Thymeleaf with Spring Boot

What is Thymeleaf?

What is Thymeleaf?

- Thymeleaf is a Java templating engine

What is Thymeleaf?

- Thymeleaf is a Java templating engine

Thymeleaf

www.thymeleaf.org

What is Thymeleaf?

www.thymeleaf.org

- Thymeleaf is a Java templating engine

- Commonly used to generate the HTML views for web apps

What is Thymeleaf?

- Thymeleaf is a Java templating engine
- Commonly used to generate the HTML views for web apps
- However, it is a general purpose templating engine

What is Thymeleaf?

- Thymeleaf is a Java templating engine
- Commonly used to generate the HTML views for web apps
- However, it is a general purpose templating engine
  - Can use Thymeleaf outside of web apps (more on this later)

What is Thymeleaf?

- Thymeleaf is a Java templating engine
- Commonly used to generate the HTML views for
- However, it is a general purpose templating engine
  - Can use Thymeleaf outside of web apps (more on this later)

Thymeleaf

www.thymeleaf.org

Separate project
Unrelated to spring.io

What is a Thymeleaf template?

What is a Thymeleaf template?

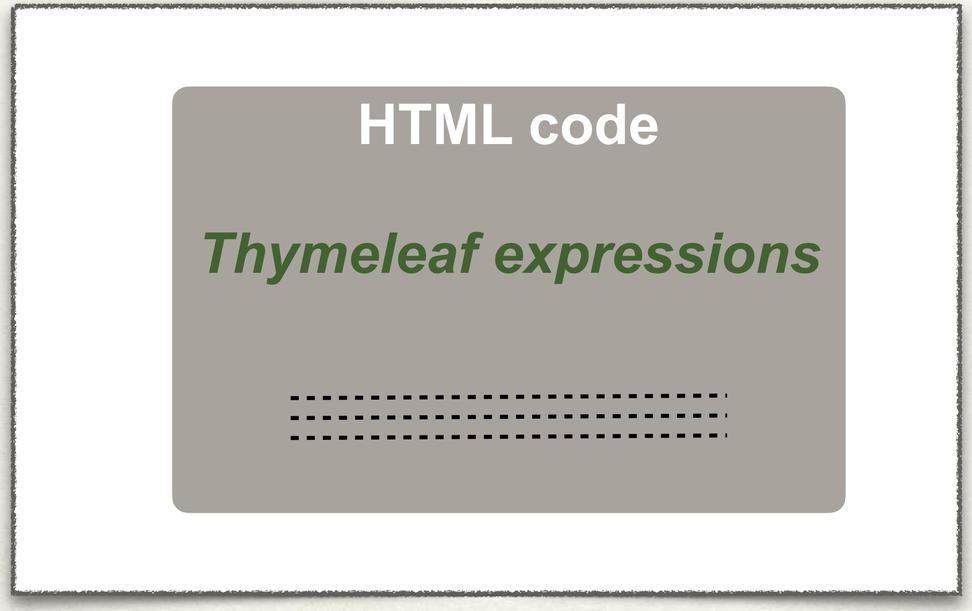
- Can be an HTML page with some Thymeleaf expressions

What is a Thymeleaf template?

- Can be an HTML page with some Thymeleaf expressions
- Include dynamic content from Thymeleaf expressions

