deti universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Technologies and Web Programming

Angular Framework

Angular Framework

Introduction to Angular

Angular Introduction (i)

- Angular Framework is a web application development platform based on TypeScript language.

- It makes the switch from MVC (Model-View-Controller) approach, used in AngularJS, to a Components-Based Web Development approach.

Angular Introduction (ii)

- Angular allows to build web applications out of components, which are UI building blocks that are easy to test and reuse.

- Components, neatly encapsulate all the style and function required for a certain feature to work, and so, creating custom HTML elements is a greatly simplified process.

Angular Introduction (iii)

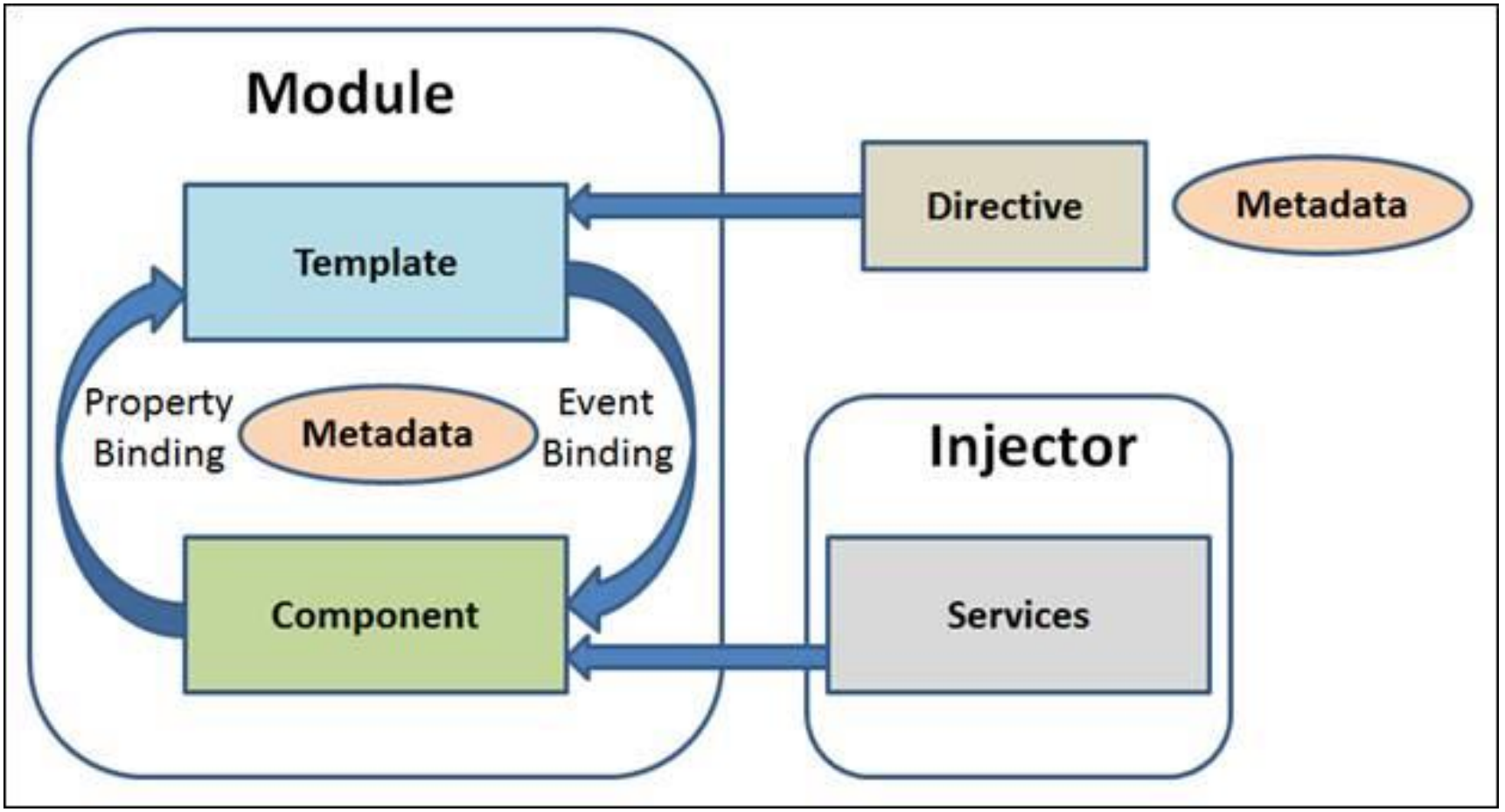
- The idea is to build declarative components that are fully encapsulated.
- They describe their own views and can be easily packaged and distributed to other developers.
- The application itself consists of a root component that contains a set of components for every UI element, screen, and route.
- This component tree lies at the core of any Angular application.

