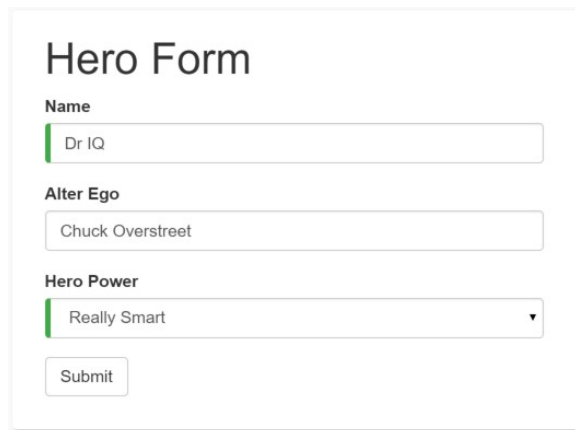


# Projeto de Interfaces WEB

## Formulários e Toastr em Angular Aula 07

# Introdução

- Abordagem Template-Driven-Form
  - <https://angular.io/guide/forms>
- Nessa aula, iremos aprender a criar formulários baseados em template.



The image shows a web form titled "Hero Form". It contains three input fields: "Name" with the value "Dr IQ", "Alter Ego" with the value "Chuck Overstreet", and "Hero Power" with the value "Really Smart". A "Submit" button is located at the bottom of the form.

Hero Form

Name  
Dr IQ

Alter Ego  
Chuck Overstreet

Hero Power  
Really Smart

Submit

Fonte: site Angular

# Classe Model Heroi

- Crie uma classe Heroi.ts, dentro da pasta models.

```
export class Heroi{  
  constructor(public id:number,  
               public nome:string,  
               public poder:string,  
               public identidadeSecreta?:string, //opcional  
              ){  
  
  }  
}
```

# Componente Heroi

- Crie um componente heroi-form:
  - ng g c heroi-form --spec=false

```
export class HeroiFormComponent{  
  
  poderes:string[] = ['Inteligência', 'Força', 'Elasticidade', 'Programa em Assembly'];  
  heroi:Heroi = new Heroi(0,"Dr. Zé",this.poderes[0]); //modelo a ser usado  
  submetido = false;  
  
  salvar(){  
    this.submetido = true;  
  }  
  
  imprimir(){  
    console.log(JSON.stringify(this.heroi));  
  }  
}
```

# App.module

- Import o FormsModule no seu módulo.

```
@NgModule({  
  declarations: [  
    AppComponent,  
    HeroiFormComponent  
  ],  
  imports: [  
    BrowserModule,  
    FormsModule ←  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

# Template do HTML

- Em heroi-form.html

```
<div class="container">
  <h1>Inserir Herói</h1>
  <form>
    <div class="form-group">
      <label>Nome</label>
      <input type="text" class="form-control" id="nome" required>
    </div>
    <div class="form-group">
      <label>Identidade Secreta</label>
      <input type="text" class="form-control" id="identidadeSecreta" required>
    </div>
    <button type="submit" class="btn btn-primary">Salvar</button>
  </form>
</div>
```

# Template do HTML

- Adicione um drop-down dos poderes:

```
...  
<div class="form-group">  
  <label for="poder">Poder do Herói</label>  
  <select class="form-control" id="poder" required>  
    <option *ngFor="let pod of poderes" [value]="pod">{{pod}}</option>  
  </select>  
</div>  
...
```

# Template do HTML

- Uso do [(ngModel)]

```
...  
<div class="form-group">  
  <label>Nome</label>  
  <input type="text" class="form-control" id="nome"  
    [(ngModel)] = "heroi.nome" name="nome"  
    required>  
</div>  
<div class="form-group">  
  <label>Identidade Secreta</label>  
  <input type="text" class="form-control" id="identidadeSecreta"  
    [(ngModel)] = "heroi.identidadeSecreta" name="identidadeSecreta"  
    required>  
</div>  
  
<div class="form-group">  
  <label for="poder">Poder do Herói</label>  
  <select class="form-control" id="poder" required [(ngModel)]="heroi.poder" name="poder">  
    <option *ngFor="let pod of poderes" [value]="pod" >{{pod}}</option>  
  </select>  
</div>  
...
```

Usando a classe model, Heroi.ts!

Não esqueça o atributo "name". Ele deve ter o Mesmo valor do atributo do seu Model.



# Validação

- Diretiva ngForm
  - **<form #heroForm="ngForm"> ...</form>**
    - What NgForm directive? You didn't add an NgForm directive, Angular did. Angular automatically creates and attaches an NgForm directive to the <form> tag.
    - The NgForm directive supplements the form element with additional features. It holds the controls you created for the elements with an ngModel directive and name attribute, and monitors their properties, including their validity. It also has its own valid property which is true only if every contained control is valid

# Validação

- O uso do ngModel é muito poderoso. Ele te diz se o usuário “tocou” em algum controle, mudou seus valores ou se os valores se tornaram **inválidos**.
- O Angular faz uso de classes CSS especiais para a o controle de aparência:

State	Class if true	Class if false
The control has been visited.	ng-touched	ng-untouched
The control's value has changed.	ng-dirty	ng-pristine
The control's value is valid.	ng-valid	ng-invalid

# Validação

- Faça o seguinte teste:

```
<input type="text" class="form-control" id="nome"  
  [(ngModel)] = "heroi.nome" name="nome"  
  required #spy>  
  {{spy.className}}
```

Depois a gente apaga isso.

- Siga os passos:
  - Look but don't touch.
  - Click inside the name box, then click outside it.
  - Add slashes to the end of the name.
  - Erase the name.

# Validação

- O que acontece?

Dr IQ	TODO: remove this: form-control ng-untouched ng-pristine ng-valid	Untouched
Dr IQ	TODO: remove this: form-control ng-pristine ng-valid ng-touched	Touched
Dr IQ////	TODO: remove this: form-control ng-valid ng-touched ng-dirty	Changed
	TODO: remove this: form-control ng-touched ng-dirty ng-invalid	Invalid

# Validação

- Criação do CSS para válido e inválido (heroi-form.css)

```
.ng-valid[required], .ng-valid.required {  
  border-left: 5px solid #42A948; /* green */  
}
```

```
.ng-invalid:not(form) {  
  border-left: 5px solid #a94442; /* red */  
}
```

# Validação

- Crie o efeito da mensagem (exemplo do input):

```
<div class="form-group">  
  <label>Nome</label>  
  <input type="text" class="form-control" id="nome" required  
    [(ngModel)] = "heroi.nome" name="nome"  
    #nome="ngModel">  
  <div [hidden]="nome.valid || nome.pristine" class="alert alert-danger">  
    Nome é obrigatório!  
  </div>  
</div>
```

Não esqueça do "required"

Acessa os valores do ngModel

Mensagem de erro.

# Validação

- Progresso:

## Inserir Herói

Nome

Nome é obrigatório!

Identidade Secreta

Poder do Herói

Programa em Assembly ▼

Salvar

A div é [hidden] apenas quando o campo é valid (tem algo escrito) ou pristine (nunca foi tocado).

# Resetando o Formulário

- Crie uma nova função no componente heroi-form.

```
novoiHerói(form: NgForm){  
  this.heroi = new Heroi(0, "Dr. Zé", this.poderes[3]);  
  form.reset();  
}
```

- Em heroi-form.html, o botão de limpar:

```
<button type="button" class="btn btn-secondary ml-2"  
  (click)="novoiHerói(heroiForm);">Limpar</button>
```



# ngSubmit

- Submetendo o form com o `ngSubmit`. Deve-se primeiro atrelar o form a um evento de submissão:
  - `<form #heroiForm="ngForm" (ngSubmit)="salvar()">`
    - Iremos chamar o método `salvar()` ao submeter esse form.
- Depois, devemos implementar o método `salvar`:

```
salvar(){  
  this.submetido = true;  
  console.log(`Herói salvo: ${this.imprimir()}`);  
}
```

# Extra

- Caso você não use uma classe Model, você pode pegar os atributos **manualmente**:

## heroi-form.html

```
...  
<div class="form-group">  
  <label>Identidade Secreta</label>  
  <input type="text" class="form-control" name="identidadeSecreta" ngModel>  
</div>  
...
```

**heroi-form.ts** com salvar modificado para receber um NgForm.

```
...  
salvar(form: NgForm){  
  console.log(form.value.identidadeSecreta);  
}  
...
```

# Notificações Toastr

- <https://www.npmjs.com/package/ngx-toastr>
  - `npm install ngx-toastr --save`
- `npm install @angular/animations --save`
- Importe em angular.json:

```
"styles": [  
  "src/styles.css",  
  "./node_modules/bootstrap/dist/css/bootstrap.css",  
  "./node_modules/ngx-toastr/toastr.css"  
],
```

# Notificações Toastr

- Em app.module.ts

```
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { ToastrModule } from 'ngx-toastr';
...
imports: [
  BrowserModule,
  FormsModule,
  BrowserAnimationsModule,
  ToastrModule.forRoot(),
],
...
```

# Notificações Toastr

- No componente heroi-form.ts

```
import { ToastrService } from 'ngx-toastr';  
...  
constructor(private toasty: ToastrService){}  
...  
salvar(){  
  this.submetido = true;  
  console.log(`Herói salvo: ${this.imprimir()}`);  
  this.toasty.success("Herói salvo com sucesso.")  
}  
...
```

# Notificações Toastr

- Caso queira mudar a posição:

```
this.toasty.success("Herói salvo com sucesso.",  
                    "Sucesso",  
                    {positionClass: 'toast-bottom-left'});
```

## Inserir Herói

Nome

Dr. Zé

Identidade Secreta

Poder do Herói

Programa em Assembly

Salvar

Limpar



**Sucesso**

Herói salvo com sucesso.