Samuel Ronce

Angular

version 18 et suivantes

Le framework JavaScript open source de Google

Angular

Éditions EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75005 Paris
www.editions-eyrolles.com

© Éditions Eyrolles, 2024, ISBN : 978-2-416-01732-2

Table des matières

PARTIE 8

Annexes 261

ANNEXE A

Rappels sur TypeScript 263

ANNEXE B

Les promesses 273

ANNEXE C

Jasmine 277

ANNEXE D

Les modules 281

ANNEXE E

Opérateurs courants sur RxJS 285

Index 291

Avant-propos

Pourquoi utiliser Angular ?

Angular offre une architecture complète qui inclut tout ce dont vous avez besoin pour construire une application complexe, en fournissant des solutions intégrées pour la gestion de l'état, le routage, les formulaires, et plus encore. Il promeut également les bonnes pratiques de développement, comme la modularité, l'injection de dépendances, et le développement par composants, facilitant ainsi le travail en équipe et la maintenance du code.

Angular est également bien connu pour son système de synchronisation en temps réel entre le modèle et la vue. Cela réduit considérablement le code nécessaire pour les mises à jour de l'interface utilisateur, rendant le développement plus rapide et plus intuitif.

En outre, Angular bénéficie d'un large écosystème et d'une communauté active, offrant une multitude de bibliothèques, d'extensions et d'outils de développement pour étendre les fonctionnalités de vos applications.

À qui s'adresse cet ouvrage ?

L'objectif principal de ce livre est de vous guider à travers les fondamentaux d'Angular, en couvrant toutes les versions à partir d'Angular 18, et en mettant en lumière les meilleures pratiques et les nouvelles fonctionnalités introduites. Que vous soyez de niveau débutant, intermédiaire, ou expert, ce livre est conçu pour vous offrir une compréhension approfondie d'Angular, en vous permettant de progresser à votre rythme.

- Débutant : nous utiliserons des exemples simples et proposerons une vulgarisation pour expliquer les concepts élémentaires d'Angular. Cela permettra à ceux qui découvrent Angular de comprendre les bases nécessaires pour commencer à construire leurs premières applications web.

- Intermédiaire : nous explorerons des concepts plus avancés et des techniques de développement spécifiques à Angular, permettant ainsi aux lecteurs d'élargir leurs compétences et de construire des applications plus complexes.

- Expert : nous plongerons dans des stratégies avancées de test et d'optimisation pour garantir la fiabilité et la performance des applications Angular.

Comment est structuré ce livre ?

Nous commencerons par une **Introduction à Angular**, où vous découvrirez les fondations du framework, son architecture et sa philosophie. Cela posera les bases nécessaires pour comprendre pourquoi et comment Angular transforme le développement d'applications web.

Ensuite, nous plongerons au cœur d'Angular avec les **composants** qui sont les blocs de construction de toute application Angular. Vous apprendrez à les créer, à les organiser et à les utiliser pour construire une interface utilisateur interactive et réactive.

L'**Injection de dépendances** est un concept clé dans Angular, facilitant la modularité et la réutilisabilité du code. Nous explorerons comment Angular utilise ce modèle de conception pour fournir des instances de classes nécessaires à vos composants et services.

Avec les **routeurs**, nous verrons comment Angular gère la navigation dans votre application, permettant de créer des applications à page unique (SPA) avec des URL propres et une navigation fluide.

Les **observables** et les **signaux** introduisent un modèle de programmation réactif dans Angular, offrant une approche puissante pour gérer les événements asynchrones, comme les données provenant d'une API web.

Nous aborderons également les **requêtes HTTP** pour communiquer avec des serveurs et des API externes, un aspect vital pour les applications modernes qui nécessitent souvent d'interagir avec des ressources sur le Web.

La gestion des **formulaires** est une autre fonctionnalité puissante d'Angular, permettant de collecter et de valider les entrées utilisateur de manière efficace et flexible.

Pour compléter votre apprentissage, le livre comprend cinq annexes dédiées à des concepts et outils complémentaires :

1. **Rappels sur TypeScript**, pour solidifier votre compréhension du langage principal utilisé avec Angular.
2. **Les promesses**, un mécanisme clé pour gérer l'asynchronisme dans les applications web modernes.
3. **Jasmine**, pour découvrir les tests unitaires dans vos applications Angular.
4. **Les modules**, essentiels pour structurer et organiser votre code Angular.
5. **Opérateurs courants sur RxJS**, pour approfondir votre maîtrise des observables et des opérations réactives.

À travers cet apprentissage, vous construirez une application complète, enrichissant progressivement ses fonctionnalités au fur et à mesure de votre progression dans le livre. Cet ouvrage est conçu pour vous offrir une expérience d'apprentissage complète et pratique, vous permet-

tant non seulement de comprendre Angular, mais aussi de l'appliquer efficacement dans vos propres projets de développement web.

