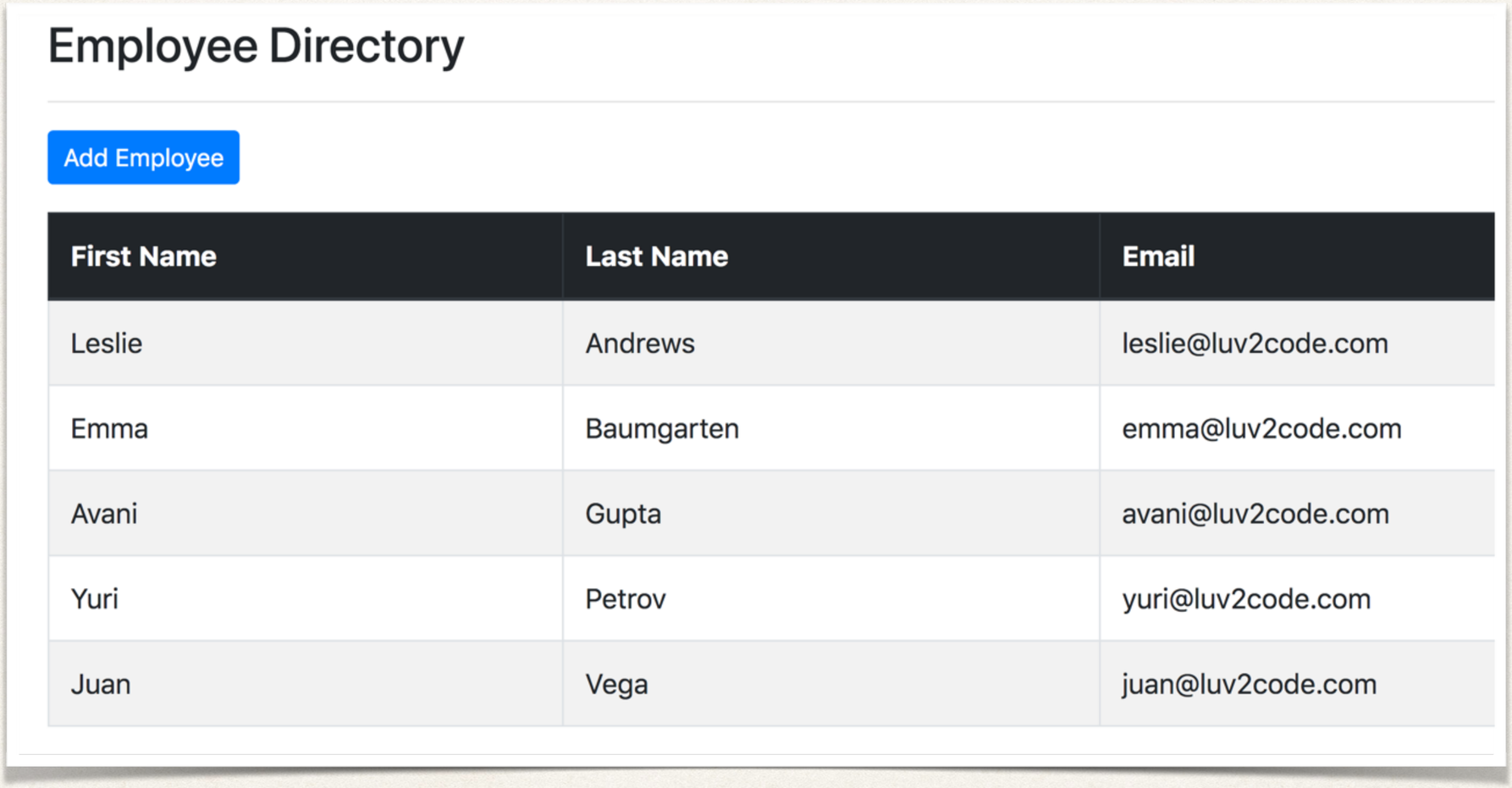
Thymeleaf - Add Employee

Add Employee - DEMO



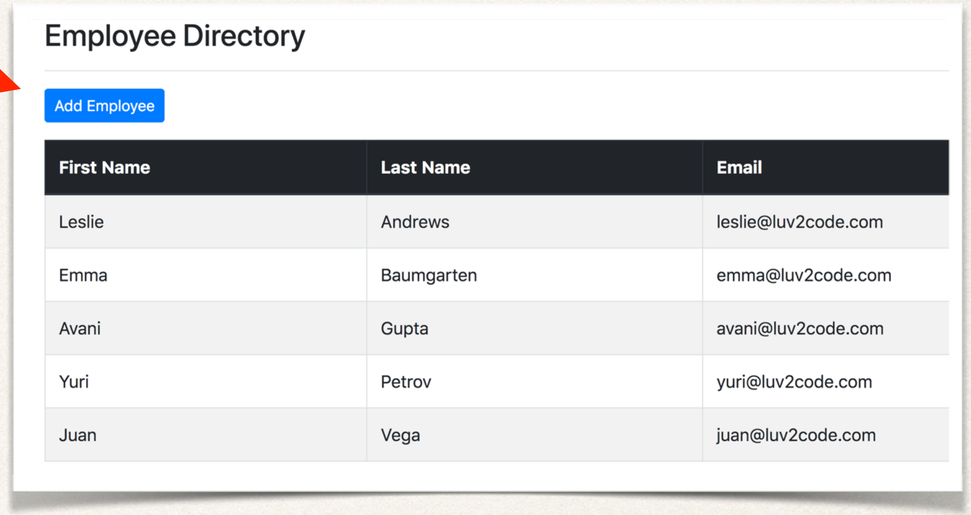
Add Employee

Step-By-Step

Add Employee

Step-By-Step

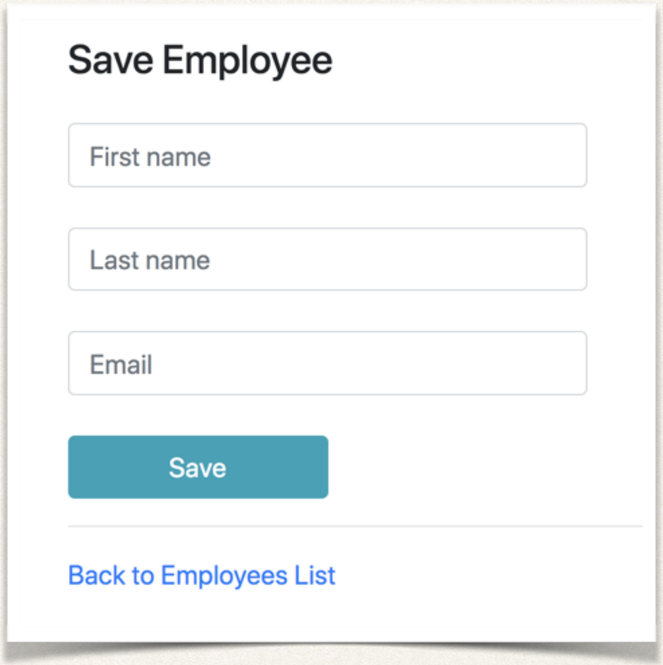
1. New **Add Employee** button for list-employees.html



Add Employee

1. New **Add Employee** button for list-employees.html

2. Create HTML form for new employee



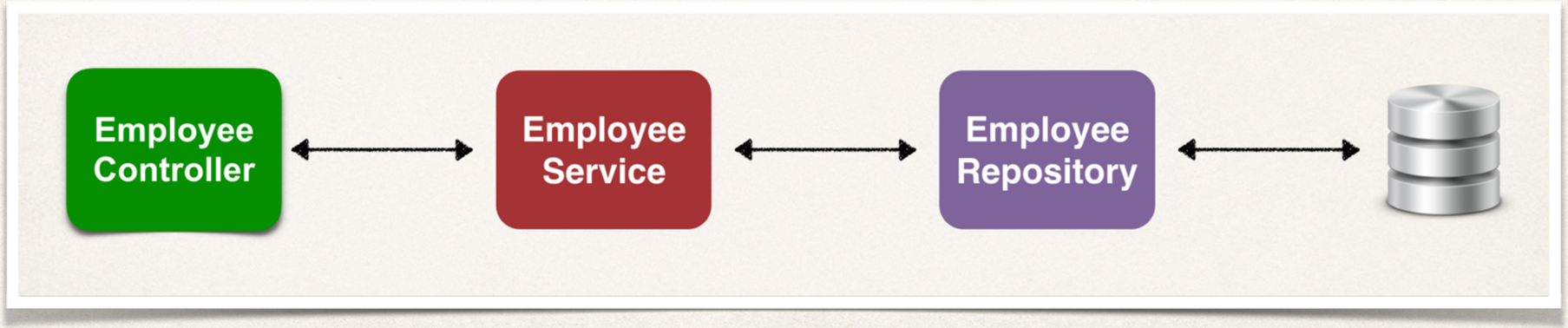
Add Employee

Step-By-Step

1. New **Add Employee** button for list-employees.html

2. Create HTML form for new employee

3. Process form data to save employee



Step 1: New "Add Employee" button

Step 1: New "Add Employee" button

- **Add Employee** button will href link to
  - request mapping ***`/employees/showFormForAdd`***

Step 1: New "Add Employee" button

- **Add Employee** button will href link to
  - request mapping ***/employees/showFormForAdd***

```
<a th:href="@{/employees/showFormForAdd}">
  Add Employee
</a>
```

Add Employee

Step 1: New "Add Employee" button

- **Add Employee** bu
  - request mapping

@ symbol
Reference context path of your application
(app root)

```
<a th:href="@{/employees/showFormForAdd}">
  Add Employee
</a>
```

Add Employee

Step 1: New "Add Employee" button

- **Add Employee** button will href link to
  - request mapping ***/employees/showFormForAdd***

```
<a th:href="@{/employees/showFormForAdd}">
