Dev Tools Manual Angular Setup

This guide will help you set up your project with Builder.io for Angular. With Builder Develop, this setup is necessary if you plan on using component mapping and want to preview your components in Builder.io's Visual Editor.

Prerequisites

Before you begin, make sure you have:

- Node.js installed on your machine (download from nodejs.org)
- An Angular project
- A Builder.io account with an API key

Setup Steps

1. Install the required packages

First, you need to install the necessary packages for Builder.io and Angular:

```
npm install --save-dev concurrently @builder.io/dev-tools
@builder.io/angular
```

2. Configure the development script

Next, add a development task to your npm scripts. This will run both the Angular development server and Builder.io Dev Tools concurrently:

```
"scripts": {
  // ...existing code...
  "dev": "concurrently \"ng serve\" \"builder-dev-tools\""
}
```

3. Add your Builder API key to environment.ts

Add your Builder API key to the `environment.ts` file:

```
export const environment = {
  production: false,
  builderApiKey: "<YOUR_BUILDER_API_KEY>",
};
```

You can find your API key in the Builder.io dashboard under Account Settings > API Keys.

4. Create a builder-registry.ts file

Create a new file at `src/app/builder-registry.ts` to register your custom components with Builder.io:

```
import { Builder } from "@builder.io/angular";

// Import your custom components
// import { YourCustomComponent } from './your-custom-component';

// Register your custom components
// Builder.registerComponent(YourCustomComponent, {
//   name: 'Your Custom Component',
//   inputs: [
//     { name: 'text', type: 'string' }
//   ]
// });

// Export your custom components for use in the Figma imports page
export const CUSTOM_COMPONENTS = [
  // YourCustomComponent
];
```

For more details on registering components, see the Builder.io documentation on component setup.

5. Create a Figma imports component

Create a new component for previewing Figma imports at `src/app/figma-imports/figma-imports.component.ts`:

```
import { Component, Input } from "@angular/core";
import { fetchOneEntry, type BuilderContent } from "@builder.io/sdk-angular";
import { Content } from "@builder.io/sdk-angular";
import { CommonModule } from "@angular/common";
import { environment } from "../../environments/environment";
import { CUSTOM_COMPONENTS } from "../builder-registry";

@Component({
  selector: "app-figma-imports",
  standalone: true,
  imports: [Content, CommonModule],
  template: `
    <builder-content
      [model]="model"
      [content]="content"
      [apiKey]="apiKey"
      [customComponents]="customComponents"
    ></builder-content>
  `,
})
export class FigmaImportsPage {
  @Input() model = "figma-imports";

  apiKey = environment.builderApiKey;
  content: BuilderContent | null = null;
  customComponents = CUSTOM_COMPONENTS;

  async ngOnInit() {
    const urlPath = window.location.pathname || "/";

    const builderContent = await fetchOneEntry({
      model: this.model,
      apiKey: this.apiKey,
      userAttributes: {
        urlPath,
      },
    });

    if (!builderContent) {
```

```
      return;
    }

    this.content = builderContent;
  }
}
```

This component allows you to:

- Preview designs imported from Figma within your Angular application
- See how your custom components render with Builder.io content
- Test the integration between Builder.io and your Angular app

6. Add the Figma imports component to your router

Add the `figma-imports` component as a route in your Angular router, but only in development mode:

```
import { NgModule } from "@angular/core";
import { RouterModule, Routes } from "@angular/router";
import { RootComponent } from "./root/root.component";
import { environment } from "../environments/environment";
import { FigmaImportsPage } from "./figma-imports/figma-imports.component";

const routes: Routes = [
  {
    path: "",
    component: RootComponent,
  },
  // ...existing routes...
];

// Only add the Figma plugin route in development
if (environment.production === false) {
  import("./builder-registry");

  routes.push({
    path: "figma-imports",
    component: FigmaImportsPage,
  });
}

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

Testing Your Setup

After completing the setup:

1. Start your development server with:

```
npm run dev
```

2. Navigate to `http://localhost:4200/figma-imports` in your browser
3. If everything is set up correctly, you should see a blank page or any existing Builder.io content for the “figma-imports” model
4. You can now configure your components in Builder.io’s Visual Editor and they will appear in your Angular application

Troubleshooting

- **Blank Page**: If you see a blank page, check that your API key is correct and that you’ve created content for the “figma-imports” model in Builder.io
- **404 Error**: Ensure that the route is correctly configured in your router
- **Component Not Rendering**: Verify that your components are properly registered in `builder-registry.ts`
- **Build Errors**: Make sure all dependencies are installed and that your Angular version is compatible with Builder.io

Next Steps

Now that you’ve set up Builder.io Dev Tools for Angular, you can:

1. Register your custom components in the `builder-registry.ts` file
2. Create new content in Builder.io using the Visual Editor
3. Import designs from Figma and map them to your components
4. Configure your content models in Builder.io
5. Integrate Builder.io content into your main application

For more detailed information on using Builder.io with Angular, refer to the Builder.io Angular documentation.

Cookbook

Registering web components in Angular

```
import { Component, CUSTOM_ELEMENTS_SCHEMA } from "@angular/core";
import type { RegisteredComponent } from "@builder.io/sdk-angular";

@Component({
  standalone: true,
  template: `<custom-button>{{ textContent }}</custom-button>`,
  inputs: ["textContent"],
  schemas: [CUSTOM_ELEMENTS_SCHEMA],
})
class CustomButtonComponent {
  textContent = "Click me!";
}

export const CUSTOM_COMPONENTS: RegisteredComponent[] = [
  {
    name: "Custom Button",
    component: CustomButtonComponent,
    inputs: [
      { name: "textContent", type: "string", defaultValue: "Click me!" },
    ],
  },
];
```

Currently the Builder.io Angular SDK only supports Angular components, but you can use Angular elements to wrap web components so they show up in the Visual Editor. The inputs in the registered component map Builder.io fields to Angular component inputs. The Angular component is a thin wrapper over the web component, which is the actual component that will be rendered in the Visual Editor.