Wednesday: NgModel

Two-Way Data Binding

Now that we have created a form that we will use to add goals in our application, let's make use of it. We want our users to see what they will be typing in real time, that is as they type it. This is the concept of two-way data binding. With two-way data binding, data moves from our template(the view) to our component class(the model) and vice versa. The essence is that when data changes on the template(view) it simultaneously changes in the component class(model) and if it also changes in the model, the view is automatically updated with the changes.

The FormsModule that we imported has a feature that will help us do the two-way data binding, the `ngModel` directive. This form has to create a goal the same way we defined it in the blueprint, so let's write code to make it create goals in the same format.

src/app/goal-form/goal-form.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Goal } from '../goal';

@Component({
  selector: 'app-goal-form',
  templateUrl: './goal-form.component.html',
  styleUrls: ['./goal-form.component.css']
})
export class GoalFormComponent implements OnInit {

  newGoal = new Goal(0,"","",new Date());
  constructor() { }

  ngOnInit() {
  }

}
```

At the top, we have imported the `Goal` blueprint class. Inside the component definition class, we have created a `newGoal` property and assigned it to the `Goal` class that takes in the format we have been using in creating goals. This new goal object will be changed by the form inputs.

src/app/goal-form/goal-form.component.html

```
<div class="container-fluid">
  <h2 class="text-center">Create a new Goal</h2>
  <hr>
  <form #goalForm='ngForm'>
      <div class="form-group">

          <label for="name">Name</label>
          <input type="text" required class="form-control" id="name" [(ngModel)]="newGoal.name" nam
e="name">

<!-- Testing to see if we get any data -->
          Display {{newGoal.name}}
      </div>
      <div class="form-group">

          <label for="description">Description</label>
          <textarea class="form-control" id="description" rows="4" required></textarea>

      </div>
```

```
      <div class="form-group">

          <label for="complete">Completion</label>
          <input type='date' id="complete" required>

      </div>
      <button type="submit" class="btn btn-success">Add Goal</button>
  </form>
</div>
```

We have added a template reference variable `goalForm` to our form tag and equated it to `ngForm`. This will provide the form element with additional features and monitor the changes and validity of input elements.

We have then added the `[(ngModel)]` attribute to our name input and equated it to the name attribute of the `newGoal` object we created in our component definition class. We have then defined a `name` attribute which is a requirement when using `[(ngModel)]` and a form. The `Display` `{{newGoal.name}}` line will temporarily display the data being received.

If our server is still running, we can type anything in the form-input Name, and see two-way data binding in action.

Two-way data binding is one of the key things that Angular makes easy for us to implement. If you want this kind of experience in your app, Angular should be a preferred tool.