

NGX TinyMCE Integration

A short guide to integrating tinymce in an Angular Application



+233 302 717 577

+233 502 588 736

+233 242 286 304

@ ka@jojoaddison.net

www.jojoaddison.net

✓ Ankobra River Street #5

Teshie Nungua Estates,

Accra, Ghana

Kojo Ampia-Addison

08.05.2020

ka@jojoaddison.net

INTRODUCTION

In this guide, I will show you how to integrate tinymce editor in an angular application. The application will be created using the popular Spring Boot Framework generated by JHipster.

FINAL






























Here is what we will be building.

Create or edit a Task

Name

Description

File ▾ Edit ▾ View ▾ Insert ▾ Format ▾ Table ▾

Paragraph ▾ **B** *I* ~~S~~ A A ▾                             

PROCEDURE

1. Generate your JHipster Application
2. NPM install TinyMCE 4.9.0
3. NPM install NGX-TinyMCE 9.0.0
4. Create Editor Widget
5. Inject Editor Widget in an Entity

Generate a JHipster Application

- Navigate to the folder where you want to create your application
 - Mine is in `D:\workspace\tinymce-demo-app`

```
D:\workspace>mkdir tinymce-demo-app  
D:\workspace>cd tinymce-demo-app  
D:\workspace\tinymce-demo-app>
```

- JHipster is a Yeoman scaffolding generator so yo and generator-jhipster are required.
 - `npm i -g yo generator-jhipster`

```
C:\Users\kojoa>npm i -g yo generator-jhipster
```

- Create the application by the simple command `jhipster`

```

D:\workspace\tinymce-demo-app>jhipster
INFO! Using JHipster version installed globally
INFO! Running default command
INFO! Executing jhipster:app
INFO! Options: from-cli: true

JHIPSTER

https://www.jhipster.tech

Welcome to JHipster v5.8.2
Application files will be generated in folder: D:\workspace\tinymce-demo-app

Documentation for creating an application is at https://www.jhipster.tech/creating-an-app/
If you find JHipster useful, consider sponsoring the project at https://opencollective.com/generator-jhipster

WARNING! Java 1.8 is not found on your computer. Your Java version is: 11.0.4

JHipster update available: 6.8.0 (current: 5.8.2)

Run npm install -g generator-jhipster to update.

? Which *type* of application would you like to create? Monolithic application (recommended for simple projects)
? What is the base name of your application? editor
? What is your default Java package name? io.jojoaddison
? Do you want to use the JHipster Registry to configure, monitor and scale your application? No
? Which *type* of authentication would you like to use? JWT authentication (stateless, with a token)
? Which *type* of database would you like to use? MongoDB
? Do you want to use the Spring cache abstraction? Yes, with the Ehcache implementation (local cache, for a single node)
? Would you like to use Maven or Gradle for building the backend? Maven
? Which other technologies would you like to use? WebSockets using Spring Websocket
? Which *Framework* would you like to use for the client? Angular
? Would you like to enable *SASS* stylesheet preprocessor? Yes
? Would you like to enable internationalization support? No
? Besides JUnit and Jest, which testing frameworks would you like to use? Gatling
? Would you like to install other generators from the JHipster Marketplace? No
Git repository initialized.

KeyStore 'src/main/resources/config/tls/keystore.pl2' generated successfully.

```

This will finally generate a fully functional application based on the spring boot framework. Find out more about this wonderful project at <https://jhipster.tech>

```

Application successfully committed to Git.

If you find JHipster useful consider sponsoring the project https://www.jhipster.tech/sponsors/

Server application generated successfully.

Run your Spring Boot application:
./mvnw (mvnw if using Windows Command Prompt)

Client application generated successfully.

Start your Webpack development server with:
npm start

> editor@0.0.0 cleanup D:\workspace\tinymce-demo-app
> rimraf target/{aot,www}

INFO! Congratulations, JHipster execution is complete!

```

NPM install TinyMCE 4.9.0

- Install tinymce from npm with:

```
npm i -S tinymce@4.9.0
```

- Copy tinymce from node_modules.

```
cp -r node_modules\tinymce src\main\webapp\content\.
```

This step is only necessary if you have to serve tinymce yourself.

The alternative is to use the public cdn.

