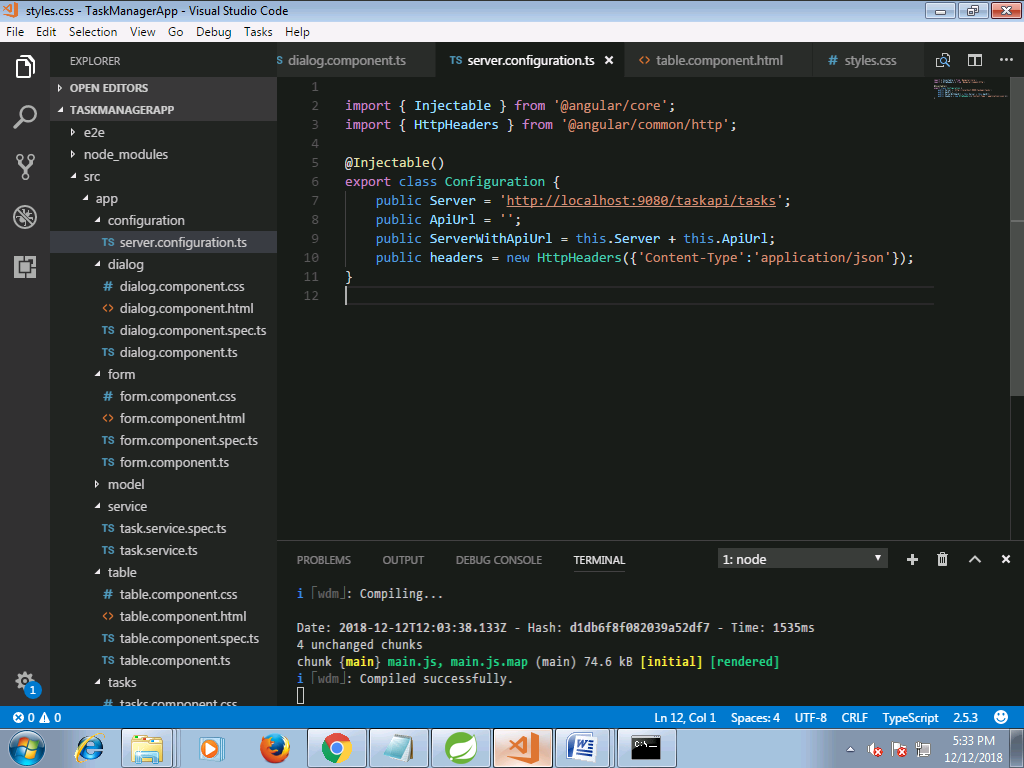
Project Structure



server.configuration.ts

import { Injectable } from '@angular/core';

import { HttpHeaders } from '@angular/common/http';

@Injectable()

export class Configuration {

public Server = 'http://localhost:9000/taskapi/tasks';

public ApiUrl = '';

public ServerWithApiUrl = this.Server + this.ApiUrl;

public headers = new HttpHeaders({'Content-Type':'application/json'});

}

dialog.component.css

/\*\*\*Dialog Box\*\*/

@media (min-width: 768px){

.col-md-4 {

flex: 0 0 100%;

max-width: 100%;

}

}

dialog.component.html

/\*\*\*Dialog Box\*\*/

@media (min-width: 768px){

.col-md-4 {

flex: 0 0 100%;

max-width: 100%;

}

}

form.component.css

.calIcon {

width: 40px;

padding-top: 25px;

cursor: pointer;

}

form.component.html

<div class="p-3 bg-light" [ngClass]="isModalUpdateForm ?'':'mat-elevation-z8 w-50'">

<form [formGroup]="angularForm" >

<div class="row">

<div class="col-xs-12 col-12 col-sm-10 form-group">

<label for="taskValue">Task:

<input class="form-control form-control-sm" formControlName="taskValue">

</label>

</div>

</div>

<div \*ngIf="angularForm.controls['taskValue'].invalid && (angularForm.controls['taskValue'].dirty || angularForm.controls['taskValue'].touched)" class="alert alert-danger col-xs-12 col-12 col-sm-6">

<div \*ngIf="angularForm.controls['taskValue'].errors.required">

Task is required.

</div>

</div>

<div class="row">

<div class="col-xs-12 col-12 col-sm-10 form-group">

<label for="priorityValue">Priority: <span class="display-box">{{angularForm.controls['priorityValue'].value}}</span>

<input type="range" class="form-control form-control-sm" min="0" max="30" formControlName="priorityValue">

</label>

</div>

</div>

<!--<slider-formatting ></slider-formatting>-->

<div class="row">

<div class="col-xs-12 col-12 col-sm-10 form-group">

<label for="parentTaskValue">Parent Task:

<input class="form-control form-control-sm" formControlName="parentTaskValue">

</label>

</div>

</div>

<div class="row">

<div class="col-xs-12 col-12 col-sm-10 form-group">

<label for="startDate">Start Date:

<input type="text" class="form-control" placeholder="YYYY-MM-DD, HH:mm:ss"

#dp1 = "bsDatepicker"

bsDatepicker

[minDate]="startMinDate"

[maxDate]="startMaxDate"

(onHidden)="openEndDatePicker('onShown')"

placement = "top"

[bsConfig]="{ dateInputFormat: 'YYYY-MM-DD, HH:mm:ss' }"

formControlName="startDate"/>

</label>

</div>

<div class="col-xs-12 col-12 col-sm-2 form-group">

<img src="../../assets/images/calendar-icon.svg" class="calIcon" (click)="dp1.toggle()" [attr.aria-expanded]="dp1.isOpen">

</div>

</div>

<div \*ngIf="angularForm.controls['startDate'].invalid && (angularForm.controls['startDate'].dirty || angularForm.controls['startDate'].touched)" class="alert alert-danger col-xs-12 col-12 col-sm-6">

<div \*ngIf="angularForm.controls['startDate'].errors.required">

Start Date is required.

</div>

</div>

<div class="row">

<div class="col-xs-12 col-12 col-sm-10 form-group">

<label for="endDate">End Date:

<input type="text" class="form-control" placeholder="YYYY-MM-DD, HH:mm:ss"

#dp = "bsDatepicker"

bsDatepicker

[minDate]="endMinDate"

[maxDate]="endMaxDate"

placement = "top"

[bsConfig]="{ dateInputFormat: 'YYYY-MM-DD, HH:mm:ss' }"

formControlName="endDate"/>

</label>

</div>

<div class="col-xs-12 col-12 col-sm-2 form-group">

<img src="../../assets/images/calendar-icon.svg" class="calIcon" (click)="dp.toggle()" [attr.aria-expanded]="dp.isOpen">

</div>

</div>

<div \*ngIf="angularForm.controls['endDate'].invalid && (angularForm.controls['endDate'].dirty || angularForm.controls['endDate'].touched)" class="alert alert-danger col-xs-12 col-12 col-sm-6">

<div \*ngIf="angularForm.controls['endDate'].errors.required">

End Date is required.

</div>

</div>

<div class="form-group" \*ngIf="!isModalUpdateForm">

<div class="padRight10 float-left">

<button type="submit" (click) ="onSubmit($event);"

[disabled]="angularForm.pristine || angularForm.invalid || isSubmitted" class="btn btn-primary">

{{submitButtonLabel}}

</button>

</div>

<div>

<button type="submit" (click)="resetClick();" class="btn btn-secondary">

Reset

</button>

</div>

</div>

<div class="modal-footer" \*ngIf="isModalUpdateForm">

<button type="button" class="btn btn-primary" (click) ="onUpdate($event);" [disabled]="valueChnages">UPDATE</button>

<button type="button" class="btn btn-outline-dark" (click)="cancelClick($event)">CANCEL</button>

</div>

</form>

</div>

form.component.ts

import { Component, OnInit, Input,Output,EventEmitter, ViewChild, ElementRef, OnChanges} from "@angular/core";

import { FormControl, FormGroup,FormBuilder,Validators} from '@angular/forms';

import { Task} from '../model/task';

@Component({

selector:'add-task-form',

styleUrls:['./form.component.css'],

templateUrl : './form.component.html'

})

export class FormComponent implements OnInit ,OnChanges{

// startDatepickerModel :Date; //Selected date in start date

startMinDate :Date; //Set Start date min as current date

endMinDate :Date; // Set start date end as after date selected in start date

@Input('taskToUpdate') task :Task;

\_tempTask :Task;

valueChnages :boolean =true;

@Input() submitButtonLabel :string;

//To check if form submitted

@Input('isSubmitted') isSubmitted :boolean = false;

//For Modal Form for Update

@Input('isModalUpdateForm') isModalUpdateForm : boolean;

//Ends

angularForm = new FormGroup ({

taskValue: new FormControl(),

priorityValue : new FormControl(),

parentTaskValue: new FormControl(),

startDate: new FormControl(),

endDate: new FormControl(),

taskId: new FormControl(),

parenttaskId: new FormControl()

});

constructor(private fb: FormBuilder){

this.startMinDate = new Date();

}

ngOnInit(){

this.createForm();

this.submitButtonLabel = "Add Task";

console.log('Input Changes------ >');

if(this.isModalUpdateForm){

this.onChanges();

}

}

onChanges(): void {

this.angularForm.valueChanges.subscribe(val => {

this.valueChnages = false;

});

}

ngOnChanges(){

console.log("Is submitted :"+this.isSubmitted);

}

//This handler will be called when end date date picker will be shown

openEndDatePicker(value:string) :void{

let endateMin :Date = this.angularForm.controls["startDate"].value;

endateMin.setDate(endateMin.getDate() + 7) ;

this.endMinDate = endateMin;

}

private createForm() {

if(this.task != null){

this.\_tempTask = this.task;

this.angularForm = this.fb.group({

taskValue: [this.\_tempTask.taskName,Validators.required ],

parentTaskValue: this.\_tempTask.parentTask != null ? this.task.parentTask.parentTaskName:null,

priorityValue:this.\_tempTask.priority,

startDate:this.\_tempTask.startDate,

endDate:this.\_tempTask.endDate,

taskId:this.\_tempTask.taskId,

parentTaskId:this.\_tempTask.parentTask != null ? this.task.parentTask.parentId:null,

});

}else{

this.angularForm = this.fb.group({

taskValue: ['',Validators.required ],

parentTaskValue: '',

priorityValue:0,

startDate:['',Validators.required],

endDate:['',Validators.required]

});

}

}

//Emit the event in Parent Component

@Output() outPutToParent: EventEmitter<FormGroup> = new EventEmitter();

public onSubmit($event){

this.isSubmitted = true;

this.outPutToParent.emit(this.angularForm);

this.angularForm.disable();

}

public onUpdate($event){

console.log('Update Form Clicked'+$event);

this.isSubmitted = true;

this.outPutToParent.emit(this.angularForm);

this.angularForm.disable();

}

@Output() cancelUpdate: EventEmitter<any> = new EventEmitter();

public cancelClick($event){

this.cancelUpdate.emit($event);

}

public resetClick($event){

console.log('Reset Form Clicked'+$event);

this.angularForm.reset();

this.\_tempTask = this.task;

this.isSubmitted = false;

this.angularForm.enable();

}

}

task.ts

import { ParentTask } from "./parenttask";

export class Task{

taskId :number;

taskName :string;

priority :number;

parentTask :ParentTask;

startDate:string;

endDate:string;

isActive:boolean;

}

parenttask.ts

export class ParentTask{

parentId :number;

parentTaskName :string;

taskId :number;

isActive:boolean;

}

taskvo.ts

export class TaskVo{

id :number;

taskName :string;

priority :number;

parentTask :string;

startDate:Date;

endDate:Date;

isActive:boolean;

}

taskservice.ts

import { Injectable} from '@angular/core';

import {Logger} from '../utillity/logger.service';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import {Configuration} from '../configuration/server.configuration';

import { DomSanitizer } from '@angular/platform-browser';

import {Task} from '../model/task';

@Injectable()

export class TasksService{

private actionUrl: string;

private headers :HttpHeaders;

constructor(private \_logger: Logger,private http: HttpClient, public sanitizer: DomSanitizer, \_configuration: Configuration) {

this.actionUrl = \_configuration.ServerWithApiUrl;

this.headers = \_configuration.headers;

}

// Get all Tasks

getTasks(): Promise<Task[]> {

console.log(this.actionUrl);

return this.http.get(this.actionUrl)

.toPromise()

.then(response => response as Task[])

.catch(this.handleError);

}

getTask(id: number): Promise<Task> {

return this.http.get(`${this.actionUrl}/${id}`)

.toPromise()

.then(response => response as Task)

.catch(this.handleError);

}

createTask(task: Task): Promise<Task> {

this.\_logger.log('Task Name :'+JSON.stringify(task));

return this.http

.post(this.actionUrl, JSON.stringify(task),{headers:this.headers})

.toPromise()

.then(res => res as Task)

.catch(this.handleError);

}

updateTask(task: Task): Promise<Task> {

const url = `${this.actionUrl}/${task.taskId}`;

return this.http

.put(url, JSON.stringify(task), {headers: this.headers})

.toPromise()

.then(() => task)

.catch(this.handleError);

}

deleteTask(id: number): Promise<void> {

const url = `${this.actionUrl}/${id}`;

return this.http.delete(url, {headers: this.headers})

