

Error Handling

Error Handling



Using exception handling +
an error partial to catch and
report errors

Select Amount ▼

☐ Paypal
 ☐ Direct

Donate

```

<form action="/donate" method="POST">
  <div class="ui dropdown" name="amount">
    <input type="hidden" name="amount">
    <div class="text">Select Amount</div>
    <i class="ui dropdown icon"></i>
    <div class="menu">
      <div class="item">50</div>
      <div class="item">100</div>
      <div class="item">1000</div>
    </div>
  </div>
  <div class="grouped inline fields">
    <div class="field">
      <div class="ui radio checkbox">
        <input type="radio" name="method" value="paypal">
        <label>Paypal</label>
      </div>
    </div>
    <div class="field">
      <div class="ui radio checkbox">
        <input type="radio" name="method" value="direct">
        <label>Direct</label>
      </div>
    </div>
  </div>
  <button class="ui blue submit button">Donate</button>
</form>

```

```
{ method: 'POST', path: '/donate', config: Donations.donate },
```

```
const donationSchema = new Schema({
  amount: Number,
  method: String
});
```

```

donate: {
  handler: async function(request, h) {
    const data = request.payload;
    const newDonation = new Donation({
      amount: data.amount,
      method: data.method
    });
    await newDonation.save();
    return h.redirect('/report');
  }
}

```

- No validation
- No error handling
- No error reporting

```

donate: {
  handler: async function(request, h) {
    const data = request.payload;
    const newDonation = new Donation({
      amount: data.amount,
      method: data.method
    });
    await newDonation.save();
    return h.redirect('/report');
  }
}

```

- Terminate mongod after user has logged in (Ctrl-C)

```

2. bash
0.0.1:62017 #1 (1 connection now open)
2019-01-31T11:40:13.155+0000 I NETWORK [conn1] received client metadata from 127.0.0.1:62017 conn1: { driver: { name: "nodejs", version: "3.1.10" }, os: { type: "Darwin", name: "darwin", architecture: "x64", version: "18.2.0" }, platform: "Node.js v10.15.0, LE, mongodb-core: 3.1.9" }
2019-01-31T11:57:00.934+0000 I NETWORK [conn1] end connection 127.0.0.1:62017 (0 connections now open)
^C2019-01-31T11:57:03.045+0000 I CONTROL [signalProcessingThread] got signal 2 (Interrupt: 2), will terminate after current cmd ends
2019-01-31T11:57:03.045+0000 I NETWORK [signalProcessingThread] shutdown: going to close listening sockets...
2019-01-31T11:57:03.045+0000 I NETWORK [signalProcessingThread] removing socket file: /tmp/mongodb-27017.sock
2019-01-31T11:57:03.046+0000 I CONTROL [signalProcessingThread] Shutting down free monitoring
2019-01-31T11:57:03.046+0000 I FTDC [signalProcessingThread] Shutting down full-time diagnostic data capture
2019-01-31T11:57:03.048+0000 I STORAGE [signalProcessingThread] WiredTigerKVEngine shutting down
2019-01-31T11:57:03.109+0000 I STORAGE [signalProcessingThread] shutdown: removing fs lock...
2019-01-31T11:57:03.110+0000 I CONTROL [signalProcessingThread] now exiting
2019-01-31T11:57:03.110+0000 I CONTROL [signalProcessingThread] shutting down with code:0
iMac:dev edelestar$

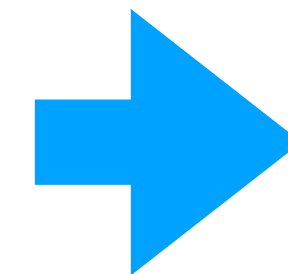
```

- Attempt Donation

50 ▼

☒ Paypal
 ☐ Direct

Donate



```

localhost:3000/donate
{
  statusCode: 500,
  error: "Internal Server Error",
  message: "An internal server error occurred"
}

```

Javascript Exceptions

- You can throw exceptions using the throw statement and handle them using the try...catch statements.
- Use the throw statement to throw an exception. When you throw an exception, you specify the expression containing the value to be thrown
- The try...catch statement marks a block of statements to try, and specifies one or more responses should an exception be thrown. If an exception is thrown, the try...catch statement catches it.

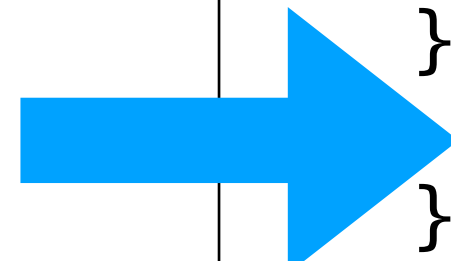
```
function getMonthName(mo) {  
  mo = mo - 1; // Adjust month number for array index (1 = Jan, 12 = Dec)  
  var months = ['Jan', 'Feb', 'Mar', 'Apr',  
                'May', 'Jun', 'Jul', 'Aug',  
                'Sep', 'Oct', 'Nov', 'Dec'];  
  
  if (months[mo]) {  
    return months[mo];  
  } else {  
    throw 'InvalidMonthNo'; //throw keyword is used here  
  }  
}  
  
try {  
  // statements to try  
  monthName = getMonthName(myMonth); // function could throw exception  
} catch (e) {  
  monthName = 'unknown';  
  logMyErrors(e); // pass exception object to error handler -> your own function  
}
```

```

donate: {
  handler: async function(request, h) {
    const data = request.payload;
    const newDonation = new Donation({
      amount: data.amount,
      method: data.method
    });
    await newDonation.save();
    return h.redirect('/report');
  }
}

```

- Wrap function body in try/catch
- If exception generated, display 'main' view and pass errors object structured to view



