CRUD Operations With ASP.NET Core Using Angular 5 And ADO.NET

Ankit Sharma Updated date May 17 2018 80.8k 36 16

Download Free .NET & JAVA Files API

Try Free File Format APIs for Word/Excel/PDF

Introduction

In this article, I am going to explain how to create an MVC web application in ASP.NET Core 2.0 with Angular 5. We will be creating a sample Employee Record Management system using Angular 5 at frontend, Web API on backend and ADO.NET to fetch data from the database. We will use an Angular form with required field validations for the input fields to get data from the user. We will also learn how to deploy this application on IIS.

We will be using Visual Studio 2017 (Version 15.3.5 or above) and SQL Server 2008.

Prerequisites

- Install .NET Core 2.0.0 or above SDK from here.
- Install Visual Studio 2017 Community Edition (Version 15.3.5 or above) from here.
- Download and install the latest version of Node.js from here.

Source Code

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Now let’s proceed to create our tables and stored procedures using SQL Server. I am using SQL server 2008. You can use any version above that also.

Creating Table and Stored Procedures

We will be using a DB table to store all the records of employees.

Open SQL Server and use the following script to create *tblEmployee* table.

```sql
Create table tblEmployee(
    EmployeeId int IDENTITY(1,1) NOT NULL,
    Name varchar(20) NOT NULL,
    City varchar(20) NOT NULL,
    Department varchar(20) NOT NULL,
    Gender varchar(6) NOT NULL
)
```

Now, we will create stored procedures to add, delete, update, and get employee data.

To insert an Employee Record

```sql
Create procedure spAddEmployee
(
    @Name VARCHAR(20),
    @City VARCHAR(20),
    @Department VARCHAR(20),
    @Gender VARCHAR(6)
)
as
Begin
    Insert into tblEmployee (Name,City,Department, Gender)
    Values (@Name,@City,@Department, @Gender)
End
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```sql
Create procedure spUpdateEmployee
(
   @EmpId INTEGER ,
   @Name VARCHAR(20),
   @City VARCHAR(20),
   @Department VARCHAR(20),
   @Gender VARCHAR(6)
)
as
begin
   Update tblEmployee
   set Name=@Name,
   City=@City,
   Department=@Department,
   Gender=@Gender
   where EmployeeId=@EmpId
End
```

To delete an Employee Record

```sql
Create procedure spDeleteEmployee
(
   @EmpId int
)
as
begin
   Delete from tblEmployee where EmployeeId=@EmpId
End
```

To view all Employee Records

```sql
Create procedure spGetAllEmployees
as
Begin
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    order by EmployeeId
End
```

Now, our Database part has been completed. So, we will proceed to create the MVC application using Visual Studio 2017.

Create MVC Web Application

Open Visual Studio and select File >> New >> Project

After selecting the project, a "New Project" dialog will open. Select .NET Core inside Visual C# menu from the left panel.

Then, select “ASP.NET Core Web Application” from available project types. Put the name of the project as *ASPCoreWithAngular* and press OK. Refer to the image below.

After clicking on OK, a new dialog will open asking to select the project template. You can observe two drop-down menus at the top left of the template window. Select “.NET Core” and “ASP.NET Core 2.0” from these dropdowns. Then, select “Angular” template and press OK.

Now, our project will be created. You can observe the folder structure in solution explorer as shown in below image.

Here we have our *Controllers* and *Views* folders. We won’t be touching the *Views* folders for this tutorial since we will be using Angular to handle the UI. The *Controllers* folders will contain our Web API Controller. The point of interest for us is the ClientApp folder where

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

tutorial, we will delete *fetchdata* and *counter* folders from *ClientApp/app/components.*

Adding the Model to the Application

You can also observe that there is no Models folder in our application. So, we will create one by Right-click on the *solution name* and then Add >> New Folder and name the folder as *Models.*

Right click on *Models* folder and select Add >> Class. Name your class *Employee.cs.* This class will contain our Employee model properties.

Add one more class file to *Models* folder. Name it as *EmployeeDataAccessLayer.cs* . This class will contain our Database related operations.

Now, the Models folder has the following structure.