Angular Framework

Angular Architecture

Angular Architecture

- The main eight blocks of an Angular Application



Angular – Modules

- A Module is a block of code which is designed to perform a single task.
- It can be exported in form of class.
- Angular applications can have several modules. Although, every Angular application must have at least one.
- Every Angular application has one root module and can have many more featured modules.

Angular – Components

- A component is the combination of a class, containing the core logic for a page, and an associated template that deals with its View.
- The application logic is written inside the class which is used by the View. The class interacts with the View through methods and properties of its API.
- Angular creates and updates the required components, and destroys the unused ones as the user moves through an application.

Angular – Metadata

- Metadata is the way to define how Angular process a class, a method or a property, for particular reasons.
- In TypeScript, metadata is defined by using decorators.
- For example, if we want to define a component in an Angular application, we need to tell Angular that a particular class is a component.
  - This is done associating metadata to the class using @Component decorator.

Angular – Template

- A template is the component View that tells Angular how to display the component.

- It looks like normal HTML.

Angular – Data Binding

- Data binding is a powerful feature of software development technologies.
- It is the connection bridge between View and the business logic of the application.
- There are four types of data binding supported by Angular:
  - Interpolation - used to display the component value within HTML.
  - Property Binding - It passes the property's value of a parent component to its child's property.

Angular – Data Binding

- Data binding, continuation:
  - Event Binding – used to fire an event when we click on a component's name, or some change occurs in an input component.
  - Two-way Binding – it combines event and property binding in single notation by using ngModel directive, to have automatic change between the data model and the view.

Angular – Directives

- Directives extend HTML attributes.
- These are markers on the DOM elements which provides some special behavior to them and tell Angular HTML compiler to attach it.
- There are three types of directives:
  - Decorator Directive - it decorates (@Directive) the elements using additional behavior. There are many built-in directives like ngModel, and others.
  - Component Directive - it is extended from @Directive decorator with template-oriented features.
  - Template Directive - it converts HTML into a reusable template. It is also known as structural directive.

Angular – Service

- A service is a broad category encompassing any value, function, or feature that an app needs.
- A service is typically a class with a narrow, well-defined purpose.
- Angular distinguishes components from services to increase modularity and reusability.
- Typical examples of services are logging service, data service, message service, etc.
  There is no base class to define a service.

Angular – Dependency Injection

- Dependency Injection is a software design pattern in which objects are passed as dependencies.
- It helps us remove the hard coded dependencies and makes dependencies configurable.
- Using Dependency Injection, we can make components more maintainable, reusable, and testable.
- Dependency Injection (DI) is wired into the Angular framework and used everywhere to provide new components with the services or other things they need.

