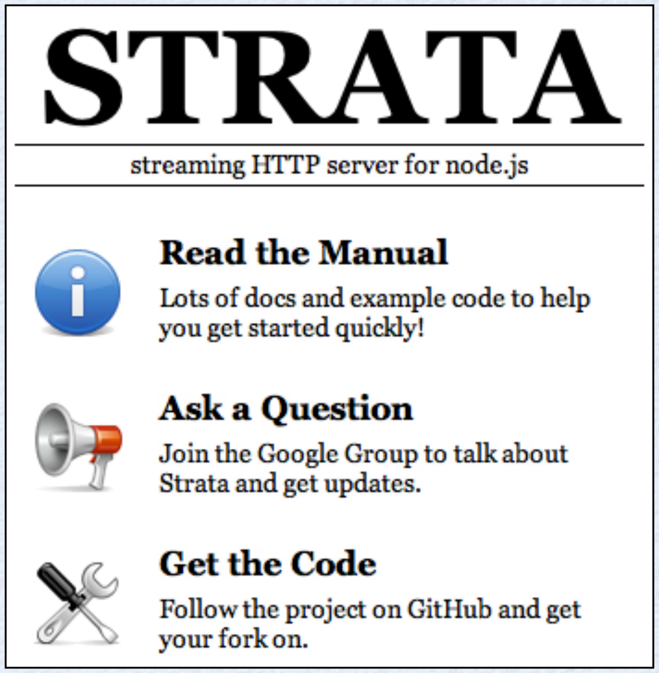
Strata

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Overview

```
npm install [-g] strata
var strata = require('strata');
```

- Node.js streaming HTTP server
- Based on
  - Web Server Gateway Interface (WSGI) - a Python standard at http://wsgi.org
  - Rack - a Ruby Webserver interface at http://rack.github.com
- Developed by Michael Jackson
  - a web developer at Twitter
  - also contributes to Mustache templating library
- Used by Twitter
- http://stratajs.org


STRATA

streaming HTTP server for node.js

Read the Manual
Lots of docs and example code to help you get started quickly!

Ask a Question
Join the Google Group to talk about Strata and get updates.

Get the Code
Follow the project on GitHub and get your fork on.

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

“Middleware”

- A JavaScript function that can
  modify requests before they are processed and
  modify responses before they are returned
- To register a middleware,
  `strata.use(`*`middleware-fn`*`, [`*`args`*`]);`
- Much more on this later!

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Benefits

over other Node HTTP frameworks

- Supports streams
  - important for working with large requests and/or responses
    so all the data doesn’t have to be in memory at once
- Middleware can operate in request/response flow
  - can operate before (upstream) or after (downstream) main processing
  - not easy to do in Express (popular alternative to Strata)
- Middleware abstraction makes mock testing easy
  - provides mock for server operations
  - not covered

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Serving Static Files

- To serve static files from a directory
  - approach #1
    - register `strata.file` middleware, typically before other middleware that contains application logic
    - `strata.use(strata.file, dirPath);`
    - takes optional 3rd argument to specify file names to use when URL leads to a directory
      - ex. `'index.html'`
      - can be a string or an array of strings to be attempted in order
      - if none are found, processing continues with next registered middleware
  - approach #2
    - pass main app to `strata.file` middleware
    - `strata.file(app, dirPath);`
    - takes same optional 3rd argument
- To serve a directory listing for paths that lead to a directory
  - register both the `strata.file` and `strata.directory` middlewares
    - `strata.use(strata.file, dirPath);`
    - `strata.use(strata.directory, dirPath);`

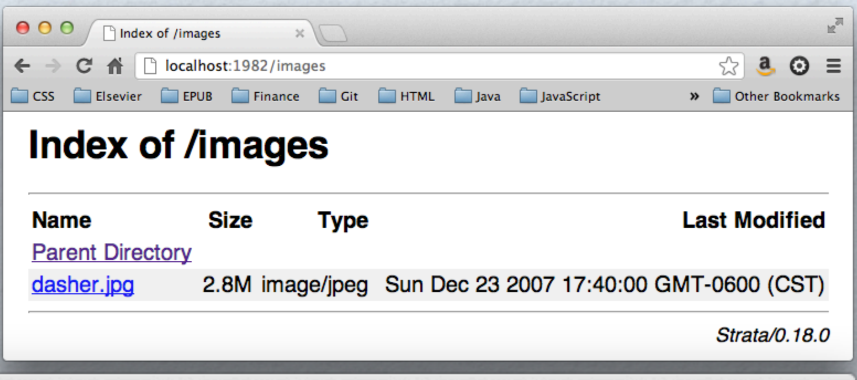
Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

