FREE eBook

# LEARNING angular-cli

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#angular-cli

# Table of Contents

# About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: angular-cli

It is an unofficial and free angular-cli ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official angular-cli.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapter 1: Getting started with angular-cli

## Remarks

This section provides an overview of what angular-cli is, and why a developer might want to use it.

It should also mention any large subjects within angular-cli, and link out to the related topics. Since the Documentation for angular-cli is new, you may need to create initial versions of those related topics.

## Examples

### Installation or Setup

**Note:** Angular CLI versions are under rapid development. This documentation targets for the latest version.

---

**Prerequisites**

To execute and work with the latest version, Angular CLI and the project have dependencies that require node v6.9.0 or higher.

---

**Setup**

Make sure that a node version is installed which is compatible with the CLI

Install the Angular CLI globally. It installs the latest version.

`npm i @angular/cli -g` or `yarn global add @angular/cli`, depending on the package manager in use.

---

`ng help` command will provide the executable commands

### Generating and serving an Angular project via a development server

To create a new project `ng new [project-name]` which initializes git.Install packages for tooling via npm. It creates the project successfully.

`cd [project-name]` and execute either `npm start` or `ng serve`

It'll run the server in the given default port.

Application will refresh according to the changes made in the directory. The default HTTP port and the one used by the LiveReload server can be changed using below command

```
ng serve --host 0.0.0.0 --port 4201 --live-reload-port 49153
```

## Angular CLI - The Basic Steps

1. You will need to install node.js - https://nodejs.org/en/
2. npm install -g @angular/cli - install the CLI by executing this command in the terminal
3. ng new projectname - after executing this command in the terminal, you will create a new sub folder titled projectname in your current folder.
4. cd projectname - go to the sub folder, where your project is
5. ng serve - run the project. It will be available at http://localhost:4200

Read Getting started with angular-cli online: https://riptutorial.com/angular-cli/topic/9275/getting-started-with-angular-cli

# Chapter 2: angular-cli project deployment on apache tomcat 8.0.14 server

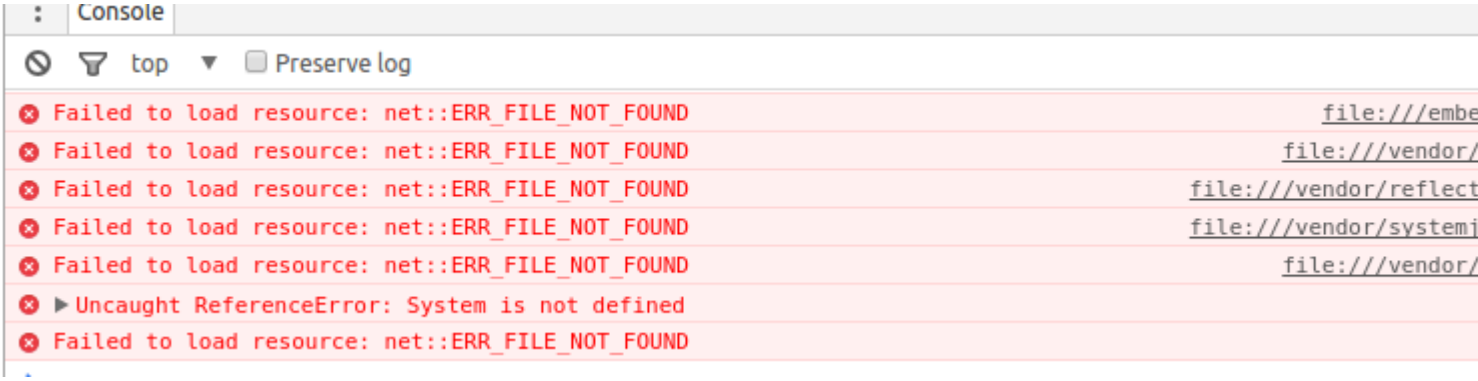
## Introduction

This topic would cover how angular-cli project is ready for production build, what all necessary steps taken before deploying, how to create war file for project deployment and finally how to configure the apache tomcat for angular-cli project deployment.

## Examples

### Neccessary steps taken before deploying the angular-cli project for production build.

Before deploying the angular project in server we need to build angular project for production. We also need to change the routing path in index.html file from `<base href="/">` to `<base href="./">` if it is not done then your project wouldn't get loaded properly there will be some routing error saying 404 file not found.