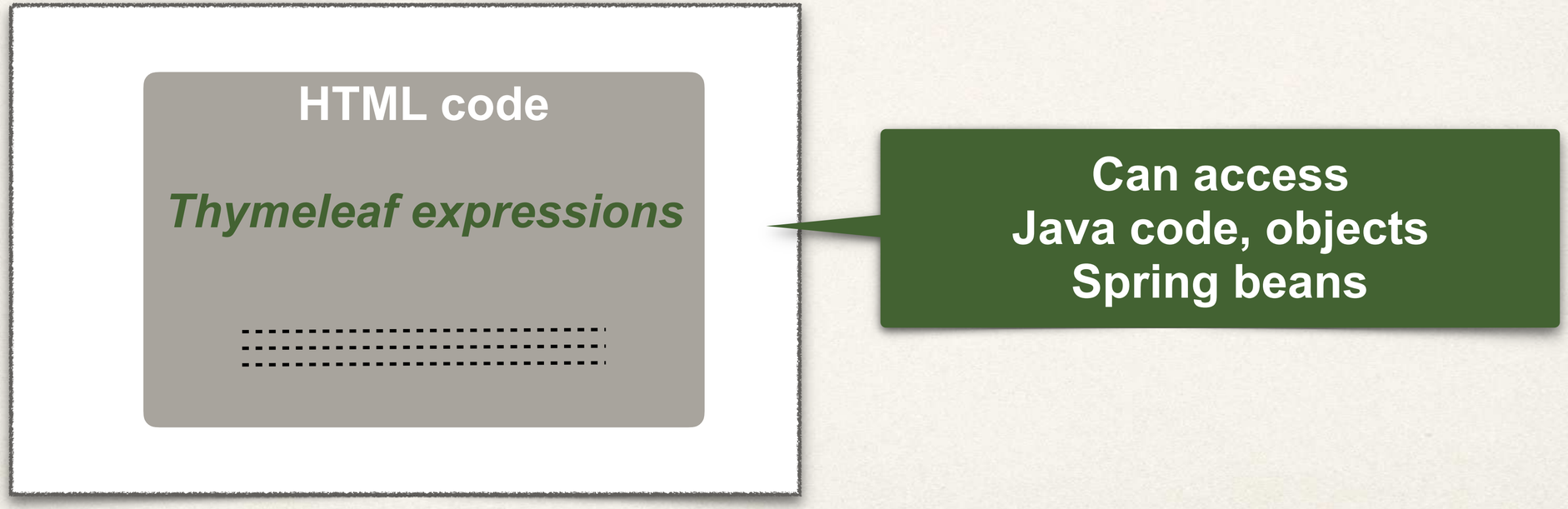
What is a Thymeleaf template?

- Can be an HTML page with some Thymeleaf expressions
- Include dynamic content from Thymeleaf expressions



What is a Thymeleaf template?

- Can be an HTML page with some Thymeleaf expressions
- Include dynamic content from Thymeleaf expressions



Where is the Thymeleaf template processed?

Where is the Thymeleaf template processed?

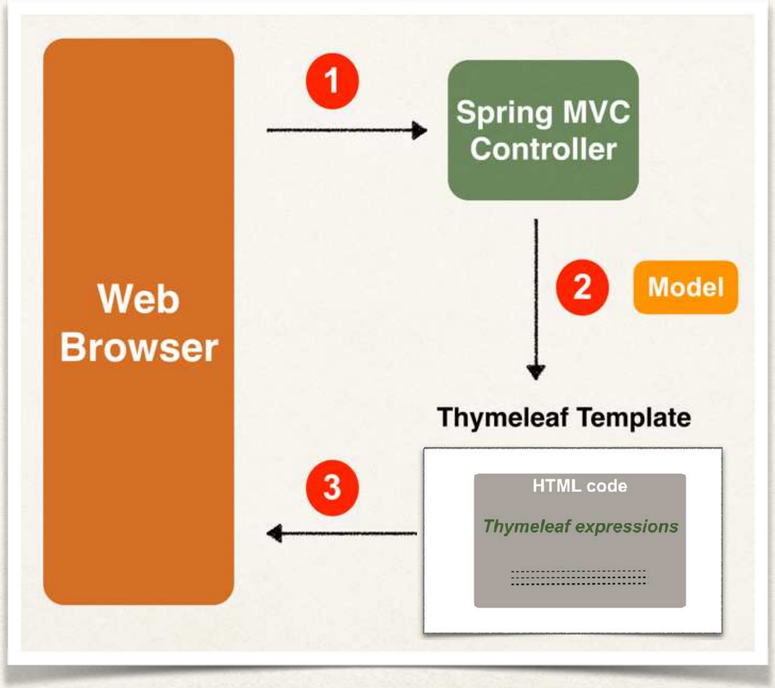
- In a web app, Thymeleaf is processed on the server

Where is the Thymeleaf template processed?

