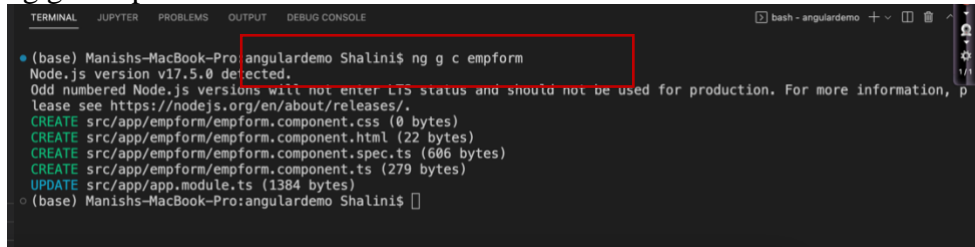
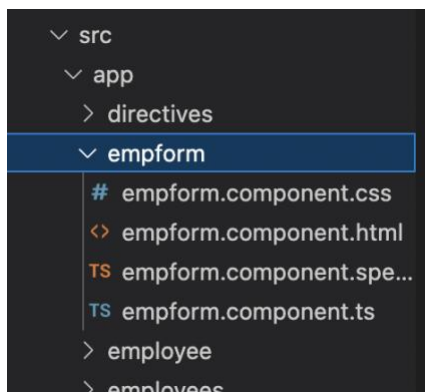


STEP 11: Template Driven Forms

1. Make sure to add FormsModule in app.module.ts file
2. Add empform component from within the angulardemo folder as follows:
`ng g c empform`



```
(base) Manishs-MacBook-Pro:angulardemo Shalini$ ng g c empform
Node.js version v17.5.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/releases/.
CREATE src/app/empform/empform.component.css (0 bytes)
CREATE src/app/empform/empform.component.html (22 bytes)
CREATE src/app/empform/empform.component.spec.ts (606 bytes)
CREATE src/app/empform/empform.component.ts (279 bytes)
UPDATE src/app/app.module.ts (1384 bytes)
(base) Manishs-MacBook-Pro:angulardemo Shalini$
```



3. Add an employee property in employee.component.ts file:
`emp={ename:''}`

4. Add the below code in employee.component.html file:
`<form>`
`<input type="text" class="form-control" required/>`
`</form>`

5. Add the following on the form tag :

```
{{empform.value | json}}
<form #empform ="ngForm">
```

It prints nothing as angular ngForm does not tracks by default for all the controls.

6. To tell which controls to track add ngModel to all the controls:
`<input type="text" class="form-control" required ngModel name='ename'/>`
With ngModel name attribute is mandatory .

7. To see the state, angular tracks for, modify the input element as follows:

REMOVE ngModel from above code and add below:

```
<input type="text" class="form-control" required [(ngModel)] = 'emp.ename'
name='ename ' #ename >
<br>TODO: remove this: {{eid.className}}
```

8. Add custom CSS, to display for error messages:
`<input type="text" class="form-control is-invalid" required [(ngModel)] = 'emp.ename' name='ename' #ename>`

9. But red border should be applied conditionally only if control is invalid
Use class binding of angular instead and remove **is-invalid** applied above
`<input type="text" class="form-control"
[class.is-invalid]="ename.invalid"
required [(ngModel)] = 'emp.ename' name='ename' #ename="ngModel">`

COMMENT {{ename.className}}

10. Now we don't want red border to be applied on page load without user interaction.
So let's add the following as well :

`[class.is-invalid]="ename.invalid && ename.touched"`

11. Let's validate name with only alphabets

`pattern="^[A-Za-z]+$"`

With the above code and `[class.is-invalid]` attribute it displays red border on the control.

12. Let's display error messages. Add the following in the div of name input element
`<small class="text-danger" [class.d-none]="ename.valid || ename.untouched">
Name required
</small>`

With class `text-danger` of bootstrap class, it displays error message in red.

Now change the error message to display for required and pattern validation

`<small class="text-danger"
[class.d-none]="ename.valid || ename.untouched">`

Name required and it should be only alphabets</small>

Here it displays both the message for anything not valid.