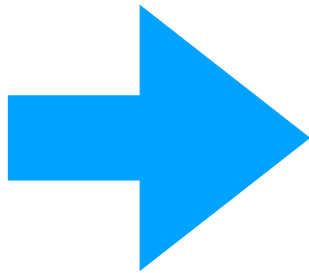
```

donate: {
  handler: async function(request, h) {
    try {
      const data = request.payload;
      const newDonation = new Donation({
        amount: data.amount,
        method: data.method,
      });
      await newDonation.save();
      return h.redirect('/report');
    } catch (err) {
      return h.view('main', { errors: [{ message: err.message }] });
    }
  }
}

```



error.hbs

```
{{#if errors}}
  <div class="ui negative message transition">
    <i class="close icon"></i>
    <div class="header">
      There was a problem...
    </div>
    <ul class="list">
      {{#each errors}}
        <li>{{message}}</li>
      {{/each}}
    </ul>
  </div>
{{/if}}
```



```
{{> welcomemenu }}

<section class="ui stacked segment">
  <div class="ui grid">
    <aside class="six wide column">
      
    </aside>
    <article class="ten wide column">
      <header class="ui header"> Help Me Run Springfield</header>
      <p> Donate what you can now - No Bitcoins accepted! </p>
      {{> error }}
    </article>
  </div>
</section>
```



Help Me Run Springfield

Donate what you can now - No Bitcoins accepted!

There was a problem...

- failed to reconnect after 30 attempts with interval 1000 ms

- Errors appear in semantic ui message if provided by handler





Help Me Run Springfield

Donate what you can now - No Bitcoins accepted!

There was a problem...

- failed to reconnect after 30 attempts with interval 1000 ms

```
donate: {  
  handler: async function(request, h) {  
    try {  
      const data = request.payload;  
      const newDonation = new Donation({  
        amount: data.amount,  
        method: data.method,  
      });  
      await newDonation.save();  
      return h.redirect('/report');  
    } catch (err) {  
      return h.view('main', { errors: [{ message: err.message }] });  
    }  
  }  
}
```

- Error Reporting Library
- General purpose HTTP error helper Library



HTTP-friendly error objects

build passing npm v7.3.0

Lead Maintainer: [Eran Hammer](#)

- [Boom](#)
 - `reformat(debug)`
 - [Helper Methods](#)
 - `new Boom(message, [options])`
 - `boomify(err, [options])`
 - `isBoom(err)`
 - [HTTP 4xx Errors](#)
 - `Boom.badRequest([message], [data])`
 - `Boom.unauthorized([message], [scheme], [attributes])`
 - `Boom.paymentRequired([message], [data])`
 - `Boom.forbidden([message], [data])`
 - `Boom.notFound([message], [data])`
 - `Boom.methodNotAllowed([message], [data], [allow])`
 - `Boom.notAcceptable([message], [data])`
 - `Boom.proxyAuthRequired([message], [data])`
 - `Boom.clientTimeout([message], [data])`
 - `Boom.conflict([message], [data])`
 - `Boom.resourceGone([message], [data])`

HTTP 4xx Errors

`Boom.badRequest([message], [data])`

Returns a 400 Bad Request error where:

- `message` - optional message.
- `data` - optional additional error data.

```
Boom.badRequest('invalid query');
```

Generates the following response payload:

```
{
  "statusCode": 400,
  "error": "Bad Request",
  "message": "invalid query"
}
```

HTTP 4xx Errors

Boom.badRequest([message], [data])

Returns a 400 Bad Request error where:

- `message` - optional message.
- `data` - optional additional error data.

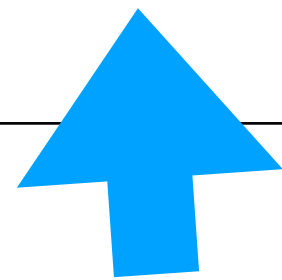
```
Boom.badRequest('invalid query');
```

Generates the following response payload:

```
{
  "statusCode": 400,
  "error": "Bad Request",
  "message": "invalid query"
}
```

```
userSchema.methods.comparePassword = function(candidatePassword) {  
  const isMatch = this.password === candidatePassword;  
  if (!isMatch) {  
    throw new Boom('Password mismatch');  
  }  
  return this;  
};
```

```
handler: async function(request, h) {  
  const { email, password } = request.payload;  
  try {  
    let user = await User.findByEmail(email);  
    if (!user) {  
      const message = 'Email address is not registered';  
      throw new Boom(message);  
    }  
    user.comparePassword(password);  
    request.cookieAuth.set({ id: user.id });  
    return h.redirect('/home');  
  } catch (err) {  
    return h.view('login', { errors: [{ message: err.message }] });  
  }  
}
```



Catching Exceptions

- Error in standard format from all sources

Throwing Exceptions

- If password doesn't match
- If no user registered with given email
- Database failure

```
handler: async function(request, h) {
  const { email, password } = request.payload;
  try {
    let user = await User.findByEmail(email);
    if (!user) {
      const message = 'Email address is not registered';
      throw new Boom(message);
    }
    user.comparePassword(password);
    request.cookieAuth.set({ id: user.id });
    return h.redirect('/home');
  } catch (err) {
    return h.view('login', { errors: [{ message: err.message }] });
  }
}
```

```
<article class="ten wide column">
  <header class="ui header"> Help Me Run Springfield</header>
  <p> Donate what you can now – No Bitcoins accepted! </p>
  {{> error }}
</article>
```

```
{{#if errors}}
  <div class="ui negative message transition">
    <i class="close icon"></i>
    <div class="header">
      There was a problem...
    </div>
    <ul class="list">
      {{#each errors}}
        <li>{{message}}</li>
      {{/each}}
    </ul>
  </div>
{{/if}}
```

