Method Override

Implementing the edit feature

Two Requests to Implement the Edit feature

Implementing the edit feature requires two requests.

1. Users use `http://localhost:5500/edit/11` to edit a post (ID 11 as an example).
2. Server returns a HTML page to edit the post 11.
3. Users update the post, and send it back again to the server for updating the post.

We need to make a PUT request to update a post, as a POST request is used to make a new post; this is the recommended and standard REST style.

Issues of Making the PUT request

However, we cannot make a PUT request using a web browser; there are two solutions.

1. Use POST request with edit endpoint: simple but not recommended for REST.
2. Use method override so that the server can responds to the POST request.

Implementing Edit Feature

Get Request + edit endpoint with ID

This is the code to process the ID (11 in the example) with users input `http://localhost:5500/edit/11`.

```
app.get('/edit/:id', async function (req, resp) {
  try {
    const posts = db.collection(POSTS)
    let res = await posts.findOne({ _id: parseInt(req.params.id) })
    console.log({ data: res })
    if (res != null) {
      resp.render('edit.ejs', { data: res })
    }
    else {
      resp.status(500).send({ error: 'result is null' })
    }
  }
  catch (error) {
    console.log(error)
    resp.status(500).send({ error: 'Error from db.collection().findOne()' })
  }
});
```

edit.ejs

```
<form action="/edit?_method=PUT" method="POST">
  <input value="<%= data._id %>" name="id" style="display : none">
  <div class="form-group">
    <label>Update: What to do today</label>
    <input type="text" class="form-control" name="title" value=<%=data.title %> />
  </div>
  <div class="form-group">
    <label>Update: Date</label>
    <input type="text" class="form-control" name="date" value=<%=data.date %> />
  </div>
  <button type="submit" class="btn btn-outline-secondary">Submit</button>
</form>
```

For a form, we should use the POST method; however, we prepend `?_method=PUT` after the `/edit` endpoint.

Method Override Middleware

We can use the method override middleware to transform this POST request with `?_method=PUT` into the PUT request.

```
const methodOverride = require('method-override')
app.use(methodOverride('_method'))
```

In this code, we instruct the method override middleware that any request with `_method` should be replaced with what follows (in this example, PUT).

1. Browser sends a POST request:

```
POST /edit?_method=PUT
content: id=11&title=NewTitle
```

2. methodOverride checks _method:

```
app.use(methodOverride('_method'));
```

If the request has _method=PUT

3. It replaces the HTTP method internally:

```
POST → PUT
```

4. Express receives:

```
PUT /edit
```

PUT request + edit endpoint

Then, Express invokes this method to update the post and redirect to `/list`.

```
app.put('/edit', async function (req, resp) {
  try {
    const posts = db.collection(POSTS);

    await posts.updateOne(
      { _id: parseInt(req.body.id) },
      { $set: { title: req.body.title, date: req.body.date } }
    );

    console.log('app.put.edit: Update complete');
    resp.redirect('/list');    // ✅ Correct
  } catch (e) {
    console.error(e);
    resp.status(500).send('Update error');
  }
});
```