- In a web app, Thymeleaf is processed on the server
- Results included in HTML returned to browser

Where is the Thymeleaf template processed?

- In a web app, Thymeleaf is processed on the server
- Results included in HTML returned to browser



Hmmm …

This sounds similar to JSP …

Thymeleaf vs JSP

Thymeleaf vs JSP

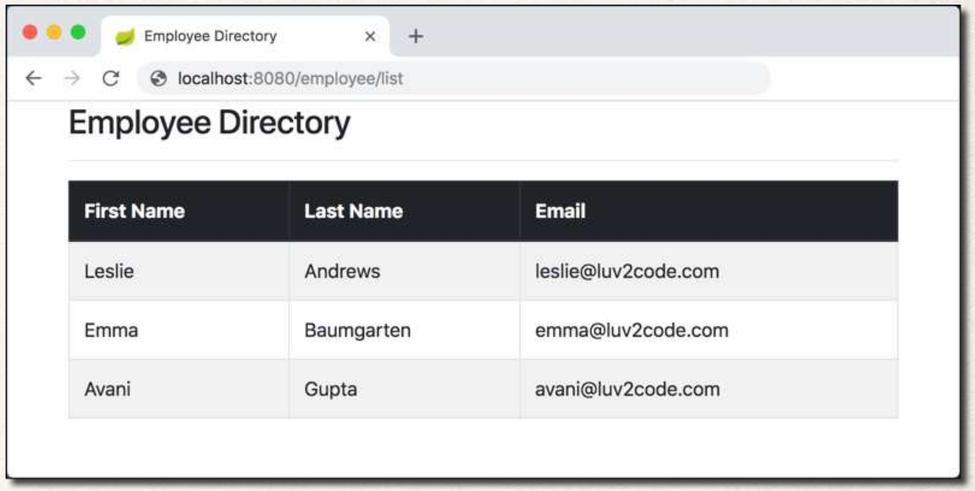
- Yes, Thymeleaf is similar to JSP

Thymeleaf vs JSP

- Yes, Thymeleaf is similar to JSP
  - Can be used for web view templates

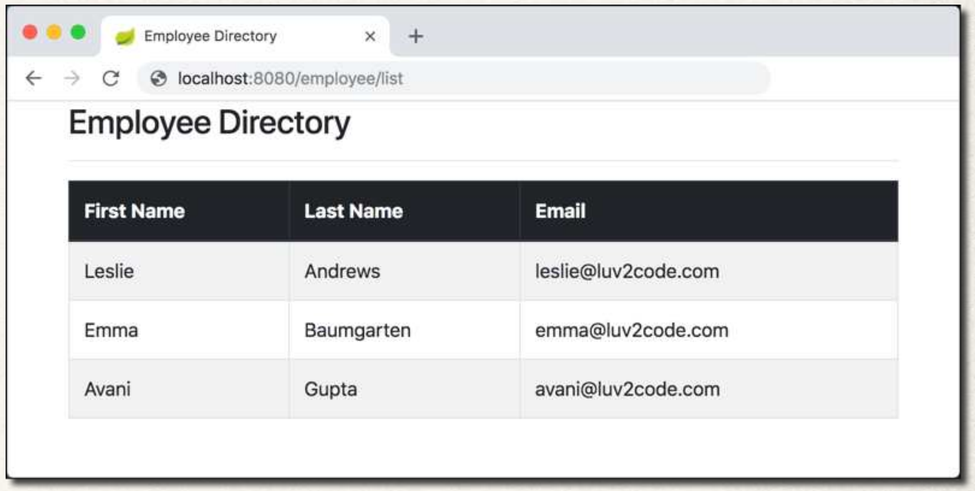
Thymeleaf vs JSP

- Yes, Thymeleaf is similar to JSP
  - Can be used for web view templates



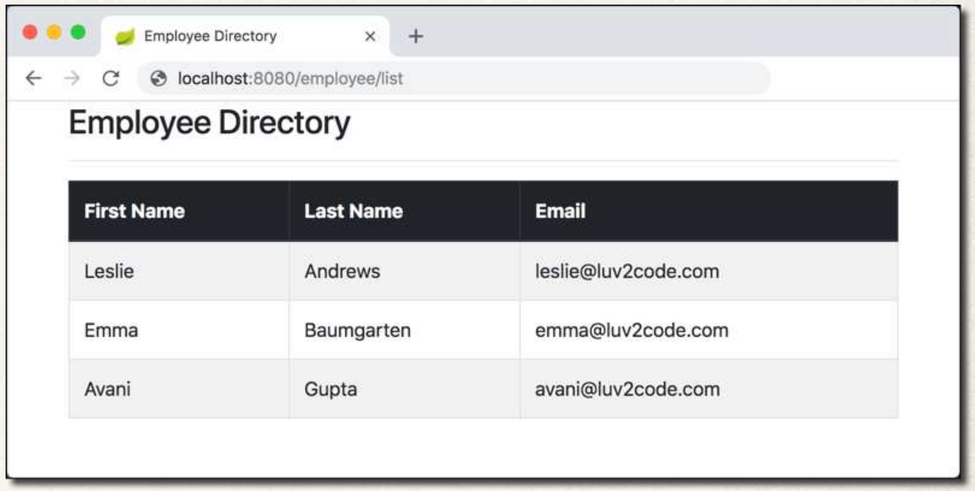
Thymeleaf vs JSP

- Yes, Thymeleaf is similar to JSP
  - Can be used for web view templates