NPM install NGX-TinyMCE 9.0.0

- Install ngx-tinymce from npm with:

```
npm i -S ngx-tinymce@9.0.0
```

Create Editor Widget

- Create a widget module with:
 - Navigate to the shared folder

```
cd src\main\webapp\app\shared
```

- Generate the Component. Skip import to avoid adding the component to the Shared modules.

```
ng g component EditorWidget --skipImport=true
```

- Generate the Module

```
ng g module EditorWidget
```

- Configure NgxTinyMceModule in EditorWidgetModule as follows:

```
import { NgModule, CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { NgxTinyMceModule } from 'ngx-tinymce';
import { EditorWidgetComponent } from './editor-widget.component';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

@NgModule({
  declarations: [
    EditorWidgetComponent
  ],
  imports: [
    CommonModule,
    FormsModule,
    NgxTinyMceModule.forRoot({
      baseURL: '/content/tinymce/'
    })
  ],
  exports: [
    EditorWidgetComponent
  ],
  schemas: [CUSTOM_ELEMENTS_SCHEMA]
})
export class EditorWidgetModule {}
```

- Configure the EditorWidgetComponent HTML

```
<tinyMce [(ngModel)]="content" [config]="config" (ready)="onReady($event)"></tinyMce>
```

- Configure the EditorWidgetComponent Typescript

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

@Component({
  selector: 'jhi-editor-widget',
  templateUrl: './editor-widget.component.html'})
export class EditorWidgetComponent implements OnInit {

  @Input() content: string;
  @Input() config: any;
  @Output() onDataChanged = new EventEmitter<any>();
  @Output() onDataBlur = new EventEmitter<any>();
  @Output() onKeyup = new EventEmitter<any>();
  editor: any;

  constructor() {
    this.config = {
      height: 250,
      theme: 'modern',
      plugins: 'print preview fullpage searchreplace autolink directionality visualblocks visualchars fullscreen
image imagetools link media template codesample table charmap hr pagebreak nonbreaking anchor
insertdatetime advlist lists textcolor wordcount contextmenu colorpicker textpattern',
      toolbar: 'formatselect | bold italic strikethrough forecolor backcolor | link | alignleft aligncenter alignright
alignjustify | numlist bullist outdent indent | removeformat',
      image_advtab: true,
      imagetools_toolbar: 'rotatleft rotateright | flipv fliph | editimage imageoptions',
      templates: [
        { title: 'Test template 1', content: 'Test 1' },
        { title: 'Test template 2', content: 'Test 2' }
      ],
      content_css: [
        '//fonts.googleapis.com/css?family=Lato:300,300i,400,400i',
        '//www.tinymce.com/css/codepen.min.css'
      ],
      style_formats_merge: true,
      branding: false,
      setup: (editor: any) => this.setup(editor),
    };
  }

  ngOnInit() {
    if (!this.content) {
      this.content = 'test content';
    }
  }

  onReady() {
    console.log('editor ready!');
  }

  private setup(editor: { on: (arg0: string, arg1: { (): void; (): void; (): void; }) => void; getContent: () => string }) {
    this.editor = editor;
    editor.on('blur', () => {
      this.content = editor.getContent();
      this.onDataBlur.emit(this.content);
    });
    editor.on('keyup', () => {
      this.content = editor.getContent();
      this.onKeyup.emit(this.content);
    });
    editor.on('change', () => {
      this.content = editor.getContent();
      this.onDataChanged.emit(this.content);
    });
  }
}
```

USAGE

To use the widget, create an entity and inject the editor widget into it to see it displayed. Here are the steps we shall take.

1. Create an entity with some attributes in JHipster
2. Inject the EditorWidget
3. View the entity on the client

Create a JHipster Entity

JHipster provides the entity API for creating an Entity. Let's create a Task entity.

jhipster entity Task

```
jhipster

D:\workspace\tiny-mce-demo-app>jhipster entity Task
INFO! Using JHipster version installed locally in current project's node_modules
INFO! Executing jhipster:entity Task
INFO! Options: from-cli: true

The entity Task is being created.

Generating field #1
? Do you want to add a field to your entity? Yes
? What is the name of your field? name
? What is the type of your field? String
? Do you want to add validation rules to your field? No

===== Task =====
Fields
name (String)

Generating field #2
? Do you want to add a field to your entity? Yes
? What is the name of your field? description
? What is the type of your field? String
? Do you want to add validation rules to your field? No

===== Task =====
Fields
name (String)
description (String)

Generating field #3
? Do you want to add a field to your entity? Yes
? What is the name of your field? createdAt
? What is the type of your field? Instant
? Do you want to add validation rules to your field? No

===== Task =====
Fields
name (String)
description (String)
createdAt (Instant)

Generating field #4
? Do you want to add a field to your entity? Yes
? What is the name of your field? modifiedDate
? What is the type of your field? Instant
? Do you want to add validation rules to your field? No

===== Task =====
Fields
name (String)
description (String)
createdAt (Instant)
modifiedDate (Instant)

Generating field #5
? Do you want to add a field to your entity? Yes
? What is the name of your field? lastModifiedBy
```

Answer the questions at the API prompt by providing attribute names and selecting types.

Inject the EditorWidget

Open your webapp code and navigate to:

- src/main/webapp/app/entities/task/task.module.ts

```
import { EditorSharedModule } from 'app/shared';
@NgModule({
  imports: [
    EditorSharedModule, EditorWidgetModule, RouterModule.forChild(ENTITY_STATES)
  ]
  ...
})
...
```

- src/main/webapp/app/entities/task/task-update.component.html

Here we replace the input tag below

```
<input type="text" class="form-control" name="description" id="field_description"
[(ngModel)]="task.description" />
```

with the widget tag that follows.

