware for template based response rendering.

requestlog

Middleware for collecting log messages per HTTP request.

route

Middleware for intra-app request routing.

session

Middleware for HTTP session support.

static

Middleware for serving static resources.

upload

Middleware for file upload support.

Module stick/middleware/accept

This module provides middleware to check if a HTTP request accepts the possible response and makes content negotiation more convenient.

Functions

middleware (next, app)

middleware (next, app)

This middleware installs a accept function in the application object. If the client request does not accept one of the specified content types, the middleware returns a 406 - Not Acceptable response.

Applying this middleware adds an accepted property to the request object. This contains an array with the client's accepted media types sorted from highest to the lowest quality.

Example

```
app.configure("accept", "route");
app.accept(["text/plain", "text/html"]);
app.get("/", function(req) {
  if (req.accepted[0].subType === "html") {
    return response.html("");
  } else {
    return response.text("foo");
  }
});
```

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

Module stick/middleware/basicauth

Basic Authentication middleware.

To apply authentication to parts of your website configure this middleware and call the app's basicauth method with the URI path, user name, and SHA1 digest of the user's password as arguments for each path you want to protect:

Example

```
app.configure("basicauth");
app.basicauth('/protected/path', 'admin',
        '30B93F320076DE1304C34673F9F524F7EA7DB709');
```

Functions

middleware (next, app)

middleware (next, app)

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function — a JSGI middleware function

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/metho
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

Module stick/middleware/continuation

Provide support for JavaScript 1.7 generator actions.

This middleware supports two types of yield values from generators: Promises and JSGI response objects.

If a generator action yields a promise, this middleware adds a listener to that promise that will feed the value back to the generator once the promise is resolved. If the promise resolves to an error, the error is thrown in the generator.

For example, if promise is a promise, the yield statement will interrupt execution of the action until the promise is resolved, at which point the generator is resumed with the value of the promise being assigned to the resolved variable.

```
var resolved = yield promise;
```

If a generator action yields a JSGI response, the response is sent to the client. To be able to yield more than one response from the same generator, the generator has to be associated with a continuation id and stored in the user's session. This is done by calling continuation.activate() before yielding the first response. The activate() method tells the middleware to store the generator in the user's session and returns a contination id.

For subsequent invocations of the generator, the continuation id has to be set as query string parameter with name _c. When suspended generator is resumed, the new request object is passed in as value for the last yield statement.

```
function continuation(request) {
    var c = app.continuation.activate();
    while(true) {
        request = yield response.html(linkTo(app, {_c: c}));
    }
}
```


stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/methoc
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

See http://blog.ometer.com/2010/11/28/a-sequential-actor-like-api-for-server-side-javascript/ for background.

Functions

middleware (next, app)

middleware (next, app)

Parameters

next
app

Module stick/middleware/cookies

This module provides middleware for reading cookies from the request.

Functions

middleware (next, app)

Properties

request.cookies

middleware (next, app)

This middleware provides support for cookie access.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function	a JSGI middleware function

---

request.cookies

A cookies object for the current request.

See

servletRequest

Module stick/middleware/cors

Cross-site HTTP Request headers

Ability to configure all CORS headers. For a detailed explanation on what the different headers do see MDN on CORS

Even if allowOrigin is defined as "*" (allowOrigin: ['*']) this middleware will always respond with an "Access-Control-Allow-Origin" header with the value of the "Origin" header.

Example

```
app.configure("cors");
app.cors({
  allowOrigin: ['http://example.com', 'https://example.com'],
  allowMethods: ['POST', 'GET'],
  allowHeaders: ['X-PingOther'],
  exposeHeaders: []
  maxAge: 1728000,
  allowCredentials: true
})
```

Functions

middleware (next, app)

middleware (next, app)

Example

```
app.configure("cors");
app.cors({
  allowOrigin: ['http://example.com', 'https://example.com'],
  allowMethods: ['POST', 'GET'],
  allowHeaders: ['X-PingOther'],
  exposeHeaders: []
  maxAge: 1728000,
  allowCredentials: true
})
```

Parameters

next
app

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/methoc
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

Module stick/middleware/csrf

A middleware for CSRF mitigation, using either the synchronizer token or the double submission cookie pattern (see OWASP CSRF Prevention Cheat Sheet for a detailed explanation).