File/Directory Serving Example

server.js

```
var strata = require('strata');

strata.use(strata.file, 'public',
  'index.html');
strata.use(strata.directory, 'public');
strata.run(); // port defaults to 1982
```

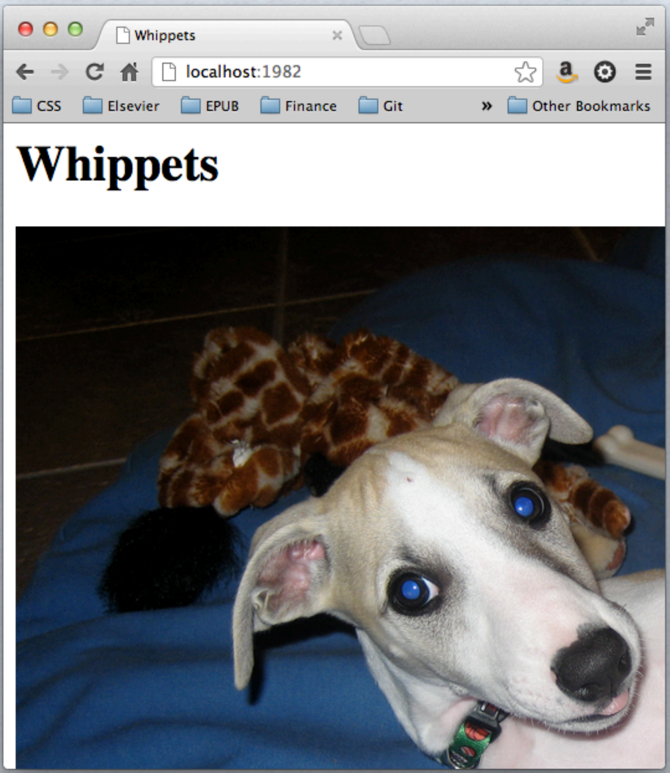


localhost:1982/images

Index of /images

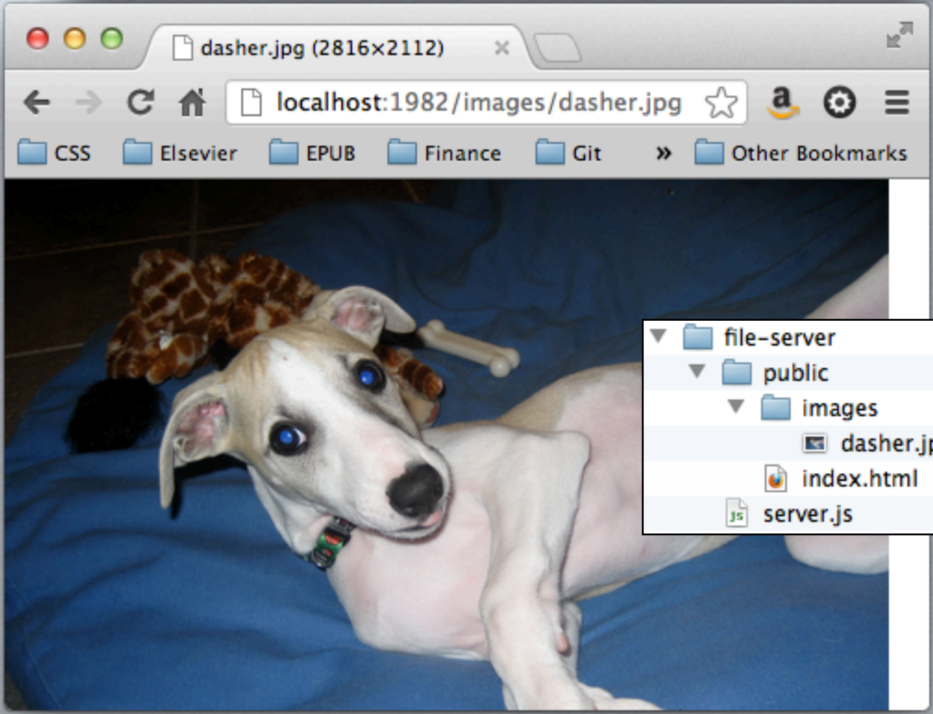
Name	Size	Type	Last Modified
Parent Directory			
dasher.jpg	2.8M	image/jpeg	Sun Dec 23 2007 17:40:00 GMT-0600 (CST)

