

Configurer HTTPS Angular



Comment configurer du https dans angular.json

Pour configurer HTTPS dans un projet Angular en local, tu peux modifier le fichier `angular.json` et utiliser la commande `ng serve` avec des options spécifiques. Voici comment procéder :

1. Générer un certificat auto-signé (si nécessaire)

Si tu n'as pas déjà un certificat SSL, tu peux en générer un avec OpenSSL :

```
sh

openssl req -x509 -newkey rsa:4096 -keyout localhost-key.pem -out localhost-cert.pem -days 365 -nodes
```

Cela va créer deux fichiers :

- `localhost-key.pem` (clé privée)
- `localhost-cert.pem` (certificat)

2. Modifier `angular.json`

Ajoute ou modifie les options du serveur de développement dans le fichier `angular.json` :

```
json

{
  "projects": {
    "mon-projet": {
      "architect": {
        "serve": {
          "options": {
            "ssl": true,
            "sslKey": "localhost-key.pem",
            "sslCert": "localhost-cert.pem"
          }
        }
      }
    }
  }
}
```

3. Lancer Angular avec HTTPS

Utilise la commande suivante :

```
sh

ng serve --ssl
```

Si tu veux spécifier directement les fichiers de clé et de certificat :

```
sh

ng serve --ssl --ssl-key localhost-key.pem --ssl-cert localhost-cert.pem
```

Cela activera HTTPS pour ton serveur de développement Angular.

Notes :

- Le navigateur peut afficher un avertissement de sécurité, car il s'agit d'un certificat auto-signé.
- Pour éviter cet avertissement, il faut ajouter le certificat aux autorités de certification de confiance de ton système.
- En production, utilise un certificat valide issu d'une autorité de certification comme Let's Encrypt.

Tu veux un exemple plus détaillé sur la gestion des certificats dans ton environnement de développement ?



Comment debugger une application Outlook web addin en utilisant angular.json et non du webpack

Pour déboguer une application Outlook Web Add-in basée sur Angular et configurée avec `angular.json` (sans Webpack direct), voici les étapes à suivre :

1. Activer le Debugging avec HTTPS

Outlook impose HTTPS pour charger un Web Add-in. Si tu ne l'as pas encore fait, configure HTTPS dans `angular.json` :

Ajoute ces options dans `angular.json` sous `"serve"` :

```
json

"options": {
  "ssl": true,
  "sslKey": "localhost-key.pem",
  "sslCert": "localhost-cert.pem"
}
```

Puis lance ton serveur Angular avec :

```
sh

ng serve --ssl
```

2. Modifier `manifest.xml`

Dans ton fichier `manifest.xml`, mets l'URL du serveur Angular (exécuté en local sur `https://localhost:4200`) :

```
xml

<SourceLocation DefaultValue="https://localhost:4200/index.html"/>
```

Vérifie aussi les permissions et scopes nécessaires.

3. Charger le Web Add-in dans Outlook

Ouvre un terminal et utilise la commande `office-addin-dev-settings` pour charger le manifest :

```
sh

npx office-addin-dev-settings appsource add --manifest-path ./manifest.xml
```

Puis ouvre Outlook Web ou Desktop, et active le chargement d'add-ins personnalisés.

4. Lancer le Debugging avec Visual Studio Code

Si tu utilises **VS Code**, ajoute une configuration de debug dans `.vscode/launch.json` :

```
json

{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "chrome",
      "request": "launch",
      "name": "Launch Outlook Add-in",
      "url": "https://localhost:4200",
      "webRoot": "${workspaceFolder}",
      "sourceMaps": true
    }
  ]
}
```

Lance le debug avec `F5`.

5. Utiliser Edge pour Debugger (via DevTools)

Si tu préfères Edge :

1. Ouvre ton Web Add-in dans Outlook Web (<https://outlook.office.com>).
 2. Appuie sur `F12` pour ouvrir les DevTools.
 3. Va dans l'onglet **Sources** pour explorer ton code Angular.
 4. Mets des **breakpoints** et recharge ton add-in.
-

6. (Facultatif) Debugger avec `office-addin-debugging`

Tu peux aussi utiliser `office-addin-debugging` pour attacher un debugger :

```
sh

npx office-addin-debugging start --manifest-path ./manifest.xml
```

Cela ouvre Outlook et active le mode debug.

Résumé

- ⚡ `ng serve --ssl` pour exécuter ton app Angular en HTTPS.
- ✏️ Mets l'URL locale dans `manifest.xml`.
- 📄 Charge le manifest avec `office-addin-dev-settings`.
- 🛠️ Utilise VS Code, Edge ou `office-addin-debugging` pour debugger.

Tu veux un exemple concret avec un projet Angular minimal ?



Oui

Je vais te fournir un projet minimal d'Outlook Web Add-in basé sur Angular 18 en utilisant `angular.json` (sans Webpack). L'objectif est d'avoir une structure fonctionnelle avec :

- ✓ Un `manifest.xml` bien configuré.
- ✓ Un projet Angular 18 avec des **Standalone Components**.
- ✓ Le support du **debugging avec HTTPS**.
- ✓ Un exemple basique d'interface avec Fluent UI.