Open *Employee.cs* and put the following code in it. This is our Employee class having five properties of Employees.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ASPCoreWithAngular.Models
{
    public class Employee
    {
        public int ID { get; set; }

        public string Name { get; set; }
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            public string Department { get; set; }

            public string City { get; set; }
        }
}
```

Open *EmployeeDataAccessLayer.cs* and put the following code to handle database operations. Make sure to put your connection string.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;

namespace ASPCoreWithAngular.Models
{
    public class EmployeeDataAccessLayer
    {
        string connectionString = "Put Your Connection string here";

        //To View all employees details
        public IEnumerable<Employee> GetAllEmployees()
        {
            try
            {
                List<Employee> lstemployee = new List<Employee>();

                using (SqlConnection con = new SqlConnection(connectionString))
                {
                    SqlCommand cmd = new SqlCommand("spGetAllEmployees", con);
                    cmd.CommandType = CommandType.StoredProcedure;

                    con.Open();
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                    while (rdr.Read())
                    {
                        Employee employee = new Employee();

                        employee.ID = Convert.ToInt32(rdr["EmployeeID"]);
                        employee.Name = rdr["Name"].ToString();
                        employee.Gender = rdr["Gender"].ToString();
                        employee.Department = rdr["Department"].ToString();
                        employee.City = rdr["City"].ToString();

                        lstemployee.Add(employee);
                    }
                    con.Close();
                }
                return lstemployee;
            }
            catch
            {
                throw;
            }
        }

        //To Add new employee record
        public int AddEmployee(Employee employee)
        {
            try
            {
                using (SqlConnection con = new SqlConnection(connectionString))
                {
                    SqlCommand cmd = new SqlCommand("spAddEmployee", con);
                    cmd.CommandType = CommandType.StoredProcedure;

                    cmd.Parameters.AddWithValue("@Name", employee.Name);
                    cmd.Parameters.AddWithValue("@Gender", employee.Gender);
                    cmd.Parameters.AddWithValue("@Department", employee.Department);
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();
                }
                return 1;
            }
            catch
            {
                throw;
            }
        }

        //To Update the records of a particluar employee
        public int UpdateEmployee(Employee employee)
        {
            try
            {
                using (SqlConnection con = new SqlConnection(connectionString))
                {
                    SqlCommand cmd = new SqlCommand("spUpdateEmployee", con);
                    cmd.CommandType = CommandType.StoredProcedure;

                    cmd.Parameters.AddWithValue("@EmpId", employee.ID);
                    cmd.Parameters.AddWithValue("@Name", employee.Name);
                    cmd.Parameters.AddWithValue("@Gender", employee.Gender);
                    cmd.Parameters.AddWithValue("@Department", employee.Department);
                    cmd.Parameters.AddWithValue("@City", employee.City);

                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();
                }
                return 1;
            }
            catch
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                }
            }

            //Get the details of a particular employee
            public Employee GetEmployeeData(int id)
            {
                try
                {
                    Employee employee = new Employee();

                    using (SqlConnection con = new SqlConnection(connectionString))
                    {
                        string sqlQuery = "SELECT * FROM tblEmployee WHERE EmployeeID= " + id;
                        SqlCommand cmd = new SqlCommand(sqlQuery, con);

                        con.Open();
                        SqlDataReader rdr = cmd.ExecuteReader();

                        while (rdr.Read())
                        {
                            employee.ID = Convert.ToInt32(rdr["EmployeeID"]);
                            employee.Name = rdr["Name"].ToString();
                            employee.Gender = rdr["Gender"].ToString();
                            employee.Department = rdr["Department"].ToString();
                            employee.City = rdr["City"].ToString();
                        }
                    }
                    return employee;
                }
                catch
                {
                    throw;
                }
            }

```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            {
                try
                {
                    using (SqlConnection con = new SqlConnection(connectionString))
                    {
                        SqlCommand cmd = new SqlCommand("spDeleteEmployee", con);
                        cmd.CommandType = CommandType.StoredProcedure;

                        cmd.Parameters.AddWithValue("@EmpId", id);

