



Angular Advanced Using multiple Modules



Peter Kassenaar
info@kassenaar.com

Why devs and enterprises like Angular...



CLI



Router



HTTP



Forms



Animations



i18n



Testing



Language
Services



Universal



Material &
CDK

Generic 'Advanced' Github repo

The screenshot shows the GitHub repository page for **PeterKassenaar / AngularAdvanced**. The repository is described as "Labs, exercises and example code on the training Angular Advanced by Peter Kassenaar, info@kassenaar.com" with a link to <https://www.angulartraining.nl/>. It has 135 commits, 1 branch, 0 releases, and 2 contributors. The latest commit by PeterKassenaar is "Updated examples to Angular V8" from 2 days ago. The file list includes `examples`, `.angulardoc.json`, `.gitignore`, and `README.md`.

Search or jump to... Pull requests Issues Marketplace Explore

PeterKassenaar / AngularAdvanced Unwatch 1 Star 3 Fork 8

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Labs, exercises and example code on the training Angular Advanced by Peter Kassenaar, info@kassenaar.com <https://www.angulartraining.nl/> Edit

angular training Manage topics

135 commits 1 branch 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

PeterKassenaar Updated examples to Angular V8 Latest commit fe74999 2 days ago

examples	Updated examples to Angular V8	2 days ago
.angulardoc.json	Updated angular.json	last year
.gitignore	Update .gitignore	2 years ago
README.md	Update README.md	27 days ago

README.md

<https://github.com/PeterKassenaar/AngularAdvanced>



Multiple modules

Splitting your application into separate, reusable modules

Default application – 1 module

The image displays the Angular CLI web interface on the left and a file explorer window on the right, illustrating the default application structure.

Angular CLI Interface:

- Welcome** header with a Twitter icon.
- multiple-modules app is running!** notification banner.
- Resources** section: "Here are some links to help you get started:"
 - [Learn Angular >](#)
 - [CLI Documentation >](#)
 - [Angular CLI >](#)
- Next Steps** section: "What do you want to do next with your app?"
 - [+ New Component](#)
 - [+ Angular Material](#)
 - [+ Add Dependency](#)
 - [+ Build for Production](#)
- Terminal** snippet: `ng generate component xyz`
- Footer**: "Love Angular? Give our repo a star. ★ Star >"

File Explorer (customProject):

- Project** (C:\Users\Peter Kassenaar\Desktop\custo...)
- Project Files** view:
- customProject** folder:
 - e2e** folder
 - node_modules** library root
 - src** folder:
 - app** folder:
 - app.component.css
 - app.component.html
 - app.component.spec.ts
 - app.component.ts
 - app.module.ts
 - assets** folder
 - .gitkeep** file
 - environments** folder:
 - environment.prod.ts
 - environment.ts
 - favicon.ico
 - index.html
 - main.ts
 - polyfills.ts
 - styles.css
 - test.ts
 - tsconfig.app.json
 - tsconfig.spec.json
 - typings.d.ts
 - External Libraries** folder:
 - .angular-cli.json
 - .editorconfig
 - .gitignore
 - karma.conf.js
 - package.json
 - protractor.conf.js
 - README.md
 - tsconfig.json
 - tslint.json
 - yarn.lock

(228 MB)

Bigger applications – multiple modules

The image displays a screenshot of an Angular application project structure and code. On the left, a file explorer shows the project layout with folders like `src`, `app`, `home`, `layout`, and `training`. The `app` folder contains `app-routing.module.ts`, `app.component.css`, `app.component.html`, `app.component.ts`, `app.module.ts`, `assets`, `environments`, `favicon.ico`, `index.html`, `main.ts`, `polyfills.ts`, `styles.css`, `test.ts`, `tsconfig.app.json`, `tsconfig.spec.json`, `typings.d.ts`, `.angular-cli.json`, `.editorconfig`, `.gitignore`, `karma.conf.js`, `package.json`, `README.md`, `tsconfig.json`, and `tslint.json`.

The main area shows the `app.module.ts` file with the following code:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.module#HomeModule';
import { TrainingModule } from './training/training.module#TrainingModule';

const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: 'home', loadChildren: './home/home.module#HomeModule' },
  { path: 'training', loadChildren: './training/training.module#TrainingModule' },
];

const config: ExtraOptions = {
  enableTracing: true,
  preloadingStrategy: PreloadAllModules,
};