  Add Employee
</a>
```

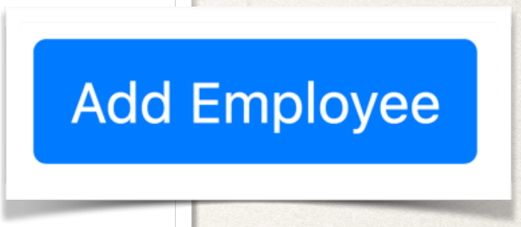
Add Employee

Apply Bootstrap styles

Step 1: New "Add Employee" button

- **Add Employee** button will href link to
  - request mapping ***/employees/showFormForAdd***

```
<a th:href="@{/employees/showFormForAdd}"
 class="btn btn-primary btn-sm mb-3">
 Add Employee
</a>
```

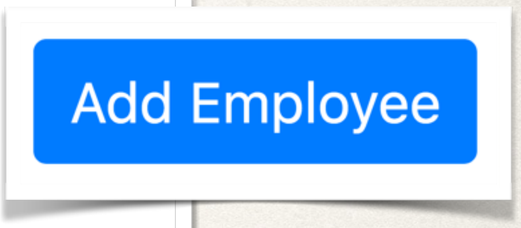


Apply Bootstrap styles

Step 1: New "Add Employee" button

- **Add Employee** button will href link to
  - request mapping ***/employees/showFormForAdd***

```
<a th:href="@{/employees/showFormForAdd}"
 class="btn btn-primary btn-sm mb-3">
 Add Employee
</a>
```



Apply Bootstrap styles

Button
Button Primary
Button Small
Margin Bottom, 3 pixels

Step 1: New "Add Employee" button

- **Add Employee** button will href link to
  - requ

Docs on Bootstrap styles: www.getbootstrap.com

```
<a th:href="@{/employees/showFormForAdd}"
 class="btn btn-primary btn-sm mb-3">
 Add Employee
</a>
```

Add Employee

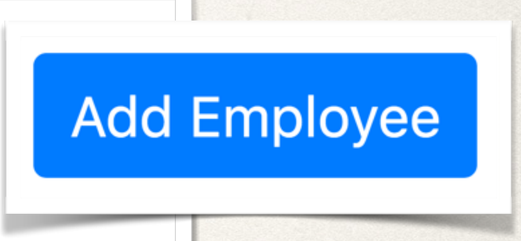
Apply Bootstrap styles

Button
Button Primary
Button Small
Margin Bottom, 3 pixels

Step 1: New "Add Employee" button

- **Add Employee** button will href link to
  - request mapping ***/employees/showFormForAdd***

```
<a th:href="@{/employees/showFormForAdd}"
 class="btn btn-primary btn-sm mb-3">
 Add Employee
</a>
```