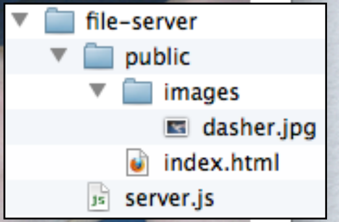
Strata/0.18.0



localhost:1982

Whippets





```
▼ file-server
  ▼ public
    ▼ images
        dasher.jpg
      index.html
    server.js
```

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Running Servers

- Enter "**`node server.js`**"
- Browse http://localhost:{port}
  - default port is 1982; year author was born
  - to listen on a different port, **`strata.run({port: `*`port`*`});`**

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Routing

- Maps URL patterns and request methods (GET, PUT, ...) to app functions
  - patterns must be strings or **`RegExp`** objects
- Each route specifies a pattern, app and optional request method
  - app is function that will process request
- To configure a route that is
  - only used for one request method
    - **`strata.`*`method`*`(`*`pattern`*`, `*`app`*`);`** ← implemented using **`strata.route()`**
    - ***`method`*** is **`get`**, **`post`**, **`put`**, **`delete`** or **`head`**
  - used for more than one request method
    - **`strata.route(`*`pattern`*`, `*`app`*`, `*`method-array`*`);`**
    - if ***`method-array`*** is omitted, it matches any request method
    - if ***`method-array`*** is a string, it is assumed to be a single request method
- If no matching route is found,
  the app passed to the following is used
  - **`var server = strata.run(`*`app`*`);`**

Basic Examples

```
var strata = require('strata');
strata.get('/', function (env, cb) {
  var headers = {
    'Content-Type': 'text/plain',
    'Content-Length': '12'
  };
  cb(200, headers, 'Hello world!');
});
strata.run();
```

the anonymous function in each of these examples is referred to as the **"downstream app."**

- Environment
  - in **`env`** parameter; described on slide 14
- Callback is passed
  - HTTP status code
  - object containing HTTP headers
  - response data; a string or readable **`Stream`**

```
var strata = require('strata');
strata.use(strata.contentType,
  'text/plain'); // default
strata.use(strata.contentLength);
strata.get('/', function (env, cb) {
  cb(200, {}, 'Hello, World!');
});
strata.run();
```

same, but using provided middleware to determine content type and calculate content length

```
var strata = require('strata');
strata.run(function (env, cb) {
  var content = 'Hello, World!';
  var res = strata.Response(content);
  res.contentLength =
    Buffer.byteLength(content);
  res.contentType = "text/plain";
  res.send(cb);
});
```

this approach handles any path not handled by a specific route

`strata.Response` objects provide an alternative to directly invoking **`cb`** to specify the response. They have a **`headers`** property, the methods **`setHeader`** and **`addHeader`**, and many convenience methods for getting and setting headers.

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Executable

- Provides an alternate way of writing and running Strata apps
- Write a module that exports an app function
  - `module.exports = function (env, cb) { ... };`
- To run
  - `strata -p` ***`port module-name`***`.js`
- During development, to cause server to reload code changes at some time interval
  - `strata -p` ***`port`*** `-r` ***`seconds module-name`***`.js`
- **`strata`** script is in **`node_modules/strata/bin`**

