Building RESTful APIs with Express

So, in this section, you learned that:

- REST defines a set of conventions for creating HTTP services:
  - POST: to create a resource
  - PUT: to update it
  - GET: to read it
  - DELETE: to delete it
- Express is a simple, minimalistic and lightweight framework for building web servers.

```
// Build a web server
const express = require('express');
const app = express();

// Creating a course
app.post('/api/courses', (req, res) => {
    // Create the course and return the course object
    resn.send(course);
});

// Getting all the courses
app.get('/api/courses', (req, res) => {
     // To read query string parameters (?sortBy=name)
     const sortBy = req.query.sortBy;

     // Return the courses
```

```
    res.send(courses);
});


// Getting a single course
app.get('/api/courses/:id', (req, res) => {
    const courseId = req.params.id;

    // Lookup the course
    // If not found, return 404
    res.status(404).send('Course not found.');

    // Else, return the course object
    res.send(course);
});

// Updating a course
app.put('/api/courses/:id', (req, res) => {
    // If course not found, return 404, otherwise update it
    // and return the updated object.
});

// Deleting a course
app.delete('/api/courses/:id', (req, res) => {
    // If course not found, return 404, otherwise delete it
    // and return the deleted object.
});

// Listen on port 3000
app.listen(3000, () => console.log('Listening...'));
```

- We use **Nodemon** to watch for changes in files and automatically restart the node process.
- We can use environment variables to store various settings for an application. To read an environment variable, we use **process.env.**

```
// Reading the port from an environment variable
const port = process.env.PORT || 3000;
app.listen(port);
```

- You should never trust data sent by the client. Always validate! Use **Joi** package to perform input validation.