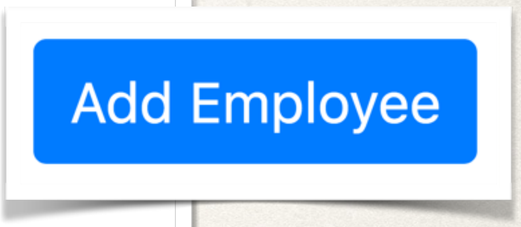


Step 1: New "Add Employee" button

- **Add Employee** button will href li
  - request mapping ***/employees/***

TODO:
Add controller request mapping for
`/employees/showFormForAdd`

```
<a th:href="@{/employees/showFormForAdd}"
 class="btn btn-primary btn-sm mb-3">
 Add Employee
</a>
```



Showing Form

Showing Form

In your Spring Controller

Showing Form

In your Spring Controller

- Before you show the form, you must add a ***model attribute***

Showing Form

In your Spring Controller

- Before you show the form, you must add a ***model attribute***

- This is an object that will hold form data for the ***data binding***

Controller code to show form

Controller code to show form

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {
```

Controller code to show form

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

  @GetMapping("/showFormForAdd")
  public String showFormForAdd(Model theModel) {
```

Controller code to show form

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

  @GetMapping("/showFormForAdd")
  public String showFormForAdd(Model theModel) {

    // create model attribute to bind form data
    Employee theEmployee = new Employee();
```

Controller code to show form

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

  @GetMapping("/showFormForAdd")
  public String showFormForAdd(Model theModel) {

    // create model attribute to bind form data
    Employee theEmployee = new Employee();

    theModel.addAttribute("employee", theEmployee);
```

Controller code to show form

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

  @GetMapping("/showFormForAdd")
  public String showFormForAdd(Model theModel) {

    // create model attribute to bind form data
    Employee theEmployee = new Employee();

    theModel.addAttribute("employee", theEmployee);
```

Our Thymleaf template will access this data for binding form data

Controller code to show form

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

  @GetMapping("/showFormForAdd")
  public String showFormForAdd(Model theModel) {

    // create model attribute to bind form data
    Employee theEmployee = new Employee();

    theModel.addAttribute("employee", theEmployee);

    return "/employees/employee-form";
  }
  ...
}
```

Our Thymleaf template will access this data for binding form data

Controller code to show form

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

  @GetMapping("/showFormForAdd")
  public String showFormForAdd(Model theModel) {

    // create model attribute to bind form data
    Employee theEmployee = new Employee();

    theModel.addAttribute("employee", theEmployee);

    return "/employees/employee-form";
  }
  ...
}
```

Our Thymleaf template will access this data for binding form data

src/main/resources/templates/employees/employee-form.html

Thymeleaf and Spring MVC Data Binding

Thymeleaf and Spring MVC Data Binding

- Thymeleaf has special expressions for binding Spring MVC form data

Thymeleaf and Spring MVC Data Binding

- Thymeleaf has special expressions for binding Spring MVC form data
- Automatically setting / retrieving data from a Java object

Thymeleaf Expressions

Thymeleaf Expressions

- Thymeleaf expressions can help you build the HTML form :-)

Thymeleaf Expressions

- Thymeleaf expressions can help you build the HTML form :-)

Expression	Description
`th:action`	Location to send form data

Thymeleaf Expressions

- Thymeleaf expressions can help you build the HTML form :-)

Expression	Description
`th:action`	Location to send form data
`th:object`	Reference to model attribute

Thymeleaf Expressions

- Thymeleaf expressions can help you build the HTML form :-)

Expression	Description
`th:action`	Location to send form data
`th:object`	Reference to model attribute
`th:field`	Bind input field to a property on model attribute

Thymeleaf Expressions

- Thymeleaf expressions can help you build the HTML form :-)

Expression	Description
`th:action`	Location to send form data
`th:object`	Reference to model attribute
`th:field`	Bind input field to a property on model attribute
more ….	*See - **www.luv2code.com/thymeleaf-create-form***

Step 2: Create HTML form for new employee

Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
      th:object="${employee}" method="POST">


</form>
```

Step 2: Create HTML form for new employee

Empty place holder
Thymeleaf will handle real work

```
<form action="#" th:action="@{/employees/save}"
      th:object="${employee}" method="POST">


</form>
```

Step 2: Create HTML form for new employee

Empty place holder
Thymeleaf will handle real work

Real work
Send form data to
/employees/save

```
<form action="#" th:action="@{/employees/save}"
      th:object="${employee}" method="POST">


</form>
```