                        con.Open();
                        cmd.ExecuteNonQuery();
                        con.Close();
                    }
                    return 1;
                }
                catch
                {
                    throw;
                }
            }
        }
    }
```

Now, we will proceed to create our Web Api Controller.

Adding the Web API Controller to the Application

Right click on Controllers folder and select Add >> New Item

An "Add New Item" dialog box will open. Select *ASP.NET* from the left panel, then select "Web API Controller Class" from templates panel, and put the name as *EmployeeController.cs*. Click Add.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

This will create our Web API *EmployeeController* class. We will put all our business logic in this controller. We will call the methods of *EmployeeDataAccessLayer* to fetch data and pass on the data to Angular frontend.

Open *EmployeeController.cs* file and put the following code into it

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using System.Data.SqlClient;
using System.Data;
using ASPCoreWithAngular.Models;
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
namespace ASPCoreWithAngular.Controllers
{
    public class EmployeeController : Controller
    {
        EmployeeDataAccessLayer objemployee = new EmployeeDataAccessLayer();

        [HttpGet]
        [Route("api/Employee/Index")]
        public IEnumerable<Employee> Index()
        {
            return objemployee.GetAllEmployees();
        }

        [HttpPost]
        [Route("api/Employee/Create")]
        public int Create([FromBody] Employee employee)
        {
            return objemployee.AddEmployee(employee);
        }

        [HttpGet]
        [Route("api/Employee/Details/{id}")]
        public Employee Details(int id)
        {
            return objemployee.GetEmployeeData(id);
        }

        [HttpPut]
        [Route("api/Employee/Edit")]
        public int Edit([FromBody]Employee employee)
        {
            return objemployee.UpdateEmployee(employee);
        }

        [HttpDelete]
        [Route("api/Employee/Delete/{id}")]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                return objemployee.DeleteEmployee(id);
            }
        }
}
```

We are done with our backend logic. So, we will now proceed to code our frontend using Angular 5.

Create the Angular Service

We will create an Angular service which will convert the Web API response to JSON and pass it to our component

Right click on *ClientApp/app* folder and then Add >> New Folder and name the folder as *Services*.

Right click on Sevices folder and select Add >> New Item. An “Add New Item” dialog box will open. Select *Scripts* from the left panel, then select “*TypeScript File*” from templates panel, and put the name as *empservice.service.ts*. Click Add

Open *empservice.service.ts* file and put the following code into it

```
import { Injectable, Inject } from '@angular/core';
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import { Router } from '@angular/router';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';
import 'rxjs/add/observable/throw';

@Injectable()
export class EmployeeService {
    myAppUrl: string = "";

    constructor(private _http: Http, @Inject('BASE_URL') baseUrl: string) {
```

```
    getEmployees() {
        return this._http.get(this.myAppUrl + 'api/Employee/Index')
            .map((response: Response) => response.json())
            .catch(this.errorHandler);
    }

    getEmployeeById(id: number) {
        return this._http.get(this.myAppUrl + "api/Employee/Details/" + id)
            .map((response: Response) => response.json())
            .catch(this.errorHandler)
    }

    saveEmployee(employee) {
        return this._http.post(this.myAppUrl + 'api/Employee/Create', employee)
            .map((response: Response) => response.json())
            .catch(this.errorHandler)
    }

    updateEmployee(employee) {
        return this._http.put(this.myAppUrl + 'api/Employee/Edit', employee)
            .map((response: Response) => response.json())
            .catch(this.errorHandler);
    }

    deleteEmployee(id) {
        return this._http.delete(this.myAppUrl + "api/Employee/Delete/" + id)
            .map((response: Response) => response.json())
            .catch(this.errorHandler);
    }

    errorHandler(error: Response) {
        console.log(error);
        return Observable.throw(error);
    }
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

At this point of time, you might get an error “Parameter 'employee' implicitly has an 'any' type” in *empservice.service.ts* file.

If your encounter this issue then add the following line inside *tsconfig.json* file

"noImplicitAny": false

Refer to below image,

Now we will proceed to create our components.

Creating Angular Components

We will be adding two Angular components to our application

- fetchemployee component - to display all the employee data and delete an existing employee data
- addemployee component - to add a new employee data as well as to edit an existing employee data.

Right click on *ClientApp/app/components* folder and select Add >> New Folder and name the folder as *addemployee*. Right click on *addemployee* folder and select Add >> New Item. An “Add New Item” dialog box will open. Select *Web* from the left panel, then select “*TypeScript File*” from templates panel, and put the name as *Addemployee.component.ts*. Click Add. This will add a typescript file inside *addemployee* folder

Right click on *addemployee* folder and select Add >> New Item. An “Add New Item” dialog box will open. Select *ASP.NET Core* from the left panel, then select “HTML Page” from templates panel, and put the name as *Addemployee.component.html.* Click Add. This will add a HTML file inside *addemployee* folder.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Similarly add *fetchemployee.component.ts* typescript and *fetchemployee.component.html* HTML file inside *fetchemployee* folder.

Now our *ClientApp/app/components* will look like the image below,

Open *fetchemployee.component.ts* file and put the following code into it