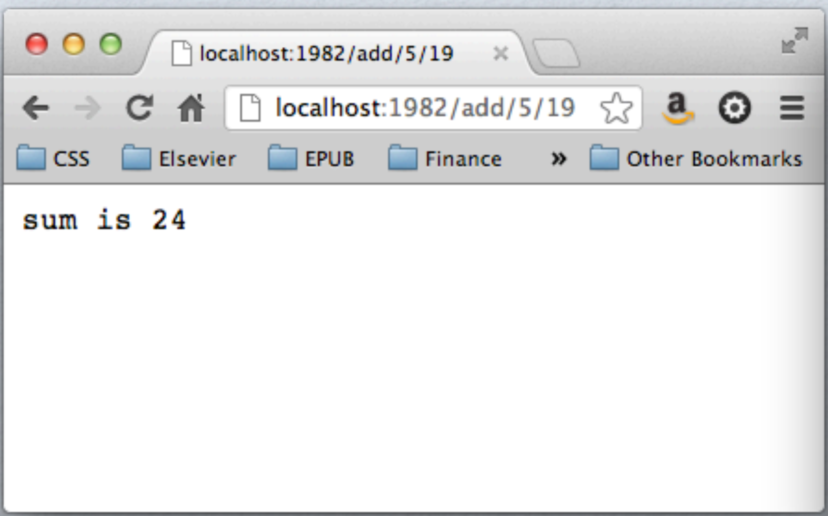
Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Advanced Routes

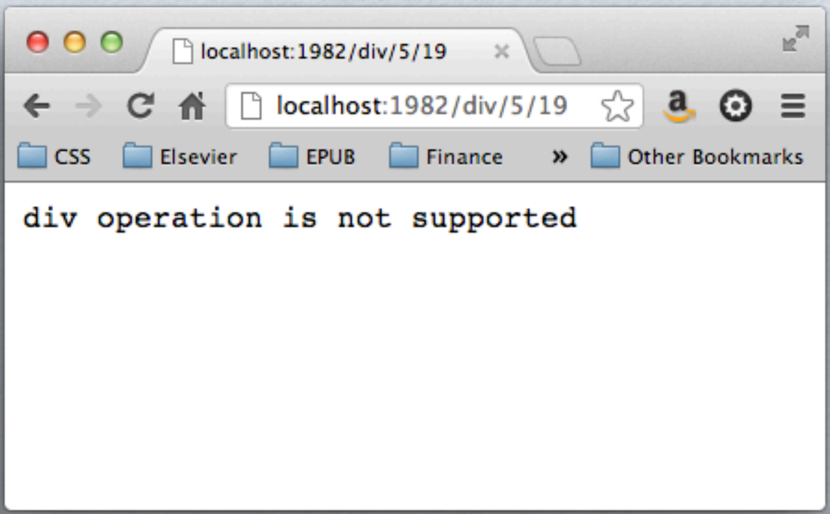
- To match URLs with parts that are data to be extracted
  - specify the parts with names preceded by colons
  - access values of parts with **`env.route.`*`name`***
- Example
  - specify route with **`strata.get('/student/:id', `*`app`*`);`**
  - get value of id inside app with **`env.route.id`**
- Can also extract data from any part of path using regular expressions
  - on **`env.pathInfo`**

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Routes Example ...



```
sum is 24
```



```
div operation is not supported
```

```
var strata = require('strata');

var BAD_REQUEST = 400;
var OK = 200;

strata.use(strata.contentType, 'text/plain'); // default
strata.use(strata.contentLength);
```

continued on next slide

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

... Strata Routes Example

```
strata.get('/add/:n1/:n2', function (env, cb) {
  var n1 = parseInt(env.route.n1, 10);
  var n2 = parseInt(env.route.n2, 10);
  var result;
  var status;

  if (isNaN(n1) || isNaN(n2)) {
    result = 'path parts after "add" must be integers';
    status = BAD_REQUEST;
  } else {
    result = 'sum is ' + (n1 + n2);
    status = OK;
  }

  var headers = {};
  cb(status, headers, result);
});

strata.run(function (env, cb) {
  var path = env.pathInfo.substring(1); // removes leading slash
  var operation = path.split('/')[0];
  var msg = operation + ' operation is not supported';
  var headers = {};
  cb(BAD_REQUEST, headers, msg);
});
```

