nodejs-webservice-hw

Node JS Web Service - 6.14

version :1.0.0

nodejs webservice hw

COREMODULE

connect

Connect is a middleware framework for node, shipping with over 18 bundled middleware and a rich selection of 3rd-party middleware

Middleware:

1)Logger request logger with custom format support
2)Csrf Cross-site request forgery protection
3)Compress Gzip compression middleware
4)BasicAuth basic http authentication
5)BodyParser extensible request body parser
6)Json application/json parser
7)Urlencoded application/x-www-form-urlencoded parser
8)Multipart multipart/form-data parser
9)CookieParser cookie parser
10)Session session management support with bundled MemoryStore
11)SessionSession cookie-based session support
12)MethodOverride faux HTTP method support
13)ResponseTime calculates response-time and exposes via X-Response-Time
14)StaticCache memory cache layer for the static() middleware
15)Static streaming static file server supporting Range and more
16)Directory directory listing middleware
17)Vhost virtual host sub-domain mapping middleware
18)Favicon efficient favicon server (with default icon)
19)Limit limit the bytesize of request bodies
20)Query automatic querystring parser, populating req.query
21)ErrorHandler flexible error handler

Internals:

1)Server prototype
2)Connect utilities
3)Node monkey patches

Express

Express is used for high performance and for high class web development for Node.js. Creating A Server: To create an instance of the express.HTTPServer, simply invoke the createServer() method. With our instance app we can then define routes based on the HTTP verbs, in this example app.get(). Creating An HTTPS Server: To initialize a express.HTTPSServer we do the same as above, however we pass an options object, accepting key, cert and the others mentioned in node's https documentation. Configuration: Express supports arbitrary environments, such as production and development. Developers can use the

configure() method to setup needs required by the current environment. When configure() is called without an environment name it will be run in every environment prior to the environment specific callback.In the example below we only dumpExceptions, and respond with exception stack traces in development mode, however for both environments we utilize methodOverride and bodyParser. Note the use of app.router, which can (optionally) be used to mount the application routes, otherwise the first call to app.get(), app.post(), etc will mount the routes. Settings: Express supports the following settings out of the box: 1)Basepath Application base path used for res.redirect() and transparently handling mounted apps. 2)Views Root views directory defaulting to CWD/views 3)View engine Default view engine name for views rendered without extensions 4)View options An object specifying global view options 5)View cache Enable view caching (enabled in production) 6)Case sensitive routes Enable case-sensitive routing 7)Strict routing When enabled trailing slashes are no longer ignored 8)Jsonp callback Enable res.send() / res.json() transparent jsonp support Routing: Express utilizes the HTTP verbs to provide a meaningful, expressive routing API. For example we may want to render a user's account for the path /user/12, this can be done by defining the route below. The values associated to the named placeholders are available as req.params. Passing Route Control: We may pass control to the next matching route, by calling the third argument, the next() function. When a match cannot be made, control is passed back to Connect, and middleware continue to be invoked in the order that they are added via use(). The same is true for several routes which have the same path defined, they will simply be executed in order until one does not call next() and decides to respond.

mime

Mime is mainly used for mapping API.

nodeunit

Easy unit testing for node.js and the browser

qs

Query string parser for node supporting nesting.This module provides utilities for dealing with query strings.Query string parser for node supporting nesting.This module provides utilities for dealing with query strings.

Methods:
1)querystring.stringify(obj, sep='&', eq='=').
2)Serialize an object to a query string. Optionally override the default separator and assignment characters.
3)querystring.parse(str, sep='&', eq='=').
4)Deserialize a query string to an object. Optionally override the default separator and assignment characters. querystring.escape.
5)The escape function used by querystring.stringify, provided so that it could be

overridden if necessary. querystring.unescape.
6)The unescape function used by querystring.parse, provided so that it could be overridden if necessary.

xml2js

Simple XML to JavaScript object converter.