13. Modify the above error messages for custom message:

`<div *ngIf="ename.touched">
<div *ngIf="ename.errors && (ename.invalid)">
<small class="text-danger" *ngIf="ename.errors['required']">
<small class="text-danger" *ngIf="ename.errors['pattern']">
it can contain only alphabets and space</small>
</div>`

```
</div>
```

14. Now comment all the above code and update the ts and html respectively:

```
    saveEmployee(emp:any)
    {
        console.log(emp.value)
    });

<div class="container">
<h1>Add Employee</h1>
<!-- ngModel , ngForm => we have FormsModule in app.module.ts-->
<!-- template reference variable > local to the tmeplate-->
<!-- <p>{{ empform.value | json }}</p> -->
<form #empform="ngForm" (submit)="saveEmployee(empform)">
    <div class="mb-3">
        <input type="text" class="form-control" [class.errborder]="eid.invalid &&
eid.touched" required
            placeholder="enter id" #eid="ngModel" ngModel name="eid"/>
        <!-- <span>{{ {ename.className} }}</span> -->
        <div *ngIf="eid.touched">
            <div *ngIf="eid.errors && (eid.invalid)">
                <small class="text-danger" *ngIf="eid.errors?.required">
                    Id required </small>

                </div>
            </div>
        </div>
        <div class="mb-3">
            <input type="text" class="form-control" [class.is-invalid]="ename.invalid &&
ename.touched" minlength="5"
                required placeholder="enter name" #ename="ngModel" ngModel
                name="ename" />
            <!-- <span>{{ {ename.className} }}</span> -->
            <div *ngIf="ename.touched">
                <div *ngIf="ename.errors && (ename.invalid)">
                    <small class="text-danger" *ngIf="ename.errors?.required">
                        ename required </small>
                    <small class="text-danger" *ngIf="ename.errors?.minlength">
                        it should have minimin 5 characters</small>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

```

        <input type="email" class="form-control" [class.is-invalid]="email.invalid &&
email.touched" minlength="5"
        required placeholder="enter email" #email="ngModel" ngModel
name="email" />
        <!-- <span>{ {ename.className} }</span> -->
        <div *ngIf="email.touched">
            <div *ngIf="email.errors && (email.invalid)">
                <small class="text-danger" *ngIf="email.errors?.required">
                    email required </small>
                </div>
            </div>
        </div>
    </div>
    <div class="mb-3">
        <input type="password" class="form-control" [class.is-
invalid]="password.invalid && password.touched" minlength="5"
        required placeholder="enter password" #password="ngModel" ngModel
name="password" />
        <!-- <span>{ {ename.className} }</span> -->
        <div *ngIf="password.touched">
            <div *ngIf="password.errors && (password.invalid)">
                <small class="text-danger" *ngIf="password.errors?.required">
                    password required </small>
                </div>
            </div>
        </div>
    </div>
    <div class="mb-3">
        <input type="text" class="form-control" [class.is-invalid]="phone.invalid &&
phone.touched" minlength="5"
        required placeholder="enter phone" #phone="ngModel" ngModel
name="phone" />
        <!-- <span>{ {ename.className} }</span> -->
        <div *ngIf="phone.touched">
            <div *ngIf="phone.errors && (phone.invalid)">
                <small class="text-danger" *ngIf="phone.errors?.required">
                    Phone required </small>
                </div>
            </div>
        </div>
    </div>
    <div ngModelGroup="address">
        <div class="mb-3">
            <input type="text" class="form-control" placeholder="enter city"
#city="ngModel" ngModel name="city" />
        </div>
        <div class="mb-3">
            <input type="text" class="form-control" placeholder="enter country"
required
#country="ngModel" ngModel name="country" />
            <div *ngIf="country.touched">
                <div *ngIf="country.errors && (country.invalid)">
                    <small class="text-danger" *ngIf="country.errors?.required">

```

```
        Country required </small>
    </div>
</div>
<div class="mb-3">
    <input type="text" class="form-control"
        placeholder="enter zipcode" #zipcode="ngModel" ngModel name="zipcode"
/>
</div>

</div>
<div class="mb-3">
    <button type="submit">Add Employee</button>
</div>
</form>

</div>
```