academeez

@angular/cli - advanced

7. Creating a library

Library

academeez

- A library can contain logic that is used in many applications.
- It’s logic is not binded to one app
- A library can be version controlled when used in many apps
- A library can contain anything you want
  - Angular stuff: components, services, pipes, directives, modules, etc.
  - Styles
  - General typescript like classes and functions

Creating a library - EX

- You can create a library using the following command:
  - npx ng g library @nz/utils
  - It’s highly recommended to scope your libraries **@scope/package-name**

tsconfig paths

- In the **tsconfig.json** of the root of the workspace, when you generated a new library a new **paths** configuration was added to **compilerOptions**
- The paths configuration will map an import statement to a custom location

```
"paths": {
  "name-of-lib": [
    "dist/name-of-lib/name-of-lib",
    "dist/name-of-lib"
  ]
}
```

paths - ex

academeez

- This configuration can be translated to this:
  - If I find an import to **name-of-lib** I will resolve that import with this array
- During development of the library might be easier to point to the **projects/name-of-lib/src/public-api.ts**

```
"paths": {
  "name-of-lib": [
    "dist/name-of-lib/name-of-lib",
    "dist/name-of-lib"
  ]
}
```

library ex

- After changing the **paths** in the **tsconfig.base.json** create a component in the library and use that component in the project you created.
- Create another project and use the same component in another project
- The component in the library should just display text

Building a library

- You can build your library with the following command:
  - npx ng build @nz/utils —prod

publishing a library

academeez

- You can publish your library to npm by typing:
  - cd dist/nz/utils
  - npm publish
- Note that scoped packages will require a private npm of your own or paying npm for a private registry.

Manage dependencies - ex

academeez

- You now have 2 projects using a component from a shared library.
- It is recommended to use the trick with **paths** pointing to the library src code only while developing the library
- Once the library is shared by multiple projects, it is recommended to use NPM to manage the versions of the library
- Build the library you created
- Try to push the library to NPM

Manage dependencies - ex

academeez

- After pushing your package, the projects will start using the package from npm and not from the src code of the library.
- You can modify the **paths** in **tsconfig.base.json** to the following:
- Now you can leverage both npm and also use the src code if you want the nightly

```
"paths": {
  "name-of-lib@dev": [
    "./projects/name-of-lib/src/public-api.ts"
  ]
}
```

academeez

Thank You

Congratulations! You are now an @angular/cli expert