Middleware

Contents

Adding custom middleware

Since Luminus uses Ring for routing the application handler is a standard Ring handler and can be wrapped in middleware just like you would in any other Ring based application.

The middleware allows wrapping the handlers in functions which can modify the way the request is processed. Middleware functions are often used to extend the base functionality of Ring handlers to match the needs of the particular application.

A middleware is simply a function which accepts an existing handler with some optional parameters and returns a new handler with some added behaviour. An example of a middleware function would be:

```
(defn wrap-nocache [handler]
  (fn [request]
     (let [response (handler request)]
        (assoc-in response [:headers  "Pragma"] "no-cache"))))
```

As you can see the wrapper accepts the handler and returns a function which in turn accepts the request. Since the returned function was defined in the scope where the handler exists, it can use it internally. When called, it will call the handler with the request and add Pragma: no-cache to the response map. For detailed information please refer to the official Ring documentation.

The middleware is added in the `middleware` namespace of your project. The `wrap-base` function aggregates all the common middleware for the application. Only add middleware that you also wish to use in production in the `wrap-base` function.

```
(defn wrap-base [handler]
  (-> ((:middleware defaults) handler)
```

Build Tool: lein ▼

Topics

Your First Application
REPL Driven Developme
Application Profiles
HTML Templating
Static Assets
ClojureScript
Routing
RESTful Services
Request types
Response types
Websockets
• **Middleware**
Sessions and Cookies
Input Validation
Security
Component Lifecycle
Database Access
Database Migrations
Logging
Internationalization
Testing
Server Tuning
Environment Variables
Deployment
Useful Libraries
Sample Applications
Upgrading
Clojure Resources

Books

```
      wrap-formats
      wrap-webjars
      wrap-flash
      (wrap-session {:cookie-attrs {:http-only true}})
      (wrap-defaults
        (-> site-defaults
            (assoc-in [:security :anti-forgery] false)
            (dissoc :session)))
      wrap-context
      wrap-internal-error))
```

Any development middleware, such as middleware for showing stacktraces, should be added in the `wrap-dev` function found in the `<app>.dev-middleware` namespace. This namespace resides in the `env/dev/clj` source path and will only be included during development mode.

```
(defn wrap-dev [handler]
  (-> handler
      wrap-reload
      wrap-error-page
      wrap-exceptions))
```

Note that the order of the middleware matters as the request is modified by each middleware function. For example, any middleware functions that rely on the session must be placed before the `wrap-defaults` middleware that creates the session. The reason being that the request will pass through the outer middleware functions before reaching the inner ones.

For example, when we have the handler wrapped using `wrap-formats` and `wrap-defaults` as seen below:

```
(-> handler wrap-formats wrap-defaults)
```

The request is passed through these functions in the following order:

```
handler <- wrap-formats <- wrap-defaults <- request
```

Useful ring middleware

ring-ratelimit - Rate limiting middleware
ring-etag-middleware - Calculates etags for ring responses and returns 304 responses when appropriate
ring-gzip-middleware - Gzips ring responses for user agents which can handle it
ring-upload-progress - Provide upload progress data in ring session