This middleware extends the request object by adding two methods:

- getCsrfToken() returns the CSRF token value for the current request, creating and storing it in the current session if necessary
- rotateCsrfToken() creates a new CSRF token and stores it in the current session

Use getCsrfToken() of the request object to add the token value as hidden input to all forms generated by the application, and make sure the name of the form input is configured accordingly (defaults to "csrftoken").

The token value returned by getCsrfToken() is additionally stored either in the current session (default) or in a cookie. The POST parameter and the token value stored are compared, and in case of a mismatch this middleware returns a 403 "Forbidden" response.

For non-POST requests this middleware accepts the token sent as query parameter or custom header field.

Configuration options:

app.csrf() accepts an object as parameter containing the following properties (default values in brackets):

- tokenLength: The length of the CSRF token (32)
- checkReferrer: Enable strict referrer checking for HTTPS requests (true)
- rotate: If true tokens are only used once and modified after each validation (false)
- getToken: By default the middleware expects the submitted CSRF token in either a post or query parameter csrftoken or in a custom request header x-csrf-token (in this order). To customize this define a function that will receive the request object as single argument and is expected to return the CSRF token (or null).

- getFailureResponse: An optional function receiving the request as single argument. This method is called if the CSRF validation failed, and is expected to return a valid JSGI response (default: 403 forbidden).
- safeMethods: An array containing request method names that are considered safe, so no token validation is done (["GET", "HEAD", "OPTIONS", "TRACE"])

Cookie mode

The following options switch the middleware into "double submission cookie" mode, and allow detailed configuration of the cookie:

- useCookie: If true the CSRF token is stored in a cookie
- cookieName: The name of the cookie to set ("csrftoken")
- cookieHttpOnly: If true the httpOnly flag of the cookie is set (true)
- cookieSecure: If true the secure flag of the cookie is set (false)

Example

```
app.configure("csrf");
app.csrf({
    "tokenLength": 64,
    "rotate": true,
    "getToken": function(req) {
        return req.headers["x-custom-field"];
    }
});
```

Functions

middleware (next, app)

middleware (next, app)

Stick middleware factory for CSRF mitigation

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function	a JSGI middleware function

Module stick/middleware/error

Middleware to catch errors and generate simple error pages.

By default, resource stick/middleware/error.html is used as page template. This can be set through the app.error.template property. This is the complete list of properties that influence the behaviour of this middleware:

- **template** the error page template (*string*)
- **message** static error message to use instead of actual message (*string*)
- **location** whether to report any information about the code location of the error (*boolean*)
- **stack** whether to include a JavaScript stack trace (*boolean*)
- **javaStack** whether to include a Java stack trace (*boolean*)

Example

```
app.configure("error");
app.error.template = module.resolve("500.html");
app.error.location = true;
```

Functions

middleware (next, app)

middleware (next, app)

Stick middleware factory to display error messages and stack traces.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

Module stick/middleware/etag

Middleware for conditional HTTP GET request based on response body message digests.

The response body must implement a digest() method for this middleware to work.

Functions

middleware (next, app)

middleware (next, app)

Middleware for conditional HTTP GET request based on response body message digests.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function	a JSGI middleware function

Module stick/middleware/gzip

Middleware for on-the-fly GZip compression of response bodies.

By default only text content types are compressed. This can be controlled using the gzip.contentTypes property:

Example

```
app.configure("gzip");
app.gzip.contentTypes = /^text|xml|json|javascript/;
```

Functions

middleware (next, app)

middleware (next, app)

JSGI middleware for GZIP compression.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

Module stick/middleware/method

JSGI middleware for HTTP method override.

Since older browsers are not able to send XMLHttpRequest with methods other than GET and POST, this middleware allows the method of POST requests to be overridden based on the value of a HTTP form parameter. The default name for the override parameter is _method. This can be configured through the method.key property.

Example

```
app.configure("method");
app.method.key = "__method";
```

Functions

middleware (next, app)

middleware (next, app)

JSGI middleware for HTTP method override.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

Module stick/middleware/mount

This module provides middleware for mounting other applications on a specific URI path or virtual host.

Applying this middleware adds a mount method to the application. The mount method takes a path or virtual host specification and an application as arguments. If the spec is a string, it is interpreted as the URI path on which the app will be mounted. If it is an object, it may contain path or host properties that will be matched against the URI path and Host header of incoming requests. *Note that virtual host based mounting has not been tested so far.*

The mount method accepts an optional third boolean noRedirect argument. If set to true it will disable redirecting GET requests to the mount base URL without a trailing slash to the same URL with trailing slash. By default, mount middleware will send a redirect to the mount URL with trailing slash.