Step 2: Create HTML form for new employee

Empty place holder
Thymeleaf will handle real work

Real work
Send form data to
/employees/save

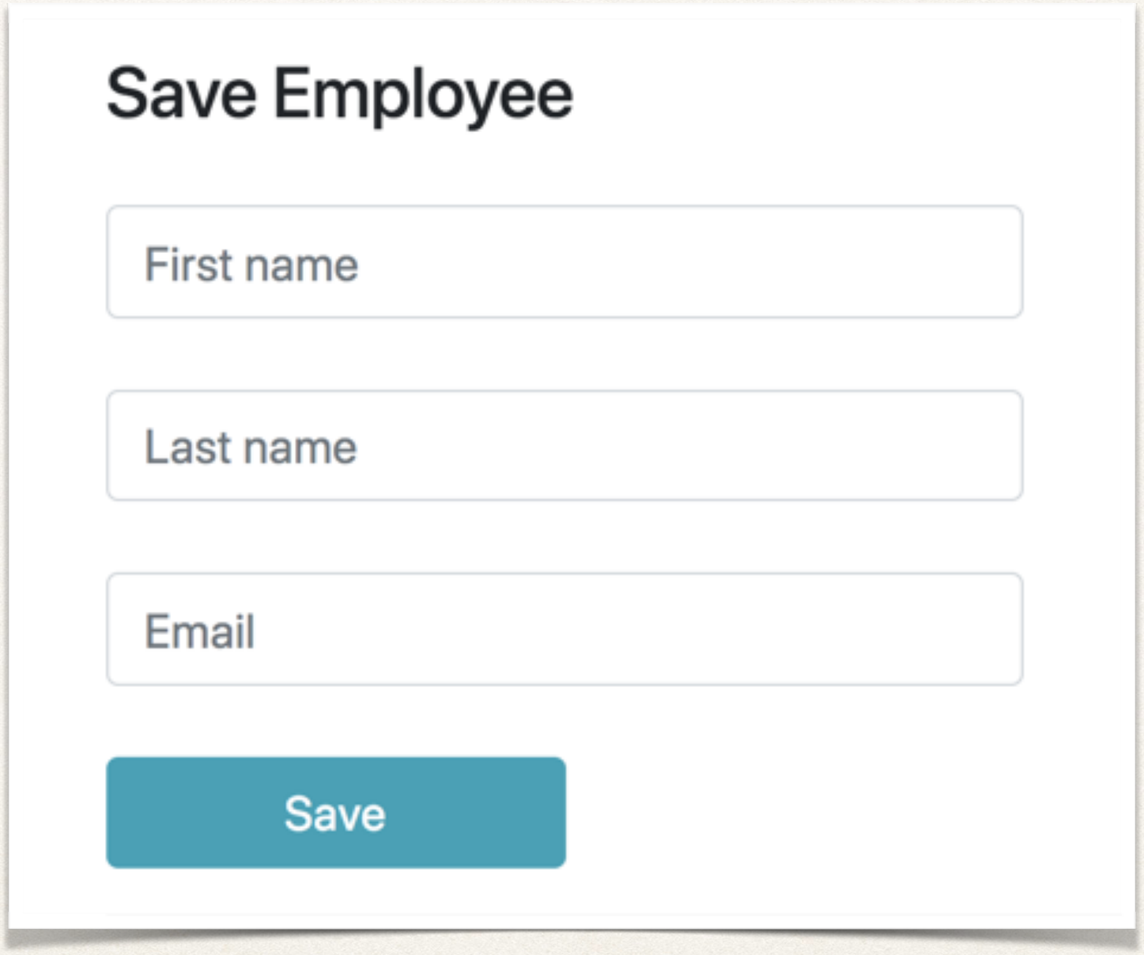
```
<form action="#" th:action="@{/employees/save}"
      th:object="${employee}" method="POST">


</form>
```

Our model attribute

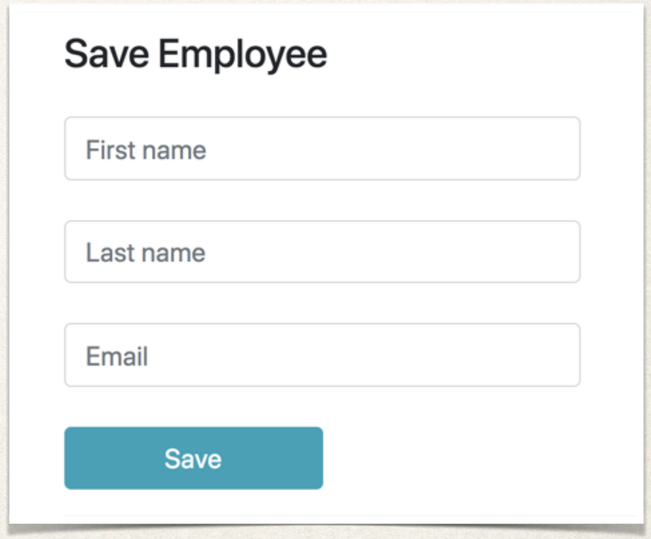
```
theModel.addAttribute("employee", theEmployee);
```

Step 2: Create HTML form for new employee

Step 2: Create HTML form for new employee

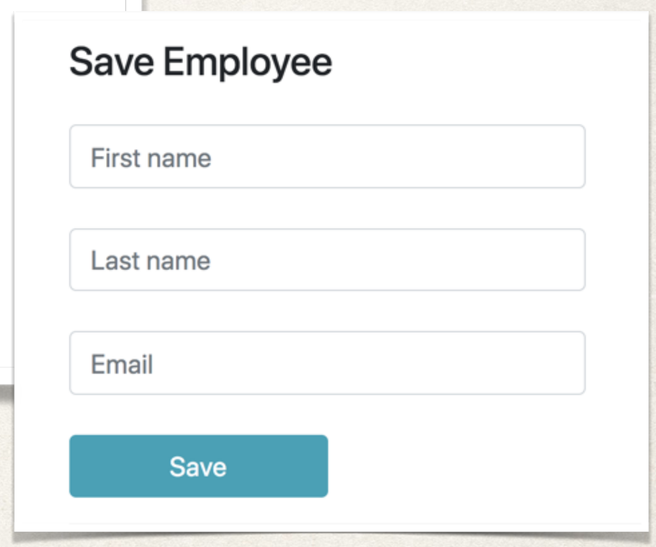


Step 2: Create HTML form for new employee



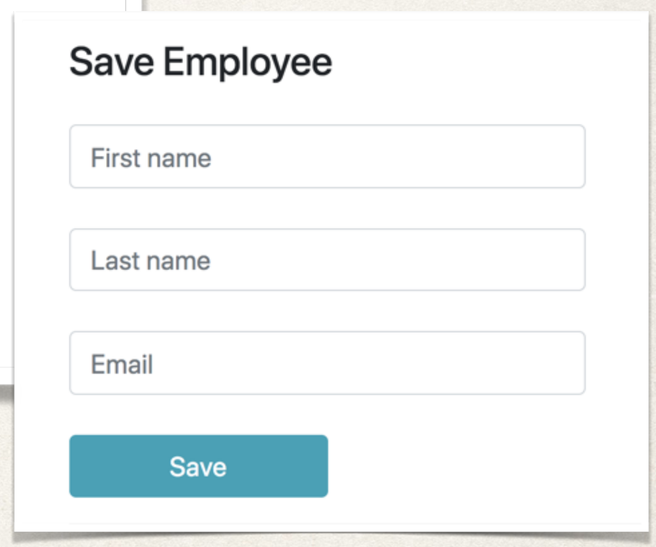
Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
            th:object="${employee}" method="POST">
```



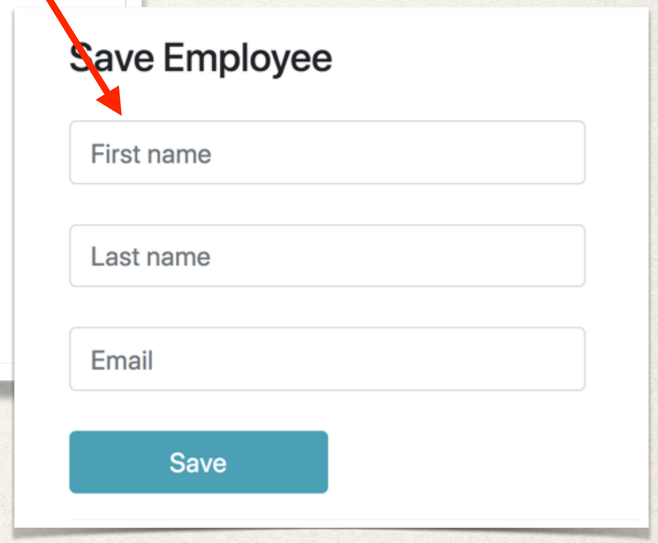
Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">
```



Step 2: Create HTML form for new employee

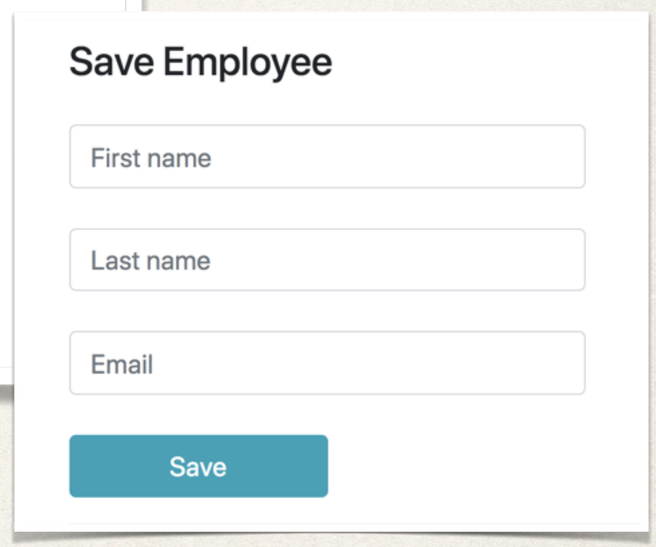
```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">
```



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">
```

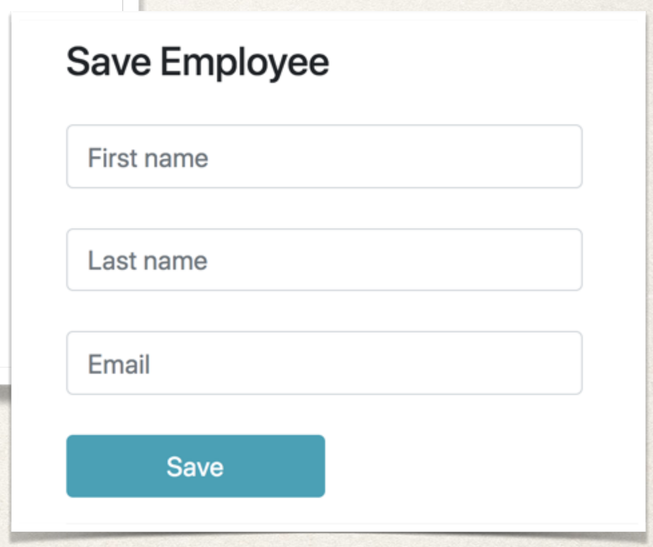
*{...}
Selects property on referenced
th:object



Step 2: Create HTML form for new employee

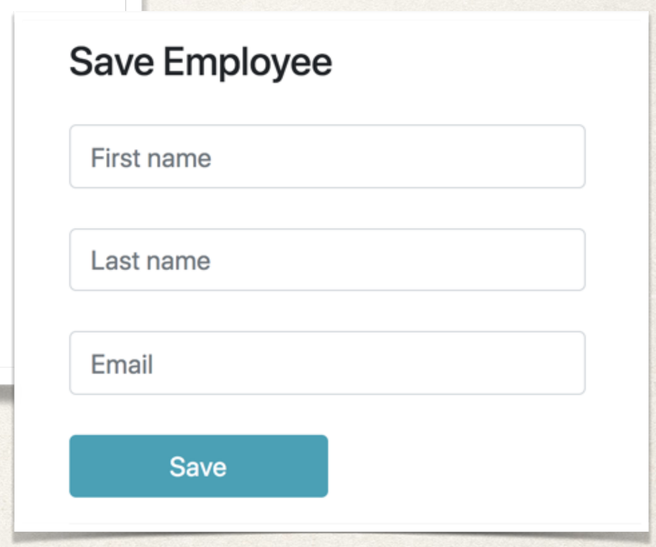
```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">
```

*{...}
Selects property on referenced
th:object



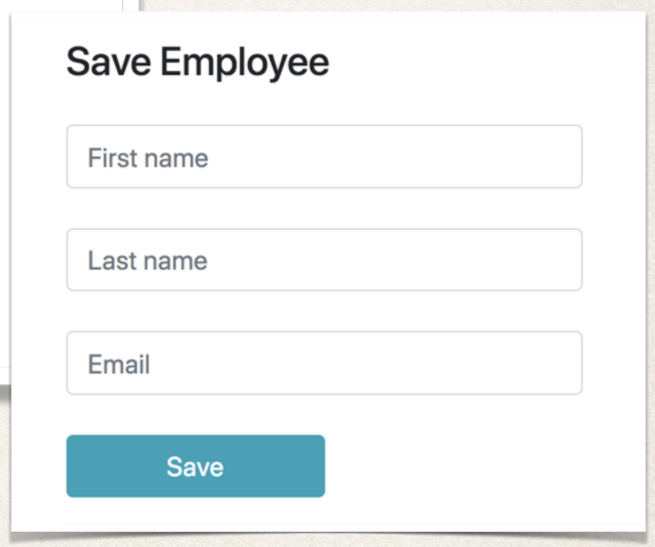
Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">


  <input type="text" th:field="*{lastName}" placeholder="Last name">


  <input type="text" th:field="*{email}" placeholder="Email">
```



