Checkout Form
Populate Country-State Drop-Down Lists

Populating Drop-Down Lists

© luv2code LLC

Populating Drop-Down Lists

- Backend work for REST API is done ... now let's handle front-end

© luv2code LLC

Populating Drop-Down Lists

- Backend work for REST API is done ... now let's handle front-end
- The user will select a country

© luv2code LLC

Populating Drop-Down Lists

- Backend work for REST API is done ... now let's handle front-end
- The user will select a country
- Depending on country selected, populate list of states

© luv2code LLC

Populating Drop-Down Lists

- Backend work for REST API is done ... now let's handle front-end
- The user will select a country
- Depending on country selected, populate list of states

© luv2code LLC

Populating Drop-Down Lists

- Backend work for REST API is done ... now let's handle front-end
- The user will select a country
- Depending on country selected, populate list of states

© luv2code LLC

Development Process - Frontend

Step-By-Step

© luv2code LLC

Development Process - Frontend

Step-By-Step

© luv2code LLC

Development Process - Frontend

Step-By-Step

1. Create TypeScript classes for: `Country` and `State`

© luv2code LLC

Development Process - Frontend

Step-By-Step

1. Create TypeScript classes for: `Country` and `State`

2. Add methods to the form service for countries and states

© luv2code LLC

Development Process - Frontend

Step-By-Step

1. Create TypeScript classes for: `Country` and `State`
2. Add methods to the form service for countries and states
3. Update checkout component to retrieve the countries from service

© luv2code LLC

Development Process - Frontend

Step-By-Step

1. Create TypeScript classes for: `Country` and `State`
2. Add methods to the form service for countries and states
3. Update checkout component to retrieve the countries from service
4. Update HTML template to populate drop-down lists for countries

© luv2code LLC

Development Process - Frontend

Step-By-Step

1. Create TypeScript classes for: `Country` and `State`
2. Add methods to the form service for countries and states
3. Update checkout component to retrieve the countries from service
4. Update HTML template to populate drop-down lists for countries
5. Add event handler for checkout component

© luv2code LLC

Development Process - Frontend

Step-By-Step

1. Create TypeScript classes for: `Country` and `State`
2. Add methods to the form service for countries and states
3. Update checkout component to retrieve the countries from service
4. Update HTML template to populate drop-down lists for countries
5. Add event handler for checkout component
    1. Read the selected country, retrieve list of states based on selected country

© luv2code LLC

Development Process - Frontend

Step-By-Step

1. Create TypeScript classes for: `Country` and `State`
2. Add methods to the form service for countries and states
3. Update checkout component to retrieve the countries from service
4. Update HTML template to populate drop-down lists for countries
5. Add event handler for checkout component
    1. Read the selected country, retrieve list of states based on selected country
6. Update HTML template to populate drop-down lists for states

© luv2code LLC

Backend REST APIs

http://localhost:8080/api/countries


```
{
  "_embedded" : {
    "countries" : [ {
      "id" : 1,
      "code" : "BR",
      "name" : "Brazil",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/1"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/1"
        }
      }
    }, {
      "id" : 2,
      "code" : "CA",
      "name" : "Canada",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/2"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/2"
        }
      }
    }, {
```


http://localhost:8080/api/states/search/findByCountryCode?code=IN


```
{
  "_embedded" : {
    "states" : [ {
      "id" : 57,
      "name" : "Andhra Pradesh",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/states/57"
        },
        "state" : {
          "href" : "http://localhost:8080/api/states/57"
        },
        "country" : {
          "href" : "http://localhost:8080/api/states/57/country"
        }
      }
    }, {
      "id" : 58,
      "name" : "Arunachal Pradesh",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/states/58"
        },
        "state" : {
          "href" : "http://localhost:8080/api/states/58"
        },
        "country" : {
          "href" : "http://localhost:8080/api/states/58/country"
        }
      }
    }, {
```


© luv2code LLC

Step 1: Create TypeScript classes for: Country and State

© luv2code LLC

Step 1: Create TypeScript classes for: Country and State

File: country.ts

```
export class Country {
    id: number;
    code: string;
    name: string;
}
```


© luv2code LLC

Step 1: Create TypeScript classes for: Country and State

File: country.ts

```
export class Country {
    id: number;
    code: string;
    name: string;
}
```

