Practical exercise on Express

Monday, June 18, 2018 10:10 PM

Practical exercises: the to do list

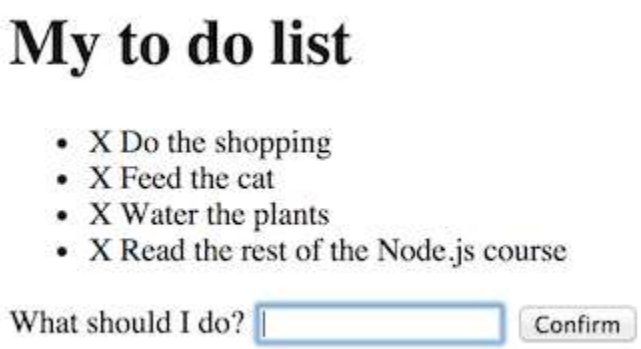
I think that it's high time we moved on to some practical exercises. We've done the rounds of a large number of basic Node.js features and we've learnt how to use the Express micro-framework.

But you only get a true feeling of understanding once you've practiced and completed your first real app. This is what I'm going to suggest you do in this practical exercise.


We're going to create a to do list. The visitor will simply be able to add and remove tasks.

So we'll start with something simple. Then you'll be able to improve it in any way you want to!

Before going any further, I'm going to show you the page that we're going to create (next figure).



The beautiful app we're going to create

- We can add elements to the to do list via the form.
- We can delete items by clicking on the crosses in the list.
- The list is stored in the visitor's session. If someone else connects to the site, they will have their own list.

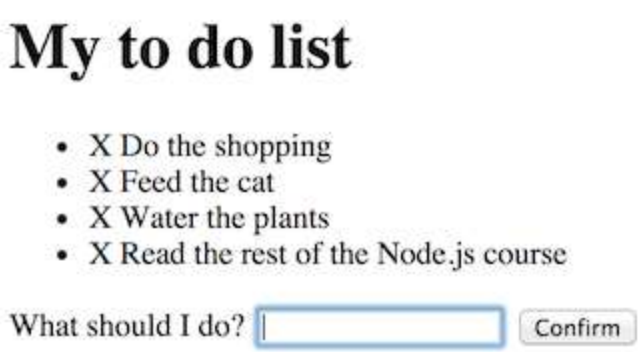
The instructions are simple and the final code won't be very long. However, if it's your first more complex app using Node.js, you'll probably fumble around a bit and move forward slowly. Don't worry, it's perfectly normal! Keep at it and if you really need to, read the "A bit of help" section before comparing your work with mine.


Don't forget to look at the Express documentation. I would never have been able to complete the exercise myself without referring to the documentation.
You've got all the information, now it's over to you!

Need help?

So you need a bit of help? You're lost and don't know where to start?


Remember how we want our final application to look like. I showed you a screenshot of it earlier (see next figure).



The app we're going to create. Still beautiful!

Modules and package.json

Let's take a deep breath and look at things in the right order. What do we need?

- **Express**: without the Express framework the job will be very difficult. Now that you know how to use this framework, it would be a shame to leave it out.
- **EJS** (or another template system): this will allow us to easily construct the HTML page that displays the to do list and the form.

In theory, these modules are enough. A good first exercise would be to create apackage.jsonfile in our project's folder:

```
{
"name": "my-todolist",
"version": "0.1.0",
"dependencies": {
"express": "~4.16.3",
"ejs": "~2.6.1",
"cookie-session": "~1.1.0",
"body-parser": "~1.10.1",
"morgan": "~1.9.0"
},
"author": "Enrique <enrique@ortuno.net>",
"description": "A very basic to do list manager"
}
```

Obviously, you can obviously replace the name, the author and the description with whatever you like.

For the Express and EJS version numbers, I based myself on the versions available when this course was written. As Node.js advances very quickly, you will most likely have more recent versions. The tilde (~) allows for future patches on these modules, but not for new minor or major versions, which guarantees that their API won't change and that our code will continue to work even with these updates.

Now that you have yourpackage.json, install the dependencies using a simple:

```
npm install
```

We're ready to start working!

Routes

I said earlier that the definition of your routes was important when constructing a web app. If you're hesitating and don't know where to start, I would suggest you list the routes of your app. What should it do?

- List the tasks.
- Add a task.

- Delete a task.

In theory, we can associate a route to each of these features:

- /todo: list of tasks.
- /todo/add: add a task.
- /todo/delete/:id: delete task n°id.

So you're going to write the routes in your app.js file like this:

```
.get('/todo', function(req, res) {
});
// ...
```

However, it would be a good idea to take a quick look at the form. Generally, forms send data using the POST method and not the GET method. So adding tasks is going to be done on the /todo/add route but with the POST method. That's why instead of calling for.get(), we'll call for.post()for this route:

```
.post('/todo/add/', function(req, res) {
})
```

The right way to chain calls to middleware

We know that we need a session system. The documentation on the cookie-session piece of middleware I mentioned to you earlier tells us how to use the sessions.

Another thing: we would need to retrieve the data from the form in/todo/add. We've learnt how to retrieve the settings from the URL but not from forms. It's actually really easy. You just need to include the middleware body-parser. You'll then have access toreq.body.nameOfField.

Therefore, the start of our JavaScript code should look something like this:

```
var express = require('express');
var session = require('cookie-session'); // Loads the piece of middleware for sessions
var bodyParser = require('body-parser'); // Loads the piece of middleware for managing the settings
var urlencodedParser = bodyParser.urlencoded({ extended: false });
var app = express();
/* Using sessions */
app.use(session({secret: 'todotopsecret'}))
/* Route management below
.... */
```

The secret setting sent to the session module is compulsory: it allows the session cookies to be secured. Send the value of your choice. Note that you can send other options, such as the lifespan of your session cookie (by default, the session will last as long as the browser is open).