```
import { Component, Inject } from '@angular/core';
import { Http, Headers } from '@angular/http';
import { Router, ActivatedRoute } from '@angular/router';
import { EmployeeService } from '../../services/empservice.service'

@Component({
    selector: 'fetchemployee',
    templateUrl: './fetchemployee.component.html'
})

export class FetchEmployeeComponent {
    public empList: EmployeeData[];

    constructor(public http: Http, private _router: Router, private _employeeService: EmployeeServi
        this.getEmployees();
    }

    getEmployees() {
        this._employeeService.getEmployees().subscribe(
            data => this.empList = data
        )
    }

    delete(employeeID) {
        var ans = confirm("Do you want to delete customer with Id: " + employeeID);
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                    this.getEmployees();
                }, error => console.error(error))
            }
        }
}

interface EmployeeData {
    id: number;
    name: string;
    gender: string;
    department: string;
    city: string;
}
```

Let’s understand this code. At the very top we have imported Angular modules and EmployeeService references. After this we have @Component decorator to define the selector and template URL for our component

Inside the FetchEmployeeComponent class we have declared an array variable *empList* of type *EmployeeData* where *EmployeeData* is an interface having the properties same as our Employee Model class. Inside the *getEmployees* method we are calling the getEmployees method of our service EmployeeService, which will return an array of Employees to be stored in empList variable. The *getEmployees* method is called inside the constructor so that the employee data will be displayed as the page loads.

Next, we have delete method which accepts employeeID as parameter. This will prompt the user with a confirmation box and if the user selects yes then it will delete the employee with this employeeID.

Open *fetchemployee.component.html* file and paste the following code to it

```
<h1>Employee Data</h1>

<p>This component demonstrates fetching Employee data from the server.</p>

<p *ngIf="!empList"><em>Loading...</em></p>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    <a [routerLink]="['/register-employee']">Create New</a>
</p>

<table class='table' *ngIf="empList">
    <thead>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Gender</th>
            <th>Department</th>
            <th>City</th>
        </tr>
    </thead>
    <tbody>
        <tr *ngFor="let emp of empList">
            <td>{{ emp.id }}</td>
            <td>{{ emp.name }}</td>
            <td>{{ emp.gender }}</td>
            <td>{{ emp.department }}</td>
            <td>{{ emp.city }}</td>
            <td>
            <td>
                <a [routerLink]="['/employee/edit/', emp.id]">Edit</a> |
                <a [routerLink]="" (click)="delete(emp.id)">Delete</a>
            </td>
        </tr>
    </tbody>
</table>
```

The code for this html file is pretty simple. On the top it has a link to create new employee record and after that it will have a table to display employee data and two link for editing and deleting each employee record.

Now open *Addemployee.component.ts* file and put the following code into it.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
import { NgForm, FormBuilder, FormGroup, Validators, FormControl } from '@angular/forms';
import { Router, ActivatedRoute } from '@angular/router';
import { FetchEmployeeComponent } from '../fetchemployee/fetchemployee.component';
import { EmployeeService } from '../../services/empservice.service';

@Component({
    selector: 'createemployee',
    templateUrl: './AddEmployee.component.html'
})

export class createemployee implements OnInit {
    employeeForm: FormGroup;
    title: string = "Create";
    id: number;
    errorMessage: any;

    constructor(private _fb: FormBuilder, private _avRoute: ActivatedRoute,
        private _employeeService: EmployeeService, private _router: Router) {
        if (this._avRoute.snapshot.params["id"]) {
            this.id = this._avRoute.snapshot.params["id"];
        }

        this.employeeForm = this._fb.group({
            id: 0,
            name: ['', [Validators.required]],
            gender: ['', [Validators.required]],
            department: ['', [Validators.required]],
            city: ['', [Validators.required]]
        })
    }

    ngOnInit() {
        if (this.id > 0) {
            this.title = "Edit";
            this._employeeService.getEmployeeById(this.id)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            }
        }

        save() {

            if (!this.employeeForm.valid) {
                return;
            }

            if (this.title == "Create") {
                this._employeeService.saveEmployee(this.employeeForm.value)
                    .subscribe((data) => {
                        this._router.navigate(['/fetch-employee']);
                    }, error => this.errorMessage = error)
            }
            else if (this.title == "Edit") {
                this._employeeService.updateEmployee(this.employeeForm.value)
                    .subscribe((data) => {
                        this._router.navigate(['/fetch-employee']);
                    }, error => this.errorMessage = error)
            }
        }

        cancel() {
            this._router.navigate(['/fetch-employee']);
        }

        get name() { return this.employeeForm.get('name'); }
        get gender() { return this.employeeForm.get('gender'); }
        get department() { return this.employeeForm.get('department'); }
        get city() { return this.employeeForm.get('city'); }
    }
```