JSON from REST API


```
{
  "id" : 4,
  "code" : "IN",
  "name" : "India",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/api/countries/4"
    },
    "country" : {
      "href" : "http://localhost:8080/api/countries/4"
    },
    "states" : {
      "href" : "http://localhost:8080/api/countries/4/states"
    }
  }
}
```


© luv2code LLC

Step 1: Create TypeScript classes for: Country and State

File: country.ts

```
export class Country {
    id: number;
    code: string;
    name: string;
}
```

File: state.ts

```
export class State {
    id: number;
    name: string;
}
```

JSON from REST API


```
{
  "id" : 4,
  "code" : "IN",
  "name" : "India",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/api/countries/4"
    },
    "country" : {
      "href" : "http://localhost:8080/api/countries/4"
    },
    "states" : {
      "href" : "http://localhost:8080/api/countries/4/states"
    }
  }
}
```


© luv2code LLC

Step 1: Create TypeScript classes for: Country and State

File: country.ts

```
export class Country {
    id: number;
    code: string;
    name: string;
}
```

File: state.ts

```
export class State {
    id: number;
    name: string;
}
```

JSON from REST API


```
{
  "id" : 4,
  "code" : "IN",
  "name" : "India",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/api/countries/4"
    },
    "country" : {
      "href" : "http://localhost:8080/api/countries/4"
    },
    "states" : {
      "href" : "http://localhost:8080/api/countries/4/states"
    }
  }
}
```


JSON from REST API


```
{
  "id" : 57,
  "name" : "Andhra Pradesh",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/api/states/57"
    },
    "state" : {
      "href" : "http://localhost:8080/api/states/57"
    },
    "country" : {
      "href" : "http://localhost:8080/api/states/57/country"
    }
  }
}
```


© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor(private httpClient: HttpClient) { }
```


© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor(private httpClient: HttpClient) { }

  getCountries(): Observable<Country[]> {

    return this.httpClient.get<GetResponseCountries>(this.countriesUrl).pipe(
      map(response => response._embedded.countries)
    );
  }
  ...
}
```

Get Countries

Returns an observable

Map the JSON data from Spring Data REST to `Country` array

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor(private httpClient: HttpClient) { }

  getCountries(): Observable<Country[]> {

    return this.httpClient.get<GetResponseCountries>(this.countriesUrl).pipe(
      map(response => response._embedded.countries)
    );
  }
  ...
}

interface GetResponseCountries {
  _embedded: {
    countries: Country[];
  }
}
```

Get Countries

Returns an observable

Map the JSON data from Spring Data REST to `Country` array

Unwraps the JSON from Spring Data REST `_embedded` entry

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormServ

  private countriesUrl = 'htt
  private statesUrl = 'http:/

  constructor(private httpCli

  getCountries(): Observable<

    return this.httpClient.ge                                  pipe(
      map(response => respons
    );
  }
  ...
}

interface GetResponseCountries {
  _embedded: {
    countries: Country[];
  }
}
```

```
{
  "_embedded" : {
    "countries" : [ {
      "id" : 1,
      "code" : "BR",
      "name" : "Brazil",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/1"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/1"
        }
      }
    }, {
      "id" : 2,
      "code" : "CA",
      "name" : "Canada",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/2"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/2"
        }
      }
    }, {
```

Get Countries

Returns an observable

Map the JSON data from Spring Data REST to `Country` array

Unwraps the JSON from Spring Data REST `_embedded` entry

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormServ

  private countriesUrl = 'htt
  private statesUrl = 'http:/

  constructor(private httpCli

  getCountries(): Observable<

    return this.httpClient.ge                                  pipe(
      map(response => respons
    );
  }
  ...
}

interface GetResponseCountries {
  _embedded: {
    countries: Country[];
  }
}
```


```
{
  "_embedded" : {
    "countries" : [ {
      "id" : 1,
      "code" : "BR",
      "name" : "Brazil",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/1"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/1"
        }
      }
    }, {
      "id" : 2,
      "code" : "CA",
      "name" : "Canada",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/2"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/2"
        }
      }
```


Get Countries

Returns an observable

Map the JSON data from Spring Data REST to `Country` array

Unwraps the JSON from Spring Data REST `_embedded` entry

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor(private httpClient: HttpClient) { }

  getCountries(): Observable<Country[]> {

    return this.httpClient.get<GetResponseCountries>(this.countriesUrl).pipe(
      map(response => response._embedded.countries)
    );
  }
  ...
}

interface GetResponseCountries {
  _embedded: {
    countries: Country[];
  }
}
```

Get Countries

Returns an observable

Map the JSON data from Spring Data REST to `Country` array

Unwraps the JSON from Spring Data REST `_embedded` entry

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor() { }

  getStates(theCountryCode: string): Observable<State[]> {

    // search url
    const searchStatesUrl = `${this.statesUrl}/search/findByCountryCode?code=${theCountryCode}`;

    return this.httpClient.get<GetResponseStates>(searchStatesUrl).pipe(
      map(response => response._embedded.states)
    );
  }
  ...
}

interface GetResponseStates {
  _embedded: {
    states: State[];
  }
}
```


© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor() { }

  getStates(theCountryCode: string): Observable<State[]> {

    // search url
    const searchStatesUrl = `${this.statesUrl}/search/findByCountryCode?code=${theCountryCode}`;

    return this.httpClient.get<GetResponseStates>(searchStatesUrl).pipe(
      map(response => response._embedded.states)
    );
  }
  ...
}

interface GetResponseStates {
  _embedded: {
    states: State[];
  }
}
```

Get States

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor() { }

  getStates(theCountryCode: string): Observable<State[]> {

    // search url
    const searchStatesUrl = `${this.statesUrl}/search/findByCountryCode?code=${theCountryCode}`;

    return this.httpClient.get<GetResponseStates>(searchStatesUrl).pipe(
      map(response => response._embedded.states)
    );
  }
  ...
}

interface GetResponseStates {
  _embedded: {
    states: State[];
  }
}
```

Get States

Returns an observable

Map the JSON data from Spring Data REST to `State` array

© luv2code LLC

Step 2: Add methods to the form service for countries and states

File: luv2-shop-form.service.ts

```
export class Luv2ShopFormService {

  private countriesUrl = 'http://localhost:8080/api/countries';
  private statesUrl = 'http://localhost:8080/api/states/';

  constructor() { }

  getStates(theCountryCode: string): Observable<State[]> {

    // search url
    const searchStatesUrl = `${this.statesUrl}/search/findByCountryCode?code=${theCountryCode}`;

    return this.httpClient.get<GetResponseStates>(searchStatesUrl).pipe(
      map(response => response._embedded.states)
    );
  }
  ...
}

interface GetResponseStates {
  _embedded: {
    states: State[];
  }
}
```

Get States

Returns an observable

Map the JSON data from Spring Data REST to `State` array

Unwraps the JSON from Spring Data REST `_embedded` entry

© luv2code LLC

Step 3: Update checkout component to retrieve countries from service

File: checkout.component.ts

© luv2code LLC

Step 3: Update checkout component to retrieve countries from service

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  ...
```


© luv2code LLC

Step 3: Update checkout component to retrieve countries from service

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  ...
  countries: Country[] = [];

  constructor(private formBuilder: FormBuilder,
```


© luv2code LLC

Step 3: Update checkout component to retrieve countries from service

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  ...
  countries: Country[] = [];

  constructor(private formBuilder: FormBuilder,
              private luv2ShopFormService: Luv2ShopFormService) { }

  ngOnInit(): void {
```


© luv2code LLC

Step 3: Update checkout component to retrieve countries from service

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  ...
  countries: Country[] = [];

  constructor(private formBuilder: FormBuilder,
              private luv2ShopFormService: Luv2ShopFormService) { }

  ngOnInit(): void {
    ...

    // populate countries
```


© luv2code LLC

Step 3: Update checkout component to retrieve countries from service

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  ...
  countries: Country[] = [];

  constructor(private formBuilder: FormBuilder,
              private luv2ShopFormService: Luv2ShopFormService) { }

  ngOnInit(): void {
    ...

    // populate countries
    this.luv2ShopFormService.getCountries().subscribe(
      data => {
        console.log("Retrieved countries: " + JSON.stringify(data));
        this.countries = data;
      }
    );

  }

}
```


© luv2code LLC

Step 3: Update checkout component to retrieve countries from service

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  ...
  countries: Country[] = [];

  constructor(private formBuilder: FormBuilder,
              private luv2ShopFormService: Luv2ShopFormService) { }

  ngOnInit(): void {
    ...

    // populate countries
    this.luv2ShopFormService.getCountries().subscribe(
      data => {
        console.log("Retrieved countries: " + JSON.stringify(data));
        this.countries = data;
      }
    );

  }

}
```

When form is initially displayed, populate the countries

© luv2code LLC

Step 4: Update HTML template to populate drop-down lists for countries

File: checkout.component.html

```
<div class="col-md-2"> <label>Country</label></div>

<div class="input-space">
     <select formControlName="country">
        <option *ngFor="let country of countries" [ngValue]="country" >
          {{ country.name }}
        </option>
     </select>
</div>
```


© luv2code LLC

Step 4: Update HTML template to populate drop-down lists for countries

File: checkout.component.html

```
<div class="col-md-2"> <label>Country</label></div>

<div class="input-space">
     <select formControlName="country">
        <option *ngFor="let country of countries" [ngValue]="country" >
          {{ country.name }}
        </option>
     </select>
</div>
```

Loop over array

© luv2code LLC

Step 5: Update checkout component, add event handler

File: checkout.component.html

```
<div class="col-md-2"> <label>Country</label></div>

<div class="input-space">
     <select formControlName="country" (change)="getStates('shippingAddress')">
        <option *ngFor="let country of countries" [ngValue]="country" >
          {{ country.name }}
        </option>
     </select>
</div>
```


© luv2code LLC

Step 5: Update checkout component, add event handler

For `change` event call method `getStates(..)`

File: checkout.component.html

```
<div class="col-md-2"> <label>Country</label></div>

<div class="input-space">
     <select formControlName="country" (change)="getStates('shippingAddress')">
        <option *ngFor="let country of countries" [ngValue]="country" >
          {{ country.name }}
        </option>
     </select>
</div>
```


© luv2code LLC

Step 5: Update checkout component, add event handler

File: checkout.component.ts

© luv2code LLC

Step 5: Update checkout component, add event handler

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  checkoutFormGroup: FormGroup;
  shippingAddressStates: State[] = [];
  billingAddressStates: State[] = [];
```


© luv2code LLC

Step 5: Update checkout component, add event handler

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  checkoutFormGroup: FormGroup;
  shippingAddressStates: State[] = [];
  billingAddressStates: State[] = [];

  getStates(formGroupName: string) {

    const formGroup = this.checkoutFormGroup.get(formGroupName);
    const countryCode = formGroup.value.country.code;
```

Read the selected country code from the form

© luv2code LLC

Step 5: Update checkout component, add event handler

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  checkoutFormGroup: FormGroup;
  shippingAddressStates: State[] = [];
  billingAddressStates: State[] = [];

  getStates(formGroupName: string) {

    const formGroup = this.checkoutFormGroup.get(formGroupName);
    const countryCode = formGroup.value.country.code;

    this.formDataService.getStates(countryCode).subscribe(
      data => {

        if (formGroupName === 'shippingAddress') {
          this.shippingAddressStates = data;
        }
        else {
          this.billingAddressStates = data;
        }
```

Read the selected country code from the form

Retrieve the states for the selected country code

© luv2code LLC

Step 5: Update checkout component, add event handler

File: checkout.component.ts

```
export class CheckoutComponent implements OnInit {

  checkoutFormGroup: FormGroup;
  shippingAddressStates: State[] = [];
  billingAddressStates: State[] = [];

  getStates(formGroupName: string) {

    const formGroup = this.checkoutFormGroup.get(formGroupName);
    const countryCode = formGroup.value.country.code;

    this.formDataService.getStates(countryCode).subscribe(
      data => {

        if (formGroupName === 'shippingAddress') {
          this.shippingAddressStates = data;
        }
        else {
          this.billingAddressStates = data;
        }

        // select first item by default
        formGroup.get('state').setValue(data[0]);
      }
    );
  }
```

Read the selected country code from the form

Retrieve the states for the selected country code

© luv2code LLC

Step 6: Update HTML template to populate drop-down lists for states

File: checkout.component.html

```
<div class="col-md-2"> <label>State</label></div>

<div class="input-space">
     <select formControlName="state">
        <option *ngFor="let state of shippingAddressStates" [ngValue]="state" >
          {{ state.name }}
        </option>
     </select>
</div>
```


© luv2code LLC

Step 6: Update HTML template to populate drop-down lists for states

File: checkout.component.html

```
<div class="col-md-2"> <label>State</label></div>

<div class="input-space">
     <select formControlName="state">
        <option *ngFor="let state of shippingAddressStates" [ngValue]="state" >
          {{ state.name }}
        </option>
     </select>
</div>
```

Loop over array

© luv2code LLC