### Angular-cli build command to build project bundle for production deployment

```
ng build –prod
```

Above given command with extra option like **–prod** would generate the production build project bundle. Once the above command gets executed in the root directory of your project would appear a directory called **dist**. In which all the production build bundle of your project appears in it.

### Creating the war file for production deployment of angular-cli project on apache tomcat server

Once the **dist** directory is ready with your production built bundles. Just open the **dist** directory and open the command prompt type the following command to create the **war** file to deploy your project on apache tomcat server.

```
jar cvf dist.war .
```

Once the above jar commands gets executed. It would generate a **dist.war** file within the **dist** directory.

## Configuring the apache tomcat for angular-cli project deployment.

1. Cut/Copy the **dist.war** file from **dist** directory and place it in apache tomcat **webapp** directory.
2. Go to apache tomcat **bin** folder and double click on **startup.bat** file.
3. Now tomcat server will execute **dist.war** file and startup the tomcat **catalina** server.
4. Once the tomcat **catalina** server gets started open web browser and type the **localhost:8080/dist** and tap on enter key your project gets executed on the web browser window.

Read angular-cli project deployment on apache tomcat 8.0.14 server online: https://riptutorial.com/angular-cli/topic/9780/angular-cli-project-deployment-on-apache-tomcat-8-0-14-server

# Chapter 3: Working with angular-cli: Generating components, directives, pipes, services, etc.

## Introduction

The angular-cli tool can help you to scaffold different parts of an angular application (components, directives, pipes, services, classes, guards, interfaces, enums and modules).

## Syntax

- ng generate [component | directive | service | pipe | class | enum | interface | guard | module] [name] [flags...]
- ng g [c | d | s | p | cl | e | i | g | m] [name] [flags...]

## Parameters

| Parameter | Description |
|---|---|
| `component` or `c` | Used to generate component |
| `directive` or `d` | Used to generate directives |
| `service` or `s` | Used to generate services |
| `pipe` or `p` | Used to generate pipes |
| `class` or `cl` | Used to generate classes |
| `enum` or `e` | Used to generate enums |
| `interfaces` or `i` | Used to generate interfaces |
| `guard` or `g` | Used to generate guards |
| `module` or `m` | Used to generate modules |
| `--flat` or `-f` | Used to enable/disable directory creation |
| `--inline-template` or `-it` | Used to enable/disable inline html templates in components |
| `--inline-style` or `-is` | Used to enable/disable inline styles in components |
| `--prefix` or `-p` | Used to disable or change prefix |

| Parameter | Description |
|---|---|
| `--spec` or `-s` | Used to enable/disable `.spec` files creation |
| `--skip-import` | Used to skip the module import |
| `--app` or `-a` | Used to specify app name to use |
| `--module` or `-m` | Used to specify the declaring module |
| `--view-encapsulation` or `-ve` | Used to specify the view encapsulation strategy in components |
| `--change-detection` or `-cd` | Used to specify the change detection strategy in components |
| `--routing` or `-r` | Used to specify if routing module file should be generated |

# Examples

### "Generate command" usage

You can use the `ng generate` or `ng g` command to generate Angular building blocks (components, services, pipes, etc.).

You can find all possible **blueprints** in the table below:

| Scaffold | Usage | Shortened |
|---|---|---|
| Component | `ng generate component component-name` | `ng g c component-name` |
| Directive | `ng generate directive directive-name` | `ng g d directive-name` |
| Pipe | `ng generate pipe pipe-name` | `ng g p pipe-name` |
| Service | `ng generate service service-name` | `ng g s service-name` |
| Class | `ng generate class class-name` | `ng g cl class-name` |
| Guard | `ng generate guard guard-name` | `ng g g guard-name` |
| Interface | `ng generate interface interface-name` | `ng g i interface-name` |
| Enum | `ng generate enum enum-name` | `ng g e enum-name` |
| Module | `ng generate module module-name` | `ng g m module-name` |

So, for example, if you run `ng generate component user-list` - angular-cli will:

- create `user-list` directory in `src/app` folder or folder where you have run the command.
- inside that directory generate 4 files (`user-list.component.ts`, `user-list.component.html`, `user-`

`list.component.css` and `user-list.component.spec.ts`)

- add `user-list` as a declaration in the `@NgModule` decorator of the nearest module.

