БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ и ИНФОРМАТИКИ
Кафедра многопроцессорных сетей и систем

Рафеенко Е.Д.

Web- программирование

Single Page Applications (SPA). Фреймворк Angular

Содержание

Angular - data binding

Data binding (привязка данных) в html template.

1. Интерполяция (text interpolation) - позволяет включать динамические строковые значения в шаблоны HTML. Синтаксис: {{ }}.

Текст между фигурными скобками - это шаблонное выражение, которое Angular сначала вычисляет, а затем преобразует в строку:

```
<p>The sum of 1 + 1 is {{1 + 1}}</p>
```

Интерполяция используется для просмотра значения свойства компонента на экране.

Объявление свойства myBike в *src/app/app.component.ts*

myBike="Ninja250R";

В *src/app/app.component.html* можно вывести значение этой переменной:

```
<h3>My bike: {{myBike}}</h3>
```

Angular - data binding

2. Привязка свойств (property binding) в Angular позволяет устанавливать значения для свойств элементов HTML или директив, передавать значения между компонентами.

Синтаксис – [property]=“value”.

<app-item-detail [childItem]="parentItem"></app-item-detail>

Скобки заставляют Angular оценивать правую часть присваивания как динамическое выражение.

Без скобок Angular обрабатывает правую часть как строковый литерал и устанавливает для свойства (childItem) это статическое значение.

<app-item-detail childItem="parentItem"></app-item-detail>

Angular - data binding

2. Привязка свойств (передача значения между компонентами – от parent к child)

<parent-component>
    <child-component>
    </child-component>
</parent-component>

Пусть parent-component – это *app.component.ts (app.component.html),*
<child-component> - это *bike-info.component.ts (bike-info.component.html*)

Используем child selector <app-bike-info> в *src/app/app.component.html,*

Используем привязку свойств (property binding), чтобы привязать свойство **bike** в child – компоненте свойству selectedBike в родительском компоненте:

<app-bike-info **[bike]**="selectedBike"></app-bike-info>

Объявление в *src/app/app.component.ts:* selectedBike: Bike;

Объявление в *src/app/ bike-info.component.ts:* @Input() bike: Bike;

Angular - data binding

3. Привязка событий (Event Binding) используется для генерации события от определенного элемента, например, при нажатии кнопки ввода и т. д.

Синтаксис: (event)= "handler";

Событие щелчок по ссылке в *src/app/app.component.html*:

(click)="onSelect(bike)"

Обработка события в *src/app/app.component.ts:*

```
onSelect(bike: Bike): void {
    this.selectedBike = bike;
  }
```

Angular - data binding

4. Двунаправленная привязка данных (T**wo-way data binding)**

Синтаксис:

[(ngModel)]="property“

Такой вид привязки данных чаще всего используется в формах (template-driven forms) – свойство компонента отображается на элемент управления формы.

В результате всякий раз при обновлении значения на экране, оно будет автоматически обновляться в компоненте и наоборот.

[(ngModel)] – комбинация синтаксиса для input property binding [] и output event binding ().

Angular - data binding

4. Двунаправленная привязка данных (T**wo-way data binding)**

Пример.

Поле ввода в *src/app/app-bike-info.html*

```
<div>
   <label>model: </label>
   <input [(ngModel)]="bike.model" placeholder="model" />
 </div>
```

Объявление в *src/app/ bike-info.component.ts:* @Input() bike: Bike;

Angular metadata

Metadata – это декораторы (аннотации) в ts (typescript) классах, которые определяют каким образом Angular должен обработать класс. Например, аннотации:

@Component,
@NgModule,
@Injectable,
@Directive

Angular metadata

Directives — это классы, которые добавляют дополнительное поведение к элементам в приложениях Angular.

- Attribute directives— директивы, которые изменяют внешний вид или поведение элемента, компонента или другой директивы:
  [ngClass]

- Structural directives— директивы, которые изменяют макет DOM, добавляя и удаляя элементы DOM:
  например *ngFor, *ngSwitch, *ngIf

Angular metadata

Services (службы) - используются как многократно используемые службы данных для совместного использования между компонентами в приложении.

Помечаются аннотацией @Injectable().

Сервисы обязательно асинхронны. Например, можно вернуть данные, используя Observable RxJS и httpClient.

```
getBikes(): Observable<Bike[]>
        {
            return this.httpClient.get(this.bikesUrl1);
         }
```

Angular Routing

Angular Router используется чтобы управлять переходом от одного представления (view) к другому.

В Angular лучше всего загружать и настраивать **Router** в отдельном модуле верхнего уровня, который предназначен для маршрутизации и импортируется корневым модулем AppModule.

По умолчанию, имя класса модуля: AppRoutingModule. Он находится в файле app-routing.module.ts в папке src/app

Сгенерировать его можно командой Angular cli:

ng generate module app-routing --flat --module=app

--flat - сгенерированный модуль будет находиться в папке src/app (собственная папка для модуля создаваться не будет)

--module=app - CLI добавит модуль в массиве imports в корневом модуле AppModule

Angular Routing

В классе AppRoutingModule:

1. сделать импорт нужных модулей:

```
import { RouterModule, Routes } from '@angular/router';
```

2. сконфигурировать маршруты (routes).

```
const routes: Routes = [
 {path: '', component: BikeListComponent},
 {path: 'bikes', component: BikeListComponent}
];
```

Обычный Angular маршрут (Route) имеет два свойства:

path: строка, соответствующая URL в адресной строке браузера;

component: компонент, который обработчик должен создать при переходе на этот маршрут.

Angular Routing

Аннотация (метаданные) @NgModule инициализирует маршрутизатор и запускает его для прослушивания изменений.

Необходимо добавить RouterModule в массив импорта AppRoutingModule и настроить его с нужными маршрутами, вызывая RouterModule.forRoot():

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

Метод *forRoot*() предоставляет поставщиков услуг и директивы, необходимые для маршрутизации, и выполняет начальную навигацию на основе текущего URL-адреса браузера.

Angular Routing

<router-outlet> элемент добавляется в html template,

он сообщает маршрутизатору, где отображать нужные представления (views).

RouterOutlet - это одна из директив маршрутизатора

Angular Routing – lazy loading

Ленивая загрузка функциональных модулей — шаблон проектирования, который загружает NgModules по мере необходимости.

Использует loadChildren (вместо компонент) в маршрутах AppRoutingModule.

```
const routes: Routes = [
  {
    path: 'bike-center',
    loadChildren: () => import('./bikes/bikes.module').then(m => m. BikesModule)
  }
];
```

Angular Routing – lazy loading

В модуле с отложенной загрузкой сгенерировать модуль маршрутизации,

добавить в модуль маршрутизации маршрут для компонента.

```
const routes: Routes = [
  {
    path: '',
    component: BikeCenterComponent  }
];
```

Angular Forms

Angular предоставляет два разных подхода к обработке пользовательского ввода через формы:

reactive forms

template-driven forms.

Оба вида форм

фиксируют события пользовательского ввода,

валидируют пользовательский ввод,

создают модель формы и модель данных для обновления,

предоставляют способ отслеживания изменений.