Fil rouge du livre

Tout au long de ce livre, notre fil rouge sera la création d'une application web. Ce projet pratique permettra d'appliquer et de consolider les connaissances acquises à chaque étape, offrant ainsi une expérience d'apprentissage complète et engageante.

À la fin de quelques chapitres, vous aurez accès à un lien contenant du code fonctionnel correspondant aux parties du projet abordées. Cela vous permettra de visualiser concrètement votre progression et d'appliquer directement les concepts étudiés dans un contexte réel.

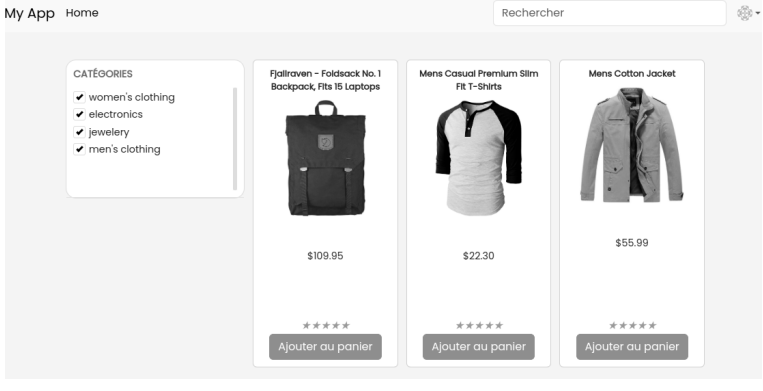


Figure 0–1
Prévisualisation de l'application

Figure 0–2
Prévisualisation de la page de connexion

Ensemble, nous allons découvrir, apprendre et maîtriser Angular, en construisant une application e-commerce robuste, efficace et moderne. Prêts pour cette aventure ? Alors, plongeons dans le monde d'Angular !

1

Introduction

Angular est un acteur incontournable dans l'univers du développement web, se positionnant comme le deuxième framework frontend le plus utilisé au monde, juste derrière des géants comme React. Mais qu'est-ce qui rend Angular si spécial et pourquoi continue-t-il à captiver une large communauté de développeurs ?

La vision d'Angular

Angular, un nom qui résonne avec force dans l'univers du développement web, a été lancé en 2010, sous l'égide de Google. Initialement conçu pour répondre aux défis du développement d'applications web dynamiques, AngularJS dans sa première version, a révolutionné la manière dont les développeurs et les entreprises concevaient leurs applications, introduisant le concept de composants avant qu'il ne devienne une norme.

En 2016, le monde a assisté à une évolution majeure avec la sortie d'Angular 2. Plus qu'une simple mise à jour, Angular 2 représentait une réécriture complète du framework, introduisant un changement de paradigme avec l'adoption de TypeScript et une architecture orientée composants et services. Cette version a non seulement amélioré la performance et la scalabilité, mais a aussi modernisé le développement web avec des pratiques et des standards plus avancés.

Depuis lors, Angular a continué d'évoluer, avec un nouveau cycle de versions majeures tous les six mois. Chaque version apporte son lot d'améliorations, de nouvelles fonctionnalités, et de simplifications, rendant Angular plus performant, plus facile à utiliser, et mieux adapté aux défis du développement web moderne. Angular a embrassé la réactivité avec RxJS, renforcé sa sécurité, et optimisé son rendu.

L'introduction de concepts comme les signaux dans Angular 16 et le contrôle de flux amélioré dans Angular 17 témoigne de l'engagement constant du framework à simplifier le développement web tout en restant à la pointe de la technologie. Angular ne cesse de repousser les limites, rendant le développement d'applications web à la fois plus accessible aux novices et plus agréable pour les experts.

Une des transformations majeures est l'élimination progressive des modules, au profit des composants standalone. Cette évolution signifie que vous pouvez désormais créer des applications avec moins de configurations et une structure plus légère. De plus, Angular s'éloigne doucement de RxJS pour les opérations courantes, se dirigeant vers l'utilisation des signaux. Ces derniers représentent une abstraction réactive simplifiée, permettant de gérer les états et les flux de données avec une facilité déconcertante.

Aujourd'hui, Angular est l'un des frameworks les plus utilisés et respectés dans le monde du développement web. Grâce à sa communauté dynamique et au soutien de Google, Angular continue de prospérer, en s'adaptant aux besoins changeants des développeurs et en restant pertinent dans un paysage technologique en constante évolution.