.toPromise()

.then(() => null)

.catch(this.handleError);

}

private handleError(error: any): Promise<any> {

//this.\_logger.error('Error'); // for demo purposes only

return Promise.reject(error.message || error);

}

}

table.component.html

<div class="mat-elevation-z8 p-3 bg-light">

<form class="">

<div class="row col-xs-12 col-12">

<div class="col-6 form-group no-padding-left">

<label>Task :</label>

<input type="text" class="w-77 ml-1" (keyup)="applyFilterByTaskName($event.target.value)">

</div>

<div class="col-6 form-group no-padding-left">

<label>Parent Task :</label>

<input type="text" class="w-77 ml-1" (keyup)="applyFilterByParentTaskName($event.target.value)">

</div>

</div>

<div class="row col-xs-12 col-12 marTop20">

<div class="col-3 form-group no-padding">

<label>Priority From :

<input type="number" class="w-29" min="0" max="30" value="0" [(ngModel)]="priorityFrom" name="priorityFrom" (change) = "applyFilterByPriorityRange($event.target.value)">

</label>

</div>

<div class="col-3 form-group no-padding">

<label>Priority To :

<input type="number" class="w-29" min="0" max="30" value="30" [(ngModel)]="priorityTo" name="priorityTo" (change) = "applyFilterByPriorityRange($event.target.value)">

</label>

</div>

<div class="col-3 form-group no-padding">

<label for="startDate">Start Date: </label>

<input type="text" class="w-48" placeholder="YYYY-MM-DD"

#dp = "bsDatepicker"

bsDatepicker [(ngModel)] = "startDate"

[minDate]="startMinDate"

[maxDate]="startMaxDate"

(onHidden)="openEndDatePicker(startDate)"

[bsConfig]="{ dateInputFormat: 'YYYY-MM-DD' }"

name="startDate"/>

<img src="../../assets/images/calendar-icon.svg" class="calIcon" (click)="dp.toggle()">

</div>

<div class="col-3 form-group no-padding">

<label for="endDate">End Date: </label>

<input type="text" class="w-48" placeholder="YYYY-MM-DD"

#dp1 = "bsDatepicker"

bsDatepicker [(ngModel)] = "endDate"

[minDate]="endMinDate"

[maxDate]="endMaxDate"

(onHidden) ="applyFilterByEndDate(endDate)"

[bsConfig]="{ dateInputFormat: 'YYYY-MM-DD' }"

name="endDate"/>

<img src="../../assets/images/calendar-icon.svg" class="calIcon" (click)="dp1.toggle()">

</div>

</div>

</form>

</div>

<div class="dataContainer">

<table mat-table [dataSource]="dataSource" matSort matSortDisableClear class="mat-elevation-z8">

<!-- Progress Column -->

<ng-container matColumnDef="taskName">

<th mat-header-cell \*matHeaderCellDef mat-sort-header> Task </th>

<td mat-cell \*matCellDef="let row"> {{row.taskName}} </td>

</ng-container>

<!-- Name Column -->

<ng-container matColumnDef="priority">

<th mat-header-cell \*matHeaderCellDef mat-sort-header> Priority </th>

<td mat-cell \*matCellDef="let row"> {{row.priority}} </td>

</ng-container>

<!-- Color Column -->

<ng-container matColumnDef="parentTask">

<th mat-header-cell \*matHeaderCellDef mat-sort-header> Parent Task </th>

<td mat-cell \*matCellDef="let row" > {{row.parentTask}} </td>

</ng-container>

<!-- Start Date Column -->

<ng-container matColumnDef="startDate">

<th mat-header-cell \*matHeaderCellDef mat-sort-header> Start Date </th>

<td mat-cell \*matCellDef="let row" > {{row.startDate | date: 'dd-MM-yyyy'}} </td>

</ng-container>

<!-- End date Column -->

<ng-container matColumnDef="endDate">

<th mat-header-cell \*matHeaderCellDef mat-sort-header> End Date </th>

<td mat-cell \*matCellDef="let row" > {{row.endDate | date: 'dd-MM-yyyy'}} </td>

</ng-container>

<!-- Action Column -->

<ng-container matColumnDef="customColumn">

<th mat-header-cell \*matHeaderCellDef> Actions </th>

<td mat-cell \*matCellDef="let row" >

<button [disabled]="!row.isActive" class="btn btn-success" (click) = "editOrEndTask(row.id,'edit');"> Edit Task</button>

<button [disabled]="!row.isActive" class="btn btn-primary marLeft20" (click)="editOrEndTask(row.id,'end');">End Task</button>

</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

<tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

</table>

<mat-paginator [pageSizeOptions]="[5, 10, 25, 50]"></mat-paginator>

</div>

table.component.css

table {

width: 100%;

}

.mat-form-field {

font-size: 14px;

width: 100%;

}

td, th {

width: 25%;

}

th.mat-sort-header-sorted {

color: black;

}

.calIcon {

width: 30px;

cursor: pointer;

}

.w-77 {

width: 77% !important;

}

.w-29 {

width: 29% !important;

}

.w-48 {

width: 48% !important;

}

.marTop20 {

margin-top: 20px !important;

}

table.component.ts

import {Component, OnInit, ViewChild, Input, Output, EventEmitter} from '@angular/core';

import {MatPaginator, MatSort, MatTableDataSource, MatSortable} from '@angular/material';

import { Task } from '../model/task';

import { TaskVo } from '../model/taskvo';

import { Logger } from '../utillity/logger.service';

import { NgxSpinnerService } from 'ngx-spinner';

import { DatePipe } from '@angular/common';

/\*\*

\* @title Data table with sorting, pagination, and filtering.

\*/

@Component({

selector: 'table-task',

templateUrl: './table.component.html',

styleUrls:['./table.component.css'],

})

export class TableTaskComponent implements OnInit {

displayedColumns: string[] = ['taskName', 'priority', 'parentTask','startDate','endDate','customColumn'];

dataSource: MatTableDataSource<TaskVo>;

public tasks : Array<TaskVo>;

startMinDate :Date; //Set Start date min as current date

endMinDate :Date; // Set start date end as after date selected in start date

startDate :Date;

priorityFrom : number = 0;

priorityTo : number =30;

@ViewChild(MatPaginator) paginator: MatPaginator;

@ViewChild(MatSort) sort: MatSort;

//bind this property from parent injected data List on change of this

// ngOnChanges will be called which is invild feature of Angular 2

@Input('data') data : Task[];

constructor(private datePipe: DatePipe, private spinner: NgxSpinnerService,private \_logger : Logger) {

}

ngOnInit() {

this.sort.sort(<MatSortable>{

id: 'startDate',

start: 'asc'

}

);

}

// Called when "data" is changed in Parent Component

ngOnChanges(){

this.tasks = this.getTasksView(this.data);

this.dataSource = new MatTableDataSource(this.tasks);

this.dataSource.paginator = this.paginator;

this.dataSource.sort = this.sort;

//this.spinner.hide();

}

//This handler will be called when end date date picker will be shown

openEndDatePicker(filterValue: string) :void{

let endateMin :Date = this.startDate;

endateMin.setDate(endateMin.getDate()) ;

this.endMinDate = endateMin;

this.applyFilterByStartDate(filterValue);

}

applyFilterByTaskName(filterValue: string) {

this.dataSource.filterPredicate =

(data: TaskVo, filter: string) => data.taskName.indexOf(filter) != -1;

this.dataSource.filter = filterValue.trim();

if (this.dataSource.paginator) {

this.dataSource.paginator.firstPage();

}

}

applyFilterByParentTaskName(filterValue: string) {

this.dataSource.filterPredicate =

(data: TaskVo, filter: string) => data.parentTask.indexOf(filter) != -1;

this.dataSource.filter = filterValue.trim();

if (this.dataSource.paginator) {

this.dataSource.paginator.firstPage();

}

}

applyFilterByStartDate(filterValue: string) {

console.log(filterValue);

this.dataSource.filterPredicate =

(data: TaskVo, filter: string) :boolean =>{

//console.log("Date Date :"+data.startDate);

let inputValue = this.datePipe.transform(filterValue,'yyyy-MM-dd')

let dataValue = this.datePipe.transform(data.startDate,'yyyy-MM-dd')

return inputValue == dataValue;

}

this.dataSource.filter = filterValue;

if (this.dataSource.paginator) {

this.dataSource.paginator.firstPage();

}

}

applyFilterByEndDate(filterValue: string) {

this.dataSource.filterPredicate =

(data: TaskVo, filter: string) :boolean =>{

//console.log("Date Date :"+data.startDate);

let inputValue = this.datePipe.transform(filterValue,'yyyy-MM-dd')

let dataValue = this.datePipe.transform(data.endDate,'yyyy-MM-dd')

return inputValue == dataValue;

}

this.dataSource.filter = filterValue;

if (this.dataSource.paginator) {

this.dataSource.paginator.firstPage();

}

}

applyFilterByPriorityRange(filterValue: string) {

this.dataSource.filterPredicate =

(data: TaskVo, filter: string) :boolean =>{

//console.log("Date Date :"+data.startDate);

let pFrom = this.priorityFrom;

let pTo = this.priorityTo;

let retValue = data.priority >= pFrom && data.priority <= pTo;

console.log("Return Value :"+retValue);

return retValue;

}

this.dataSource.filter = filterValue;

if (this.dataSource.paginator) {

this.dataSource.paginator.firstPage();

}

}

//Emit the event in Parent Component

@Output() updateTask: EventEmitter<any> = new EventEmitter();

public editOrEndTask(id:number, action:string){

this.updateTask.emit({action:action,id:id});

}

public getTasksView(data: Task[]): Array<TaskVo> {

let taskVoList = new Array<TaskVo>();

let taskVo : TaskVo;

if(data != null){

for (let i = 0; i < data.length; i++){

taskVo = new TaskVo();

taskVo.id = data[i].taskId;

taskVo.taskName = data[i].taskName.trim();

taskVo.priority = data[i].priority;

taskVo.parentTask = data[i].parentTask != null ? data[i].parentTask.parentTaskName.trim():'';

taskVo.startDate = data[i].startDate != null ? new Date(data[i].startDate):null;

taskVo.endDate = data[i].endDate != null ? new Date(data[i].endDate):null;

taskVo.isActive = data[i].isActive;

taskVoList.push(taskVo);

}

}

//this.spinner.hide();

return taskVoList;

}

}

tasks.component.css

.taskContainer{

padding:10px 5%;

}

.display-box{

background-color:antiquewhite;

width: 20px;

height: 20px;

}

task.component.html

<div class="taskContainer">

<!-- <h3>{{app\_heading}}</h3> -->

<div class="taskContent">

<tabset>

<tab heading="View Task">

<table-task [data]="data" (updateTask)="updateTask($event);"></table-task>

</tab>

<tab heading="Add Task">

<div>

<add-task-form [isSubmitted] = "isSubmitted" (outPutToParent)="addUpdateTask($event);"

></add-task-form>

</div>

</tab>

</tabset>

<div>

</div>

</div>

<dialog-form-tag (initiateDialogContent) ="initiateDialogContent($event)" (okClickEmitter) = "okClickCall($event)"

[deleteConfirmBtn] ="deleteConfirmBtn"

[popupHeading]="popupHeading" [popupMessage]="popupMessage"

[updateForm] ="updateForm"

[taskToUpdate] = "task"

(updateTaskEvent) = "updateTaskDetails($event);"></dialog-form-tag>

</div>

<ngx-spinner

bdColor="rgba(51,51,51,0.8)"

size="medium"

color="#fff"

loadingText="Loading..."

type="ball-scale-multiple">

</ngx-spinner>

task.component.spec.ts

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { TasksComponent } from './tasks.component';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { CUSTOM\_ELEMENTS\_SCHEMA, NO\_ERRORS\_SCHEMA } from '@angular/core';

import { NgxSpinnerService } from 'ngx-spinner';

import { Logger } from '../utillity/logger.service';

import { TasksService } from '../service/task.service';

import { HttpClient, HttpHandler } from '@angular/common/http';

import { Configuration } from '../configuration/server.configuration';

import { DatePipe } from '@angular/common';

describe('TasksComponent', () => {

let component: TasksComponent;

let fixture: ComponentFixture<TasksComponent>;

beforeEach(async(() => {

TestBed.configureTestingModule({

declarations: [ TasksComponent ],

schemas:[CUSTOM\_ELEMENTS\_SCHEMA,NO\_ERRORS\_SCHEMA],

imports:[FormsModule,ReactiveFormsModule],

providers:[NgxSpinnerService,Logger,TasksService,HttpClient,HttpHandler,Configuration,DatePipe]

})

.compileComponents();

}));

beforeEach(() => {

fixture = TestBed.createComponent(TasksComponent);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

tasks.component.ts

import { Component, OnInit, ViewChild, AfterViewInit, Input, Output, ElementRef, TemplateRef, ViewChildren, Renderer2} from "@angular/core";

import {TasksService} from "../service/task.service";

import { FormGroup } from "@angular/forms";

import {Logger} from '../utillity/logger.service';

import {TableTaskComponent} from '../table/table.component';

import { NgbModalRef } from '@ng-bootstrap/ng-bootstrap';

import { Task } from "../model/task";

import { ParentTask } from "../model/parenttask";

import { DatePipe } from "@angular/common";

import { NgxSpinnerService } from "ngx-spinner";

import { TabsetComponent } from "ngx-bootstrap";

import { DialogComponent } from "../dialog/dialog.component";

