Getting Mean

Practical Perspective

Prof. Dr. Alejandro Zunino Prof. Dr. Alfredo Teyseyre

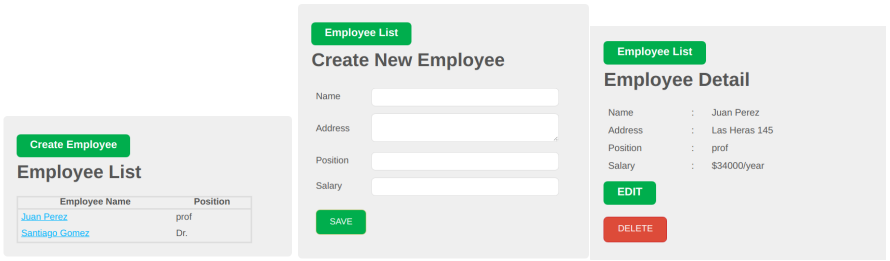
ISISTAN
Department of Computer Science
UNICEN

2018

Outline

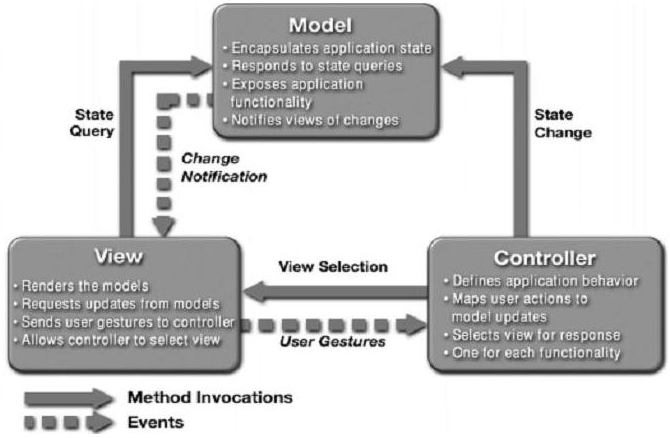
Introducing the example application

Node.js, Express.js and MongoDB CRUD Web Application

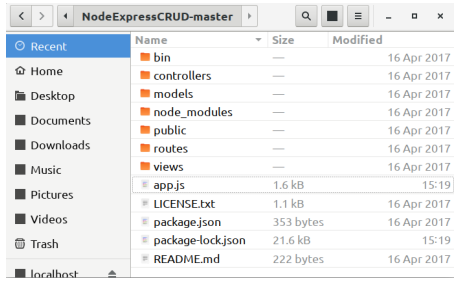


MVC Pattern

Model–view–controller is commonly used for developing software that divides an application into three interconnected parts



Creating Web Application Project



Create project:

```
$ express node-crud --view=ejs
```

Install dependencies:

```
$npm install
npm install mongoose --save
sudo npm install -g nodemon
```

Try it out

```
nodemon
```


Mongoose.js Model

It expresses the application's behavior in terms of the problem domain, independent of the user interface

models/Employee.js:

```
var mongoose = require('mongoose');

var EmployeeSchema = new mongoose.Schema({
  name: String,
  address: String,
  pos: String,
  salary: Number,
  updated_at: { type: Date, default: Date.now }, });

module.exports = mongoose.model('Employee', EmployeeSchema);
```


Controller for CRUD Operations

The controller responds to the user input and performs interactions on the data model objects.

controllers/EmployeeController.js:

```
var mongoose = require("mongoose");
//Add model require.
var Employee = require("../models/Employee");

//Create controller object for CRUD operations.
var employeeController = {};

// Show list of employees employee
Controller.list = function(req, res) {
  Employee.find({}).exec(function (err, employees) {
    if (err) {
    console.log("Error:", err);     }
    else {
    res.render("../views/employees/index",
          {employees: employees});     }   }); };
```


Controller for CRUD Operations

The controller responds to the user input and performs interactions on the data model objects.

controllers/EmployeeController.js:

```
// Show employee by id employee
Controller.show = function(req, res) {
  Employee.findOne({_id: req.params.id}).exec(function (err,
       employee) {
    if (err) { console.log("Error:", err);     }
    else { res.render("../views/employees/show", {employee:
        employee});     }   }); };

// Create new employee employee
Controller.create = function(req, res) {
  res.render("../views/employees/create"); };
```

Controller for CRUD Operations

The controller responds to the user input and performs interactions on the data model objects.

controllers/EmployeeController.js:

```
// Save new employee employee
Controller.save = function(req, res) {
  var employee = new Employee(req.body);
  employee.save(function(err) {
  if(err) {
   console.log(err);
   res.render("../views/employees/create");      }
  else {
   console.log("Successfully created an employee.");
   res.redirect("/employees/show/"+employee._id);      }    })
       ; };
```


Creating Routes

To redirect the request to the controller before call query or display page

routes/employees.js:

```
var express = require('express');
var router = express.Router();

//Add require that point to Employee controller.

var employee = require("../controllers/EmployeeController.js
    ");

// Get all employees
router.get('/', function(req, res) {
  employee.list(req, res); });

// Get single employee by id
router.get('/show/:id', function(req, res) {
  employee.show(req, res); });
```