```
<jhi-editor-widget [theme]="modern" [content]="task.description"
(onDataChanged)="setContentChanged($event)"></jhi-editor-widget>
```

- src/main/webapp/app/entities/task/task-update.component.ts

Here we define the function setContentChanged(data: any) as follows

```
protected setContentChanged(data: string): void {
  this.task.description = data;
}
```

View the Entity

- Execute the JHipster maven/gradle build and run in own window

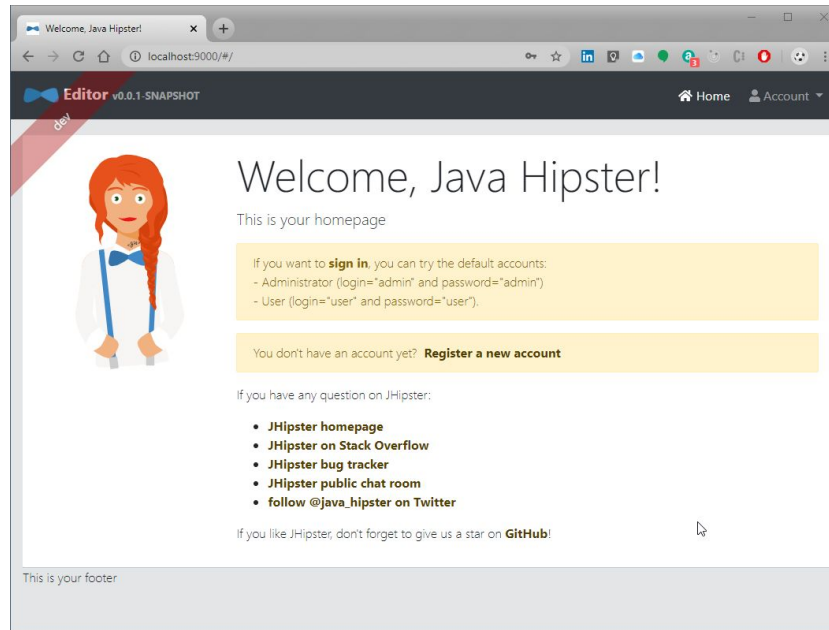
```
/mvnw
```

- Execute npm start in another

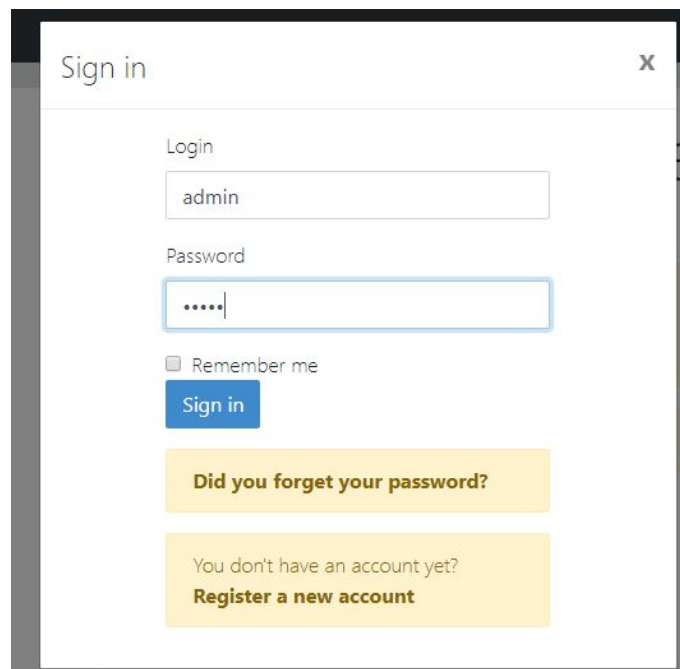