## Generating components

To add a component with a selector `[prefix]-user-list`, run:

```
  $ ng g c user-list

  installing component
    create src/app/user-list/user-list.component.css
    create src/app/user-list/user-list.component.html
    create src/app/user-list/user-list.component.spec.ts
    create src/app/user-list/user-list.component.ts
    update src/app/app.module.ts
```

**prefix** prevents element name collisions with components in other apps and with native HTML elements. So, for example, if prefix is `app` - generated component will have `app-user-list` selector.

- To prevent prefix usage add `--prefix false` or `-p false` flag

```
  $ ng g c user-list --prefix false
```

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

@Component({
  selector: 'user-list',
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.css']
})
export class UserListComponent {}
```

- To prevent `.spec` files creation add `--spec false` or `-sp false` flag

```
  $ ng g c user-list --spec false

  installing component
    create src/app/user-list/user-list.component.css
    create src/app/user-list/user-list.component.html
    create src/app/user-list/user-list.component.ts
    update src/app/app.module.ts
```

- To use inline html templates instead of external templates add `--inline-template` or `-it` flag

```
  $ ng g c user-list --inline-template

  installing component
    create src/app/user-list/user-list.component.css
    create src/app/user-list/user-list.component.spec.ts
    create src/app/user-list/user-list.component.ts
    update src/app/app.module.ts
```

- To use inline styles instead of external styles add `--inline-style` or `-is` flag

```
$ ng g c user-list --inline-style

installing component
  create src/app/user-list/user-list.component.html
  create src/app/user-list/user-list.component.spec.ts
  create src/app/user-list/user-list.component.ts
  update src/app/app.module.ts
```

- To prevent folder creation add `--flat` or `-f` flag

```
$ ng g c user-list --flat

installing component
  create src/app/user-list.component.css
  create src/app/user-list.component.html
  create src/app/user-list.component.spec.ts
  create src/app/user-list.component.ts
  update src/app/app.module.ts
```

You can also combine flags listed above. For example, to create only `.component.ts` file without `.css`, `.html`, `.spec` files and folder use the following command.

```
$ ng g c user-list -f -it -is -sp false

installing component
  create src/app/user-list.component.ts
  update src/app/app.module.ts
```

All `generate component` flags:

| Description | Flag | Shortened | Default Value |
|---|---|---|---|
| Prevent folder creation | `--flat` | `-f` | `false` |
| Prevent prefix usage | `--prefix false` | `-p false` | `true` |
| Prevent `.spec` files creation | `--spec false` | `-sp false` | `true` |
| Enable inline html templates | `--inline-template` | `-it` | `false` |
| Enable inline styles | `--inline-style` | `-is` | `false` |

## Generating directives

To add a directive with a selector `[prefix]Highlight`, run:

```
$ ng g d highlight

installing directive
  create src/app/highlight.directive.spec.ts
```

```
    create src/app/highlight.directive.ts
    update src/app/app.module.ts
```

- To prevent prefix usage add `--prefix false` or `-p false` flag

```
  $ ng g d highlight --prefix false
```

```
import { Directive } from '@angular/core';

@Directive({
  selector: '[highlight]'
})
export class HighlightDirective {}
```

- To prevent `.spec` files creation add `--spec false` or `-sp false` flag

```
  $ ng g d highlight --spec false

  installing directive
    create src/app/highlight.directive.ts
    update src/app/app.module.ts
```

- To enable folder creation add `--flat false` or `-f false` flag

```
  $ ng g d highlight --flat false

  installing directive
    create src/app/highlight/highlight.directive.spec.ts
    create src/app/highlight/highlight.directive.ts
    update src/app/app.module.ts
```

You can also combine flags listed above. For example, to create only `highlight.directive.ts` file inside `highlight` folder without `.spec` file use the following command.

