GENERAL

angular 5 basic demo project overview codeproject

Published: September 19, 2026 | Index ID: #-

When you first encounter Angular, the learning curve can feel steep. But a hands on project that walks you through the core concepts—components, services, routing, and data binding—can make the transition smoother. This article presents an **Angular 5 basic demo project overview codeproject**that will help you build a small, yet fully functional application from scratch. By the end, you'll understand how the framework's architecture works and how to structure your own projects for scalability.

Why Angular 5? A Quick Context

Angular 5, released in November 2017, introduced several performance improvements and new features such as the HttpClient module, lazy loading, and improved CLI commands. Although newer Angular versions exist, the fundamentals remain the same, making Angular 5 an excellent teaching tool. Its CLI still supports the classic ng new command, and its module system is a great way to learn how Angular organizes code.

Project Goals and Scope

Our demo will demonstrate the following:

- Setting up a clean Angular 5 environment.
- Creating a modular architecture with components, services, and routing.
- Fetching data from a public API using HttpClient.
- Implementing simple navigation and UI interaction.
- Deploying the application locally for testing.

Feel free to adapt or expand the demo as you gain confidence. The code snippets below are intentionally concise to keep the focus on structure and best practices.

Step 1: Install Node.js and Angular CLI

Before you can create an Angular 5 project, you need Node.js and the Angular CLI. Use the following commands on your terminal:

1. Verify Node.js installation: node -v and npm -v.
2. Install Angular CLI globally: npm install -g @angular/cli@1.7.4 (this is the CLI version that supports Angular 5).

Once installed, you can generate a new Angular 5 project with:

ng new angular5-demo --routing --style=scss\ncd angular5-demo\nng serve\n\n\nOpen http://localhost:4200 in your browser to confirm the default app is running.

Step 2: Understand the Project Structure

After running ng new, your project contains several key folders:

- src/app – Contains the main application logic.
- src/assets – Stores static files like images.

- src/environments – Holds environment-specific configuration.
- src/styles.scss – Global stylesheet.

The app.module.ts file is the root module where you register components, services, and other modules. The app-routing.module.ts manages navigation.

Modular Design: A Key Angular Principle

Angular encourages breaking your application into small, reusable modules. This demo will create two feature modules: PostsModule for displaying posts and AboutModule for an about page. Each module will encapsulate its own components and routing configuration.

Step 3: Create Feature Modules

Generate the modules using the CLI:

ng generate module posts --route posts --module app.module\nng generate module about --route about --module app.module\n\n\nThe --route flag automatically adds lazy loaded routes to app-routing.module.ts. Lazy loading improves initial load performance by loading modules only when needed.

Posts Module

Generate a component to display a list of posts:

ng generate component posts/posts-list\n\n\nIn posts-list.component.ts, inject a service that fetches data from a public API (e.g., JSONPlaceholder). The service will use HttpClient to retrieve posts.

About Module

Create a simple static component:

ng generate component about/about-page\n\n\nIt will display static information about the application.

Step 4: Implement the Posts Service

Generate a service that will handle HTTP requests:

ng generate service posts/posts\n\n\nIn posts.service.ts, add the following code:

import { Injectable } from '@angular/core';\nimport { HttpClient } from '@angular/common/http';\nimport { Observable } from 'rxjs';\n\n@Injectable({\n providedIn: 'root',\n})\nexport class PostsService {\n private apiUrl = 'https://jsonplaceholder.typicode.com/posts';\n\n constructor(private http: HttpClient) {}\n\n getPosts(): Observable<any[]> {\n return this.http.get<any[]>(this.apiUrl);\n }\n}\n\n\nThe providedIn: 'root' syntax ensures the service is a singleton available throughout the app.

Step 5: Build the Posts List Component

Modify posts-list.component.ts to fetch and display posts:

import { Component, OnInit } from '@angular/core';\nimport { PostsService } from '../posts.service'; \n\n@Component({\n selector: 'app-posts-list',\n templateUrl: './posts-list.component.html',\n styleUrls: ['./posts-list.component.scss'],\n})\nexport class PostsListComponent implements OnInit {\n posts: any[] = [];\n loading = true;\n error: string | null = null;\n\n constructor(private postsService: PostsService) {}\n\n ngOnInit(): void {\n this.postsService.getPosts().subscribe(\n (data) => {\n this.posts = data;\n this.loading = false;\n },

\n (err) => {\n this.error = 'Failed to load posts.';\n this.loading = false;\n }\n);\n }\n}\n\n\nIn the corresponding template posts-list.component.html, use Angular's structural directives to loop over the posts:

<div *ngIf="loading">Loading posts...</div>\n<div *ngIf="error" class="error">{{ error }}</div>\n\n<ul *ngIf="!loading && !error">\n <li *ngFor="let post of posts">\n <h3>{{ post.title }}</h3>\n <p>{{ post.body }}</p>\n \n
\n\n\nStyle the list in posts-list.component.scss:

ul {\n list-style-type: none;\n padding: 0;\n}\nli {\n border-bottom: 1px solid #ccc;\n padding: 10px 0;\n}\n.error {\n color: red;\n}\n\n\nStep 6: Configure Routing

The CLI already added lazy loaded routes. Verify app-routing.module.ts looks like this:

import { NgModule } from '@angular/core';\nimport { Routes, RouterModule } from '@angular/router';\n\nconst routes: Routes = [\n { path: 'posts', loadChildren: () => import('./posts/posts.module').then(m => m.PostsModule) },\n { path: 'about', loadChildren: () => import('./about/about.module').then(m => m.AboutModule) },\n { path: '', redirectTo: '/posts', pathMatch: 'full' },\n { path: '**', redirectTo: '/posts' }\n];\n\n@NgModule({\n imports: [RouterModule.forRoot(routes)],\n exports: [RouterModule]\n})\nexport class AppRoutingModule {}\n\n\nEach feature module will declare its own child routes. For example, posts-routing.module.ts should contain:

import { NgModule } from '@angular/core';\nimport { Routes, RouterModule } from '@angular/router';\nimport { PostsListComponent } from './posts-list/posts-list.component';\n\nconst routes: Routes = [\n { path: '', component: PostsListComponent }\n];\n\n@NgModule({\n imports: [RouterModule.forChild(routes)],\n exports: [RouterModule]\n})\nexport class PostsRoutingModule {}\n\n\nStep 7: Add Navigation Links

Update app.component.html to include navigation:

<nav>\n <a routerLink="/posts" routerLinkActive="active">Posts\n <a routerLink="/about" routerLinkActive="active">About\n</nav>\n<router-outlet></router-outlet>\n\n\nAdd minimal styling in app.component.scss:

nav {\n background: #f8f9fa;\n padding: 10px;\n a {\n margin-right: 15px;\n text-decoration: none;\n color: #007bff;\n &.active {\n font-weight: bold;\n }\n }\n}\n\n\nStep 8: Run and Test the Application

Start the development server:

ng serve\n\n\nNavigate to http://localhost:4200 and click on the **Posts** link. You should see a list of posts fetched from the API. Switching to **About**displays a static page. The navigation bar stays consistent across routes, demonstrating Angular’s component reusability.

Step 9: Common Pitfalls and Troubleshooting

Baca Juga (Artikel Terkait)

- **advanced engineering electromagnetics solution manual edition hardcover by balanis constantine a published by wiley**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-solution-manual-edition-hardcover-by-balanis-constantine-a-published-by-wiley.pdf
- **advanced engineering electromagnetics solutions manual**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-solutions-manual.pdf
- **advanced engineering electromagnetics wiley 1989 grading**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-wiley-1989-grading.pdf
- **advanced engineering fluid mechanics by biswas**
  URL: http://djjsearch.com/advanced-engineering-fluid-mechanics-by-biswas.pdf