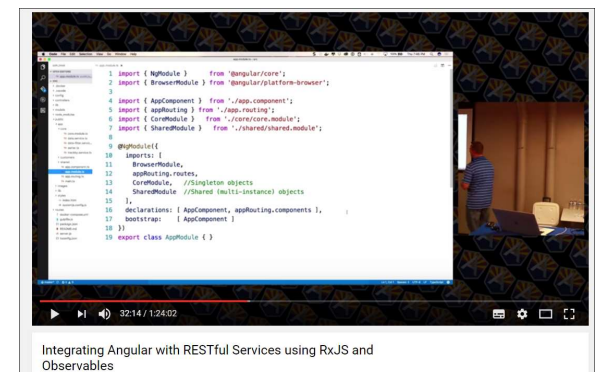
@NgModule({
  imports: [RouterModule.forRoot(routes, config)],
  exports: [RouterModule]
})
export class AppModule {}
```

Overlaid on the code is a red Angular logo with the text "World-class Angular training in Dutch and English" and "Live classrooms - focused on today's developers". Below this text are two buttons: "LEARN MORE" and "SIGN UP!".

On the right, a file explorer shows the `src` folder structure, including `app`, `admin`, `contact`, `core`, `dates`, `home`, `info`, `layout`, `shared`, `training`, `app.component.html`, `app.component.ts`, `app.module.ts`, `app-routing.module.ts`, `assets`, `environments`, `favicon.ico`, `index.html`, `main.ts`, `polyfills.ts`, `styles.css`, `test.ts`, `tsconfig.app.json`, `tsconfig.spec.json`, `typings.d.ts`, `.firebaserc`, and `.gitignore`.

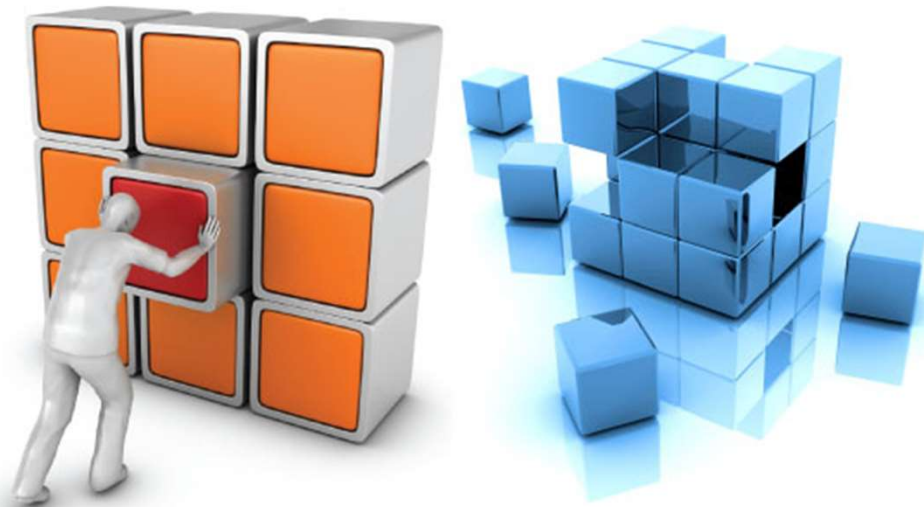
Angular Modules

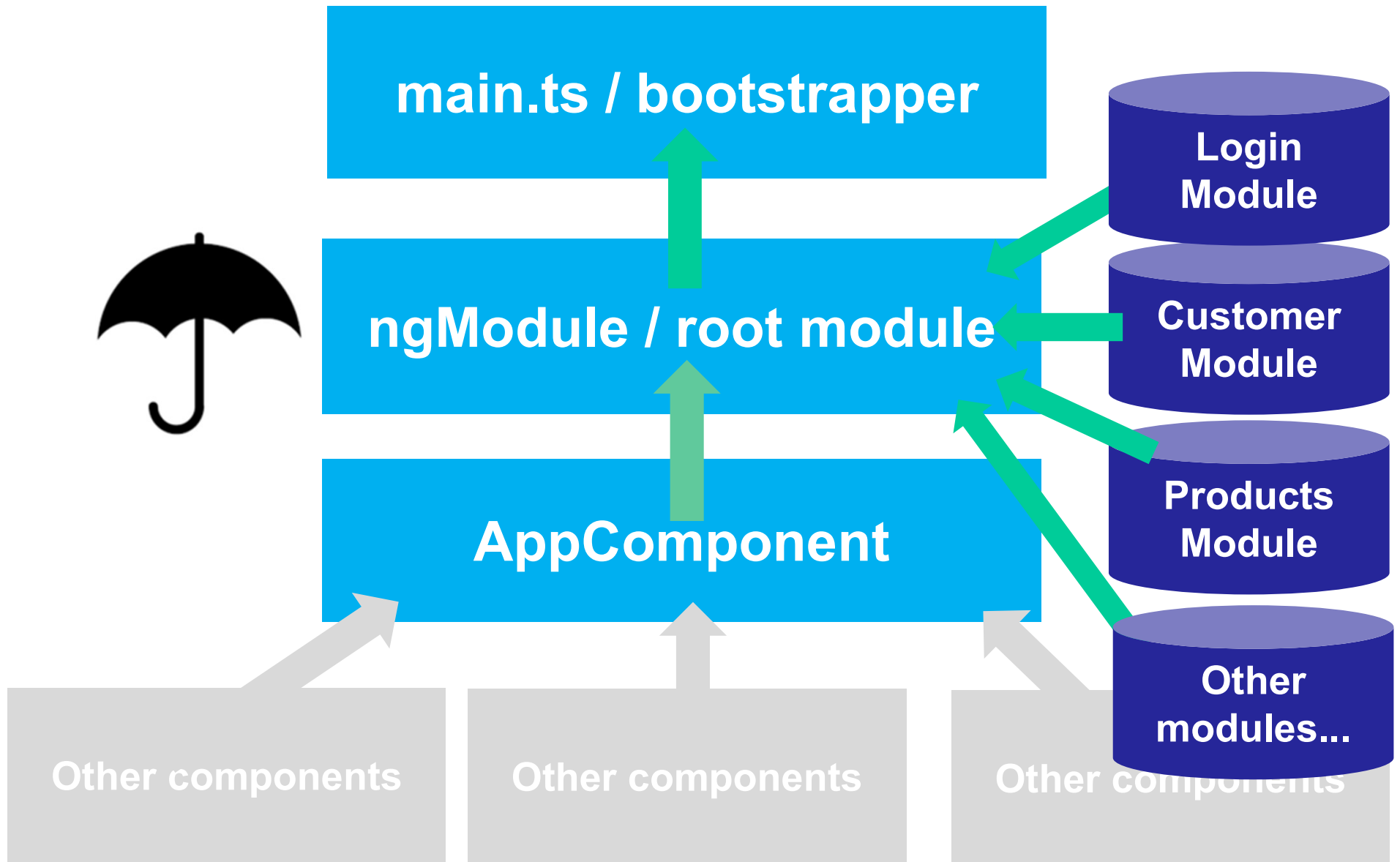
- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : **code organization**
- Use one AppModule - the root of your app
- Use one CoreModule - containing all *singletons* in your app
- Use one SharedModule - containing all shared resources, possible multiple instances
- Use additional modules *per feature*
- <https://www.youtube.com/watch?v=YxK4UW4UfCk>



Application – multiple Modules – why?

- *Reuse* of Components, Pipes, Routes and Services etc. over different apps
- *Wrap* each set of logical related components, services, etc. in its own module.





Steps

1. Create a new module

- Optional: test first with `--dry-run`
- `ng generate module customers --dry-run`

2. Create component(s) inside that module

- Again: test first with `--dry-run`
- `ng generate component customers --module customers --dry-run`

3. Apply UI, logic, etc. to your component

4. Export your component inside `customers.module.ts`

