Course › Expres... › Imple... › Handli...

Handling Various HTTP requests

Similarly, Express allows you to define other HTTP methods via its `app.{method_name}()` API:

- `app.post()`: POST requests are used for creation of new entities
- `app.put()`: PUT requests are used for complete replacements or partial updates
- `app.patch()`: PATCH requests are used for partial updates
- `app.delete()`: DELETE requests are used for removal of existing entities
- `app.head()`: HEAD requests are idential to GET requests but without the body
- `app.options()`: OPTIONS requests are used to identify allowed methods

Additionaly, the method `app.all()` will define the request handler for any HTTP method. `app.all('*', fn)` is used mostly as a final catch to show 404: Not Found. If you put this first, then nothing will be executed after this route.

Take a look at this example in which a `profile` object is read and changed via four routes with GET, POST, PUT and DELETE methods:

```
const express = require('express')
const app = express()

const profile = {
  username: 'azat',
  email: '[reducted]',
  url: 'http://azat.co'
}
app.get('/profile', (req, res)=>{
  res.send(profile)
})
app.post('/profile', (req, res) => {
  profile = req.body
  res.sendStatus(201)
})
app.put('/profile', (req, res)=>{
  Object.assign(profile, req.body)
  res.sendStatus(204)
})
app.delete('/profile', (req, res)=>{
  profile ={}
  res.sendStatus(204)
})

app.listen(3000)
```

To test this server, simply run it with `node server.js` and then use your favorite client such as CURL or Postman to submit HTTP requests.

The result is that with minimal code you can have all CRUD functionality via a RESTful HTTP interface.