@Component({

moduleId:"task-module",

selector:'app-tasks',

templateUrl : './tasks.component.html',

styleUrls:['./tasks.component.css']

})

export class TasksComponent implements OnInit {

currentUrl :any;

public app\_heading :string;

isSubmitted : boolean;

deleteConfirmBtn :boolean;

updateForm : boolean;

popupHeading :string;

popupMessage :string;

taskId : number;

\_logger :Logger = new Logger();

data :Task[];

@ViewChild(TableTaskComponent) tableComp;

@ViewChild(TabsetComponent) tabSet;

@ViewChild(DialogComponent) dialogContent :DialogComponent;

task :Task;

angularForm :FormGroup;

private modalRef: NgbModalRef;

constructor(private spinner: NgxSpinnerService,

private tasksService:TasksService,

private datePipe: DatePipe){

this.app\_heading ="Task Manager";

this.deleteConfirmBtn = false;

}

ngOnInit(){

this.spinner.show();

this.getTasks();

setTimeout(() => {

/\*\* spinner ends after 5 seconds \*/

this.spinner.hide();

}, 1000);

}

//Edit or Delete Popup Form and Message

public updateTask(passValues?:any){

this.\_logger.log("Action NAme-->"+passValues.action);

if(passValues.action == 'edit'){

this.\_logger.log('Contents :'+this.dialogContent);

this.popupHeading = "Edit Task details:";

this.updateForm = true;

this.tasksService.getTask(passValues.id).then((res) =>{

this.task = res as Task;

this.\_logger.log("Task Fetched :"+this.task.taskName);

this.dialogContent.openFormModal();

})

}

if(passValues.action == 'end'){

this.\_logger.log('Task Ended :');

this.taskId = passValues.id;

this.deleteConfirmBtn = true;

this.updateForm = false;

this.popupHeading = "End Task Confirmation!";

this.popupMessage = "Do you want to mark this task as completed ?"

this.dialogContent.openFormModal();

}

}

private getTasks(){

this.\_logger.log("Calling get All Tasks");

this.spinner.show();

return this.tasksService.getTasks().then(tasks =>

{

this.data = tasks;

this.tableComp.getTasksView();

this.spinner.hide();

},(error) => {

this.\_logger.log(error);

});

}

public addUpdateTask(form : FormGroup){

this.angularForm = form;

this.spinner.show();

this.\_logger.log('Task Added!'+form);

let parentTask :ParentTask;

if(form.controls["parentTaskValue"].value != null && form.controls["parentTaskValue"].value != ''){

parentTask = {

parentId:null,

parentTaskName:form.controls["parentTaskValue"].value,

taskId: null,

isActive:true

};

}

this.task = {

taskId:null,

taskName:form.controls["taskValue"].value,

priority :form.controls["priorityValue"].value,

parentTask: parentTask != null ? parentTask:null,

startDate:form.controls["startDate"].value != null ?this.datePipe.transform(form.controls["startDate"].value,'yyyy-MM-dd HH:mm:ss'):'',

endDate:form.controls["endDate"].value != null ? this.datePipe.transform(form.controls["endDate"].value,'yyyy-MM-dd HH:mm:ss'):'',

isActive:true

};

this.tasksService.createTask(this.task).then((res) => {

this.spinner.hide(); // end loading

this.popupHeading = "Task Added Successfully!";

this.popupMessage = "Task has been added within task list."

this.dialogContent.openFormModal(); //open successpopup

})

}

public updateTaskDetails(form : FormGroup){

this.angularForm = form;

this.spinner.show();

this.\_logger.log('Task Added Successfully!'+form);

let parentTask :ParentTask;

if(form.controls["parentTaskValue"].value != null && form.controls["parentTaskValue"].value != ''){

parentTask = {

parentId:form.controls["parentTaskId"].value != null?form.controls["parentTaskId"].value:null,

parentTaskName:form.controls["parentTaskValue"].value,

taskId: null,

isActive:true

};

}else{

parentTask = null;

}

this.task = {

taskId:form.controls["taskId"].value,

taskName:form.controls["taskValue"].value,

priority :form.controls["priorityValue"].value,

parentTask: parentTask != null ? parentTask:null,

startDate:form.controls["startDate"].value != null ?this.datePipe.transform(form.controls["startDate"].value,'yyyy-MM-dd HH:mm:ss'):'',

endDate:form.controls["endDate"].value != null ? this.datePipe.transform(form.controls["endDate"].value,'yyyy-MM-dd HH:mm:ss'):'',

isActive:true

};

this.\_logger.log(this.task);

this.tasksService.updateTask(this.task).then((res) => {

this.spinner.hide(); // end loading

this.popupHeading = "Task Updated Successfully!";

this.popupMessage = "Task has been updated with in task list."

//this.dialogContent.openFormModal(); //open successpopup

this.okClickCall({action:'OK'});

})

}

initiateDialogContent(content :any){

this.dialogContent = content;

}

//called on Ok button clik on confirmation popup.

okClickCall(actionValues?:any){

this.\_logger.log("Action name :"+actionValues.action);

if(actionValues.action == 'OK'){

this.angularForm.reset(); // Reset Add Form

this.isSubmitted = false; // Check flag for form submission click

this.angularForm.enable(); // Enable form

// this.modalRef.close();

//window.location.reload();

this.tabSet.tabs[0].active = true;

// this.spinner.show();

this.getTasks();

}else if(actionValues.action == 'Yes'){

//this.modalRef.close();

this.deleteConfirmBtn = false;

this.tasksService.deleteTask(this.taskId).then((res) => {

this.\_logger.log("dleted now");

this.getTasks();

})

}

else if(actionValues.action == 'Cancel'){

//this.modalRef.close();

this.deleteConfirmBtn = false;

this.taskId = null;

}else if(actionValues.action == 'Update'){

//this.modalRef.close();

this.deleteConfirmBtn = false;

this.taskId = null;

}

}

}

logger.service.ts

export class Logger {

log(msg: any) { console.log(msg); }

error(msg: any) { console.error(msg); }

warn(msg: any) { console.warn(msg); }

}

app.component.html

<!--The content below is only a placeholder and can be replaced.-->

<div style="text-align:center">

<h1>

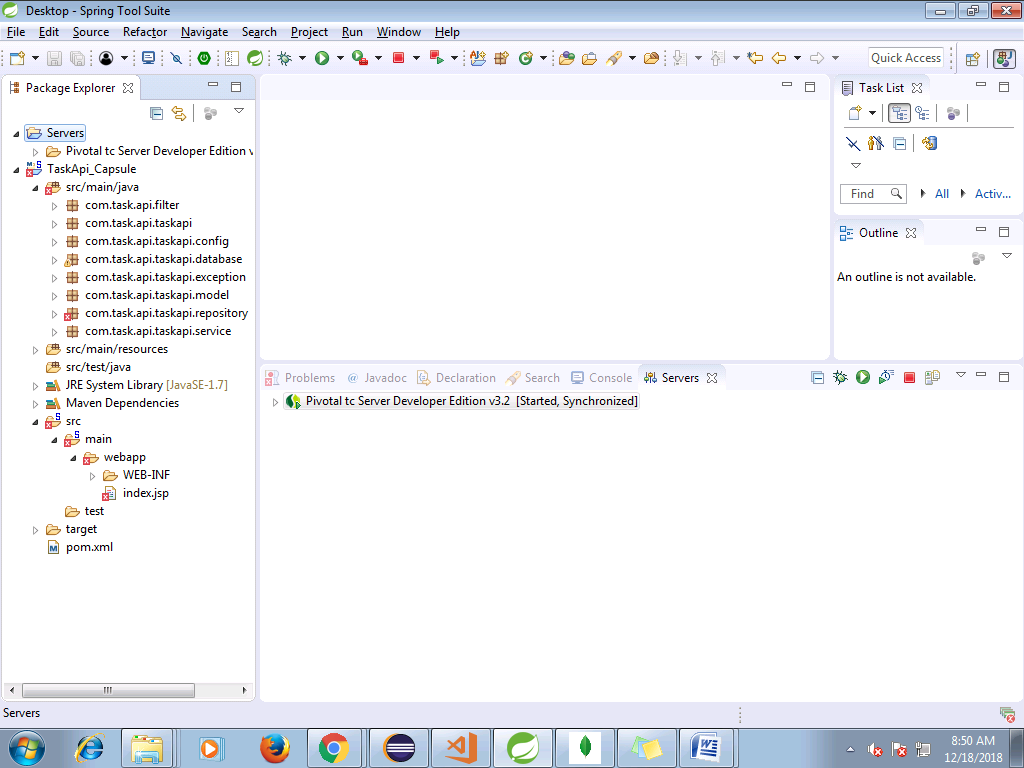
{{ title }}

</h1>

</div>

<app-tasks></app-tasks>

Project Structure - Task API



# CORSHeaderFilter.java

* package com.task.api.filter;
* import java.io.IOException;
* import javax.ws.rs.container.ContainerRequestContext;
* import javax.ws.rs.container.ContainerResponseContext;
* import javax.ws.rs.container.ContainerResponseFilter;
* import javax.ws.rs.ext.Provider;
* @Provider
* public class CORSHeaderFilter implements ContainerResponseFilter{
* @Override
* public void filter(ContainerRequestContext request, ContainerResponseContext response) throws IOException {
* System.out.println("Response Headers :"+response.getHeaders());
* System.out.println("Request Headers :"+request.getHeaders());
* response.getHeaders().add("Access-Control-Allow-Origin", "\*");
* //response.getHeaders().add("Access-Control-Allow-Credentials", "false");
* response.getHeaders().add("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT, OPTIONS, HEAD");
* response.getHeaders().add("Access-Control-Allow-Headers", "Content-Type, Authorization, Accept, X-Requested-With");

* }
* }

# SpringMongoConfig.java

* package com.task.api.taskapi.config;
* import org.springframework.context.annotation.Bean;
* import org.springframework.data.mongodb.MongoDbFactory;
* import org.springframework.data.mongodb.core.MongoTemplate;
* import org.springframework.data.mongodb.core.SimpleMongoDbFactory;
* import com.task.api.taskapi.repository.ITaskRepository;
* import com.task.api.taskapi.repository.TaskRepositoryImpl;
* import com.mongodb.MongoClient;
* public class SpringMongoConfig {
* public @Bean
* MongoDbFactory mongoDbFactory() throws Exception {
* return new SimpleMongoDbFactory(new MongoClient(), "task\_db");
* }
* public @Bean
* MongoTemplate mongoTemplate() throws Exception {
* MongoTemplate mongoTemplate = new MongoTemplate(mongoDbFactory());
* return mongoTemplate;
* }
* public @Bean
* ITaskRepository TaskRepository() throws Exception{
* ITaskRepository taskRepo = new TaskRepositoryImpl();
* return taskRepo;
* }
* }

# StaticApplicationContext.java

* package com.task.api.taskapi.controller;
* import org.springframework.beans.BeansException;
* import org.springframework.context.ApplicationContext;
* import org.springframework.context.ApplicationContextAware;
* import org.springframework.context.annotation.Configuration;
* @Configuration
* public class StaticApplicationContext implements ApplicationContextAware{
* private static ApplicationContext applicationContext = null;
* public void setApplicationContext(ApplicationContext context) throws BeansException {
* applicationContext = context;
* }
* /\*\*
* \* Note that this is a static method which expose ApplicationContext
* \*\*/
* public static ApplicationContext getContext(){
* return applicationContext;
* }
* }

# TaskResouces.java

* package com.task.api.taskapi.controller;
* import java.util.Collection;
* import java.util.Date;
* import javax.ws.rs.Consumes;
* import javax.ws.rs.DELETE;
* import javax.ws.rs.GET;
* import javax.ws.rs.POST;
* import javax.ws.rs.PUT;
* import javax.ws.rs.Path;
* import javax.ws.rs.PathParam;
* import javax.ws.rs.Produces;
* import javax.ws.rs.core.MediaType;
* import org.slf4j.Logger;
* import org.slf4j.LoggerFactory;
* import com.task.api.taskapi.model.Task;
* import com.task.api.taskapi.service.ITaskService;
* import com.task.api.taskapi.service.TaskServiceImpl;
* @Path("/tasks")
* public class TaskResouces {
* private static final ITaskService taskService = (TaskServiceImpl) StaticApplicationContext.getContext().getBean("taskService");
* private static final Logger logger = LoggerFactory.getLogger(TaskResouces.class);
* @GET
* @Path("/{taskId}")
* @Produces(MediaType.APPLICATION\_JSON)
* public Task getTask(@PathParam("taskId") Long id){
* return taskService.getTask(id);
* }
* @GET
* @Produces(MediaType.APPLICATION\_JSON)
* public Collection<Task> getTasks(){
* logger.debug("Fetching all task list--->");
* if(taskService.getTasks().size() == 0)
* return null;
* else
* return taskService.getTasks();
* }