Angular Framework

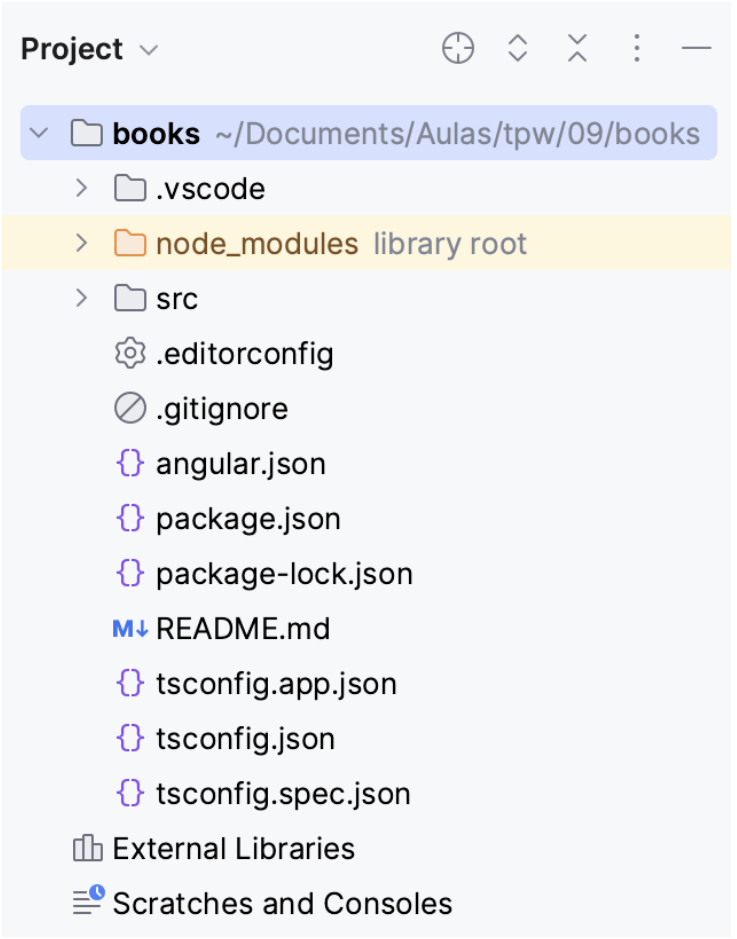
Developing

Angular – Developing

- Installation:
  - Install Node.js and npm if they are not already on your machine.
  - Install Angular CLI (Command Line Interface) globally.
    - command: npm install -g @angular/cli
- Creating a project
  - Use a known IDE, like Pycharm or WebStorm to create and develop an Angular project, or use the following command:
    - command: ng new angular-app
- Running the project
  - Use IDE to run it or use the following command, inside project folder:
    - command: ng serve

Angular – Project Books

- Create a new Angular project, named “books”, using the command line or the IDE, load it and run it.



Angular – Developing books (i)

- Change title in main component.
  - Open “src/app/app.ts” and change it.

app.ts

```ts
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';

@Component({  Show usages  Helder Zagalo *
  selector: 'app-root',
  imports: [CommonModule],
  templateUrl: './app.html',
  styleUrl: './app.css'
})
export class App {
  title : string  = 'My Books';
}
```

  - Open “src/app/app.html” and change it.

app.html

```html
<h1>{{ title }}</h1>
```

Angular – Developing books (ii)

- Create application styles and save them on file “src/styles.css”.

```
h1 {
  color: #369;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 250%;
}
h2, h3 {
  color: #444;
  font-family: Arial, Helvetica, sans-serif;
  font-weight: lighter;
}
body {
  margin: 2em;
}
body, input[text], button {
  color: #888;
  font-family: Cambria, Georgia;
}
* {
  font-family: Arial, Helvetica, sans-serif;
}
```

Angular – Developing books (iii)

- Create a component named “authors”, running the command inside project folder:
“ng generate component authors”

authors.ts

```
1   import { Component } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3
4   @Component({  Show usages
5     selector: 'app-authors',
6     imports: [CommonModule],
7     templateUrl: './authors.html',
8     styleUrl: './authors.css',
9   })
10  export class Authors {
11  }
```

Angular – Developing books (iv)

- Open “src/app/app.ts” and change it.

app.ts

```ts
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { Authors } from './authors/authors';

@Component({  Show usages  Helder Zagalo *
  selector: 'app-root',
  imports: [CommonModule, Authors],
  templateUrl: './app.html',
  styleUrl: './app.css',
})
export class App {
  title : string  = 'My Books';
}
```

- Open “src/app/app.html” and change it.

app.html

```html
<h1>{{ title }}</h1>
<app-authors></app-authors>
```

Angular – Developing books (v)

- Create Author interface in file “src/app/author.ts”

author.ts

```
1
2   export interface Author {  Show usages
3     id: number;
4     name: string;
5     email: string;
6   }
```

Angular – Developing books (vi)

- Create authors list file “src/app/authorslist.ts”

authorslist.ts

```
1
2   import { Author } from './author';
3
4   export const AUTHORS: Author[] = [  Show usages
5     { id: 1, name: 'Fernando Pessoa', email: 'fpessoa@mail.pt' },
6     { id: 2, name: 'J. K. Rowling', email: 'jkrowling@mail.uk' },
7     { id: 3, name: 'Arthur Conan Doyle', email: 'jkrowling@aol.pt' },
8     { id: 4, name: 'J. K. Rowling', email: 'acdoyle@mail.net' },
9     { id: 5, name: 'John Green', email: 'jgreen@mail.com' },
10  ];
```