```
npm start
```

- A successful execution opens the browser with a welcome page.

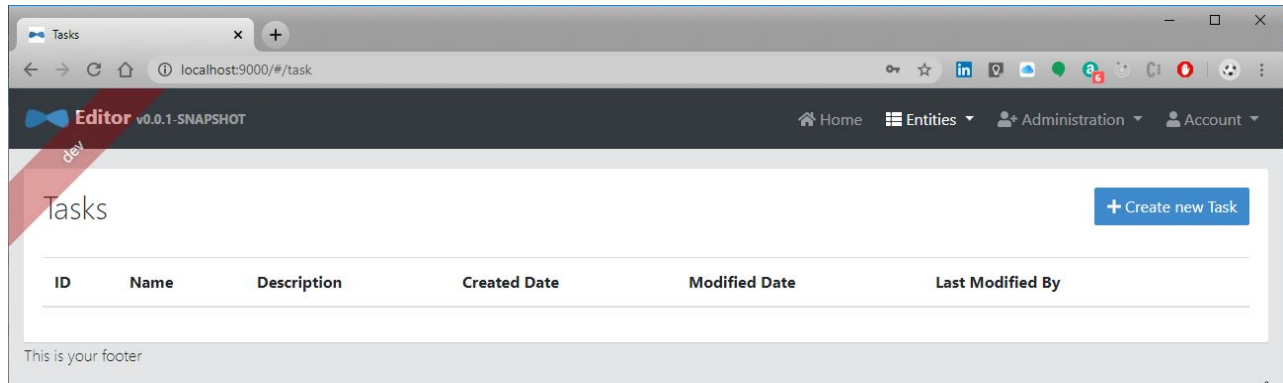
Welcome Page



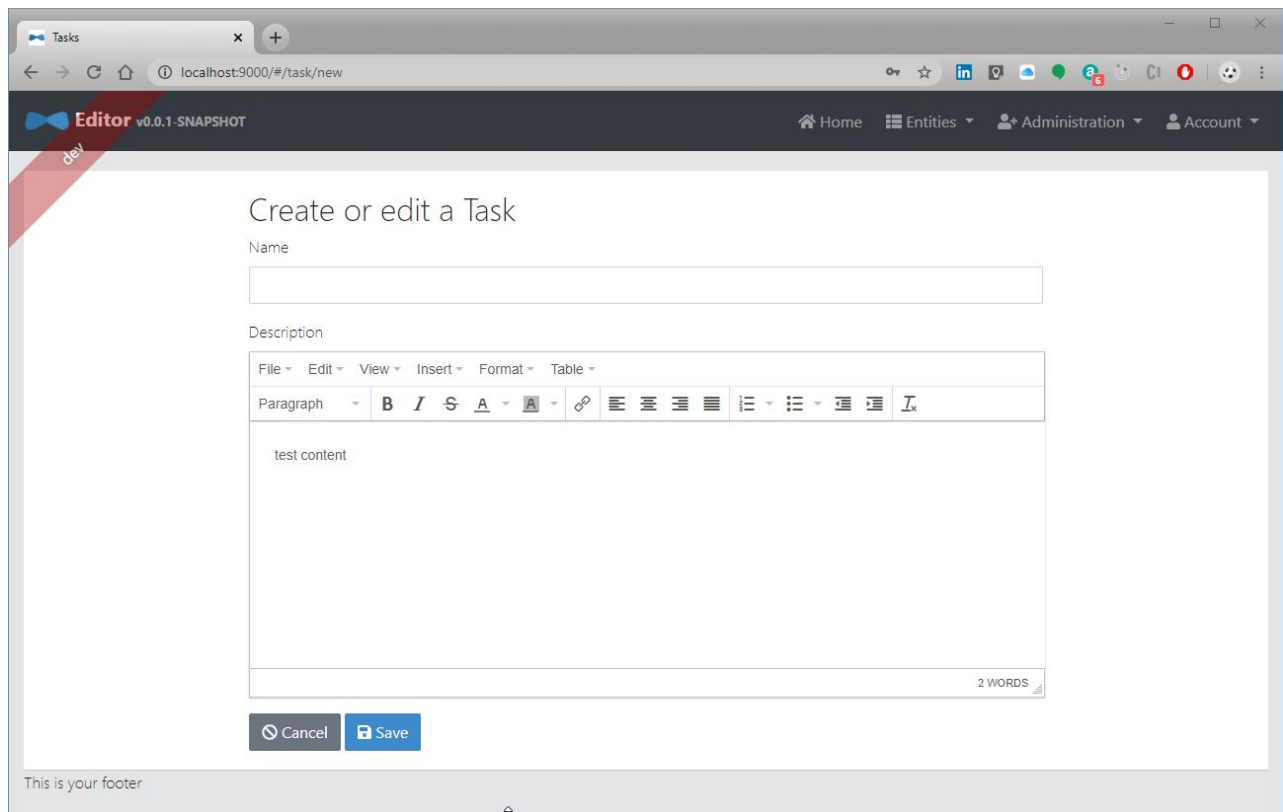
Login Page



Task List Page



Task List Create



CONCLUSION

Find this demo project on github:

<https://github.com/jojoaddison/tinymce-demo-app>

I would like to read your feedback and how you use this app.

REFERENCES

1. JHipster - <https://www.jhipster.tech>
2. NGX-TinyMCE - <https://cipchk.github.io/ngx-tinymce/>
3. TinyMCE - <https://github.com/tinymce/tinymce>