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">


  <input type="text" th:field="*{lastName}" placeholder="Last name">


  <input type="text" th:field="*{email}" placeholder="Email">


  <button type="submit">Save</button>

</form>
```



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">


  <input type="text" th:field="*{lastName}" placeholder="Last name">


  <input type="text" th:field="*{email}" placeholder="Email">


  <button type="submit">Save</button>

</form>
```

Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">

  <input type="text" th:field="*{lastName}" placeholder="Last name">

  <input type="text" th:field="*{email}" placeholder="Email">

  <button type="submit">Save</button>

</form>
```

1

When form is **loaded**, will call:

employee.getFirstName()
...
employee.getLastName

Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
            th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">

  <input type="text" th:field="*{lastName}" placeholder="Last name">

  <input type="text" th:field="*{email}" placeholder="Email">

  <button type="submit">Save</button>

</form>
```

1

When form is **loaded**, will call:

employee.getFirstName()
...
employee.getLastName

2

When form is **submitted**, will call:

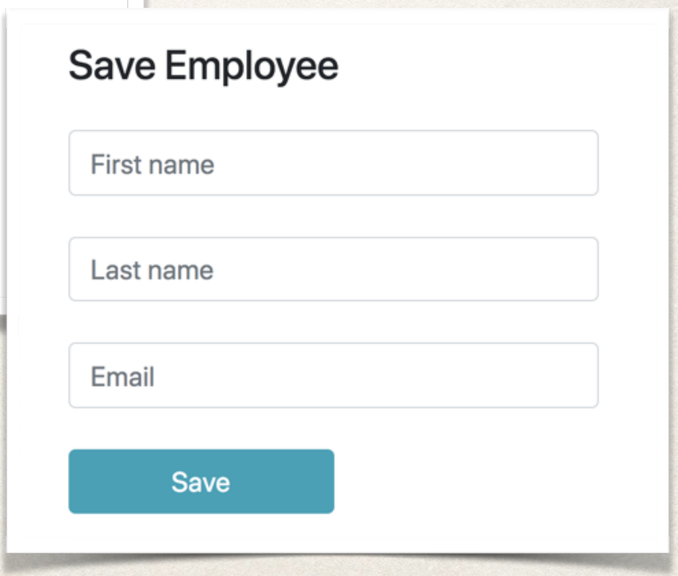
employee.setFirstName(...)
...
employee.setLastName(...)

Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">


  <input type="text" th:field="*{lastName}" placeholder="Last name">


  <input type="text" th:field="*{email}" placeholder="Email">


  <button type="submit">Save</button>

</form>
```

Step 2: Create HTML form for new employee

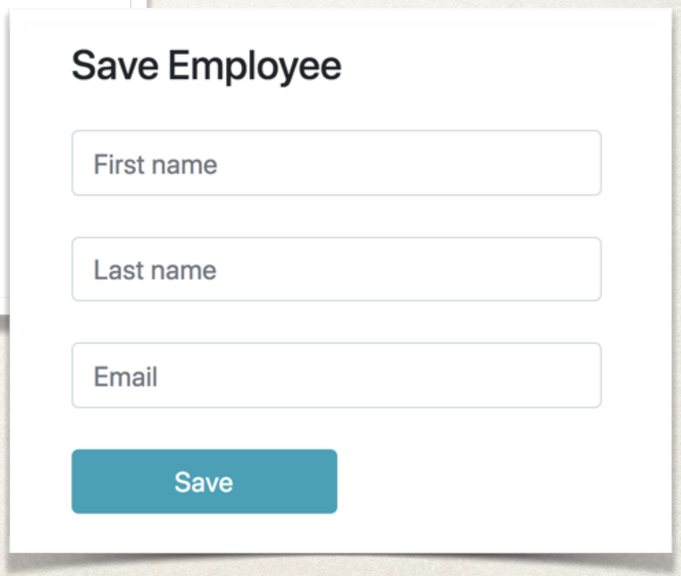
```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">


  <input type="text" th:field="*{lastName}" placeholder="Last name">


  <input type="text" th:field="*{email}" placeholder="Email">


  <button type="submit">Save</button>

</form>
```



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name">


  <input type="text" th:field="*{lastName}" placeholder="Last name">


  <input type="text" th:field="*{email}" placeholder="Email">

  <button type="submit">Save</button>

</form>
```

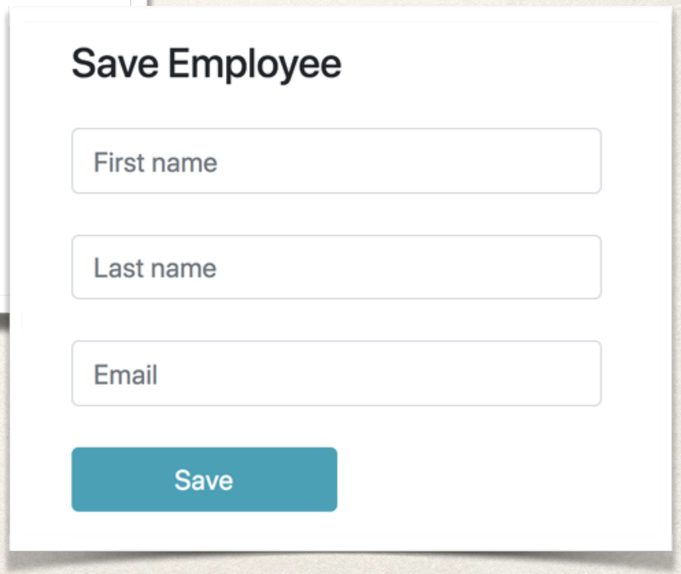
Apply Bootstrap styles



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name"
         class="form-control mb-4 col-4">