* @POST
* @Consumes(MediaType.APPLICATION\_JSON)
* @Produces(MediaType.APPLICATION\_JSON)
* public Task createTask(Task task) {
* logger.debug("Task Creation Called : "+task.getTaskName());
* task.setCreatedDate(new Date());
* Task rtnTask = taskService.createTask(task);
* return rtnTask;
* }
* @PUT
* @Path("/{taskId}")
* @Consumes(MediaType.APPLICATION\_JSON)
* @Produces(MediaType.APPLICATION\_JSON)
* public Task updateTask(@PathParam("taskId") Long taskId, Task task) {
* task.setModifiedDate(new Date());
* task.setTaskId(taskId);
* Task rtnTask = taskService.updateTask(task);
* return rtnTask;
* }
* @DELETE
* @Path("/{taskId}")
* @Consumes(MediaType.APPLICATION\_JSON)
* @Produces(MediaType.APPLICATION\_JSON)
* public Task removeTask(@PathParam("taskId") Long taskId) {
* logger.debug("Calling delete Task end task....");
* Task retTask = taskService.removeTask(taskId);
* return retTask;
* }
* }

# Database.java

* package com.task.api.taskapi.dao;
* import java.util.Date;
* import org.springframework.context.ApplicationContext;
* import org.springframework.context.annotation.AnnotationConfigApplicationContext;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.Query;
* import com.task.api.taskapi.config.SpringMongoConfig;
* import com.task.api.taskapi.model.ParentTask;
* import com.task.api.taskapi.model.SequenceGenerator;
* import com.task.api.taskapi.model.Task;
* public class Database {
* public static void main(String[] args) {
* ApplicationContext ctx = new AnnotationConfigApplicationContext(SpringMongoConfig.class);
* MongoOperations mongoOperation = (MongoOperations)ctx.getBean("mongoTemplate");
* SequenceGenerator obj = new SequenceGenerator();
* obj.setCounter(1l);
* obj.setId("tbl\_task.taskid");
* // mongoOperation.save(obj);
* obj.setCounter(101l);
* obj.setId("tbl\_parenttask.parentid");
* mongoOperation.save(obj);
* // Query searchUserQuery = new Query(Criteria.where("parentId").is(101l));
* /\* ParentTask obj2 = new ParentTask();
* obj2.setParentId(101l);
* obj2.setParentTaskName("Parenet Task");
* Task task =new Task();
* task.setTaskId(1l);
* task.setTaskName("First Task");
* task.setParentTask(obj2);
* task.setStartDate(new Date());
* task.setEndDate(null);
* task.setCreatedDate(new Date());
* mongoOperation.save(obj2);
* mongoOperation.save(task);
* Query searchUserQuery = new Query(Criteria.where("parentTaskMap.parentId").is(102l));
* // find the saved user again.
* Task savedUser = mongoOperation.findOne(searchUserQuery, Task.class);
* System.out.println("1. find - ParentTaskMap : " + mongoOperation.findAll(SequenceGenerator.class).size());
* System.out.println("1. find - ParentTaskMap : " + savedUser.getTaskName());\*/
* }
* }

# MongoDBSingleton.java

* package com.task.api.taskapi.dao;
* import com.mongodb.DB;
* import com.mongodb.MongoClient;
* public class MongoDBSingleton {
* private static MongoDBSingleton mDbSingleton;
* private static MongoClient mongoClient;
* private static DB db ;

* private static final String dbHost = "localhost";
* private static final int dbPort = 27017;
* private static final String dbName = "taskapidb";
* private MongoDBSingleton(){};
* public static MongoDBSingleton getInstance(){
* if(mDbSingleton == null){
* mDbSingleton = new MongoDBSingleton();
* }
* return mDbSingleton;
* }
* public DB getTestdb(){
* if(mongoClient == null){
* try {
* mongoClient = new MongoClient(dbHost , dbPort);
* } catch (Exception e) {
* return null;
* }
* }
* if(db == null) {
* db = mongoClient.getDB(dbName);
* }
* return db;
* }
* }

# SequenceException.java

* **package** com.task.api.taskapi.exception;
* **public** **class** SequenceException **extends** RuntimeException {
* **private** **static** **final** **long** ***serialVersionUID*** = 1L;
* **private** String errCode;
* **private** String errMsg;

* /\*\*
* \* **@return** the errCode
* \*/
* **public** String getErrCode() {
* **return** errCode;
* }
* /\*\*
* \* **@param** errCode the errCode to set
* \*/
* **public** **void** setErrCode(String errCode) {
* **this**.errCode = errCode;
* }
* /\*\*
* \* **@return** the errMsg
* \*/
* **public** String getErrMsg() {
* **return** errMsg;
* }
* /\*\*
* \* **@param** errMsg the errMsg to set
* \*/
* **public** **void** setErrMsg(String errMsg) {
* **this**.errMsg = errMsg;
* }
* **public** SequenceException(String errMsg) {
* **this**.errMsg = errMsg;
* }
* }

# ParentTask.java

* package com.task.api.taskapi.model;
* import javax.xml.bind.annotation.XmlRootElement;
* import org.springframework.data.annotation.Id;
* import org.springframework.data.mongodb.core.mapping.Document;
* import org.springframework.data.mongodb.core.mapping.Field;
* /\*\*
* \* @author 422892
* \*
* \*/
* @XmlRootElement
* @Document(collection="tbl\_parenttask")
* public class ParentTask {
* @Id
* private Long parentId;
* private Long taskId;
* private String parentTaskName;
* @Field
* private Boolean isActive = true;
* /\*\*
* \* @return the parentId
* \*/
* public Long getParentId() {
* return parentId;
* }
* /\*\*
* \* @param parentId the parentId to set
* \*/
* public void setParentId(Long parentId) {
* this.parentId = parentId;
* }
* /\*\*
* \* @return the taskId
* \*/
* public Long getTaskId() {
* return taskId;
* }
* /\*\*
* \* @param taskId the taskId to set
* \*/
* public void setTaskId(Long taskId) {
* this.taskId = taskId;
* }
* /\*\*
* \* @return the parentTaskName
* \*/
* public String getParentTaskName() {
* return parentTaskName;
* }
* /\*\*
* \* @param parentTaskName the parentTaskName to set
* \*/
* public void setParentTaskName(String parentTaskName) {
* this.parentTaskName = parentTaskName;
* }
* /\*\*
* \* @return the isActive
* \*/
* public Boolean getIsActive() {
* return isActive;
* }
* /\*\*
* \* @param isActive the isActive to set
* \*/
* public void setIsActive(Boolean isActive) {
* this.isActive = isActive;
* }

* }

# SequenceGenerator.java

* package com.task.api.taskapi.model;
* import org.springframework.data.annotation.Id;
* import org.springframework.data.mongodb.core.mapping.Document;
* @Document(collection="tbl\_sequence")
* public class SequenceGenerator {
* @Id
* private String id;
* private Long counter;
* /\*\*
* \* @return the id
* \*/
* public String getId() {
* return id;
* }
* /\*\*
* \* @param id the id to set
* \*/
* public void setId(String id) {
* this.id = id;
* }
* /\*\*
* \* @return the counter
* \*/
* public Long getCounter() {
* return counter;
* }
* /\*\*
* \* @param counter the counter to set
* \*/
* public void setCounter(Long counter) {
* this.counter = counter;
* }


* }

# Task.java

* package com.task.api.taskapi.model;
* import java.util.Date;
* import javax.json.bind.annotation.JsonbDateFormat;
* import javax.xml.bind.annotation.XmlRootElement;
* import org.springframework.data.annotation.Id;
* import org.springframework.data.mongodb.core.mapping.DBRef;
* import org.springframework.data.mongodb.core.mapping.Document;
* import org.springframework.data.mongodb.core.mapping.Field;
* /\*\*
* \* @author 422892
* \*
* \*/
* @XmlRootElement
* @Document(collection="tbl\_task")
* public class Task {
* @Id
* private Long taskId;
* @DBRef
* private ParentTask parentTask;
* private String taskName;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date startDate;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date endDate;
* private int priority;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date createdDate;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date modifiedDate;
* @Field
* private Boolean isActive = true;
* /\*\*
* \* @return the taskId
* \*/
* public Long getTaskId() {
* return taskId;
* }
* /\*\*
* \* @param taskId the taskId to set
* \*/
* public void setTaskId(Long taskId) {
* this.taskId = taskId;
* }
* /\*\*
* \* @return the parentTask
* \*/
* public ParentTask getParentTask() {
* return parentTask;
* }
* /\*\*
* \* @param parentTask the parentTask to set
* \*/
* public void setParentTask(ParentTask parentTask) {
* this.parentTask = parentTask;
* }
* /\*\*
* \* @return the taskName
* \*/
* public String getTaskName() {
* return taskName;
* }
* /\*\*
* \* @param taskName the taskName to set
* \*/
* public void setTaskName(String taskName) {
* this.taskName = taskName;
* }
* /\*\*
* \* @return the startDate
* \*/
* public Date getStartDate() {
* return startDate;
* }
* /\*\*
* \* @param startDate the startDate to set
* \*/
* public void setStartDate(Date startDate) {
* this.startDate = startDate;
* }
* /\*\*
* \* @return the endDate
* \*/
* public Date getEndDate() {
* return endDate;
* }
* /\*\*
* \* @param endDate the endDate to set
* \*/
* public void setEndDate(Date endDate) {
* this.endDate = endDate;
* }
* /\*\*
* \* @return the priority
* \*/
* public int getPriority() {
* return priority;
* }
* /\*\*
* \* @param priority the priority to set
* \*/
* public void setPriority(int priority) {
* this.priority = priority;
* }
* /\*\*
* \* @return the createdDate
* \*/
* public Date getCreatedDate() {
* return createdDate;
* }
* /\*\*
* \* @param createdDate the createdDate to set
* \*/
* public void setCreatedDate(Date createdDate) {
* this.createdDate = createdDate;
* }
* /\*\*
* \* @return the modifiedDate
* \*/
* public Date getModifiedDate() {
* return modifiedDate;
* }
* /\*\*
* \* @param modifiedDate the modifiedDate to set
* \*/
* public void setModifiedDate(Date modifiedDate) {
* this.modifiedDate = modifiedDate;
* }
* /\*\*
* \* @return the isActive
* \*/
* public Boolean getIsActive() {
* return isActive;
* }
* /\*\*
* \* @param isActive the isActive to set
* \*/
* public void setIsActive(Boolean isActive) {
* this.isActive = isActive;
* }
* /\* (non-Javadoc)
* \* @see java.lang.Object#toString()
* \*/
* @Override
* public String toString() {
* return "Task [taskId=" + taskId + ", parentTaskMap=" + parentTask + ", taskName=" + taskName + ", startDate="
* + startDate + ", endDate=" + endDate + ", priority=" + priority + ", createdDate=" + createdDate
* + ", modifiedDate=" + modifiedDate + ", isActive=" + isActive + "]";
* }
* }

# IParentTaskRepository.java

* package com.task.api.taskapi.repository;
* import org.springframework.data.mongodb.repository.MongoRepository;
* import com.task.api.taskapi.model.ParentTask;
* public interface IParentTaskRepository extends MongoRepository<ParentTask, String>{
* }

# ISequenceDao.java

* **package** com.task.api.taskapi.repository;
* **import** com.task.api.taskapi.exception.SequenceException;
* **public** **interface** ISequenceDao {
* **long** getNextSequenceId(String key) **throws** SequenceException;
* }

# ITaskRepository.java

* package com.task.api.taskapi.repository;
* import org.springframework.data.mongodb.repository.MongoRepository;
* import com.task.api.taskapi.model.Task;
* public interface ITaskRepository extends MongoRepository<Task, String>{
* Task findById(Long id);
* }

# ParentTaskRepositoryImpl.java

* package com.task.api.taskapi.repository;
* import java.util.Date;
* import java.util.List;
* import java.util.Optional;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.data.domain.Example;
* import org.springframework.data.domain.Page;
* import org.springframework.data.domain.Pageable;
* import org.springframework.data.domain.Sort;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.Query;
* import org.springframework.data.mongodb.core.query.Update;
* import org.springframework.stereotype.Repository;
* import com.task.api.taskapi.model.ParentTask;
* import com.task.api.taskapi.model.Task;
* @Repository
* public class ParentTaskRepositoryImpl implements IParentTaskRepository{
* @Autowired
* private MongoOperations mongoOperation;
* @Override
* public <S extends ParentTask> List<S> saveAll(Iterable<S> entites) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public List<ParentTask> findAll() {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public List<ParentTask> findAll(Sort sort) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> S insert(S entity) {
* mongoOperation.save(entity);
* return entity;
* }
* @Override
* public <S extends ParentTask> List<S> insert(Iterable<S> entities) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> List<S> findAll(Example<S> example) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> List<S> findAll(Example<S> example, Sort sort) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Page<ParentTask> findAll(Pageable arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public long count() {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public void delete(ParentTask parentTask) {
* mongoOperation.save(parentTask);
* }
* @Override
* public void deleteAll() {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteAll(Iterable<? extends ParentTask> arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteById(String arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public boolean existsById(String arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public Iterable<ParentTask> findAllById(Iterable<String> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Optional<ParentTask> findById(String arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> S save(S parentTask) {
* System.out.println("Task details :-"+parentTask.toString());
* Query query = new Query();
* query.addCriteria(Criteria.where("parentId").is(parentTask.getParentId()));
* Update update = new Update();
* update.set("parentTaskName", parentTask.getParentTaskName());
* ParentTask retTask = mongoOperation.findAndModify(query, update, ParentTask.class);
* return (S) retTask;
* }
* @Override
* public <S extends ParentTask> long count(Example<S> arg0) {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public <S extends ParentTask> boolean exists(Example<S> arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public <S extends ParentTask> Page<S> findAll(Example<S> arg0, Pageable arg1) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> Optional<S> findOne(Example<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* }