Creating Routes

To redirect the request to the controller before call query or display page

routes/employees.js:

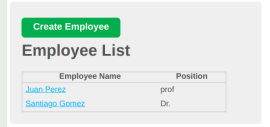
```
// Create employee
router.get('/create', function(req, res) {
  employee.create(req, res); });

// Save employee
router.post('/save', function(req, res) {   employee.save(
    req, res); });

// Edit employee
router.get('/edit/:id', function(req, res) {   employee.edit
    (req, res); });

// Edit update
router.post('/update/:id', function(req, res) {
    employee.update(req, res); });
```


Creating Views

The view means presentation of the model in a particular format.

views/employee/index.ejs:

```
<!DOCTYPE html> <html>
  <head>
    <title>Employee List</title>
    <link rel='stylesheet' href='/stylesheets/
        style.css' />
 </head>
<body>
  <div class="container">
   <h3><a href="/employees/create">Create Employee</a></h3>
   <h1>Employee List</h1>
    <% if(employees.length>0) { %>
    <table>
      <thead>
        <tr>
        <th>Employee Name</th>
        <th>Position</th>
```

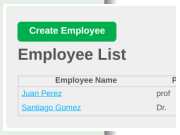


Creating Views

The view means presentation of the model in a particular format.

views/employees/index.ejs:

```
    <tbody>
          <
% for(var i=0; i<employees.length;i++) { %>
      <tr>
        <td><a href="/employees/show/<%= employees[i
            ]._id%>"><%= employees[i].name%></a></td
            >
        <td><%= employees[i].position%></td>
      </tr>           <% } %>
    </tbody>
    </table>
    <% } else { %>
      <div>No employees found.</div>
    <% } %>
  </div>
  </body>
```

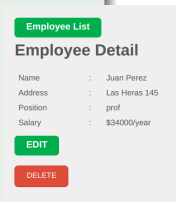


Introduction
MVC
Example

Creating Views

views/employees/show.ejs:

```
    <tr>
     <td>Address</td>
     <td>:</td>
     <td><%= employee.address %></td>
    </tr>
    ....
    <td>Salary</td>
    <td>:</td>
    <td>$<%= employee.salary %>/year</td>
    </tr>
  </tbody>
  </table>
```

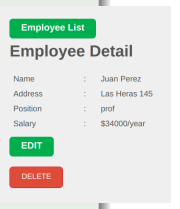


Introduction
MVC
Example

Creating Views

views/employees/show.ejs:

```
<!DOCTYPE html>
<html>
<head>
<title>Employee Detail</title>
<link rel='stylesheet' href='/stylesheets/style.css'
     />
</head>
<body>
  <div class="container">
  <h3><a href="/employees">Employee List</a></h3>
  <h1>Employee Detail</h1>
  <table>
  <tbody>
    <tr>
          <td>Name</td>
          <td>:</td>
      <td><%= employee.name %></td>
    </tr>
```

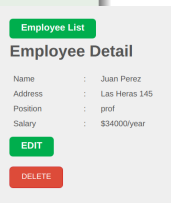


Introduction
MVC
Example

Creating Views

views/employees/show.ejs:

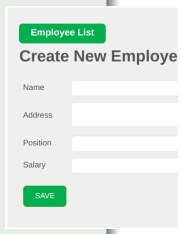
```
<h3>
<a href="/employees/edit/<%= employee._id%>">EDIT</a
    >
</h3>
<form action="/employees/delete/<%= employee._id%>"
    method="post">
  <button type="submit">DELETE</button>
</form>
</div>
</body>
</html>
```



Creating Views

views/employees/create.ejs:

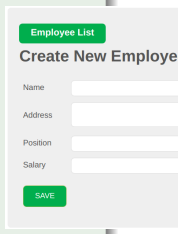
```
<html>
<head>
<title>Create Employee</title>
<link rel='stylesheet' href='/stylesheets/style.css'
     />
</head>
<body>
<div class="container">
<h3><a href="/employees">Employee List</a></h3>
<h1>Create New Employee</h1>
  <form action="/employees/save" method="post">
  <table>
  <tbody>
  <tr>
  <td>Name</td>
  <td><input type="text" name="name" /></td>
  </tr>
  <tr>
  <td>Address</td>
```



Creating Views

views/employees/create.ejs:

```
  <tr>
  <td>Position</td>
  <td><input type="text" name="position" /></td>
  </tr>
  <tr>
  <td>Salary</td>
  <td><input type="number" name="salary" /></td>
  </tr>
  <tr>
  <td colspan="2"><input type="submit" value="Save"
      /></td>
  </tr>
  </tbody>
</table>
</form>
</div>
</body>
</html>
```



For Further Reading I

Didin J.
https://www.djamware.com/post/58b27ce080aca72c54645983/how-to-create-nodejs-expressjs-and-mongodb-crud-web-application