</form>
```

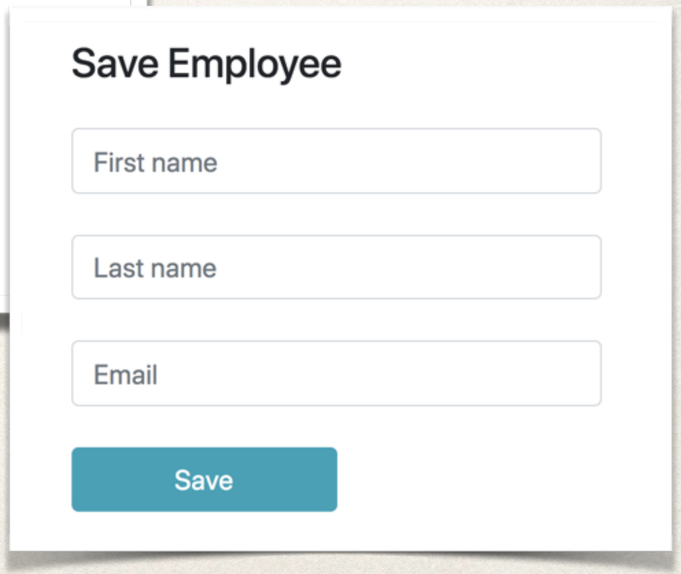
Apply Bootstrap styles



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name"
         class="form-control mb-4 col-4">




</form>
```

Apply Bootstrap styles

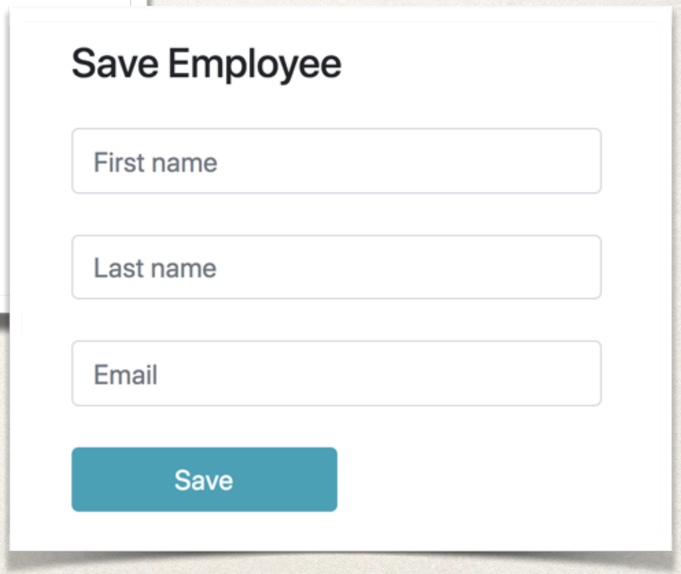


Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name"
         class="form-control mb-4 col-4">

</form>
```

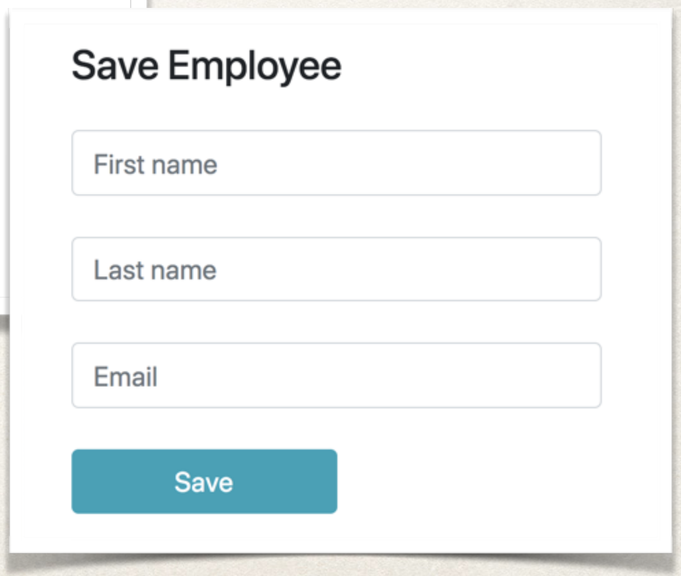
Apply Bootstrap styles

Form control
Margin Bottom, 4 pixels
Column Span 4



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name"
         class="form-control mb-4 col-4">









 </form>
```

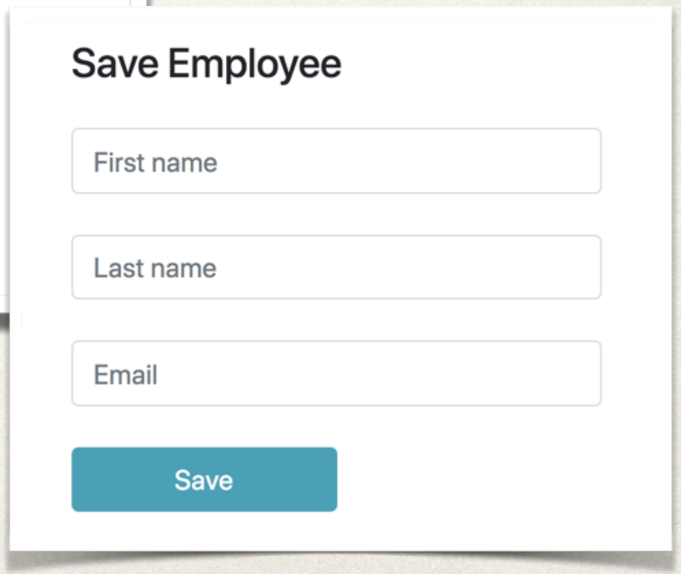


Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholc
         class="form-control mb-4 col-4">

  <input type="text" th:field="*{lastName}" placeholde
         class="form-control mb-4 col-4">

  <input type="text" th:field="*{email}" placeholder="Email"
         class="form-control mb-4 col-4">