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Environment Objects

- Passed to app function
- Plain object with these properties and more
  - **`protocol`** - part of request URL before host; **`'http:'`** or **`'https:'`**
  - **`requestMethod`** - **`'GET'`**, **`'POST'`**, **`'PUT'`**, **`'DELETE'`** or **`'HEAD'`**
  - **`serverName`** - host part of request URL; ex. 0.0.0.0 for localhost
  - **`serverPort`** - port part of request URL; ex. 1982 (the default port)
  - **`pathInfo`** - part of request URL after host and port and before **`?`**
  - **`queryString`** - part of request URL after **`?`**
  - **`headers`** - object containing header names (lowercase) and values
  - **`remoteAddr`** - client IP address
  - **`input`** - a readable **`Stream`** object for reading body
    - it’s paused; resume with **`env.input.resume();`**
  - **`error`** - a writable **`Stream`** object; defaults to stderr
  - **`session`** - object containing session data

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata REST Example ...

- Server maintains a collection of key/value pairs
- Clients can
  - PUT a key/value pair to add a key or modify an existing key

    -
      ```
      curl -XPUT http://localhost:1982/list/player \
        -H 'Content-Type: application/json' -d '{"name": "Gretzky", "number": 99}'
      ```

    -
      ```
      curl -XPUT http://localhost:1982/list/dog \
        -H 'Content-Type: text/plain' -d 'Rudy'
      ```
  - GET all the key/value pairs
    - `curl http://localhost:1982/list` `{"player":{"name":"Gretzky","number":99},"dog":"Rudy"}`
  - GET the value of a specific key
    - `curl http://localhost:1982/list/player` `{"name":"Gretzky","number":99}`
    - `curl http://localhost:1982/list/dog` `"Rudy"`
  - DELETE a specific key
    - `curl -XDELETE http://localhost:1982/list/player`
    - `curl http://localhost:1982/list/player`
    - `curl http://localhost:1982/list` `{"dog":"Rudy"}`

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

... Strata REST Example ...

```
var strata = require('strata');

var BAD_REQUEST = 400;
var NO_CONTENT = 204;
var NOT_FOUND = 404;
var OK = 200;
var list = {};
```

```
strata.put('/list/:key', function (env, cb) {
  var key = env.route.key;

  var contentType = env.headers['content-type'];
  var isJSON = contentType === 'application/json';

  var bufs = [];
  env.input.on('data', function (buf) {
    bufs.push(buf);
  });
  env.input.on('end', function () {
    var body = Buffer.concat(bufs).toString();
    try {
      list[key] = isJSON ? JSON.parse(body) : body;
      cb(NO_CONTENT, {}, '');
    } catch (e) {
      cb(BAD_REQUEST, {}, e.toString());
    }
  });
  env.input.resume();
});

strata.get('/list', function (env, cb) {
  cb(OK, {'Content-Type', 'application/json'},
    JSON.stringify(list));
});
```

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata REST Example

```
strata.get('/list/:key', function (env, cb) {
  var key = env.route.key;
  var value = list[key];
  if (value) {
    cb(OK, {'Content-Type', 'application/json'},
      JSON.stringify(value));
  } else {
    cb(NOT_FOUND, {}, key + ' not found');
  }
});

strata.delete('/list/:key', function (env, cb) {
  var key = env.route.key;
  var value = list[key];
  if (value) {
    delete list[key];
    cb(NO_CONTENT, {}, '');
  } else {
    cb(NOT_FOUND, {}, key + ' not found');
  }
});

strata.run(function (env, cb) {
  var msg = env.requestMethod + ' ' + env.pathInfo +
    ' is not supported';
  cb(BAD_REQUEST, {}, msg);
});
```

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Requests

- In addition to getting information about a request from **`env`**, a **`Request`** object can be created from it
- **`var req = strata.Request(env);`**
- Simplifies some operations
    - parsing common content types
      including multipart bodies (not covered)
    - getting query parameter values
        - **`req.query(function (err, params) { ... });`**
    - getting body parameter values
        - replace **`query`** above with **`body`**
    - getting union of parameter values with body taking precedence
        - replace **`query`** above with **`params`**

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Responses ...

