Creating Users

This lesson shows the front end implementation for creating users-- it explains the files: users.component.ts and users.component.html.

WE'LL COVER THE FOLLOWING

The first feature that we will implement will be of creating *users*.

Creating Users: Front-End Implementation

We will start with the *frontend* implementation of the `UserComponent`. As mentioned before, the implementation of this component is separated into two files:

- `users.component.ts` – Contains *TypeScript* implementation of this component.
- `users.component.html` – Contains *HTML* implementation of this component.

`users.component.html`

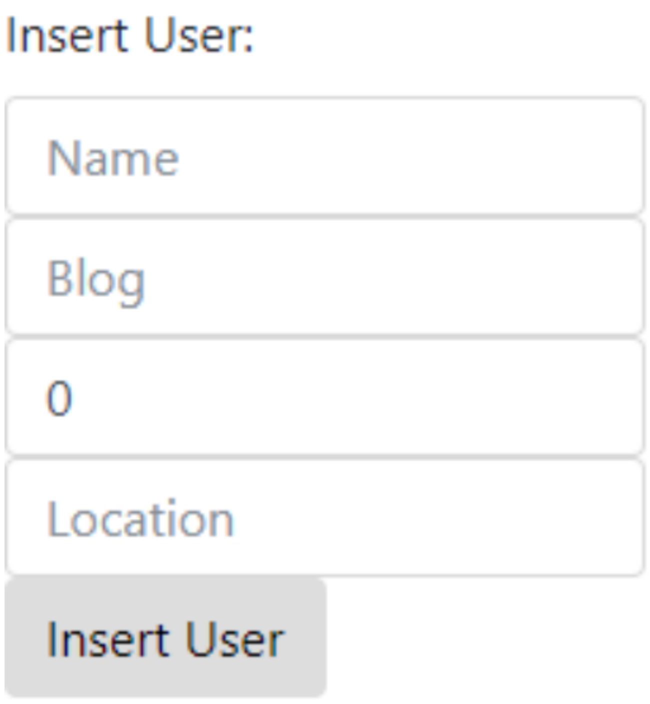
Inside `users.component.html`, one can find the part of the code, which handles the *input form* for *inserting* the user:

```
<div class="form-group col-sm-2">
    <label for="insertUser">Insert User:</label>

    <input class="form-control" type="text" placeholder="Name" [(ngModel)]="this.newUser.name
    <input class="form-control" type="text" placeholder="Blog" [(ngModel)]="this.newUser.blog
    <input class="form-control" type="text" placeholder="Age" [(ngModel)]="this.newUser.age"
    <input class="form-control" type="text" placeholder="Location" [(ngModel)]="this.newUser.
    <button class="btn btn-default" (click)="insertNewUser()"> Insert User </button>
</div>
```

/mean_frontend/src/app/users/users.component.html

In the browser that looks like this:



`[(ngmodel)]`

We use the attribute `[(ngmodel)]` to double-bind the information on the HTML with our TypeScript code.

What does this mean?

This means that changes made in those input fields will be mapped automatically to the values of the `newUser` field of the component class.

Yes, this is confusing.

`users.component.ts`

To make it more clear, let's take a look at the parts of the TypeScript implementation of this component that are in charge of implementing this feature:

```
@Component({
  selector: 'users',
  templateUrl: './users.component.html',
})
export class UserComponent implements OnInit {

  newUser: User;

  constructor(
    private userService: UserService
  ) { }

  ngOnInit() {
    this.newUser = User.CreateDefault();
  }

  insertUser() {
    this.userService
    .insertNewUser(this.newUser)
    .subscribe(
      data => {
         this.newUser._id = data.id;
         this.users.push(this.newUser);
         this.newUser = User.CreateDefault();

         console.log("Added user.");
      }
    )
  }
}
```

/mean_frontend/src/app/users/users.component.ts

Explanation

You can see that there is a field called `newUser` **(line 7)**, which will collect the data from the input form on the HTML, thanks to the `[(ngModel)]` attribute that was mentioned earlier.

Another important attribute in HTML is (`click`) which is declared inside the `button` HTML tab. This attribute defines which function will be called once the button is clicked; and we have defined the `insertUser()` function **(line 17)** for this purpose.

`insertUser()`

This function passes the value that has been added inside `newUser` to `userService` **(line 19)**.

This function is also *subscribed* to the *data* that the *service* will return **(line 21)**. `insertUser()` updates data inside `newUser` and then adds it to the `users` array **(line 23)**.

This array will be explained once we check the implementation in the next lesson.