**📁 Recommended Directory Structure**

A well-organized templates/ directory might look like this 

src/

└── templates/

├── emails/

│ ├── welcome.hbs

│ ├── password-reset.hbs

│ └── ...

├── notifications/

│ ├── system-alert.hbs

│ └── ...

└── templates.module.ts

```



- \*\*`emails/`\*\*:Contains templates for various email communications

- \*\*`notifications/`\*\*:Holds templates for system notifications or alerts

- \*\*`templates.module.ts`\*\*:Registers and provides access to the templates throughout the application

This structure aligns with common practices in web application development, facilitating easy management and localization of templates citeturn0search0

---

## 🛠️ Implementing Template Rendering with Handlebars

To render templates dynamically, you can use the Handlebars templating engine. Here's how to set it u:

### 1. \*\*Install Dependencies\*\*



```bash

npm install --save handlebars fs-extra

``





### 2. \*\*Create a Template Service\*\*



```typescript

// src/templates/template.service.ts

import { Injectable } from '@nestjs/common';

import \* as fs from 'fs-extra';

import \* as path from 'path';

import \* as Handlebars from 'handlebars';

@Injectable()

export class TemplateService {

private readonly templatesDir = path.join(\_\_dirname, 'emails');

async renderTemplate(templateName: string, context: any): Promise<string> {

const filePath = path.join(this.templatesDir, `${templateName}.hbs`);

const templateContent = await fs.readFile(filePath, 'utf-8');

const template = Handlebars.compile(templateContent);

return template(context);

}

}

``





### 3. \*\*Register the Service in a Module\*\*



```typescript

// src/templates/templates.module.ts

import { Module } from '@nestjs/common';

import { TemplateService } from './template.service';

@Module({

providers: [TemplateService],

exports: [TemplateService],

})

export class TemplatesModule {}

``





---

## 📧 Example: Sending a Welcome Emai

Assuming you have a `welcome.hbs` template, here's how you might use t:

\*\*`welcome.hbs` Template:\*



```handlebars

<!DOCTYPE html>

<html>

<body>

<h1>Welcome, {{name}}!</h1>

<p>Thank you for joining our platform.</p>

</body>

</html>

``





\*\*Email Service:\*



```typescript

// src/email/email.service.ts

import { Injectable } from '@nestjs/common';

import { TemplateService } from '../templates/template.service';

@Injectable()

export class EmailService {

constructor(private readonly templateService: TemplateService) {}

async sendWelcomeEmail(userEmail: string, userName: string) {

const htmlContent = await this.templateService.renderTemplate('welcome', { name: userName });

// Use your preferred email sending method here, e.g., Nodemailer

// await this.mailerService.sendMail({ to: userEmail, subject: 'Welcome!', html: htmlContent });

}

}

``





---

## 🌐 Best Practices

- \*\*Localization\*: Store templates in a way that supports multiple languages, facilitating internationalization citeturn0searc0.

- \*\*Modularity\*: Keep templates modular to allow for easy updates and maintenace.

- \*\*Testing\*: Regularly test your templates across different email clients to ensure consistent rendering citeturn0searc4.

--