- Alternative to directly invoking the app **`cb`** with response text
  - **`var res = strata.Response(`*`content`*`);`**
  - ***`content`*** can be a string or stream
- Properties
  - **`status`** - set to response HTTP status code
  - **`contentType`** - set to MIME type string
  - **`contentLength`** - set to body length
  - **`lastModified`** - set so clients can avoid retrieving content that hasn't changed since last request
  - **`headers`** - an object that holds header names and values
  - **`body`** - set to body text

There are properties like these for every standard HTTP header. See list in https://github.com/mjijackson/strata/blob/master/lib/response.js

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

... Responses

- Methods
  - `setHeader(`*`name`*`, `*`value`*`)` - to set one value for a header
  - `addHeader(`*`name`*`, `*`value`*`)` - to set more than one value for the same header
  - `removeHeader(`*`name`*`)`
  - `hasHeader(`*`name`*`)`

  - `setCookie(`*`name`*`, `*`value`*`)`
  - `removeCookie(`*`name`*`)`

  - `redirect(`*`url`*`, [`*`status`*`])` - status defaults to 302; means resource temporarily resides at a different URI
  - `send(`*`cb`*`)` - shorthand for `cb(res.status, res.headers, res.body);`

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

File Streaming Example

```
var fs = require('fs');
var strata = require('strata');

var NOT_MODIFIED = 304;

strata.get('/dasher', function (env, cb) {
  var path = 'dasher.jpg';
  fs.stat(path, function (err, stats) {
    if (err) {
      return strata.handleError(err, env, cb);
    }

    // If the 'If-Modified-Since' header was supplied
    // and the file has not been modified since then ...
    var ifModifiedSince = env.headers['if-modified-since'];
    if (ifModifiedSince && stats.mtime <= new Date(ifModifiedSince)) {
      // Don't bother returning the file content.
      return cb(NOT_MODIFIED, {});
    }

    var res = strata.Response(fs.createReadStream(path));
    //res.contentType = 'image/jpeg'; // browser can detect this
    res.contentLength = stats.size; // uses chunked transfer encoding without this
    res.lastModified = stats.mtime;
    res.send(cb);
  });
});

strata.run();
```

1. run "`node server.js`"
2. open browser
3. open tool to view network traffic
4. browse http://localhost:1982/dasher
5. note that response status code is 200
6. refresh page
7. note that response status code is 304

using global error handler ... explained next

from Wikipedia ...

"**Chunked transfer encoding** is a data transfer mechanism in version 1.1 of the Hypertext Transfer Protocol (HTTP) in which data is sent in a series of chunks."

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Error Handling ...

- Improvement over standard JavaScript error handling
  - works with asynchronous calls, preserving full stack trace
  - can create custom error types
- Custom error types
  - used for app and middleware callback “err” values, not meant to be thrown
  - inherit from **`strata.Error`**
  - **`cause`** can be another **`strata.Error`** object
  - **`strata.Error`** objects have a **`fullStack`** property for determining origin of errors
    - automatically populated

```
var strata = require('strata');
var util = require('util');

function MyCustomError(message, cause) {
  message = message || 'default message';
  strata.Error.call(this, message, cause);
}

util.inherits(MyCustomError, strata.Error);
```

```
cb(new MyCustomError(
  'cannot run fast', 'too hot'));
```

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

... Strata Error Handling

- Global error handler
  - **`strata.handleError`**
  - sends 500 status (Internal Server Error) to client
    and writes stack trace to **`env.error`** stream
  - recommended way to check for and handle errors passed to callbacks of asynchronous functions

```
if (err && strata.handleError(err, env, callback)) {
  return;
}
```

    - provided implementation of **`strata.handleError`** always returns true
  - custom implementations
    - assign new function to **`strata.handleError`**
    - can return a status other than 500, for example, 400 for bad request data
    - can return **`false`** to allow processing to continue after certain errors

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Middleware

- A Strata "app" is a function that conforms to rules in the Strata spec.
  - https://github.com/mjijackson/strata/blob/master/SPEC
- A Strata "middleware" is an "app" that
  - takes an app (the downstream app or another middleware)
    and optional arguments used to configure it
  - returns another function that takes an environment object (**`env`**) and a callback (**`cb`**)
    - Strata will call this function
    - **`app`** is captured via closure
  - can do things during initialization,
    before running the app passed to it (upstream) and
    after running the app passed to it (downstream)
  - can modify request in upstream part (**`env`**)
  - can modify response in downstream part (**`status`**, **`headers`** and **`body`**)
- Executed in the order in which they are passed to
  `strata.use(`*`middleware-fn, middleware-args`*`);`
  - ***`middleware-args`*** are passed to ***`middleware-fn`***, preceded by **`app`**, when Strata invokes it