  <button type="submit" class="btn btn-info col-2">Save</button>

</form>
```

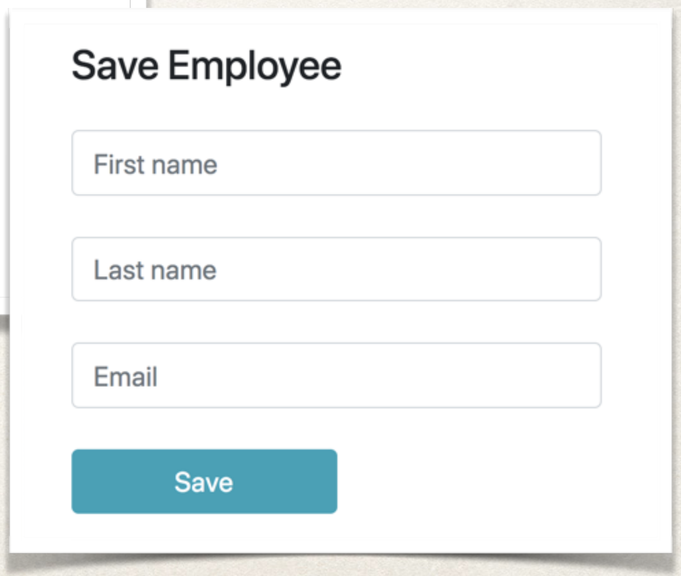
Button
Button Info
Column Span 2

Apply Bootstrap styles



Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name"
         class="form-control mb-4 col-4">

  <input type="text" th:field="*{lastName}" placeholder="Last name"
         class="form-control mb-4 col-4">

  <input type="text" th:field="*{email}" placeholder="Email"
         class="form-control mb-4 col-4">

  <button type="submit" class="btn btn-info col-2">Save</button>

 </form>
```

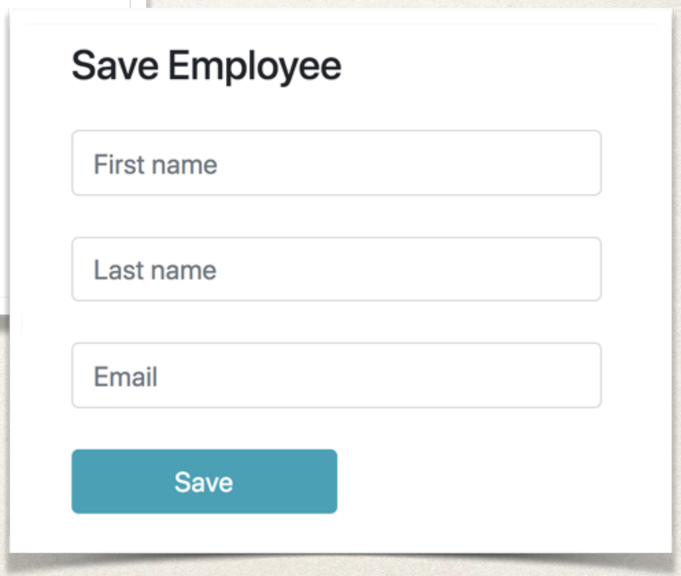


Step 2: Create HTML form for new employee

```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <input type="text" th:field="*{firstName}" placeholder="First name"
         class="form-control mb-4 col-4">

  <input type="text
         class="for

  <input type="text
         class="for

  <button type="submit" class="btn btn-info col-2">Save</button>

</form>
```

TODO:

Add controller request mapping for /employees/save



Step 3: Process form data to save employee

Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {
```

Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

   private EmployeeService employeeService;

   public EmployeeController(EmployeeService theEmployeeService) {
      employeeService = theEmployeeService;
   }
```

Constructor injection

Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/e
public class Employ

   private EmployeeService employeeService;

   public EmployeeController(EmployeeService theEmployeeService) {
      employeeService = theEmployeeService;
   }
```

Since only one constructor
`@Autowired` is optional

Constructor injection

Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

   private EmployeeService employeeService;

   public EmployeeController(EmployeeService theEmployeeService) {
      employeeService = theEmployeeService;
   }

   @PostMapping("/save")
   public String saveEmployee(@ModelAttribute("employee") Employee theEmployee) {
```

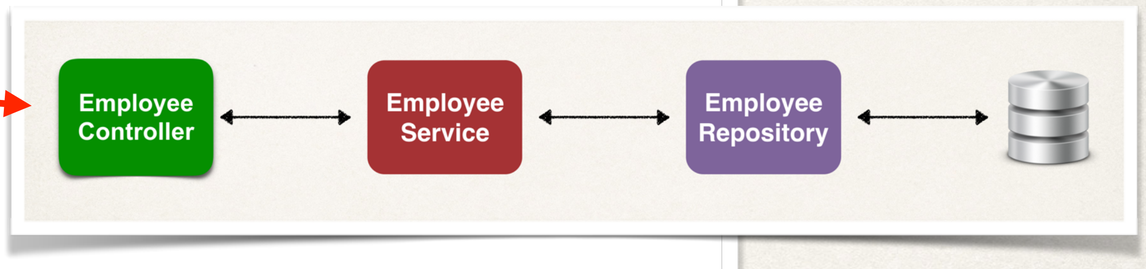
Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

   private EmployeeService emp

   public EmployeeController(E
      employeeService = theEmp
   }

   @PostMapping("/save")
   public String saveEmployee(@ModelAttribute("employee") Employee theEmployee) {
```

```
<form action="#" th:action="@{/employees/save}"
                 th:object="${employee}" method="POST">
```

Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

   private EmployeeService employeeService;

   public EmployeeController(EmployeeService theEmployeeService) {
      employeeService = theEmployeeService;
   }

   @PostMapping("/save")
   public String saveEmployee(@ModelAttribute("employee") Employee theEmployee) {

      // save the employee
      employeeService.save(theEmployee);
```



Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

   private EmployeeService employeeService;

   public EmployeeController(EmployeeService theEmployeeService) {
      employeeService = theEmployeeService;
   }

   @PostMapping("/save")
   public String saveEmployee(@ModelAttribute("employee") Employee theEmployee) {

      // save the employee
      employeeService.save(theEmployee);

      // use a redirect to prevent duplicate submissions
      return "redirect:/employees/list";
   }
   ...
}
```

Redirect to request mapping `/employees/list`

Step 3: Process form data to save employee

```
@Controller
@RequestMapping("/employees")
public class EmployeeController {

   private EmployeeService employeeService;

   public EmployeeController(EmployeeService theEmployeeService) {
      employeeService = theEmployeeService;
   }

   @PostMapping("/save")
   public String saveEmployee(@ModelAttribute("employee") Employee theEmployee) {

      // save the employee
      employeeService.save(theEmployee);

      // use a redirect to prevent duplicate submissions
      return "redirect:/employees/list";
   }
   ...
}
```

Redirect to request mapping /employees/list

"Post/Redirect/Get" pattern

For more info see www.luv2code.com/post-redirect-get