This component will be used for both adding and editing the employee data. Since we are using a form model along with clientside validation to Add and Edit customer data we have imported classes from @angular/forms. The code to create the form has been put

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

This component will handle both Add and Edit request. So how will the system will differentiate between both requests? The answer is routing. We need to define two different route parameters, one for Add employee record and another to edit employee record. We will be defining these in *app.module.shared.ts* file shortly.

We have declared variable *title* to show on the top of page and variable *id* to store the employee id passed as the parameter in case of edit request. To read the employee id from the URL we will use *ActivatedRoute.snapshot* inside the constructor and set the value of variable id.

Inside ngOnInit we will check if the id is set then we will change the title to “Edit”, get the data for that id from our service and populate the fields in our form. The value read from database will be returned as JSON and have all the properties same as we declared in our FormBuilder, hence we use setValue method to populate our form.

The save method will be called on clicking on “Save” button of our form. Based on whether it is an Add operation or an Edit operation it will call the corresponding method from our service and then upon success redirect back to fetch-employee page.

In the last we have also defined getter functions for the control names of our form to enable clientside validation.

Open *Addemployee.component.html file* and put the following code into it,

```
<h1>{{title}}</h1>
<h3>Employee</h3>
<hr />
<form [formGroup]="employeeForm" (ngSubmit)="save()" #formDir="ngForm" novalidate>
    <div class="form-group row">
        <label class=" control-label col-md-12">Name</label>
        <div class="col-md-4">
            <input class="form-control" type="text" formControlName="name" required>
        </div>
        <span class="text-danger" *ngIf="name.invalid && formDir.submitted">
            Name is required.
        </span>
    </div>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    <div class="col-md-4">
      <select class="form-control" data-val="true" formControlName="gender">
        <option value="">-- Select Gender --</option>
        <option value="Male">Male</option>
        <option value="Female">Female</option>
      </select>
    </div>
    <span class="text-danger" *ngIf="gender.invalid && formDir.submitted">
      Gender is required
    </span>
  </div>
  <div class="form-group row">
    <label class="control-label col-md-12" for="Department">Department</label>
    <div class="col-md-4">
      <input class="form-control" type="text" formControlName="department">
    </div>
    <span class="text-danger" *ngIf="department.invalid && formDir.submitted">
      Department is required
    </span>
  </div>
  <div class="form-group row">
    <label class="control-label col-md-12" for="City">City</label>
    <div class="col-md-4">
      <input class="form-control" type="text" formControlName="city">
    </div>
    <span class="text-danger" *ngIf="city.invalid && formDir.submitted">
      City is required
    </span>
  </div>
  <div class="form-group">
    <button type="submit" class="btn btn-default">Save</button>
    <button class="btn" (click)="cancel()">Cancel</button>
  </div>
</form>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Also, every input control has attribute formControlName="xyz", this is used to bind FormControl to HTML. We have also defined error message for client-side validation check and it will be invoked on form submission only.

Defining route and navigation menu for our Application

Open */app/app.module.shared.ts* file and put the following code into it.

```
import { NgModule } from '@angular/core';
import { EmployeeService } from './services/empservice.service'
import { CommonModule } from '@angular/common';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import { AppComponent } from './components/app/app.component';
import { NavMenuComponent } from './components/navmenu/navmenu.component';
import { HomeComponent } from './components/home/home.component';
import { FetchEmployeeComponent } from './components/fetchemployee/fetchemployee.component'
import { createemployee } from './components/addemployee/AddEmployee.component'

@NgModule({
    declarations: [
        AppComponent,
        NavMenuComponent,
        HomeComponent,
        FetchEmployeeComponent,
        createemployee,
    ],
    imports: [
        CommonModule,
        HttpModule,
        FormsModule,
        ReactiveFormsModule,
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            { path: 'home', component: HomeComponent },
            { path: 'fetch-employee', component: FetchEmployeeComponent },
            { path: 'register-employee', component: createemployee },
            { path: 'employee/edit/:id', component: createemployee },
            { path: '**', redirectTo: 'home' }
        ])
    ],
    providers: [EmployeeService]
})
export class AppModuleShared {
}
```

Here we have also imported all our components and defined route for our application as below,

- home which will redirect to home component
- fetch-employee to display all employee data using fetchemployee compoenent
- register-employee to add new employee record using addemployee component
- employee/edit/:id to edit existing employee record using addemployee component
- For any other path it will be redirected to home component

One last thing is to define navigation menu for our application. Open */app/components/navmenu/navmenu.component.html* file and put the following code to it.