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Request Logging

- To log all requests to console where server is running
  - `strata.use(strata.commonLogger); // writes to stderr`
  - `strata.use(strata.commonLogger, fs.createWriteStream('server.log'));`
  - **`strata.commonLogger`** is one of many provided middlewares
  - used in example on next slide

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Middleware Example

mw.js

```
module.exports = function (app, param) {
1   console.log('mw: initializing');
    console.log('mw: param =', param);

    return function (env, cb) {
2     console.log('mw: upstream');
      app(env, function (status, headers, body) {
3       console.log('mw: downstream');
        cb(status, headers, body);
      });
    };
};
```

Note how a middleware function can do things in **three places**: during initialization and before and after it calls the app function passed to it.

This means that each middleware function can effectively wrap the next middleware function, providing **AOP-like before and after functionality**.

server.js

```
var strata = require('strata');
var mw = require('mw');

strata.use(strata.commonLogger);
strata.use(mw, 'foo');

strata.run(function (env, cb) {
  console.log('server: handling request');
  console.log('pathInfo =', env.pathInfo);
  cb(200, {}, 'Hello, World!');
});
```

app function

output

```
>> Strata web server version 0.15.1 running on node 0.8.1
>> Listening on 0.0.0.0:1982, CTRL+C to stop
mw: initializing
mw: param = foo
mw: upstream
server: handling request
pathInfo = /
mw: downstream
127.0.0.1 - - [22/Jul/2012:10:05:55 -0500] "GET / HTTP/1.1" 200 13
mw: upstream
server: handling request
pathInfo = /favicon.ico
mw: downstream
127.0.0.1 - - [22/Jul/2012:10:05:55 -0500] "GET /favicon.ico HTTP/1.1" 200 13
```

```
module.exports = function (app) {
  return function (env, cb) {
    app(env, function (status, headers, body) {
      cb(status, headers, body);
    });
  };
};
```

simplest possible, no-op middleware

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Upstream vs. Downstream

mw1.js

```
module.exports = function (app, p1) {
  console.log('mw1: initializing; p1 =', p1);

  return function (env, cb) {
    var pathInfo = env.pathInfo;
    console.log('mw1: upstream for', pathInfo);
    app(env, function (status, headers, body) {
      console.log('mw1: downstream for', pathInfo);
      cb(status, headers, body);
    });
  };
};
```

mw2.js

```
module.exports = function (app, p1) {
  console.log('mw2: initializing; p1 =', p1);

  return function (env, cb) {
    var pathInfo = env.pathInfo;
    console.log('mw2: upstream for', pathInfo);
    app(env, function (status, headers, body) {
      console.log('mw2: downstream for', pathInfo);
      cb(status, headers, body);
    });
  };
};
```

server.js

```
var strata = require('strata');
var mw1 = require('mw1');
var mw2 = require('mw2');

strata.use(mw1, 'foo');
strata.use(mw2, 'bar');

strata.get('/*', function (env, cb) {
  console.log('server: for', env.pathInfo);
  var content = 'Hello, World!';
  var headers = {
    'Content-Type': 'text/plain',
    'Content-Length': content.length
  };
  cb(200, headers, content);
});

strata.run();
```

browser http://localhost:1982

output

```
>> Strata web server version 0.15.1 running on node
0.8.1
>> Listening on 0.0.0.0:1982, CTRL+C to stop
mw2: initializing; p1 = bar
mw1: initializing; p1 = foo
mw1: upstream for /
mw2: upstream for /
server: for /
mw2: downstream for /
mw1: downstream for /
mw1: upstream for /favicon.ico
mw2: upstream for /favicon.ico
server: for /favicon.ico
mw2: downstream for /favicon.ico
mw1: downstream for /favicon.ico
```

