Angular topics

Labs

2025-03-05#fe2b300

Table of contents

Lab 0: Getting started

Setting up your environment

Training on Local system

You should install the following on your system:

- Node.js version LTS
- NPM (It will be installed at the same time as Node.js)
- Git
- IDE (e.g. Visual Studio Code)

Unzip the learning materials given by your trainer.

Training on Strigo VM

Strigo Lab provides a Windows VM with the following functional environment:

- Node.js
- NPM
- Git
- Visual Studio Code (`"C:\Programs Files\Microsoft VS Code"`)

Visual Studio Code Extensions

If you use VSCode as your IDE, install the following extensions in addition:

- Angular Language Service
- Auto import (optional)
- Auto Rename Tag (optional)
- Github Theme (optional)
- vscode-icons (optional)

Version control system

Note: to use "Git Credential Manager", you might need to restart the Windows VM once all the programs have been installed.

- Open the browser and login to your favorite cloud-based version control system (Github, Gitlab, ...)
- Remotely, create a new empty repository named `angular-topics` in which to save your code
- Locally, configure your Git name and email:

```
git config --global user.name "<YOUR_NAME>"
git config --global user.email <YOUR_EMAIL>
```

Creating and running your Angular application

This app will be used along all labs.

Install the Angular CLI globally and create your app with the shell commands

```
npm i -g @angular/cli
ng new angular-topics
```

You will be displayed some options for your app.

- Choose "SCSS" as style preprocessor
- Choose "No" for SSR/SSG/Prerendering

If you can't install the Angular CLI globally, create your app with one of the following shell commands

```
npm init @angular angular-topics
```

or:

```
npx @angular/cli new angular-topics
```

In this case, to run an Angular CLI command, you will have to use NPM first `npm run ng <command>` instead of just `ng <command>`.

Run the Angular dev server

```
ng serve
```

or:

```
npm start
```

Open the Chrome browser and visit: http://localhost:4200.

You should see the app with a placeholder content. 🚀

Synchronize your repository

Push your local repository from the command line over *HTTPS* (not SSH).

Here's an example for Github:

```
git remote add origin https://github.com/[YOUR_USERNAME]/angular-topics.git
git branch -M main
git push -u origin main
```

Lab 1: PrimeNG

`src/app/*`

- In your Angular app, replace the directory `src/app` with `Exercices/solutions/projects/00_vanilla/src`

`src/styles.scss`

- Import PrimeNG icons

```scss
@import 'primeicons/primeicons.css';
```

`src/app/app.config.ts`

- Add PrimeNG provider

```ts
import { ApplicationConfig } from '@angular/core';
import { provideAnimationsAsync } from '@angular/platform-browser/animations/async';
import Aura from '@primeng/themes/aura';
import { providePrimeNG } from 'primeng/config';

export const appConfig: ApplicationConfig = {
  providers: [
    provideAnimationsAsync(),

    providePrimeNG({
      theme: {
        preset: Aura,
        options: {
          darkModeSelector: false,
        },
      },
    }),
  ],
};
```

`NavComponent`

Use the `MenuModule` to display the navigation

```
import { MenuModule } from 'primeng/menu';
```

`UserPostsComponent`

- Use `CardModule` to display the user details
- Use `ButtonModule` to display post links
- Use `DialogModule` to display the post details

```
import { ButtonModule } from 'primeng/button';
import { CardModule } from 'primeng/card';
import { DialogModule } from 'primeng/dialog';
```

`UserDetailsComponent`

- Use PrimeNG icons to enhance the card's details

Lab 2: Transloco

- Run the schematics to add Transloco to your app

```
ng add @jsverse/transloco
```

- Define translations in the translation files

```
public/assets/i18n/*.json
```

- Use the Transloco directive to translate your app

```
import { TranslocoDirective } from '@jsverse/transloco';
```

- Use the Transloco pipe to localize the current date

```
import { TranslocoDatePipe } from '@jsverse/transloco-locale';
```

- Copy the `SelectLangComponent` to your app to enable switching between available languages

```
Exercices/solutions/projects/02_transloco/src/app/select-lang
```

Lab 3: NgRx signals

- Replace the `UserService` with a `UserStore` using `signalStore` function
- Replace the `UserPostsService` with a `UserPostsStore` using `signalStore` function

Lab 4: RxResource

- Remove the `UserStore` from previous lab
- Restore `UserService` and expose the users using `rxResource`

```
import { inject, Injectable } from '@angular/core';
import { rxResource } from '@angular/core/rxjs-interop';
import { ApiService } from './api/api.service';

@Injectable({
  providedIn: 'root',
})
export class UsersService {
  private apiService = inject(ApiService);

  users = rxResource({ loader: () => this.apiService.getUsers() });
}
```

- Remove the `UserPostsStore` from previous lab
- Handle the logic directly in `UserPostsComponent`

```
import {
  Component, computed, inject, input, signal, ViewEncapsulation
} from '@angular/core';
import { rxResource } from '@angular/core/rxjs-interop';
import { TranslocoDirective } from '@jsverse/transloco';
import { ButtonModule } from 'primeng/button';
import { CardModule } from 'primeng/card';
import { DialogModule } from 'primeng/dialog';
import { ApiService } from '../shared/api/api.service';
import { User } from '../shared/api/api.types';
import { UserDetailsComponent } from './user-details/user-details.component';

@Component({
  selector: 'app-user-posts',
  imports: [
    TranslocoDirective, ButtonModule, CardModule,
    DialogModule, UserDetailsComponent
  ],
  templateUrl: './user-posts.component.html',
  encapsulation: ViewEncapsulation.None,
})
export class UserPostsComponent {
  protected apiService = inject(ApiService);

  user = input.required<User>();

  posts = rxResource({ /* Implement the resource logic */ });

  selectedPostId = signal<number | undefined>(undefined);

  selectedPost = computed(() => {
    const postId = this.selectedPostId();
    return this.posts.value()?.find(({ id }) => id === postId);
  });
}
```

Lab 5: HttpResource

- In the `UsersService`, expose the users using an `httpResource`
- In the `UserPostsComponent`, expose the user's posts using an `httpResource`
- Finally remove the `ApiService`