- `exports : [CustomersComponent],`
- Otherwise it can't be used in other components!

5. Provide new module to `app.module.ts`

- `imports: [CustomersModule]`

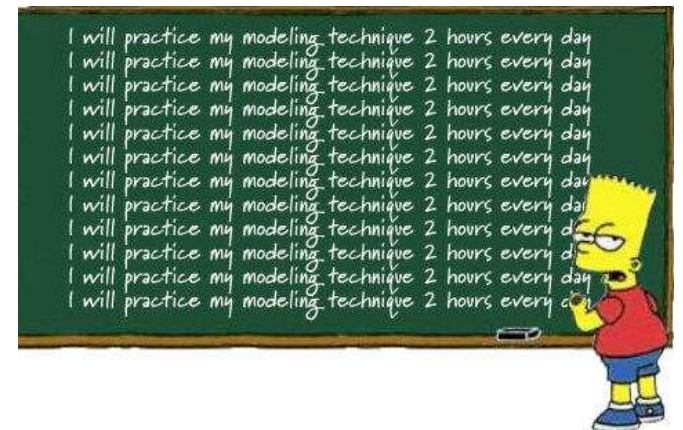
Optional : SharedModule

- Reuse components in multiple modules? Use a SharedModule
 - `ng g m shared` – shorthand notation
- Create components inside SharedModule
- Import SharedModule in other modules
- It doesn't have to be in AppModule if you don't use it directly!
- It *does* not add size to module bundles



Workshop

- Open ../100-multiple modules.
- Create a new module
- Create a new component inside this new module and give it some UI.
- Include the module in the Main Module and show it besides other modules
- Include the Search Component in your own module
- *OR:*
- Add Multiple Modules from scratch to your own application, using the steps described in this module.



How to structure feature modules



242



Why and how to structure Features in Modules in Angular

This might sound pretty basic, but I encounter these challenges over and over in customer projects and it's still an ongoing discussion internally.

A central project goal in a recent Angular project was to design features and UI components for reusability. To achieve this, we need to make sure our code is well isolated and has a simple and clear dependency model.

Prologue: Feature vs. Technical Project Structure

When building small apps and looking at common code samples in the internet a lot of devs (including myself) tend to come up with a project structure like this:

```
MYAPP
├── src
│   ├── app
│   │   ├── components
│   │   │   ├── home
│   │   │   │   ├── home.component.html
│   │   │   │   └── home.component.ts
│   │   │   └── user
│   │   │       └── user.component.html
```

<https://medium.com/@philippbauknecht/why-and-how-to-structure-features-in-modules-in-angular-d5602c6436be>