JSONP Middleware ...

```
var strata = require('strata');
var Stream = require('stream');
var util = require('util');

/**
 * A basic read/write stream.
 */
function MyStream() {
  Stream.call(this);
  this.readable = this.writable = true;
}
util.inherits(MyStream, Stream);
MyStream.prototype.end = function (data) {
  this.emit('end');
};
MyStream.prototype.pause = function () {};
MyStream.prototype.resume = function () {};
MyStream.prototype.write = function (data) {
  this.emit('data', data);
};

function streamJsonP(stream, cbName, body) {
  stream.write(cbName);
  stream.write('(');
  body.pipe(stream, {end: false});
  body.on('end', function () {
    stream.write(')');
    stream.end();
  });
}
```

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

... JSONP Middleware

```
function middleware(app) {

  return function (env, cb) {
    // Get the "callback" query parameter.
    var req = strata.Request(env);
    req.query(function (err, params) {
      if (err) return cb(err);

      var cbName = params.callback;

      app(env, function (status, headers, body) {
        if (cbName && status === 200) {
          var stream = new MyStream();
          cb(status, headers, stream);
          streamJsonP(stream, cbName, body);
        } else {
          cb(status, headers, body);
        }
      });
    });
  };
}

exports.middleware = middleware;
```

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

Strata Middleware Lint

Good idea, but I can't get it to work!

- To report errors in middleware, register **`strata.lint`** middleware before other middlewares

```
...
strata.use(strata.lint);
strata.use(someMiddleware);
strata.use(anotherMiddleware);
...
```

throws **`new strata.Error(`*`message`*`)`** for any violations

- Checks
  - app is called with two arguments, environment and callback
  - environment satisfies these checks →
  - callback is a function that takes three arguments, status, headers and body
  - status passed to callback is a number between 100 and 599
  - headers passed to callback have valid names and string values
  - body passed to callback is a string or **`EventEmitter`**

    all streams inherit from **`EventEmitter`**
  - if status passed to callback is 1xx, 204 or 304, there is no **`'Content-Type'`** header; otherwise there is
  - if status passed to callback is 1xx, 204 or 304, there is no **`'Content-Length'`** header

Environment Checks

- is an object
- has required properties
- some required properties have string values
- **`env.requestTime`** has a **`Date`** value
- **`env.protocol`** is **`'http:'`** or **`'https:'`**
- **`env.requestMethod`** is an uppercase string
- if **`env.scriptName`** exists, it starts with **`'/'`** and contains other characters
- **`env.input`** is a readable Stream
- **`env.error`** is an **`EventEmitter`** and a writeable stream
- if **`env.session`** exists, it is an object
- **`env.strataVersion`** is an array of three numbers (major, minor and patch)

Strata Advanced Topics ...

- Sessions
  - data persistence across sessions is provided via cookies
  - enabled by registering **`strata.sessionCookie`** middleware
  - to get or set a session cookie, access **`env.session.`*`cookieName`***
  - to clear all session cookies, **`env.session = {};`**
- Redirects
  - to redirect client to a new URL, **`strata.redirect(env, cb, `*`url`*`);`**
  - to redirect to a new URL and return to original URL after some action at new URL,
    `strata.redirect.forward(env, cb, `*`url`*`);` and
    `strata.redirect.back(env, cb);`
    - for example, redirecting to login page for users that haven't authenticated, and then back to original URL after successful authentication
- URL Rewriting
  - to rewrite a requested URL as a different URL,
    `strata.rewrite(app, `*`oldURL`*`, `*`newURL`*`);`

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

... Strata Advanced Topics

- Content Negotiation
  - to determine if client accepts a particular media type (ex. "text/html"),
    `var req = strata.Request(env);`
    `if (req.accepts(`*`mediaType`*`)) ...`
  - Strata doesn't help with determining the preference order for multiple accepted media types
    - expressed using "q" values in **`Accept`** header string
    - negotiator Node module at https://github.com/federomero/negotiator does this
  - also see these **`Request`** object methods
    - **`acceptsCharset, acceptsEncoding, acceptsLanguage`**
- File Uploads
  - see Strata manual for details
- Gzip Encoding
  - see Strata manual for details

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.

References

- Presentation from the author
  - http://stratajs.org/slides.pdf
- Specification for “applications”, “environment”, and other Strata topics
  - https://github.com/mjijackson/strata/blob/master/SPEC
- Source code
  - https://github.com/mjijackson/strata

Copyright © 2012-2013 by Object Computing, Inc. (OCI).
All rights reserved.