express

high class web development for nodeJS

Dennis Blöte, Rails Usergroup Hamburg, 11/10/2010

What‘s express?

- nodeJS based web framework
- inspired by Sinatra
- asynchronous

```
var app = express.createServer();

app.get('/', function(req, res){
    res.send('Hello World');
});

app.listen(3000);
```

Installation

- Install nodeJS

```
$ brew install node
```

- Install npm (node package manager)

```
$ brew install npm
```

- Install express

```
$ npm install express
```

Configuration (I)

```
app.configure(function(){
    app.use(app.router);
    app.use(express.staticProvider(__dirname + '/public'));
});

app.configure('development', function(){
    app.use(express.errorHandler({
      dumpExceptions: true,
      showStack: true
    }));
});

app.configure('production', function(){
    app.use(express.errorHandler());
});
```

Configuration (2)

```
app.configure(function(){
    app.set('views', __dirname + '/views');
    app.set('views'); // => returns views dir

    app.enable('feature');  // sets feature to true
    app.disable('feature'); // sets feature to false
});
```

Starting with a specific environment:

```
$ NODE_ENV=production node app.js
```

Routing

```
app.get('/users/:id?', function(req, res) {
    var id = req.params.id;
    res.send(id ? 'user ' + id : 'users');
});

app.post('/users/:id.:format', function(req, res) {
    var id = req.params.id;
    var format = req.params.format;
    res.send('user: ' + id + ' format: ' + format);
});

app.get('/file/*.*', function(req, res) {
    res.send('path: ' + req.params);
});
```

Connect / Middlewares

- Connect is a middleware framework
- Similar to Rack
- A middleware is simply a function with three arguments: request, response, next

```
var helloWorld = function(req, res, next) {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('hello world');
}
```

Middlewares

```
var loadUser = function(req, res, next) {
    var id = req.params.id;
    var user = db.loadUser(id); // fetch user from db
    if (user) {
        req.user = user;
        next();
    } else {
        next(new Error('Failed to load user ' + id));
    }
}

app.get('/user/:id', loadUser, function(req, res) {
    res.send('Viewing user ' + req.user.name);
});
```

HTTP POST

```
<form method="post" action="/">
    <input type="text" name="user[name]" />
    <input type="submit" value="Submit" />
</form>
```

bodyDecoder middleware for POST params:

```
app.use(express.bodyDecoder());

app.post('/', function(req, res){
    console.log(req.body.user);
    res.redirect('back');
});
```

HTTP PUT/DELETE

```
<form method="post" action="/">
  <input type="hidden" name="_method" value="put" />
  <input type="text" name="user[name]" />
  <input type="submit" value="Submit" />
</form>
```

bodyDecoder and methodOverride:

```
app.use(express.bodyDecoder());
app.use(express.methodOverride());

app.post('/', function(req, res){
    console.log(req.body.user);
    res.redirect('back');
});
```

View Rendering

- Concept: Layout, view and partials
- Template Engines: Jade, Haml, EJS, …
- View variables are passed to render

```
app.set('view engine', 'jade');

app.get('/', function(req, res){
    res.render('index.haml', {
        layout: 'app', // -> app.jade
        locals: { title: 'This is my app' }
    });
});
```

Jade - Template Engine

```
!!! 5
html(lang="en")
  head
    title= pageTitle
    :javascript
      | if (foo)
      |   bar()
  body
    h1 Jade - node template engine
    #container
      - if (youAreUsingJade)
        p You are amazing
      - else
        p Get on it!
      .comments
        = partial('comment', comments)
```

Helpers

```
app.dynamicHelpers({
  flashes: function(req, res) {
    var msg = req.flash('message');
    return msg.length ? '<p id="msg">' + msg + '</p>': '';
  }
});

app.helpers({
  linkTo: function(text, url) {
    return '<a href=' + url + '>' + text + '</a>';
  },
  name: function(user) {
    return user.firstName + ' ' + user.lastName;
  }
});
```

Sessions

- Currently depending on cookies
- Store implementations: Memory, Redis, …

```
app.use(express.cookieDecoder());
app.use(express.session());

app.post('/cart/add', function(req, res) {
    req.session.items = req.body.items;
    res.redirect('back');
});

app.get('/cart', function(req, res) {
    var items = req.session.items;
    res.render('cart', { locals: { items: items } });
});
```

Error Handling

The app.error() method receives exceptions thrown within a route, or passed to next(err)

```
app.use(express.errorHandler({ dumpExceptions: true }));

app.error(function(err, req, res, next) {
    if (err instanceof Unauthorized) {
        res.render('401.jade', { locals: { error: err } });
    } else if (err instanceof NotFound) {
        res.render('404.jade');
    } else {
        next(err);
    }
});
```

Why express?

- Full-blown feature set
- Built on Connect
- Good documentation
- Lots of examples
- Many extensions
- Nice community

Going further…

- http://expressjs.com/guide.html
- http://expressjs.com/api.html
- http://senchalabs.github.com/connect/
- http://github.com/senchalabs/connect/wiki
- http://github.com/visionmedia/express/tree/master/examples/
- https://github.com/dbloete/Codeshelver