📁 1. Structure du projet

```
pgsql

outlook-web-addin-angular/
├── src/
│   ├── app/
│   │   ├── app.component.ts
│   │   ├── app.component.html
│   │   ├── app.routes.ts
│   │   ├── home.component.ts
│   │   ├── home.component.html
│   │   └── outlook.service.ts
│   ├── index.html
│   └── main.ts
├── manifest.xml
├── angular.json
├── package.json
├── tsconfig.json
├── .vscode/launch.json
└── .gitignore
```

🛠️ 2. Initialisation du projet Angular 18

Crée le projet et ajoute les dépendances nécessaires :

```
sh

ng new outlook-web-addin-angular --standalone --style=css
cd outlook-web-addin-angular
npm install office-js @fluentui/web-components
```

3. Configuration du `manifest.xml`

Crée un fichier `manifest.xml` à la racine avec :

```
xml

<?xml version="1.0" encoding="UTF-8"?>
<OfficeApp xmlns="http://schemas.microsoft.com/office/appforoffice/1.1"
  xsi:type="TaskPaneApp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  Id="outlook-web-addin-angular"
  Version="1.0.0.0"
  ProviderName="Benutzer"
  DefaultLocale="en-US">

  <DisplayName>Angular Outlook Add-in</DisplayName>
  <Description>Un add-in Outlook avec Angular 18</Description>

  <Hosts>
    <Host Name="Mailbox"/>
  </Hosts>

  <DefaultSettings>
    <SourceLocation DefaultValue="https://localhost:4200/index.html"/>
  </DefaultSettings>

  <Permissions>ReadWriteItem</Permissions>
</OfficeApp>
```

4. Configuration `angular.json` (HTTPS)

Ajoute les options de SSL sous `"serve"` :

```
json

"serve": {
  "options": {
    "ssl": true,
    "sslKey": "localhost-key.pem",
    "sslCert": "localhost-cert.pem"
  }
}
```

Génère un certificat SSL (si nécessaire) :

```
sh

openssl req -x509 -newkey rsa:4096 -keyout localhost-key.pem -out localhost-cert.pem -days 365 -nodes
```

5. Création des composants

`app.routes.ts`

```
ts

import { Routes } from '@angular/router';
import { HomeComponent } from './home.component';

export const routes: Routes = [
  { path: '', component: HomeComponent },
];
```

`app.component.ts`

```
ts

import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';
import { routes } from './app.routes';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterModule.forRoot(routes)],
  template: `<router-outlet></router-outlet>`,
})
export class AppComponent {}
```

`home.component.ts`

```
ts

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

@Component({
  selector: 'app-home',
  standalone: true,
  template: `
    <h1>Bienvenue sur mon add-in Outlook !</h1>
    <button (click)="insertText()">Insérer du texte</button>
  `,
})
export class HomeComponent {
  insertText() {
    Office.context.mailbox.item.body.setAsync('Hello depuis Angular !');
  }
}
```

6. Lancer le projet et charger l'Add-in

1. Démarrer le serveur Angular en HTTPS :

```
sh

ng serve --ssl
```

2. Charger l'Add-in dans Outlook :

```
sh
```

```
npx office-addin-dev-settings appsource add --manifest-path ./manifest.xml
```

3. **Ouvrir Outlook Web** (<https://outlook.office.com>), aller dans "Mes compléments" et activer l'Add-in.

7. Debugging avec VS Code

Ajoute dans ``.vscode/launch.json`` :

```
json
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "chrome",
      "request": "launch",
      "name": "Launch Outlook Add-in",
      "url": "https://localhost:4200",
      "webRoot": "${workspaceFolder}"
    }
  ]
}
```

Lance le debug avec ``F5``.

Résumé

- Serveur HTTPS avec ``ng serve --ssl``
- Manifest.xml avec l'URL locale
- Ajout de l'Add-in avec ``office-addin-dev-settings``
- Debugging avec VS Code et Edge

Tu veux que j'ajoute d'autres fonctionnalités, comme la gestion des placeholders ?



Avec cette approche plus besoin de webpack?

Oui, avec cette approche basée sur ``angular.json``, **il n'y a plus besoin de Webpack directement !**

Pourquoi ?

1. **Angular CLI intègre Webpack en interne** → Tu n'as pas besoin de le configurer manuellement.
2. ``angular.json`` **gère tout** (compilation, serveur HTTPS, assets...).
3. **Le serveur** ``ng serve`` fonctionne directement avec Outlook sans build manuel.
4. **Tu peux toujours faire un build optimisé** avec ``ng build --configuration production`` pour générer les fichiers statiques (``dist/``), mais ce n'est pas nécessaire en mode dev.

Comparaison avec Webpack classique :

Critère	Avec Webpack (manuel)	Avec Angular CLI (`angular.json`)
Configuration	Fichier `webpack.config.js` à gérer	Tout est dans `angular.json`
Compilation	Nécessite des loaders spécifiques	Automatique avec Angular CLI
Serveur Dev	`webpack-dev-server` à configurer	`ng serve` suffit
HTTPS	Configuré dans Webpack	Juste `ng serve --ssl`
Debugging	Souvent plus complexe	Intégré à Angular + VS Code

Quand utiliser Webpack ?

- Si tu veux un **contrôle total sur le bundling** (rare avec Angular 18).
- Si tu dois **modifier le comportement de la compilation** (ex. charger des fichiers `.md` ou des workers Web).
- Si tu veux utiliser **Module Federation** pour du micro frontend avancé.

Mais pour un **Outlook Web Add-in avec Angular 18**, `angular.json` **suffit largement**.

Tu veux une explication plus détaillée sur un point précis ?

Cette conversation vous a-t-elle été utile jusqu'à présent ?

ChatGPT peut faire des erreurs. Envisagez de vérifier les informations importantes.