Checkout Form
Custom Validator Rule

White Space

© luv2code LLC

White Space

- Our form currently has a problem with white space

© luv2code LLC

White Space

- Our form currently has a problem with white space
- In the required fields

© luv2code LLC

White Space

- Our form currently has a problem with white space
- In the required fields
  - If you enter ONLY whitespace ... it **passes** ... YIKES!

© luv2code LLC

White Space

- Our form currently has a problem with white space
- In the required fields
  - If you enter ONLY whitespace ... it **passes** ... YIKES!
  - Should **fail**!

© luv2code LLC

Custom Validator

© luv2code LLC

Custom Validator

- We can resolve this with a custom validator: **`notOnlyWhitespace`**

© luv2code LLC

Custom Validator

- We can resolve this with a custom validator: **`notOnlyWhitespace`**
  - Check the field value

© luv2code LLC

Custom Validator

- We can resolve this with a custom validator: **`notOnlyWhitespace`**
  - Check the field value
  - If it ONLY has whitespace

© luv2code LLC

Custom Validator

- We can resolve this with a custom validator: **`notOnlyWhitespace`**
  - Check the field value
  - If it ONLY has whitespace
    - then return an error (failed)

© luv2code LLC

Custom Validator

- We can resolve this with a custom validator: **`notOnlyWhitespace`**
  - Check the field value
  - If it ONLY has whitespace
    - then return an error (failed)
    - else return null (passed)

© luv2code LLC

Development Process

Step-By-Step

© luv2code LLC

Development Process

Step-By-Step

1. Define customer validator rule

© luv2code LLC

Development Process

Step-By-Step

1. Define customer validator rule
2. Specify custom validator rule for the form controls

© luv2code LLC

Development Process

Step-By-Step

1. Define customer validator rule
2. Specify custom validator rule for the form controls
3. Update HTML template to display error messages

© luv2code LLC

Step 1: Define custom validator rule

File: luv2-shop-validators.ts

© luv2code LLC

Step 1: Define custom validator rule

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {
```


© luv2code LLC

Step 1: Define custom validator rule

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {
```

Don't allow only whitespace

© luv2code LLC

Step 1: Define custom validator rule

Don't allow only whitespace

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {
```


© luv2code LLC

Step 1: Define custom validator rule

Don't allow only whitespace

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {
```

If validation check fails then return validation error(s)

If validation check passes then return null

© luv2code LLC

Step 1: Define custom validator rule

Don't allow only whitespace

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2Sh

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {
```

Map of errors returned from failed validation checks

If validation check fails then return validation error(s)

If validation check passes then return null

© luv2code LLC

Step 1: Define custom validator rule

Don't allow only whitespace

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {

        // check if string only contains whitespace
```

If validation check fails then return validation error(s)

If validation check passes then return null

© luv2code LLC

Step 1: Define custom validator rule

Don't allow only whitespace

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {

        // check if string only contains whitespace
        if ((control.value != null) && (control.value.trim().length === 0)) {

          // invalid, return error object
          return { 'notOnlyWhitespace': true };
        }
```

If validation check fails then return validation error(s)

If validation check passes then return null

© luv2code LLC

Step 1: Define custom validator rule

Don't allow only whitespace

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {

        // check if string only contains whitespace
        if ((control.value != null) && (control.value.trim().length === 0)) {

          // invalid, return error object
          return { 'notOnlyWhitespace': true };
        }
        else {
          // valid, return null
          return null;
        }

    }
}
```

If validation check fails then return validation error(s)

If validation check passes then return null

© luv2code LLC

Step 1: Define custom validator rule

Don't allow only whitespace

File: luv2-shop-validators.ts

```
import { FormControl, ValidationErrors } from '@angular/forms';

export class Luv2ShopValidators {

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {

        // check if string only contains whitespace
        if ((control.value != null) && (control.value.trim().length === 0)) {

          // invalid, return error object
          return { 'notOnlyWhitespace': true };
        }
        else {
          // valid, return null
          return null;
        }

    }
}
```

If validation check fails then return validation error(s)

If validation check passes then return null

Validation error key

The HTML template will check for this error key

© luv2code LLC

Step 2: Specify custom validator rule for form controls

File: checkout.component.ts

```
    this.checkoutFormGroup = this.formBuilder.group({

      customer: this.formBuilder.group({
        'firstName': new FormControl('',
         [Validators.required, Validators.minLength(2), Luv2ShopValidators.notOnlyWhitespace ]
        ),

        ...
    }),
  ...
  });
```


© luv2code LLC

Step 2: Specify custom validator rule for form controls

Our custom validator method name

File: checkout.component.ts

```
    this.checkoutFormGroup = this.formBuilder.group({

      customer: this.formBuilder.group({
        'firstName': new FormControl('',
          [Validators.required, Validators.minLength(2), Luv2ShopValidators.notOnlyWhitespace ]
        ),

        ...
    }),
  ...
  });
```


© luv2code LLC

Step 2: Specify custom validator rule for form controls

File: checkout.component.ts

```
    this.checkoutFormGroup = this.formBuilder.group({

      customer: this.formBuilder.group({
        'firstName': new FormControl('',
          [Validators.required, Validators.minLength(2), Luv2ShopValidators.notOnlyWhitespace ]
        ),

        ...
    }),
  ...
  });
```

Our custom validator method name

```
export class Luv2ShopValidators {

    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {

        // check if string only contains whitespace
        if ((control.value != null) && (control.value.trim().length === 0)) {
```


© luv2code LLC

Step 3: Update HTML template to display error messages

File: checkout.component.html

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required || firstName.errors.notOnlyWhitespace">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```


© luv2code LLC

Step 3: Update HTML template to display error messages

File: checkout.component.html

If validation fails
then display error messages

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required || firstName.errors.notOnlyWhitespace">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```


© luv2code LLC

Step 3: Update HTML template to display error messages

File: checkout.component.html

If validation fails
then display error messages

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required || firstName.errors.notOnlyWhitespace">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```


© luv2code LLC

Step 3: Update HTML template to display error messages

File: checkout.component.html

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required || firstName.errors.notOnlyWhitespace">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
  First Name must be at least 2 characters long
 </div>

</div>
```

If validation fails
then display error messages

Reference to our custom
validation error key

© luv2code LLC

Step 3: Update HTML template to display error messages

File: checkout.component.html

```
<label>First Name</label>
<input formControlName="firstName" type="text">

<div *ngIf="firstName.invalid && (firstName.dirty || firstName.touched)" class="alert alert-danger">

 <div *ngIf="firstName.errors.required || firstName.errors.notOnlyWhitespace">
  First Name is required
 </div>

 <div *ngIf="firstName.errors.minlength">
```

```
    // whitespace validation
    static notOnlyWhitespace(control: FormControl) : ValidationErrors {

        // check if string only contains whitespace
        if ((control.value != null) && (control.value.trim().length === 0)) {

          // invalid, return error object
          return { 'notOnlyWhitespace': true };
        }
        else {
          // valid, return null
```

If validation fails then display error messages

Reference to our custom validation error key

© luv2code LLC