# SequenceDaoImpl.java

* package com.task.api.taskapi.repository;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.data.mongodb.core.FindAndModifyOptions;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.Query;
* import org.springframework.data.mongodb.core.query.Update;
* import org.springframework.stereotype.Repository;
* import com.task.api.taskapi.exception.SequenceException;
* import com.task.api.taskapi.model.SequenceGenerator;
* @Repository
* public class SequenceDaoImpl implements ISequenceDao {
* @Autowired
* private MongoOperations mongoOperation;
* @Override
* public long getNextSequenceId(String key) throws SequenceException {
* //get sequence id
* Query query = new Query(Criteria.where("\_id").is(key));
* //increase sequence id by 1
* Update update = new Update();
* update.inc("counter", 1);
* //return new increased id
* FindAndModifyOptions options = new FindAndModifyOptions();
* options.returnNew(true);
* options.upsert(true);// To auto create sequence
* //this is the magic happened.
* SequenceGenerator seqId =
* mongoOperation.findAndModify(query, update, options, SequenceGenerator.class);
* //if no id, throws SequenceException
* //optional, just a way to tell user when the sequence id is failed to generate.
* if (seqId == null) {
* throw new SequenceException("Unable to get sequence id for key : " + key);
* }
* return seqId.getCounter();
* }
* }

# TaskRepositoryImpl.java

* package com.task.api.taskapi.repository;
* import java.util.Date;
* import java.util.List;
* import java.util.Optional;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.data.domain.Example;
* import org.springframework.data.domain.Page;
* import org.springframework.data.domain.Pageable;
* import org.springframework.data.domain.Sort;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.CriteriaDefinition;
* import org.springframework.data.mongodb.core.query.Query;
* import org.springframework.data.mongodb.core.query.Update;
* import org.springframework.stereotype.Repository;
* import com.task.api.taskapi.model.Task;
* @Repository
* public class TaskRepositoryImpl implements ITaskRepository{
* @Autowired
* private MongoOperations mongoOperation;
* @Override
* public List<Task> findAll() {
* return mongoOperation.findAll(Task.class);
* }
* @Override
* public List<Task> findAll(Sort arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> List<S> findAll(Example<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> List<S> findAll(Example<S> arg0, Sort arg1) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> S insert(S task) {
* S savedTask = null;
* try {
* mongoOperation.save(task);
* Query searchUserQuery = new Query(Criteria.where("taskId").is(task.getTaskId()));
* CriteriaDefinition criteriaDefinition = null;
* savedTask = (S)mongoOperation.findOne(searchUserQuery, Task.class);
* }catch(Exception e) {
* System.out.println(e.getClass() +" ---------"+e.getMessage());
* }
* return savedTask;
* }
* @Override
* public <S extends Task> List<S> insert(Iterable<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> List<S> saveAll(Iterable<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Page<Task> findAll(Pageable arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public long count() {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public void delete(Task task) {
* System.out.println("Delete Task :"+task.toString());
* mongoOperation.save(task);
* }
* @Override
* public void deleteAll() {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteAll(Iterable<? extends Task> arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteById(String arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public boolean existsById(String arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public Iterable<Task> findAllById(Iterable<String> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Optional<Task> findById(String arg0) {
* // TODO Auto-generated method stub
* return null;
* }

* /\* Custom Repository method
* \* \*/
* @Override
* public Task findById(Long id) {
* Task task = null;
* try {
* Query searchUserQuery = new Query(Criteria.where("taskId").is(id));
* task = mongoOperation.findOne(searchUserQuery, Task.class);
* }catch(Exception e) {
* System.out.println("Exception in Find by Id");
* }
* return task;
* }
* @Override
* public <S extends Task> S save(S task) {
* System.out.println("Task details :-"+task.toString());
* Query query = new Query();
* query.addCriteria(Criteria.where("taskId").is(task.getTaskId()));
* Update update = new Update();
* update.set("taskName", task.getTaskName());
* update.set("priority", task.getPriority());
* update.set("modifiedDate", new Date());
* update.set("startDate", task.getStartDate());
* update.set("endDate", task.getEndDate());
* if(task.getParentTask() == null)
* update.set("parentTask", task.getParentTask());
* Task retTask = mongoOperation.findAndModify(query, update, Task.class);
* return (S) retTask;
* }
* @Override
* public <S extends Task> long count(Example<S> arg0) {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public <S extends Task> boolean exists(Example<S> arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public <S extends Task> Page<S> findAll(Example<S> arg0, Pageable arg1) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> Optional<S> findOne(Example<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* }

# ITaskService.java

* package com.task.api.taskapi.service;
* import java.util.Collection;
* import com.task.api.taskapi.model.Task;
* public interface ITaskService {
* Task createTask(Task task);
* Task getTask(Long id);
* Task updateTask(Task task);
* Task removeTask(Long taskId);
* Collection<Task> getTasks();
* }

# TaskServiceImpl.java

* package com.task.api.taskapi.service;
* import java.util.Collection;
* import java.util.Date;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Service;
* import com.task.api.taskapi.model.ParentTask;
* import com.task.api.taskapi.model.Task;
* import com.task.api.taskapi.repository.IParentTaskRepository;
* import com.task.api.taskapi.repository.ISequenceDao;
* import com.task.api.taskapi.repository.ITaskRepository;
* @Service
* public class TaskServiceImpl implements ITaskService{
* private static final String TASK\_SEQ\_KEY = "tbl\_task.taskid";
* private static final String PARENTTASK\_SEQ\_KEY = "tbl\_parenttask.parentid";
* @Autowired
* private ISequenceDao sequenceDao;
* @Autowired
* private ITaskRepository taskDao;
* @Autowired
* private IParentTaskRepository parentTaskRepo;

* @Override
* public Task createTask(Task task) {
* Task retTask = null;
* task.setTaskId(sequenceDao.getNextSequenceId(TASK\_SEQ\_KEY));
* if(task.getParentTask() != null && task.getParentTask().getParentId() == null
* && task.getParentTask().getParentTaskName() != null
* && !task.getParentTask().getParentTaskName().isEmpty()) {
* ParentTask parentTask = new ParentTask();
* parentTask.setParentId(sequenceDao.getNextSequenceId(PARENTTASK\_SEQ\_KEY));
* parentTask.setParentTaskName(task.getParentTask().getParentTaskName());
* parentTaskRepo.insert(parentTask);
* task.setParentTask(parentTask);
* }
* System.out.println("Task :"+task.toString());
* retTask = taskDao.insert(task);
* return retTask;
* }
* @Override
* public Task getTask(Long taskId) {
* Task task = taskDao.findById(taskId);
* return task;
* }
* @Override
* public Task updateTask(Task task) {
* if(task.getParentTask() != null && task.getParentTask().getParentId() != null && task.getParentTask().getParentId().longValue() != 0) {
* parentTaskRepo.save(task.getParentTask());
* }else if(task.getParentTask() != null && task.getParentTask().getParentId() != null && task.getParentTask().getParentId().longValue() == 0) {
* ParentTask parentTask = new ParentTask();
* parentTask.setParentId(sequenceDao.getNextSequenceId(PARENTTASK\_SEQ\_KEY));
* parentTask.setParentTaskName(task.getParentTask().getParentTaskName());
* parentTaskRepo.insert(parentTask);
* task.setParentTask(parentTask);
* }
* Task retTask = taskDao.save(task);
* return retTask;
* }
* @Override
* public Task removeTask(Long taskId) {
* Task task = getTask(taskId);
* if(task != null && task.getParentTask() != null) {
* ParentTask pTask = task.getParentTask();
* pTask.setIsActive(false);
* parentTaskRepo.delete(pTask);
* }
* task.setEndDate(new Date());
* task.setIsActive(false);
* taskDao.delete(task);
* return task;
* }
* @Override
* public Collection<Task> getTasks() {
* return taskDao.findAll();
* }


* }

# mongoDbConfig.properties

* #Local MongoDB config
* #spring.data.mongodb.authentication-database=admin
* #spring.data.mongodb.username=root
* #spring.data.mongodb.password=root
* spring.data.mongodb.database=task\_db
* spring.data.mongodb.port=27017
* spring.data.mongodb.host=localhost

# application-context.xml

* <beans xmlns=*"http://www.springframework.org/schema/beans"*
* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
* xmlns:context=*"http://www.springframework.org/schema/context"*
* xmlns:mongo=*"http://www.springframework.org/schema/data/mongo"*
* xsi:schemaLocation=*"http://www.springframework.org/schema/context*
* *http://www.springframework.org/schema/context/spring-context.xsd*
* *http://www.springframework.org/schema/data/mongo*
* *http://www.springframework.org/schema/data/mongo/spring-mongo-2.0.xsd*
* *http://www.springframework.org/schema/beans*
* *http://www.springframework.org/schema/beans/spring-beans.xsd"*>
* <context:component-scan base-package=*"com.task.api"*/>
* <context:property-placeholder location =*"classpath:config/mongoDbConfig.properties"* />
* <mongo:mongo-client host=*"${spring.data.mongodb.host}"* port=*"${spring.data.mongodb.port}"*></mongo:mongo-client>
* <mongo:db-factory dbname=*"${spring.data.mongodb.database}"*/>
* <bean id=*"mongoOperation"*
* class=*"org.springframework.data.mongodb.core.MongoTemplate"*>
* <constructor-arg name=*"mongoDbFactory"*
* ref=*"mongoDbFactory"* />
* </bean>
* <bean id=*"taskService"* class=*"com.task.api.taskapi.service.TaskServiceImpl"*/>
* </beans>

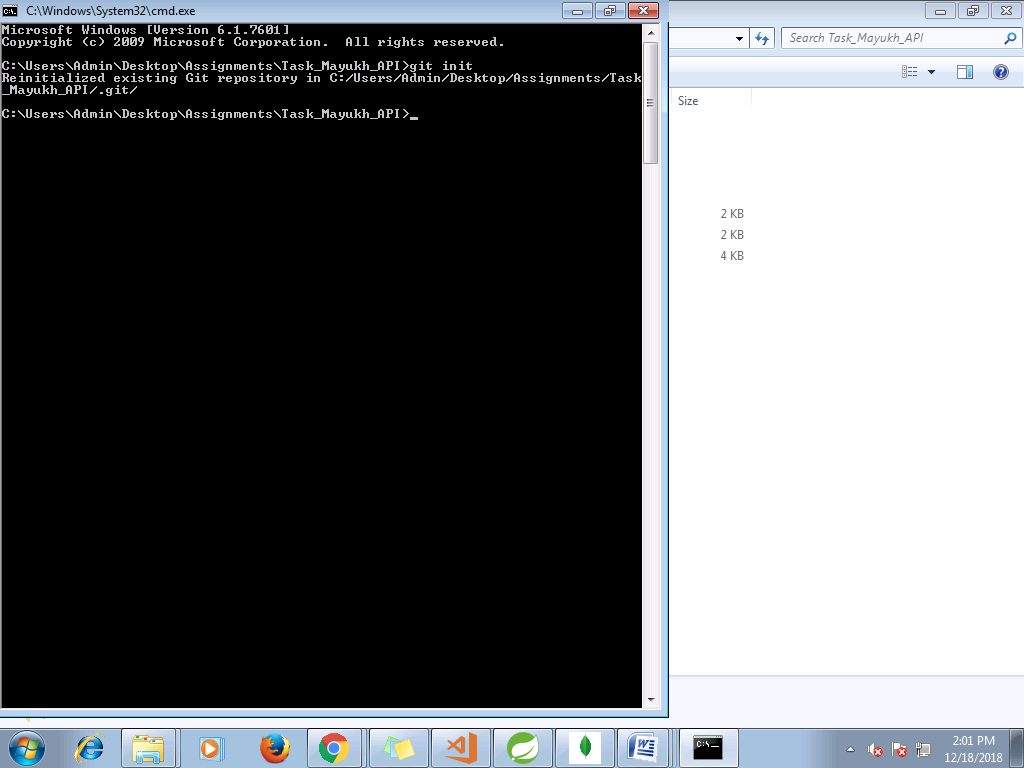
# web.xml

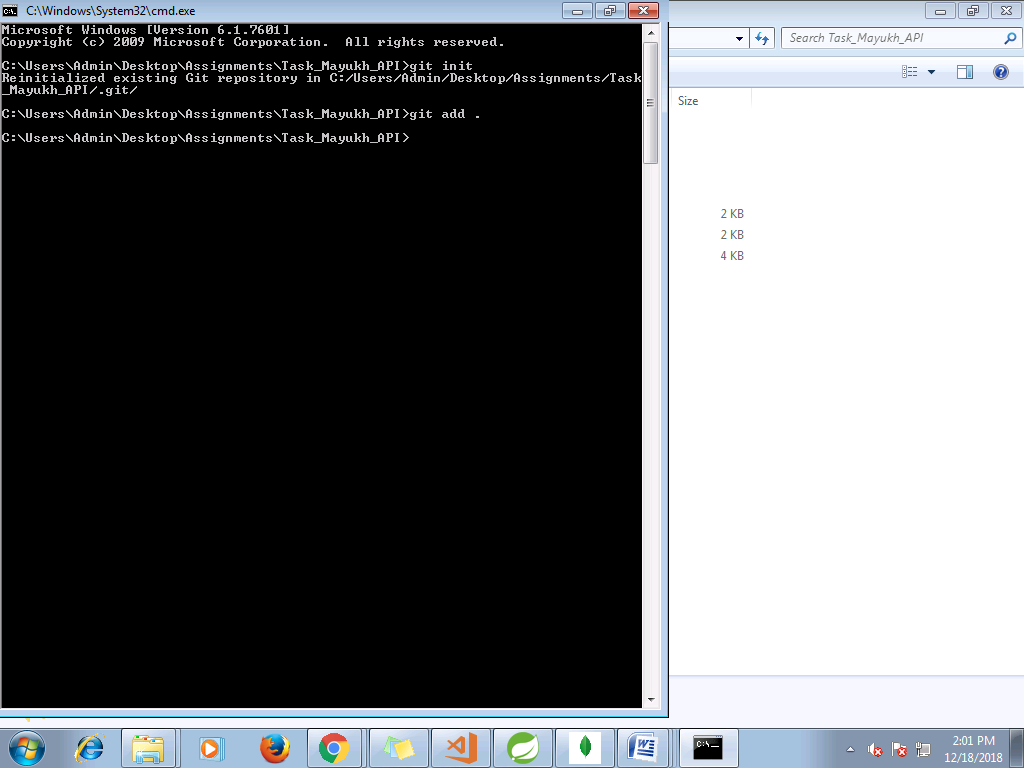
* <?xml version=*"1.0"* encoding=*"UTF-8"*?>
* <!-- This web.xml file is not required when using Servlet 3.0 container,
* see implementation details http://jersey.java.net/nonav/documentation/latest/jax-rs.html -->
* <web-app version=*"2.5"* xmlns=*"http://java.sun.com/xml/ns/javaee"*
* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>
* <context-param>
* <param-name>contextConfigLocation</param-name>
* <param-value>classpath:application-context.xml</param-value>
* </context-param>
* <listener>
* <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
* </listener>
* <servlet>
* <servlet-name>Jersey Web Application</servlet-name>
* <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
* <init-param>
* <param-name>jersey.config.server.provider.packages</param-name>
* <param-value>com.task.api</param-value>
* </init-param>
* <init-param>
* <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-name>
* <param-value>true</param-value>
* </init-param>
* <init-param>
* <param-name>com.sun.jersey.config.property.packages</param-name>
* <param-value>
* com.your.packages,
* org.codehaus.jackson.jaxrs
* </param-value>
* </init-param>
* <load-on-startup>1</load-on-startup>
* </servlet>
* <servlet-mapping>
* <servlet-name>Jersey Web Application</servlet-name>
* <url-pattern>/\*</url-pattern>
* </servlet-mapping>
* </web-app>

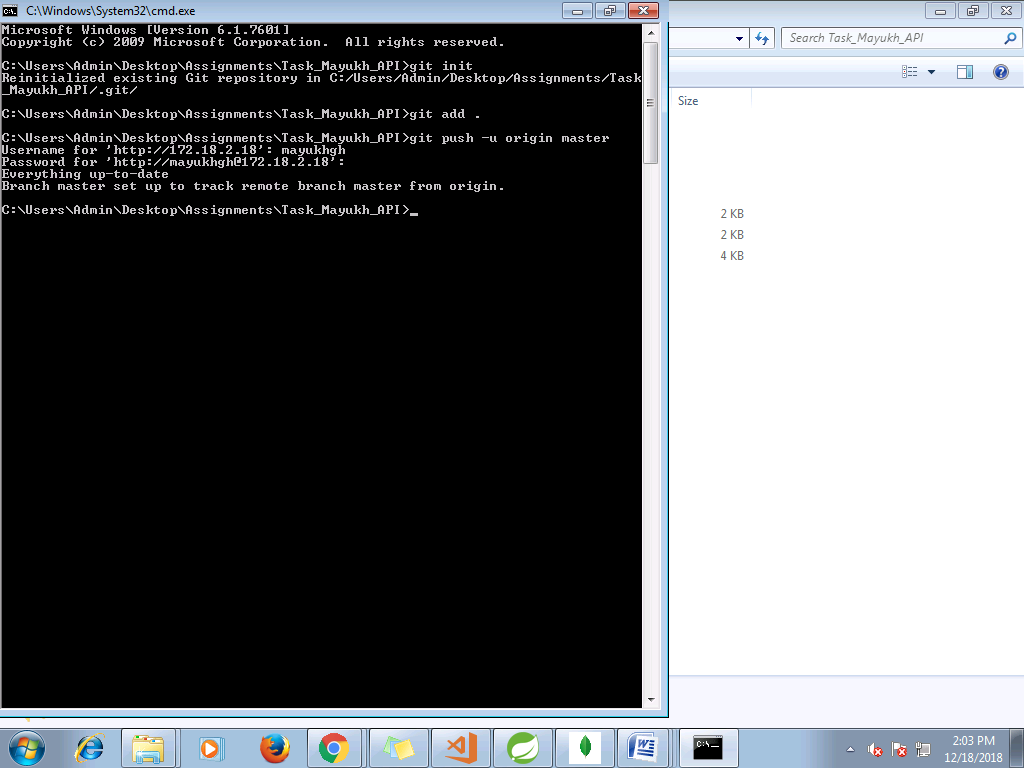
# pom.xml

* <project xmlns=*"http://maven.apache.org/POM/4.0.0"*
* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>
* <modelVersion>4.0.0</modelVersion>
* <groupId>com.task.api</groupId>
* <artifactId>taskapi</artifactId>
* <packaging>war</packaging>
* <version>0.0.1-SNAPSHOT</version>
* <name>taskapi</name>
* <build>
* <finalName>taskapi</finalName>
* <plugins>
* <plugin>
* <groupId>org.apache.maven.plugins</groupId>
* <artifactId>maven-compiler-plugin</artifactId>
* <version>2.5.1</version>
* <inherited>true</inherited>
* <configuration>
* <source>1.7</source>
* <target>1.7</target>
* </configuration>
* </plugin>
* </plugins>
* </build>
* <dependencyManagement>
* <dependencies>
* <dependency>
* <groupId>org.glassfish.jersey</groupId>
* <artifactId>jersey-bom</artifactId>
* <version>${jersey.version}</version>
* <type>pom</type>
* <scope>import</scope>
* </dependency>
* </dependencies>
* </dependencyManagement>
* <dependencies>
* <dependency>
* <groupId>org.glassfish.jersey.containers</groupId>
* <artifactId>jersey-container-servlet-core</artifactId>
* <!-- use the following artifactId if you don't need servlet 2.x compatibility -->
* <!-- artifactId>jersey-container-servlet</artifactId -->
* </dependency>
* <dependency>
* <groupId>org.glassfish.jersey.inject</groupId>
* <artifactId>jersey-hk2</artifactId>
* </dependency>
* <!-- uncomment this to get JSON support -->
* <dependency>
* <groupId>org.glassfish.jersey.media</groupId>
* <artifactId>jersey-media-json-binding</artifactId>
* </dependency>
* <dependency>
* <groupId>org.glassfish.jersey.media</groupId>
* <artifactId>jersey-media-json-jackson</artifactId>
* </dependency>
* <!-- https://mvnrepository.com/artifact/org.codehaus.jackson/jackson-mapper-asl -->
* <dependency>
* <groupId>org.codehaus.jackson</groupId>
* <artifactId>jackson-mapper-asl</artifactId>
* <version>1.9.13</version>
* </dependency>
* <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
* <!-- <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId>
* <version>2.9.6</version> </dependency> -->
* <!-- Spring framework -->
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-core</artifactId>
* <version>5.0.7.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-context</artifactId>
* <version>5.0.7.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-webmvc</artifactId>
* <version>5.0.3.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-web</artifactId>
* <version>5.0.3.RELEASE</version>
* </dependency>
* <!-- https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver -->
* <dependency>
* <groupId>org.mongodb</groupId>
* <artifactId>mongo-java-driver</artifactId>
* <version>3.8.0</version>
* </dependency>
* <dependency>
* <groupId>org.springframework.data</groupId>
* <artifactId>spring-data-mongodb</artifactId>
* <version>2.0.8.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>cglib</groupId>
* <artifactId>cglib</artifactId>
* <version>2.2.2</version>
* </dependency>
* </dependencies>
* <properties>
* <jersey.version>2.27</jersey.version>
* <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
* </properties>
* </project>

**Git Commit**

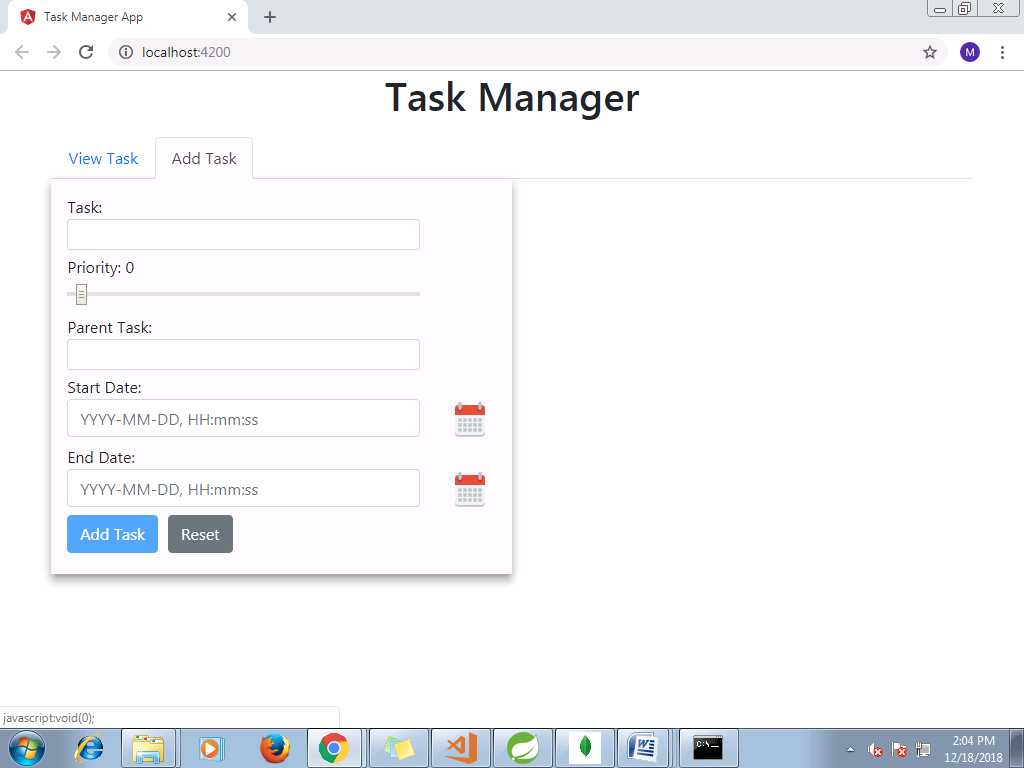




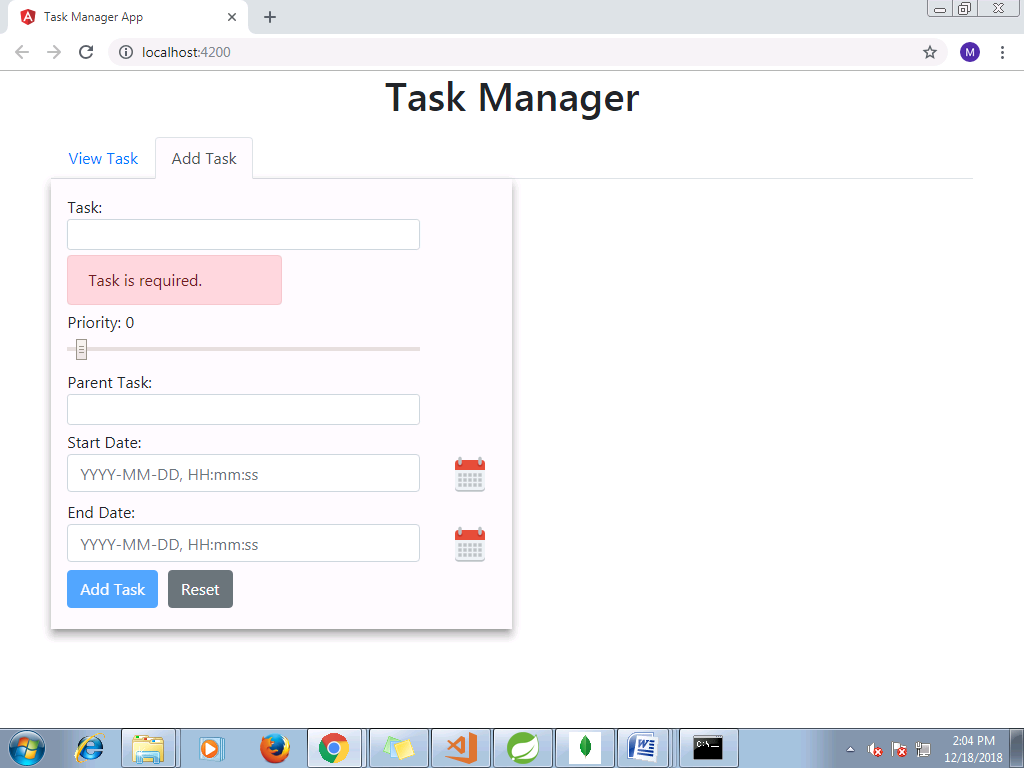


**SCREENSHOTS**

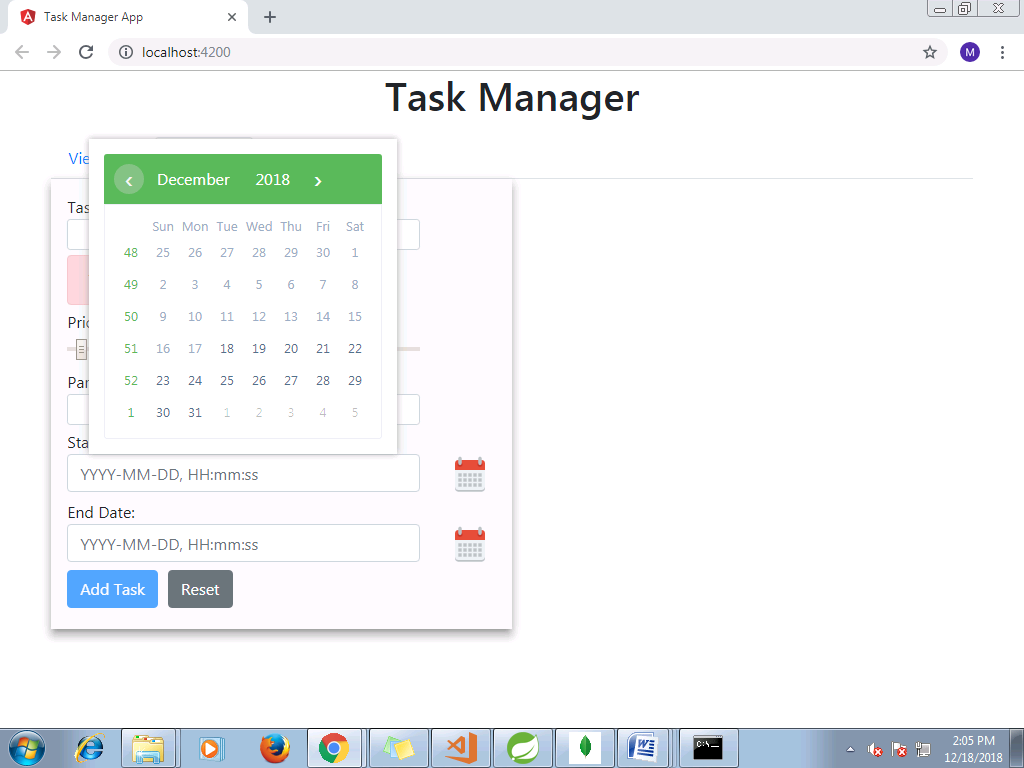
**Add Task Screen**

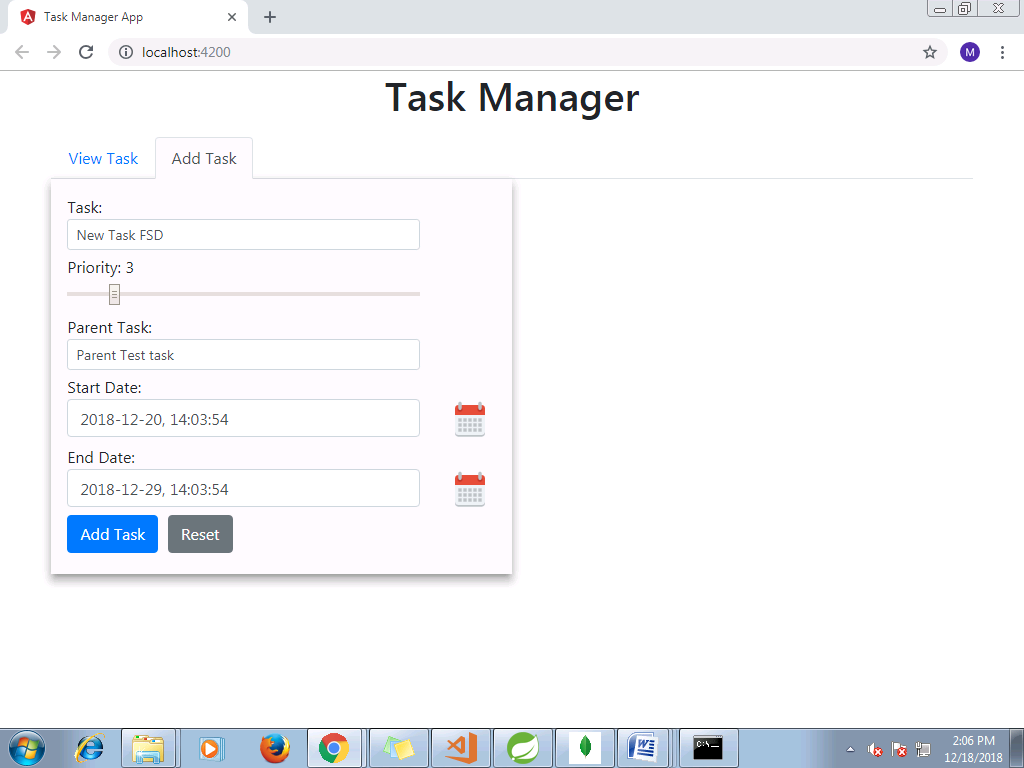


Validation

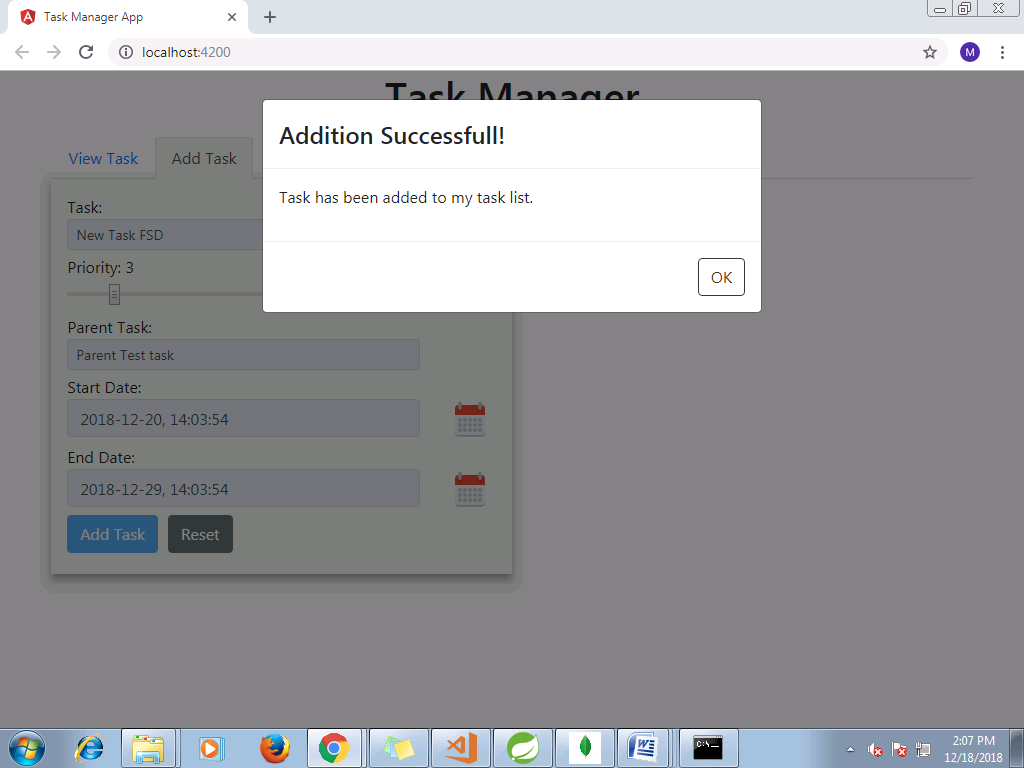


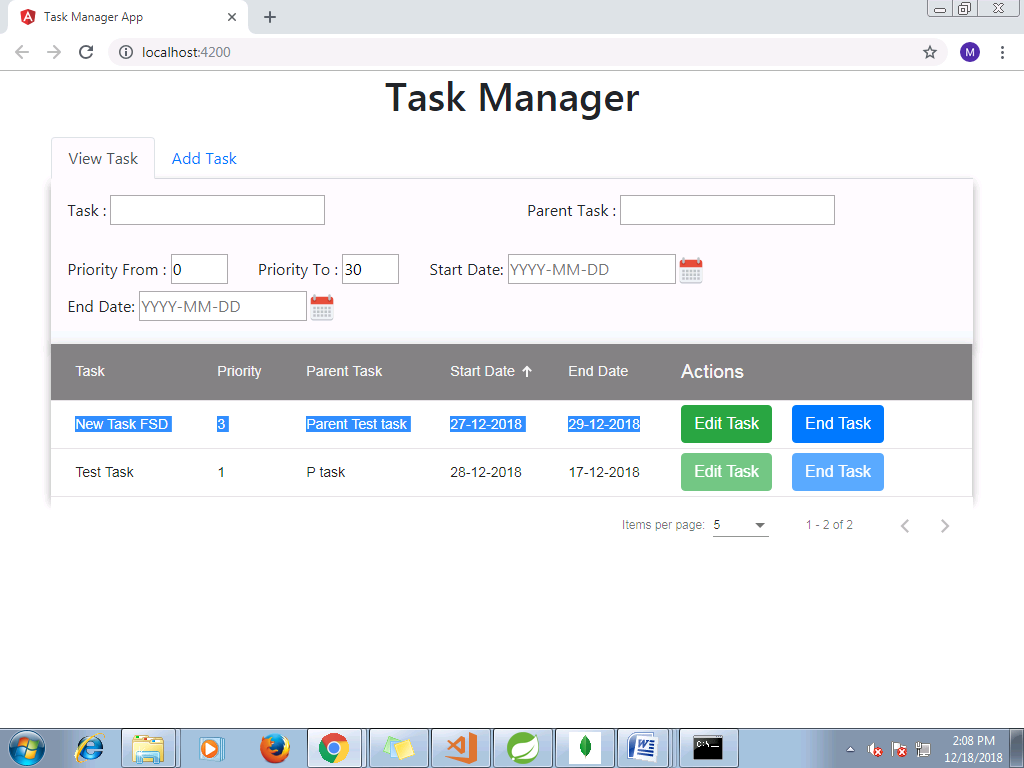
Calendar Pop up