Angular – Developing books (vii)

- Change “src/app/authors/authors.ts” file:

authors.ts

```
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { Author } from '../author';
4  import { AUTHORS } from '../authorslist';
5
6  @Component({  Show usages
7    selector: 'app-authors',
8    imports: [CommonModule],
9    templateUrl: './authors.html',
10   styleUrl: './authors.css',
11 })
12 export class Authors {
13   authors: Author[];
14
15   constructor() {  no usages
16     this.authors = AUTHORS;
17   }
18 }
```

Angular – Developing books (viii)

- Change “src/app/authors/authors.html” file:

authors.html

```
<h2>Authors</h2>
<ul class="authors">
  @for (au of authors; track au.id) {
    <li>
      <span class="badge">{{ au.id }}</span> {{au.name}}
    </li>
  }
</ul>
```

Angular – Developing books (ix)

- Create component styles and save them on file "src/app/authors/authors.css".

```
/* AuthorsComponent's private CSS styles */
.selected {
  background-color: #CFD8DC !important;
  color: white;
}
.authors {
  margin: 0 0 2em 0;
  list-style-type: none;
  padding: 0;
  width: 15em;
}
.authors li {
  cursor: pointer;
  position: relative;
  left: 0;
  background-color: #EEE;
  margin: .5em;
  padding: .3em 0;
  height: 1.6em;
  border-radius: 4px;
}
...
```

Angular – Developing books (x)

```
...
.authors li.selected:hover {
  background-color: #BBD8DC !important;
  color: white;
}
.authors li:hover {
  color: #607D8B;
  background-color: #DDD;
  left: .1em;
}
.authors .text {
  position: relative;
  top: -3px;
}
.authors .badge {
  display: inline-block;
  font-size: small;
  color: white;
  padding: 0.8em 0.7em 0 0.7em;
  background-color: #607D8B;
  line-height: 1em;
  position: relative;
  left: -1px;
  top: -4px;
  height: 1.8em;
  margin-right: .8em;
  border-radius: 4px 0 0 4px;
}
```

Angular – Developing books (xi)

- Result:

Angular – Developing books (xii)

- Change “src/app/authors/authors.html” file:

authors.html

```
1  <h2>Authors</h2>
2  <ul class="authors">
3    @for (au of authors; track au.id) {
4      <li (click)="onSelect(au)">
5        <span class="badge">{{ au.id }}</span> {{au.name}}
6      </li>
7    }
8  </ul>
```

Angular – Developing books (xiii)

- Change “src/app/authors/authors.ts” file:

authors.ts

```
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { Author } from '../author';
4  import { AUTHORS } from '../authorslist';
5  import { FormsModule} from '@angular/forms';
6
7  @Component({  Show usages
8    selector: 'app-authors',
9    imports: [CommonModule, FormsModule],
10   templateUrl: './authors.html',
11   styleUrl: './authors.css',
12 })
13 export class Authors {
14   authors: Author[];
15   selectedAuthor: Author | null = null;
16
17   constructor() {  no usages
18     this.authors = AUTHORS;
19   }
20
21   onSelect(author: Author) : void  {  Show usages
22     this.selectedAuthor = author;
23   }
24 }
```

Angular – Developing books (xv)

- Change “src/app/authors/authors.html” file:

authors.html

```
<h2>Authors</h2>
<ul class="authors">
  @for (au of authors; track au.id) {
    <li (click)="onSelect(au)">
      <span class="badge">{{ au.id }}</span> {{au.name}}
    </li>
  }
</ul>

@if (selectedAuthor) {
  <h2>Information on {{selectedAuthor.name | uppercase}}</h2>
  <div>Id: {{selectedAuthor.id}}</div>
  <div>
    <label>
      Name:
      <input [(ngModel)]="selectedAuthor.name" placeholder="name" />
    </label>
  </div>
  <div>
    <label>
      Email:
      <input [(ngModel)]="selectedAuthor.email" placeholder="email" />
    </label>
  </div>
}
```

Angular – Developing books (xvi)

- Result: