```
<div class="form-group"
    [class.has-error]="department.touched && department.invalid">
  <label for="department" class="control-label">Department</label>
  <select required #department="ngModel" name="department"
        [(ngModel)]="employee.department" id="department"
        class="form-control">
    <option [ngValue]="null">Select Department</option>
    <option *ngFor="let dept of departments" [value]="dept.id">
     {{dept.name}}
    </option>
  </select>
  <span class="help-block"
      *ngIf="department.touched && department.errors?.required">
    Department is required
  </span>
</div>
```

Code explanation :

- `<option [ngValue]="null">Select Department</option>`. Notice we are using ngValue instead of value. If you use value, null is treated as a string and not as a null. Hence the required validation does not work. Along with using ngValue, also make sure you set the department property on the employee model object to null.
- `<option *ngFor="let dept of departments" [value]="dept.id">{{dept.name}} </option>`. Here we are using value instead of ngValue, because we just want the selected department id as a string. If you want the department object itself instead of just the department id string, then use ngValue.
- `<option *ngFor="let dept of departments" [ngValue]="dept">{{dept.name}} </option>`. In this example we are using ngValue and binding it to the dept object. If we select a department now, we get the selected department object.
  "department": { "id": 3, "name": "IT" }
- Use the disabled attribute, if you do not want the user to be able to select the "Select Department" option.
  `<option **disabled** [ngValue]="null">Select Department</option>`

Please note : The built-in required validator will only work with the SELECT element, if the default option value is null.
`<option disabled [ngValue]=**"null"**>Select Department</option>`