Mounting one application within another causes the scriptName and pathInfo properties in the request object to be adjusted so that the mounted application receives the same pathInfo as if it was the main application. This means that forward and reverse request routing will usually work as expected.

This middleware maintains an index mapping applications to mount points which can be accessed using the lookup function. The stick/helpers module provides higher level functions for this which include support for the route middleware.

Example

```
app.configure("mount");
app.mount("/wiki", module.resolve("vendor/ringowiki"));
```

Functions

lookup (target)

middleware (next, app)

lookup (target)

Return the URI path of a mounted application

Parameters

target a mounted JSGI application

Returns

the URI path of the application, or ""

---

middleware (next, app)

Middleware to mount other application on specific URI paths or virtual hosts.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

Module stick/middleware/notfound

Middleware for simple Not-Found pages.

By default, resource stick/middleware/notfound.html is used as page template. This can be set through the app.notfound.template property.

- **template** the notfound page template (*string*)

Example

```
app.configure("notfound");
app.notfound.template = "templates/404.html";
```

Functions

middleware (next, app)

middleware (next, app)

Parameters

next
app

stick
stick/helpers
stick/middleware
stick/middleware/accept
stick/middleware/basicau
stick/middleware/continu
stick/middleware/cookies
stick/middleware/cors
stick/middleware/csrf
stick/middleware/error
stick/middleware/etag
stick/middleware/gzip
stick/middleware/methoc
stick/middleware/mount
stick/middleware/notfou
stick/middleware/params
stick/middleware/profile
stick/middleware/render
stick/middleware/reques
stick/middleware/route
stick/middleware/sessior
stick/middleware/static
stick/middleware/upload

Module stick/middleware/params

This module provides middleware for parsing HTTP parameters from the query string and request body.

It does not parse multipart MIME data such as file uploads which are handled by the upload module.

Functions

middleware (next, app)

Properties

request.params

request.postParams

request.queryParams

middleware (next, app)

Middleware for parsing HTTP parameters. This module handles URL-endcoded form data transmitted in the query string and request body as well as JSON encoded data in the request body.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function — a JSGI middleware function

---

request.params

An object containing the parsed HTTP parameters sent with this request.

---

request.postParams

An object containing the parsed HTTP POST parameters sent with

this request. If the content type of the request is application/json, the middleware parses the body and stores the in request.postParams.

---

request.queryParams

An object containing the parsed HTTP query string parameters sent with this request.

Module stick/middleware/profiler

This module provides profiling middleware to measure the application's runtime behaviour.

Profiler data is written to the module's logger. You have to run the application in interpreted mode (passing `-o -1` on the command line) to get meaningful results.

Functions

middleware (next, app)

middleware (next, app)

A middleware factory that runs the nested app with a runtime profiler.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function — a JSGI middleware function

Module stick/middleware/render

This module provides middleware for rendering responses using ringo/mustache.

This middleware installs a render function in the application object which is used to render HTTP responses. The behaviour of app.render can be tweaked by setting the following properties:

- render.base – the base path or repository to look for templates
- render.helpers – helper functions that will be merged in the context
- render.master – master template which is applied on the top of processing the page with template given in the .render function.
- render.contentType – MIME type to use for HTTP response
- render.charset – charset name to use for HTTP response

Example

```
app.configure("render");
app.render.base = module.resolve("templates");
app.render("index.tmpl", {data: "Hello World!"});
```

Functions

middleware (next, app)

middleware (next, app)

Middleware for template based response generation.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

Module stick/middleware/requestlog

Middleware for collecting log messages issued during execution of the current request.

This adds a requestlog property to the application object with an items property. During execution of a request items contains an array containing all the log messages issued for the request. Log messages are represented as arrays in the format [time, level, name, message].

During request execution, the requestlog property also defines a property called start containing the time the execution started.

By default, messages are appended to the response if its Content-Type is text/html. This can be controlled using the app.requestlog.append boolean flag.

Functions

middleware (next, app)

middleware (next, app)

Middleware for collecting log messages issued during execution of the current request.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function — a JSGI middleware function

Module stick/middleware/route

Middleware for HTTP method based local request routing.

This installs get, post, put, del and options methods in the application object for routing requests with the corresponding HTTP methods. These methods take a path spec as first argument and a function as second argument.

Paths and Placeholders

The path spec can consist of static parts and placeholders. Named placeholders are prefixed by : (colon) and match all characters except for / (slash) and . (dot). A named placeholder can be marked as optional by appending ? (question mark). Unnamed placeholders are denoted by the asterisk character * and match all characters including slashes and dots.

In the following example, ":id" is a named placeholder:

```
"/post/:id"
```

All placeholders are passed to the action function as positional arguments following the request object in the order in which they appear in the path spec. Unmatched optional placeholders will be undefined.

```
app.get("/post/:id", function(req, id) {...});
```

Reverse Routing

The route middleware supports generating URLs from route names and parameters required by the route.

Routes names can either be defined explicitly by passing the route name as third argument, or are derived from the route's path spec by stripping out all placeholders and removing a leading slash. For example, a path spec /post/:id.html results in route name "post.html". If a path spec does not contain any static part, its route name is "index".

Passing a valid route name and the parameters required by the route to the route.reverse method will return the URI path for the corresponding action. For example, with a route spec /post/:id.html, calling app.route.reverse({action: "post.html", id: 5}) will return the string "/post/5.html".

The stick/helpers module provides higher level helpers for reverse routing including support for mounted applications.

Example

```
app.configure("route")
app.get("/", function() {...})
app.get("/", function() {...}, "index")
app.post("/", function(req) {...})
app.get("/:id.:format?", function(req, id, format) {...})
app.del("/:id", function(req, id) {...})
app.put("/:id", function(req, id) {...})
```

Functions

middleware (next, app)

middleware (next, app)

Middleware for HTTP method based local request routing.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function	a JSGI middleware function

Module stick/middleware/session

This module provides middleware for HTTP sessions.

It adds a session property to the request object that allows to store arbitrary data on on a per-visitor basis.

The default session implementation is based on Java Servlet sessions. This can be overridden by setting the app.session.impl property to an alternative session constructor.

```
app.session.impl = MySession;
```

The session constructor will be called with the request object as only argument when the session is first accessed.

Functions

middleware (next, app)

Properties

request.session

Class ServletSession

Instance Methods

invalidate ()

Instance Properties

creationTime
data
isNew
lastAccessedTime
maxInactiveInterval
volatile

ServletSession (request)

An HTTP session object based on top of servlet sessions. Properties of the session's data object are persisted between requests of the

same client.

Parameters

request	a JSGI or servlet request object

ServletSession.prototype.creationTime

Createtime of the current session.

ServletSession.prototype.data

A container for things to store in this session between requests.

ServletSession.prototype.invalidate ()

Destroys the current session and any data bound to it.

ServletSession.prototype.isNew

True if this session was created in the current request. This can be useful to find out if the client has cookies disabled for cookie-based sessions.

ServletSession.prototype.lastAccessedTime

Time in Unix epoch milliseconds since the last client access.

ServletSession.prototype.maxInactiveInterval

A time interval in seconds, which the session will be open. If the interval is exceeded, the session gets invalidated.

ServletSession.prototype.volatile

A volatile property which survives a HTTP redirect and can be used

for warnings or error messages in forms. After a requests was handled, the property is reset to null.

---

middleware (next, app)

This middleware provides support for anonymous user sessions.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

---

request.session

A session object for the current request. If no session exists a new one will be created.

See

ServletSession

Module stick/middleware/static

Middleware for serving static resources.

This installs a static() method in the application that accepts the following arguments:

- base: the base resource directory (required)
- index: the name of a file to serve if the path matches a directory (e.g. "index.html")
- baseURI: a common prefix for a resource URI (e.g. "/static")

You can call static() multiple times to register multiple resources to be served.

Functions

middleware (next, app)

middleware (next, app)

Middleware for serving static resources.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function a JSGI middleware function

Module stick/middleware/upload

This module provides support for parsing multipart MIME messages used for file uploads.

This module behaves analogous and can be used in combination with the params middleware.

Example

```
var fileUpload = request.postParams['fooUpload'];
  var name = fileUpload.filename;
  // `filename` and `value` are user input and must be sanitized for security reasons
  // Be strict about what you allow as `filename` (e.g.: only `A-Za-z\.` or simlar)
  fs.write(join(fooPath, name), fileUpload.value);
```

Functions

middleware (next, app)

Properties

request.postParams

middleware (next, app)

Middleware factory to enable support for parsing file uploads.

Parameters

Function	**next**	the wrapped middleware chain
Object	**app**	the Stick Application object

Returns

Function	a JSGI middleware function

---

request.postParams

An object containing the parsed HTTP POST parameters sent with this request.