Angular par rapport à ses concurrents

Lorsqu'il s'agit de choisir un framework ou une bibliothèque pour le développement frontend, Angular, React et Vue.js se présentent comme des options principales, chacune avec ses propres caractéristiques et avantages. Comprendre leurs différences est crucial pour prendre une décision éclairée en fonction des besoins spécifiques du projet. Voici un tableau comparatif mettant en lumière les principales différences entre Angular, React et Vue.js :

Tableau 1–1 Comparaison entre Angular, React et Vue.js

Caractéristique	Angular	React	Vue.js
Type	Framework complet	Bibliothèque	Framework progressif
Langage principal	TypeScript	JavaScript (JSX)	JavaScript (Option API / Composition API)
Architecture	Basé sur des composants et services	Basé sur des composants	Basé sur des composants
Gestion des états	Les signaux, RxJs, NgRx ou NgXs	Redux, Zustand ou Context API	Pinia ou Composition API
Création de projets	Angular CLI	Vite ou Create React App	Vue CLI ou Vite
Performance	Optimisé avec Ivy Renderer	Dépend de l'utilisation des composants et de la gestion d'état	Performant avec Virtual DOM et optimisations fines
Popularité et communauté	Très populaire avec une grande communauté	Extrêmement populaire avec la plus grande communauté	Très populaire avec une communauté croissante
Cas d'utilisation	Grandes applications d'entreprise avec une structure complexe	Applications de toutes tailles, favorisant une approche plus fonctionnelle	Petits à moyens projets, avec une courbe d'apprentissage douce

Cette comparaison met en évidence les spécificités de chaque option, permettant aux développeurs de choisir la solution la plus adaptée à leurs besoins. Angular offre une boîte à outils complète pour les applications d'entreprise, React séduit par sa flexibilité et sa large adoption, tandis que Vue.js est apprécié pour sa simplicité et sa facilité d'intégration dans des projets de différentes tailles.

Les nouveautés depuis les dernières versions

Depuis quelques versions (Angular 15+), le framework a introduit une série de nouveautés et d'améliorations visant à simplifier le développement d'applications, à améliorer les performances et à rendre Angular plus intuitif et facile à utiliser. Voici un aperçu de certaines de ces évolutions clés :

- Composants Standalone : les composants standalone représentent une avancée significative dans la simplification de la structure des applications Angular. Cette fonctionnalité permet aux développeurs de créer des composants qui ne nécessitent pas de recourir à la déclaration dans un module Angular. Cela réduit la complexité et le *boilerplate code* (ou code passe-partout en français), facilitant ainsi le développement et la maintenance des applications.
- Moins de dépendance sur RxJS : bien que RxJS reste une partie intégrante d'Angular pour gérer les flux de données asynchrones, Angular a réduit la dépendance obligatoire sur RxJS pour certaines opérations courantes. Cette évolution est en partie due à l'introduction des signaux, offrant une alternative plus simple pour la gestion des états et des interactions réactives dans les applications.
- Contrôles de flux @if, @for : à partir de Angular 17, le panel de directives structurelles a été enrichi avec des améliorations significatives sur les contrôles de flux tels que `@if` et `@for`. Ces directives permettent au rendu conditionnel et aux listes dynamiques d'être plus performants et plus faciles à utiliser, contribuant à une meilleure lisibilité et à une maintenance simplifiée du code.
- Chargement paresseux avec @defer : la directive `@defer` est une nouveauté permettant de charger de manière paresseuse *(lazy loading)* certaines parties de l'interface utilisateur. Cette technique optimise le temps de chargement initial des applications en ne chargeant que les contenus nécessaires à l'utilisateur, améliorant ainsi la performance perçue de l'application.

Les outils pour bien démarrer avec Angular

Pour un travail efficace avec Angular, Visual Studio Code (VS Code) est l'éditeur de choix de nombreux développeurs. Il est léger, rapide et doté d'une multitude d'extensions qui facilitent le développement d'Angular. Voici quelques extensions recommandées :

- **Angular Language Service** : fournit une autocomplétion intelligente, des erreurs de diagnostic, etc. Très pratique car les templates Angular sont des fichiers HTML avec des ajouts spécifiques à Angular, et cette extension permet de travailler avec ces ajouts.

Figure 1–1
Utilisation de l'extension Angular Language Service dans un template

- **Angular Snippets** : des extraits de code pour Angular, TypeScript et HTML. En tapant un mot-clé, par exemple `a-httpclient-get`, cette fonctionnalité ajoutera le bloc de code (un *snippet*) correspondant.

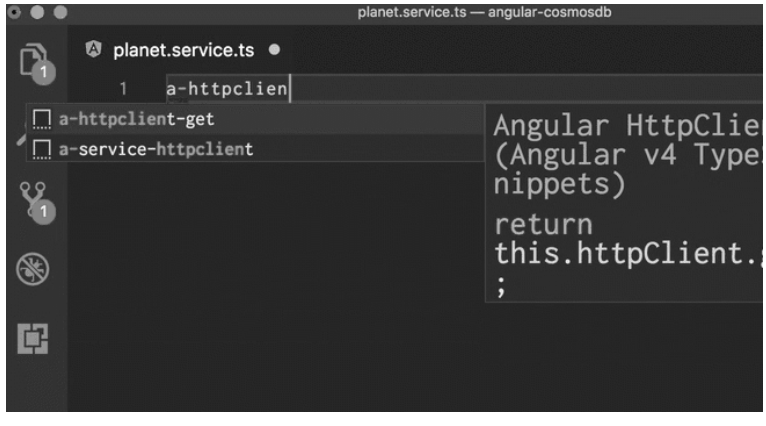


Figure 1–2
Tapez un mot-clé, Angular Snippets ajoute le bloc de code.

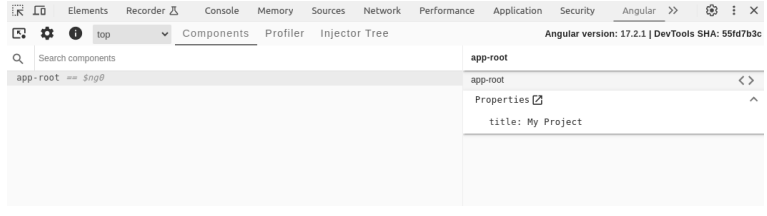
- **Prettier** : un formateur de code pour garder votre code propre et formaté uniformément.

Sur Chrome ou Firefox, vous pouvez installer Angular DevTools, extension qui permet de visualiser les composants et les débugger.

1. Cherchez *Angular DevTools* dans le Chrome Web Store ou le Firefox Add-ons.
2. Cliquez sur *Ajouter à Chrome* ou *Ajouter à Firefox* pour installer l'extension.
3. Une fois installé, vous verrez un nouvel onglet *Angular* dans les outils de développement de votre navigateur. Cliquez dessus pour voir les composants de votre application, leurs propriétés et leur état.

Figure 1–3
Installation d'Angular DevTools

Figure 1–4
Aperçu d'Angular DevTools



Installer Angular

1. Pour installer Angular, assurez-vous que Node.js est déjà présent sur votre machine. Ensuite, ouvrez votre terminal et tapez `npm install -g @angular/cli`. Cela installera l'interface de ligne de commande (CLI) d'Angular globalement sur votre système, ce qui est utile pour créer des projets, générer des fichiers et exécuter diverses tâches de développement.

> **CONSEIL Pourquoi NodeJS ?**
>
> Node.js est une plate-forme logicielle avec une architecture orientée événements. Elle est indispensable pour exécuter le package npm *(node package manager)* qui est le gestionnaire de paquets de Node.js. Angular CLI s'appuie sur npm pour gérer les dépendances de votre projet. Sans Node.js, vous ne pourrez pas installer Angular CLI, ni exécuter les commandes nécessaires pour travailler avec Angular.

2. Ensuite, pour créer un nouveau projet Angular, tapez `ng new nom-du-projet`. Cela créera un nouveau dossier nommé `nom-du-projet` contenant une application Angular prête à l'emploi.
3. Ouvrez un éditeur de code, comme VS Code, et sélectionnez le dossier de votre projet. Vous pouvez maintenant commencer à travailler sur votre application Angular.

4 Lancez le serveur de développement en utilisant la commande `ng serve`. Cela ouvrira votre application sur un serveur local, accessible à l'adresse `http://localhost:4200`.

> **CONSEIL L'alternative npm start**
>
> La commande `npm start` est un raccourci pour `ng serve`. La première lance le serveur de développement local du projet Angular, tandis que la seconde lance le serveur global Angular CLI. Si vous avez plusieurs projets avec différentes versions d'Angular, il est préférable d'utiliser `npm start` pour éviter les conflits.

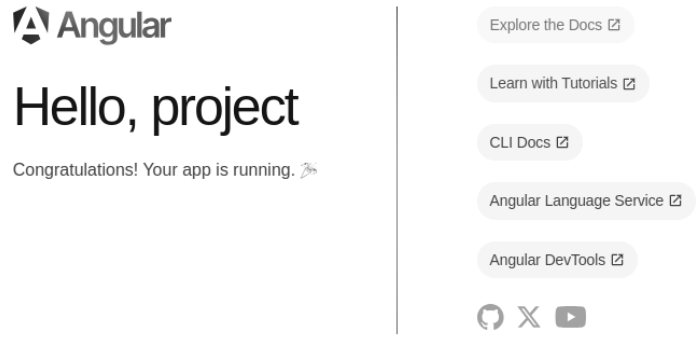


Figure 1–5
La page par défaut sur le serveur local

5 Vous pouvez commencer à travailler sur votre application Angular en modifiant les fichiers dans le dossier `src`. Les modifications effectuées seront automatiquement rechargées dans le navigateur, vous permettant de voir les changements en temps réel. Toutes les réalisations sont effectuées avec le langage TypeScript.

> **CONSEIL Le compilateur**
>
> Angular utilise un compilateur pour transformer le code TypeScript en JavaScript. L'outil ViteJS lui permet ainsi de recharger automatiquement la page lorsqu'un fichier est modifié. Vous pouvez arrêter le serveur en appuyant sur `Ctrl+C` dans le terminal.

> **CONSEIL Qu'est-ce que TypeScript ?**
>
> TypeScript est un sur-ensemble de JavaScript qui ajoute des fonctionnalités de typage statique au langage. Cela permet de détecter les erreurs de typage à la compilation plutôt qu'à l'exécution, ce qui peut aider à réduire les bogues et à améliorer la qualité du code. TypeScript est largement utilisé dans le développement Angular, car il offre une meilleure productivité et une meilleure maintenabilité du code.

Les fichiers de configuration

Lorsque vous démarrez un projet Angular, trois fichiers de configuration sont essentiels dans la structure et le comportement de votre application : `package.json`, `tsconfig.json` et `angular.json`. Chacun de ces fichiers a une fonction spécifique qui contribue à la gestion, à la

compilation et à l'exécution de votre projet. Comprendre leur rôle vous permettra de mieux configurer et optimiser votre application Angular.

Tableau 1–2 Description du fichier de configuration

Nom du fichier	Description
`package.json`	Sert de carte d'identité pour le projet. Il contient des informations telles que le nom du projet, la version, l'auteur, et détaille toutes les dépendances nécessaires pour exécuter et développer l'application.
`tsconfig.json`	Contient les réglages pour le compilateur TypeScript, spécifiant comment le code TypeScript est converti en JavaScript. Il inclut des options pour le contrôle de la sortie du compilateur, la configuration des chemins d'accès, etc.
`angular.json`	Configure les aspects de l'outil CLI d'Angular. Il définit comment l'application est construite, servie et déployée. Il inclut des configurations pour les environnements de développement et de production, ainsi que pour l'intégration de bibliothèques tierces.

Ajouter une bibliothèque CSS sur angular.json

Pour intégrer une bibliothèque CSS externe, telle que Bootstrap, dans votre projet Angular, vous devez modifier le fichier `angular.json`. Voici un exemple montrant comment ajouter Bootstrap :

1. Assurez-vous d'abord d'avoir installé Bootstrap dans votre projet via `npm` ou `yarn`.
2. Ouvrez le fichier `angular.json` dans votre éditeur de texte ou IDE.
3. Localisez la section `styles` dans les configurations de `build`. Cela se trouve généralement sous `projects>[nom_de_votre_projet]>architect>build>options`.
4. Ajoutez le chemin d'accès au fichier CSS de Bootstrap (habituellement dans `bootstrap/dist/css/bootstrap.min.css`) dans le tableau des `styles`.

```
"styles": [
 "src/styles.css",
 "bootstrap/dist/css/bootstrap.min.css" // Ajoutez cette ligne
],
```

En suivant ces étapes, Bootstrap sera inclus dans les builds de votre application, vous permettant d'utiliser ses classes CSS directement dans vos composants Angular.

Structurer l'application

Globalement, une application Angular est structurée de la manière suivante :

- **app** : contient les composants, services et autres fichiers de logique spécifiques à votre application.

Pour la création de l'application tout au long de cet ouvrage, voici une représentation visuelle de la structure de l'application :

Structure de l'application e-commerce

```
- src
 - app
    - core
      - guards
      - interceptors
      - interfaces
      - services
    - features
      - comments
      - navbar
      - products
    - pages
      - admin
      - home
      - login
      - not-found
      - product
 - shared
    - components
    - directives
    - pipes
    - shared.module.ts
 - app.component.css
 - app.component.spec.ts
 - app.component.ts
 - app.config.ts
 - app.routes.ts
```

Cette structure est choisie pour plusieurs raisons :

1. **Cohérence et modularité** : chaque partie de l'application est regroupée logiquement, ce qui facilite la compréhension et la maintenance du code. Par exemple, tout ce qui est partagé entre les différents modules de l'application, comme les composants réutilisables ou les pipes, est placé dans le dossier `shared`.
2. **Séparation des préoccupations** : les dossiers comme `core` et `features` permettent de séparer les aspects globaux de l'application (services centraux, intercepteurs, guards) des fonctionnalités spécifiques (commentaires, barre de navigation, produits), garantissant ainsi une séparation claire des préoccupations.
3. **Scalabilité** : cette structure facilite l'ajout de nouvelles fonctionnalités ou pages à l'application. Chaque nouvelle fonctionnalité peut être ajoutée comme un sous-dossier dans `features`, et chaque nouvelle page peut être placée dans `pages` sans perturber l'organisation existante.

4. **Maintenabilité** : avec des préoccupations bien séparées et une organisation logique, il devient plus facile pour les développeurs de naviguer dans le projet, d'identifier où des modifications doivent être faites, et d'assurer la maintenance de l'application à mesure qu'elle grandit.
5. **Testabilité** : une telle structure favorise également la testabilité. Les tests unitaires peuvent être plus facilement écrits et maintenus lorsque le code est bien organisé et que les dépendances sont clairement définies.

CONSEIL **Bonnes pratiques**

- **L**ocate : trouver rapidement le code.
- **I**dentify : comprendre rapidement le but du code. Un fichier doit contenir un seul concept et un dossier doit contenir des fichiers de même nature.
- **F**lattest : avoir le moins de fichiers possible.
- **T**ry to be DRY : ne pas répéter le code.

Configurer les environnements

La gestion des environnements de travail est un moyen de séparer les configurations utilisées pendant le développement, le test et la production. Ceci permet non seulement de maintenir l'ordre et la clarté dans votre code, mais aussi de sécuriser et d'optimiser votre application selon l'environnement ciblé.

Lancez la commande :

Création des environnements

```
ng generate environments
```

Angular crée par défaut deux fichiers d'environnement dans le dossier `src/environments` :

- `environment.development.ts` pour le développement ;
- `environment.ts` pour la production.

Ces fichiers contiennent des variables qui peuvent être utilisées dans votre application pour adapter son comportement selon l'environnement. Par exemple, vous pouvez avoir une URL d'API différente pour le développement et la production.

Exemple de fichier environment.development.ts

```
export const environment = {
 production: false,
 apiURL: 'http://localhost:3000/api'
};
```

Exemple de fichier environment.ts

```
export const environment = {
 production: true,
 apiURL: 'https://monapi.production/'
};
```

Pour utiliser ces variables dans votre application, vous pouvez les importer et les utiliser comme suit :

Utiliser une variable d'environnement

```
import { environment } from './environments/environment';

console.log(environment.apiURL); // Utilise l'URL selon l'environnement
```

Selon l'environnement de build, Angular remplace automatiquement le fichier `environment.ts` par `environment.*.ts`. Par exemple, lors du développement, Angular utilise `environment.development.ts`.

> CONSEIL **Choisir l'environnement de build**
>
> Vous pouvez construire votre application pour un environnement spécifique en utilisant la commande `ng build --configuration=production`.

Débogage d'une application Angular

Le débogage est une compétence essentielle pour tout développeur, car il permet de trouver et de corriger les erreurs dans le code. Dès que vous commencez à travailler sur des applications Angular, vous rencontrerez des problèmes qui nécessiteront un débogage. Voici quelques conseils pour déboguer efficacement une application Angular.

Dans le navigateur

1. **Utilisez les outils de développement** : la plupart des navigateurs modernes, comme Chrome ou Firefox, ont d'excellents outils de développement intégrés. Vous pouvez les ouvrir en effectuant un clic droit sur la page et en sélectionnant *Inspecter* ou en utilisant le raccourci `Ctrl+Shift+I`.
2. **Console** : utilisez la console pour voir les logs, les erreurs et les avertissements. Cela peut souvent vous donner des indications sur ce qui ne va pas dans votre application.
3. **Sources** : l'onglet *Sources* vous permet de voir les fichiers sources de votre application. Vous pouvez mettre des points d'arrêt directement dans le code TypeScript grâce à la cartographie de source *(source maps)* qui est généralement activée par défaut dans les projets Angular.

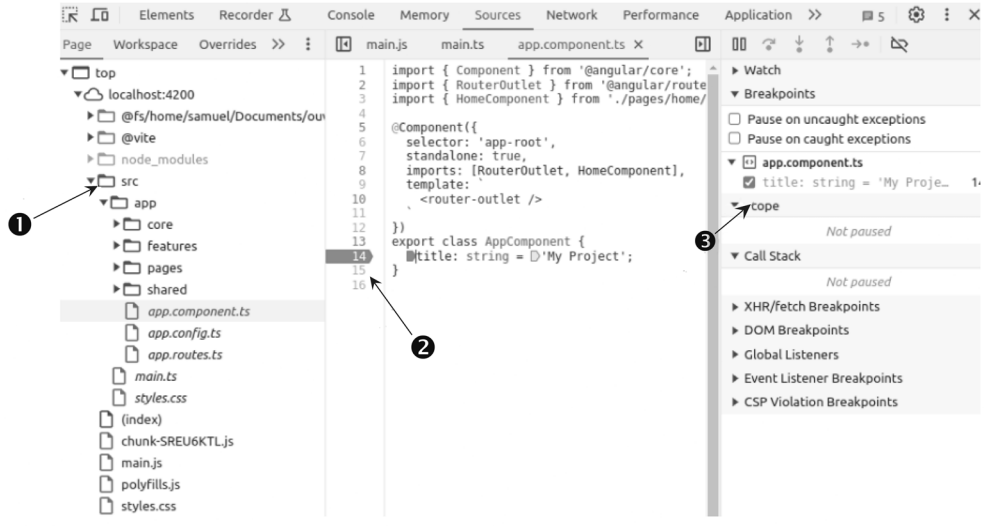


Figure 1–6 Mettre un point d'arrêt

1. Cliquez sur le dossier `src`.
2. Sélectionnez le fichier pour lequel vous voulez mettre un point d'arrêt et cliquez sur une ligne. Rafraîchissez la page pour que la modification soit prise en compte.
3. Vous pouvez maintenant voir les variables locales, les appels de fonctions, et d'autres informations utiles dans la fenêtre de débogage.

4. **Network** : l'onglet *Network* peut être très utile pour déboguer les problèmes liés aux requêtes HTTP.

Avec VS Code

1. **Extensions de débogage** : assurez-vous d'avoir installé l'extension nécessaire pour le débogage de TypeScript dans VS Code, comme Debugger for Chrome ou Debugger for Firefox.
2. **Configuration du débogueur** : dans VS Code, allez à la vue *Run and Debug* et créez un fichier de configuration de débogage (généralement `launch.json`) en sélectionnant votre environnement de débogage (Chrome ou Firefox).
3. **Lancement du débogage** : placez des points d'arrêt dans votre code TypeScript, puis lancez la session de débogage. VS Code devrait automatiquement ouvrir le navigateur et s'arrêter aux points d'arrêt que vous avez définis.

Tester votre application est une étape cruciale du développement logiciel. Angular fournit un cadre de test intégré qui utilise Karma comme *test runner* et Jasmine comme framework pour écrire les tests. Voici comment vous pouvez démarrer avec les tests dans Angular.

> **CONSEIL Prérequis**
>
> Assurez-vous que **Google Chrome** est installé sur votre machine car Karma l'utilise par défaut pour exécuter les tests.

Lancer les tests unitaires

1. **Ouvrez le terminal** dans VS Code ou votre éditeur de code préféré.
2. **Exécutez la commande de test** : dans le terminal, tapez `ng test`. Cette commande va démarrer Karma, qui à son tour lance le navigateur Chrome et exécute les tests écrits dans votre application.
3. **Rapport de tests** : une fois lancé, Karma ouvrira une nouvelle fenêtre de navigateur et exécutera les tests. Karma surveille également vos fichiers de tests pour tout changement, exécutant à nouveau les tests automatiquement dès qu'un fichier est modifié.
4. **Console de résultats** : les résultats des tests seront affichés dans le terminal. Vous verrez les tests qui ont réussi, ceux qui ont échoué et des informations sur les erreurs éventuelles.

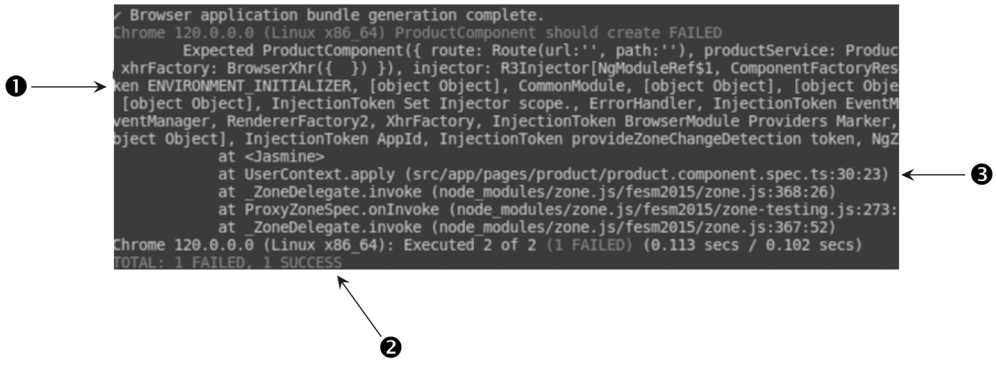


Figure 1–7 Échec des tests unitaires

1. Indication de l'erreur : « Pourquoi le test a échoué ».
2. Nombre de tests réussis/échoués.
3. Le chemin vers le fichier de test.

Mise à jour d'un projet Angular

Puisqu'Angular est en constante évolution, il est important de garder votre projet à jour pour bénéficier des dernières fonctionnalités, des correctifs de sécurité et des améliorations de performance. Les développeurs Angular maintiennent les 3 dernières versions du framework. Assurez-vous de toujours utiliser une version la plus à jour possible.

1. **Backup** : avant de commencer, assurez-vous de faire une sauvegarde de votre projet actuel ou vérifez que tout est correctement configuré dans votre système de contrôle de version.
2. **Documentation** : consultez le *changelog* d'Angular pour connaître les modifications importantes entre les versions.
3. **Angular Update Guide** : utilisez l'*Angular Update Guide* pour obtenir des instructions personnalisées pour la mise à jour de votre version spécifique.
4. **Mettre à jour Angular CLI** : dans le terminal, exécutez `npm install -g @angular/cli@latest` pour mettre à jour Angular CLI globalement.
5. **Mettre à jour les paquets** : exécutez `ng update` pour voir les paquets qui nécessitent une mise à jour. Ensuite, utilisez `ng update [package_name]` pour les mettre à jour. Faites-le pour tous les paquets qui en ont besoin.
6. **Corrections automatisées** : Angular CLI peut également exécuter des scripts de migrations qui modifieront automatiquement votre code pour qu'il soit compatible avec les nouvelles versions.
7. **Testez votre application** : après la mise à jour, exécutez tous vos tests et vérifiez manuellement le fonctionnement de votre application pour vous assurer que rien n'a été cassé.

> DANGER **Attention aux breaking changes**
>
> Les mises à jour majeures peuvent introduire des *breaking changes*. Vérifiez toujours les notes de mise à jour pour comprendre les impacts potentiels sur votre projet.

Mise en production

1. **Build de production** : utilisez la commande `ng build`. Cela va créer une version optimisée de votre application dans le dossier `dist/`.
2. **Optimisations** : le mode de production inclut des optimisations comme l'AOT *(Ahead-of-Time Compilation)*, l'arbre-shaking (pour éliminer le code inutilisé) et la minification.

> CONSEIL **AOT Compilation**
>
> L'AOT convertit le code TypeScript et les modèles HTML en JavaScript optimisé lors de la construction, ce qui améliore les performances de l'application.

3. **Serveur web** : déployez le contenu du dossier `dist/` sur votre serveur web. Cela peut être un serveur dédié, un service de cloud comme AWS, Firebase, ou une plate-forme de déploiement comme Netlify ou Vercel.
4. **Tests post-déploiement** : une fois votre application déployée, effectuez des tests pour vous assurer qu'elle fonctionne comme prévu.
5. **Surveillance et performances** : mettez en place une surveillance pour suivre les erreurs en production et utilisez des outils pour surveiller les performances de l'application.

INFO CDN et optimisations réseau

Considérez l'utilisation d'un CDN pour distribuer vos fichiers statiques et mettre en cache les contenus pour améliorer les temps de chargement pour les utilisateurs du monde entier.

Lignes de commande utiles dans Angular

Voici un tableau récapitulatif des commandes Angular courantes et de leur utilité, incluant quelques options souvent utilisées :

Tableau 1–3 Commandes courantes

Commande	Description
`ng new <name>`	Crée une nouvelle application Angular.
`ng serve`	Construit l'application et lance un serveur web.
`ng build`	Compile l'application dans le répertoire `dist/`.
`ng test`	Exécute les tests unitaires via Karma.
`ng e2e`	Exécute les tests end-to-end.
`ng generate component <name>`	Génère un nouveau composant.
`ng generate service <name>`	Génère un nouveau service.
`ng generate module <name>`	Génère un nouveau module.
`ng update`	Met à jour les dépendances de votre projet Angular.
`ng lint`	Analyse le code de votre application pour des erreurs potentielles.

DETAILS Options courantes

- `--port` (avec `ng serve`) : spécifie un port pour le serveur de développement.
- `--open` ou `-o` (avec `ng serve`) : ouvre automatiquement le navigateur après avoir lancé le serveur.
- `--include` (avec `ng test`) : inclut des tests spécifiques dans l'exécution.
- `--skip-tests` (avec `ng generate`) : génère un composant sans fichiers de test.
- `--dry-run` (avec n'importe quelle commande `ng generate`) : affiche les fichiers qui seront créés ou modifiés lors d'une commande de génération sans vraiment appliquer les changements.