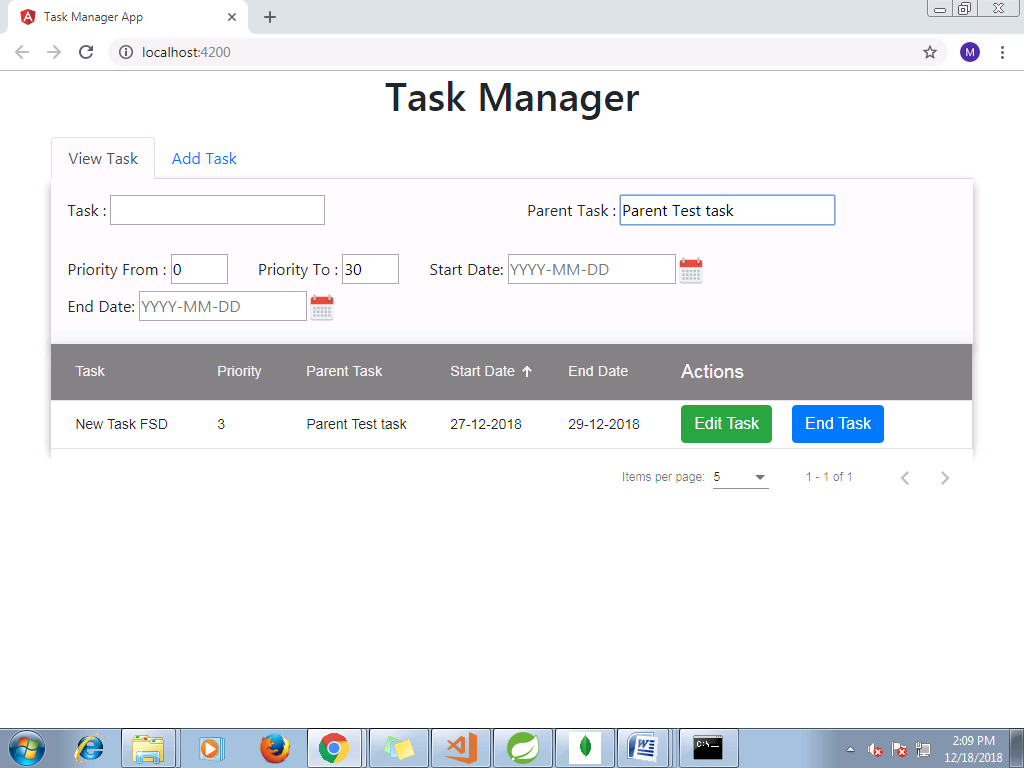


Task added successfully pop up

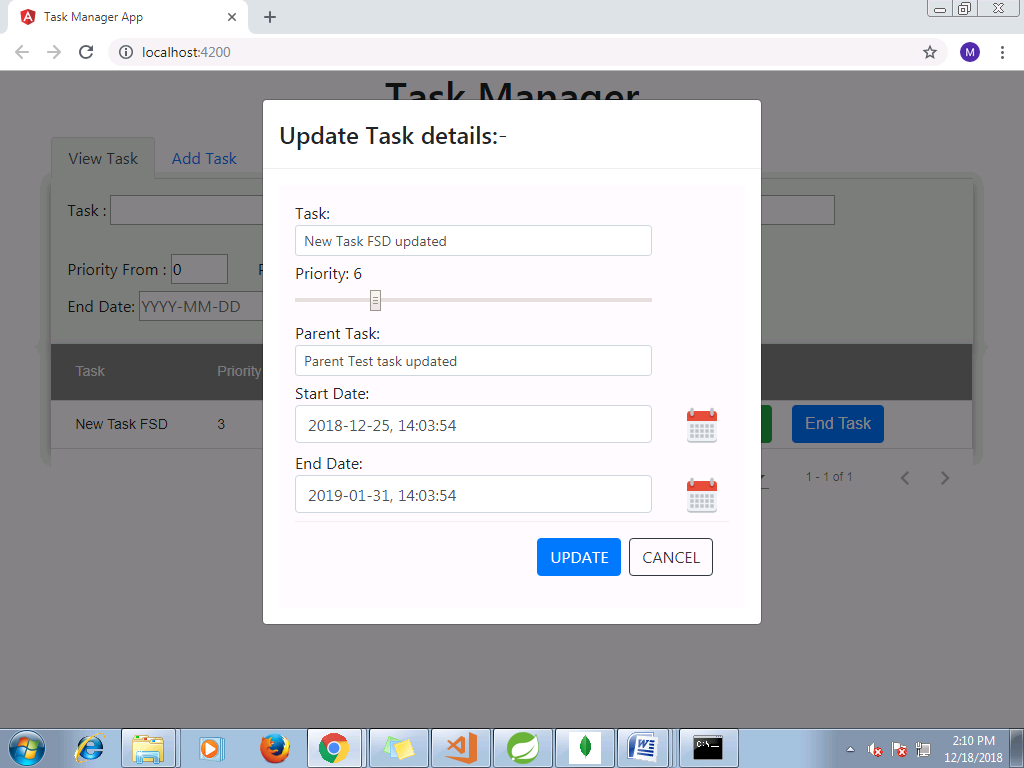




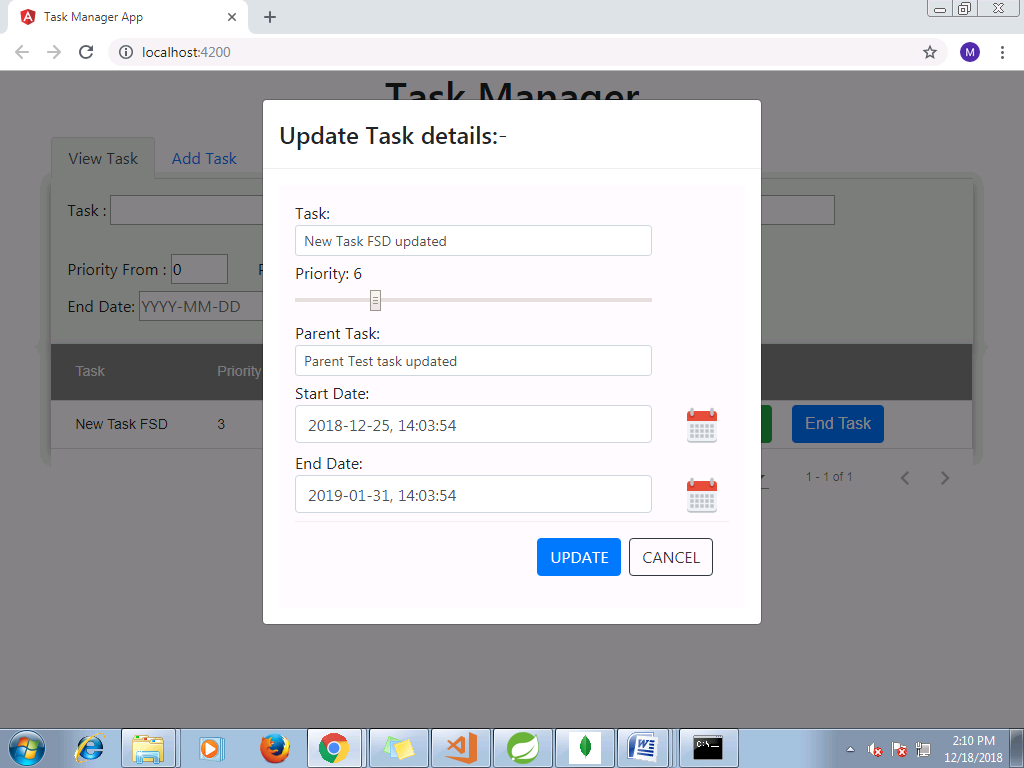
View Task Filter

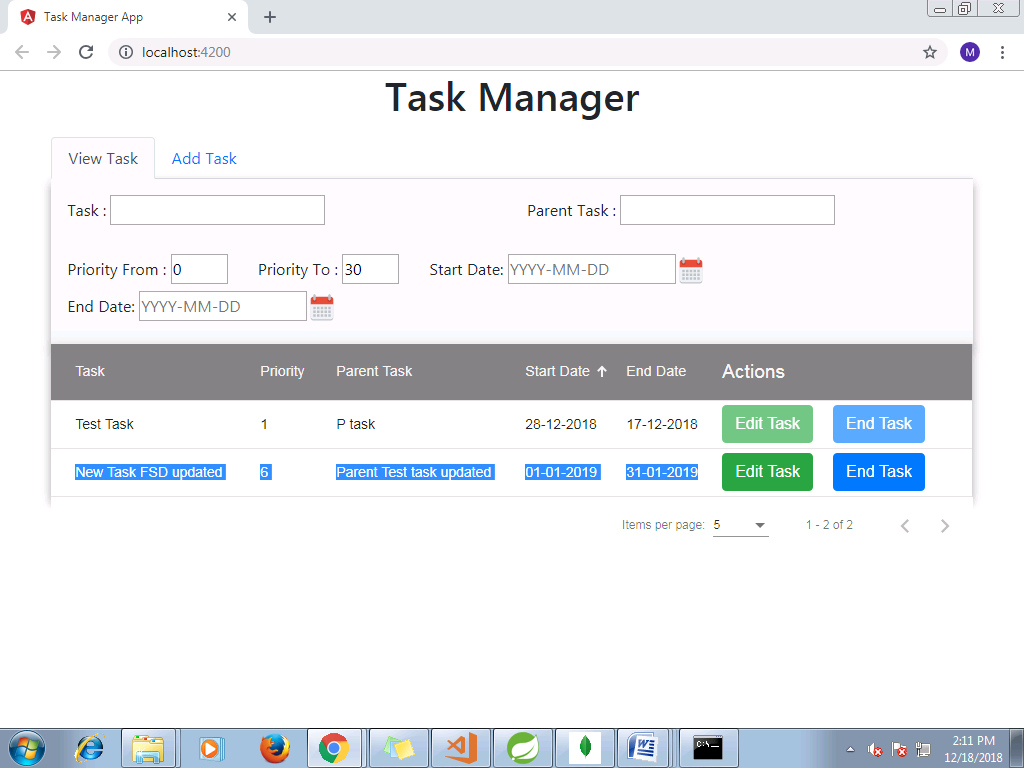


Edit Task

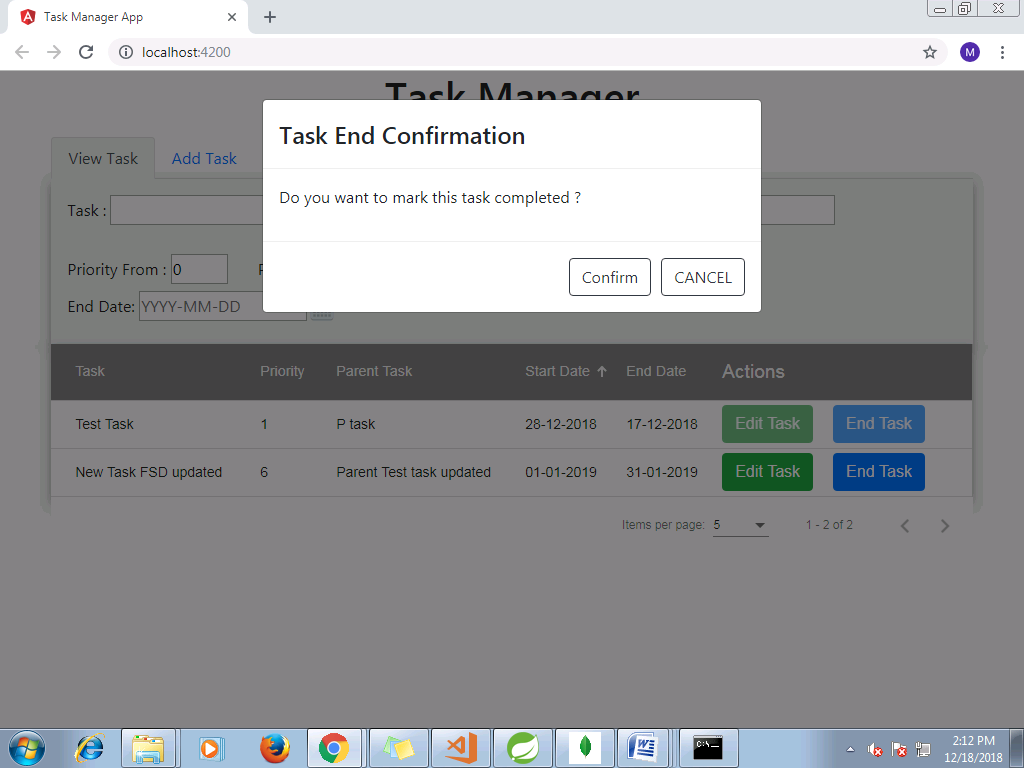


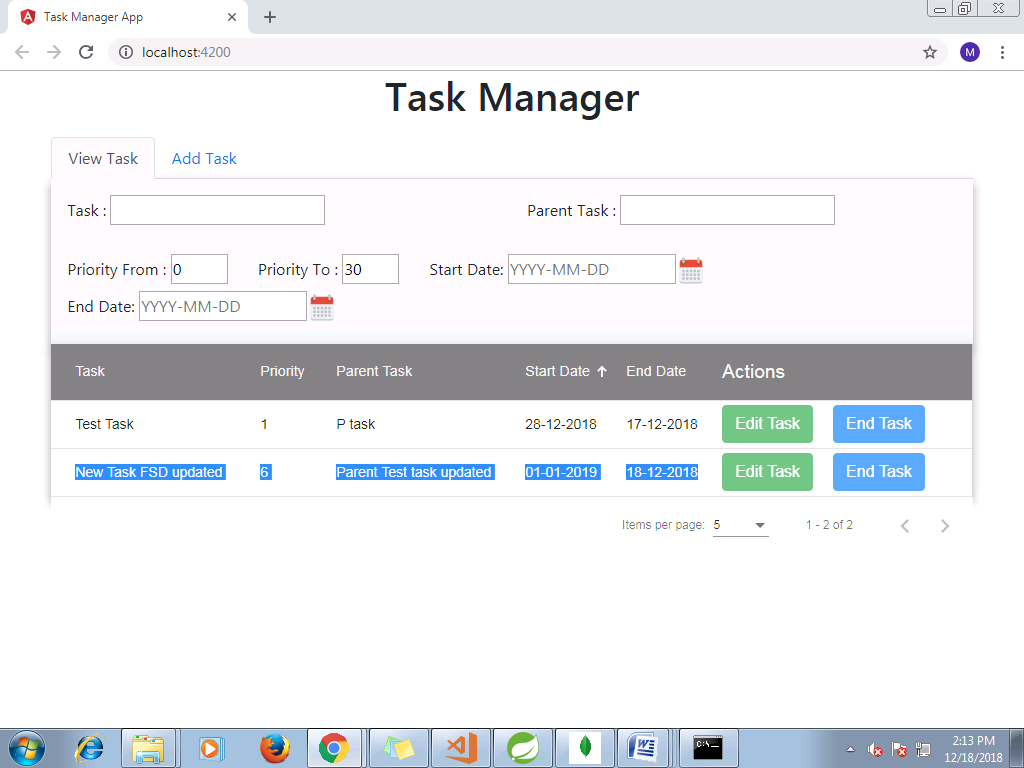
update task





End Task





Pagination