```
  $ ng g d highlight -f false -sp false

  installing directive
    create src/app/highlight/highlight.directive.ts
    update src/app/app.module.ts
```

All `generate directive` flags:

| Description | Flag | Shortened | Default Value |
|---|---|---|---|
| Enable folder creation | `--flat false` | `-f false` | `true` |
| Prevent prefix usage | `--prefix false` | `-p false` | `true` |
| Prevent `.spec` files creation | `--spec false` | `-sp false` | `true` |

## Generating services

To add a service with a name `UserService`, run:

```
$ ng g s user

installing service
  create src/app/user.service.spec.ts
  create src/app/user.service.ts
```

- To prevent `.spec` files creation add `--spec false` or `-sp false` flag

```
$ ng g s user --spec false

installing service
  create src/app/user.service.ts
```

- To enable folder creation add `--flat false` or `-f false` flag

```
$ ng g s user --flat false

installing service
  create src/app/user/user.service.spec.ts
  create src/app/user/user.service.ts
```

You can also combine flags listed above. For example, to create only `user.service.ts` file inside `user` folder without `.spec` file use the following command.

```
$ ng g s user -f false -sp false

installing service
  create src/app/user/user.service.ts
```

All `generate service` flags:

| Description | Flag | Shortened | Default Value |
|---|---|---|---|
| Enable folder creation | `--flat false` | `-f false` | `true` |
| Prevent `.spec` files creation | `--spec false` | `-sp false` | `true` |

## Generating pipes

To add a pipe with a name `searchByName`, run:

```
$ ng g p search-by-name

installing pipe
  create src/app/search-by-name.pipe.spec.ts
  create src/app/search-by-name.pipe.ts
  update src/app/app.module.ts
```

- To prevent `.spec` files creation add `--spec false` or `-sp false` flag

```
$ ng g p search-by-name --spec false

installing pipe
  create src/app/search-by-name.pipe.ts
  update src/app/app.module.ts
```

- To enable folder creation add `--flat false` or `-f false` flag

```
$ ng g p search-by-name --flat false

installing pipe
  create src/app/search-by-name/search-by-name.pipe.spec.ts
  create src/app/search-by-name/search-by-name.pipe.ts
  update src/app/app.module.ts
```

You can also combine flags listed above. For example, to create only `search-by-name.pipe.ts` file inside folder `search-by-name` folder without `.spec` file use the following command.

```
$ ng g p search-by-name -f false -sp false

installing pipe
  create src/app/search-by-name/search-by-name.pipe.ts
  update src/app/app.module.ts
```

All `generate pipe` flags:

| Description | Flag | Shortened | Default Value |
|---|---|---|---|
| Enable folder creation | `--flat false` | `-f false` | `true` |
| Prevent `.spec` files creation | `--spec false` | `-sp false` | `true` |

## Generating modules

To add a module called `GuestModule`, run:

```
$ ng g m guest

installing module
  create src/app/guest/guest.module.ts
```

- To enable `.spec` files creation add `--spec` or `-sp` flag

```
$ ng g m guest --spec

installing module
  create src/app/guest/guest.module.spec.ts
  create src/app/guest/guest.module.ts
```

- To enable routing add `--routing` or `-r` flag

```
$ ng g m guest --routing

installing module
  create src/app/guest/guest-routing.module.ts
  create src/app/guest/guest.module.ts
```

You can also combine flags listed above. For example, to create module with routing and specs use the following command.

```
$ ng g m guest -sp -r

installing module
  create src/app/guest/guest-routing.module.ts
  create src/app/guest/guest.module.spec.ts
  create src/app/guest/guest.module.ts
```

All `generate module` flags:

| Description | Flag | Shortened | Default Value |
|---|---|---|---|
| Enable `.spec` files creation | `--spec` | `-sp` | `false` |
| Enable routing | `--routing` | `-r` | `false` |

Read Working with angular-cli: Generating components, directives, pipes, services, etc. online: https://riptutorial.com/angular-cli/topic/9482/working-with-angular-cli--generating-components--directives--pipes--services--etc-

# Credits

| S. No | Chapters | Contributors |
|---|---|---|
| 1 | Getting started with angular-cli | BogdanC, Community, Daedalon, Kaloyan, Saiyaff Farouk |
| 2 | angular-cli project deployment on apache tomcat 8.0.14 server | Gurudath G |
| 3 | Working with angular-cli: Generating components, directives, pipes, services, etc. | Ketan Akbari, Saka7 |