```
<div class='main-nav'>
    <div class='navbar navbar-inverse'>
        <div class='navbar-header'>
            <button type='button' class='navbar-toggle' data-toggle='collapse' data-target='.navbar-collapse'>
                <span class='sr-only'>Toggle navigation</span>
                <span class='icon-bar'></span>
                <span class='icon-bar'></span>
                <span class='icon-bar'></span>
            </button>
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        <div class='clearfix'></div>
        <div class='navbar-collapse collapse'>
            <ul class='nav navbar-nav'>
                <li [routerLinkActive]="['link-active']">
                    <a [routerLink]="['/home']">
                        <span class='glyphicon glyphicon-home'></span> Home
                    </a>
                </li>
                <li [routerLinkActive]="['link-active']">
                    <a [routerLink]="['/fetch-employee']">
                        <span class='glyphicon glyphicon-th-list'></span> Fetch employee
                    </a>
                </li>
            </ul>
        </div>
    </div>
</div>
```

And that’s it. We have created our first ASP.NET Core application using Angular 5. Press F5 to launch the application.

A web page will open as shown in the image below. Notice the URL showing route for our home component. And navigation menu on the left showing navigation link for Home and Fetch Employee pages.

Click on *Fetch Employee* in the navigation menu. It will redirect to fetch employee component and displays all the employee data on the page.

Since we have not added any data hence it is empty.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

If we miss the data in any field while creating employee record, we will get a required field validation error message.

After inserting the data in all the fields, click on "Save" button. The new employee record will be created and you will be redirected to the */fetch-employee* page, displaying records of all the employees. Here, we can also see action methods Edit and Delete.

If we want to edit an existing employee record, then click Edit action link. It will open Edit page as below where we can change the employee data. Notice that we have passed employee id in the URL parameter.

Here we have changed the Department of employee Swati from Marketing to HR. Click on "Save" to return to the fetch-employee page to see the updated changes as highlighted in the image below.

If we miss any fields while editing employee records, then Edit view will also throw required field validation error message as shown in the image below.

Now, we will perform Delete operation on an employee named Dhiraj having Employee ID 2. Click on Delete action link which will open a javascript confirmation box asking for a confirmation to delete.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Once we click on Delete button the employee with name Dhiraj will be deleted and we can see the updated list of employees as shown below.

Deploying the application

Open tsconfig.json file and add the following line:

"strictNullChecks": false

Refer to the image below:

When we publish the application , Visual Studio will run “node node_modules/webpack/bin/webpack.js –config webpack.config.vendor.js –env.prod” command. Adding the above mentioned line in tsconfig.json file will ensure that this command will not strictly check for any null values in our application.

Please note that if you are not publishing this application then you do not need to add “strictNullChecks”: false in your tsconfig.json file.

After this follow the same steps mentioned at Deploying a Blazor Application on IIS

Conclusion

We have created an ASP.NET Core application using Angular 5 on the frontend, Web API and ADO.NET in the backend with the help of VS 2017. If you are a fan of Razor and do not want to use Angular frontend, then you can also create this same application using Razor.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

See Also

- CRUD Operation With ASP.NET Core MVC Web App Using ADO.NET
- ASP.NET Core - CRUD Using Angular 5 And Entity Framework Core
- CRUD Operation With ASP.NET Core MVC Using Visual Studio Code And ADO.NET

Next Recommended Article

CRUD Operation With Angular 5 HTTP Client And ASP.NET Core Web API

ADO.NET | Angular 5 | ASP.NET Core | ASP.NET Core CRUD | Web API

Ankit Sharma TOP 500   109   2m   3

C# Corner MVP | Dzone MVB | Author | Speaker | Senior Software Engineer | Passionate Programmer | Medium: https://medium.com/@ankitsharmablog

https://ankitsharmablogs.com/

16   36   View Previous Comments

Type your comment here and press Enter Key (Minimum 10 characters)

NodeInvocationException

sumit ganguly

• 1762 • 130 • 0

Dec 23, 2018

0   0   Reply

Create PDF in your applications with the Pdfcrowd HTML to PDF API   PDFCROWD

•1762 •130 •0 0 0 Reply

How do you do error handling in Angular other than putting it to the console log? I would like a friendly page to notify the user of an error.

First Last Dec 18, 2018

•1491 •401 •2.4k 0 0 Reply

Part 3: Is there supposed to be a node_modules folder for the Angular modules in the project?

First Last Dec 17, 2018

•1491 •401 •2.4k 0 0 Reply

Part 2: I downloaded your Github solution and 1st just tried to build it and I get the similar errors that I am experiencing in Part 1.

First Last Dec 17, 2018

•1491 •401 •2.4k 0 0 Reply

Part 1: I'm using VS 2017 community edition V 15.9.3. I chose NetCore 2.0 at application creation. I did a build and I get: Cannot find module 'C:\Dans\Work 2\Tech\Dot Net\Asp.Net Core\CRUD w Angular5 & ADO\Angular5NetcoreAdo\Angular5NetcoreAdo\node_modules\webpack\bin\webpack.js. Looks like I need webpack. However, the path here is looking for: node_modules. There is no such folder in the structure.

First Last Dec 17, 2018

•1491 •401 •2.4k 0 0 Reply

Please share code in zip format on Ajaygpt904@gmail.com

Ajay Gupta Dec 04, 2018

•1562 •330 •30.2k 0 0 Reply

I am getting this exception, when I run the web app : NodeInvocationException: Prerendering failed because of error: Error: Module parse failed, any inputs on what could be the issue?

Padmaja P Jul 10, 2018

•1890 •2 •0 0 0 Reply

I m trying to set login into that application. I made function passing two parameter name and department in EmployeeDataAccessLayer where i made a query and check if these are equal to that parameter. Now how can I set login component.? please Help

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create the component files similar to other components defined here. Read the value from the Login Form and pass it to you service to authenticate the user. This method will be similar to Create/Edit functionality defined here.

Ankit Sharma Jun 11, 2018

• 109 • 17.8k • 2m 0

while running the app. Error on vendor.js ==> Unhandled exception at line 33893, column 5 in http://localhost:54501/dist/vendor.js?v=RCvRrqPvM2Kc5BlkEQ045FeXR6gPMRIwfn51ludN14I0x800a138f - JavaScript runtime error: Unable to get property 'apply' of undefined or null reference. then i just press continue on debugger, my employee list from database showing but it show error again when u change navigation

andreas tanjaya Jun 07, 2018

• 1888 • 4 • 0 0 0 Reply

About Us Contact Us Privacy Policy Terms Media Kit Sitemap Report a Bug FAQ Partners

C# Tutorials Common Interview Questions Stories Consultants Ideas Certifications

©2020 C# Corner. All contents are copyright of their authors.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD