cookie egg

Shiro Kawai, ported to Chicken by Reed Sheridan

Table of Contents

1 About this egg

1.1 Version history

`1.1`   Fix bugs related to quoting output of construct-cookie-string

`1.0`   Initial release

1.2 Usage

Load this egg like so:

```
(require-extension cookie)
```

2 Documentation

parse-cookie-string [procedure]
> (`parse-cookie-string` string #!optional version)

Parse a cookie string `string`, which is the value of the "Cookie" request header. Usually, this information is available to a CGI program via the environment variable HTTP_COOKIE.

If the cookie version is known, via the "Cookie2" request header, the integer version must be passed to `version.`Otherwise, parse-cookie figures out the version from string.

The result has the following format.

`((<name> <value> [path: <path>] [domain: <domain>] [port: <port>]) ...)`

where `<name>` is the attribute name, and `<value>` is the corresponding value. If the attribute doesn't have value, `<value>` is #f. (Note that it differs from the attribute having null value, `""`.) If the attribute has path, domain or port options, it is given as a form of keyword-value pair.

construct-cookie-string [procedure]
> (`construct-cookie-string` specs #!optional version)

Given list of cookie specs, creates a cookie string suitable for `Set-cookie2` or `Set-cookie` header.

Optional version argument specifies cookie protocol version. 0 for the old Netscape style format, and 1 for RFC2965 style format. When omitted, version 1 is assumed.

Each cookie spec has the following format.

`(<name> <value> [comment: <comment>] [comment-url: <url>] [discard: <bool>] [domain: <domain>] [max-age: <age>] [path: <path>] [port: <port-list>] [secure: <bool>] [version: <version>] [expires: <date>])`

Where,

`<name>`   A string. Name of the cookie.

`<value>`   Value of the cookie. May be a string, or #f if no value is needed.

`<comment> <url> <domain> <path> <port-list>`
>   Strings.

`<bool>`   Boolean value

`<age> <version>`
>   Integers

`<date>`   Either an integer (seconds since Epoch) or a formatted date string following the netscape cookie specification.

The attribute values are quoted appropriately. If the specified attribute is irrelevant for the version, it is ignored. So you can pass the same specs to generate both old-style and new-style cookie strings.

Return value is a list of cookie strings, each of which stands for each cookie. For old-style protocol (using Set-cookie header) you must send each of them by individual header. For new-style protocol (using Set-cookie2 header), you can join them with comma and send it at once. See RFC2965 for further details.

3 Examples

```
(construct-cookie-string ‘(("name" "foo" domain: "foo.com" path: "/"
                            expires: ,(+ (sys-time) 86400) max-age: 86400)))
  => ("name=foo;Domain=foo.com;Path=/;Max-age=86400")

(construct-cookie-string ‘(("name" "foo" domain: "foo.com" path: "/"
                            expires: ,(+ (sys-time) 86400) max-age: 86400)) 0)
  => ("name=foo;Domain=foo.com;Path=/;Expires=Sun, 09-Sep-2001 01:46:40 GMT")
```

4 License

Index