- One key difference



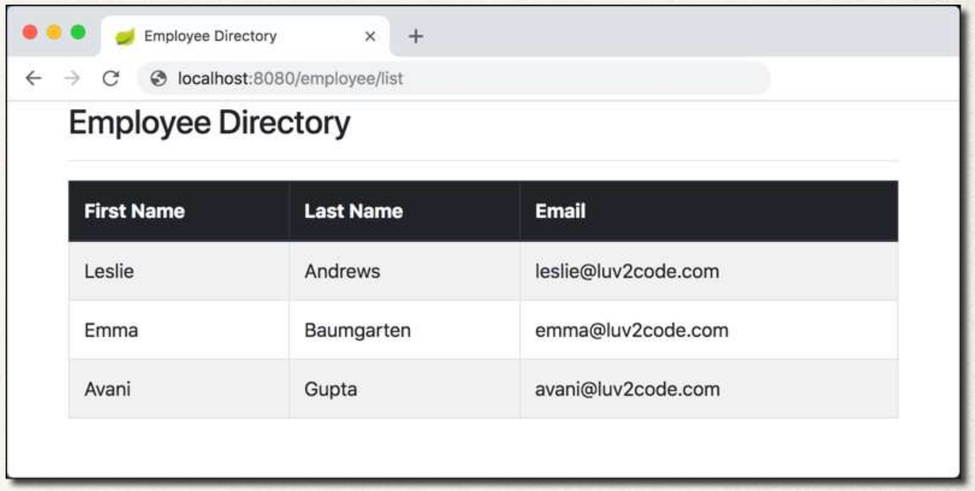
Thymeleaf vs JSP

- Yes, Thymeleaf is similar to JSP
  - Can be used for web view templates
- One key difference
  - JSP can only be used in a web environment



Thymeleaf vs JSP

- Yes, Thymeleaf is similar to JSP
  - Can be used for web view templates



- One key difference
  - JSP can only be used in a web environment
  - Thymeleaf can be used in web OR non-web environments

Thymeleaf Use Cases (non-web)

Thymeleaf Use Cases (non-web)

- Email Template

Thymeleaf Use Cases (non-web)

- Email Template
  - When student signs up for a course then send welcome email

```
Hi <<firstName>>,

Thanks for joining <<course>>!
```

Email

Thymeleaf Use Cases (non-web)

- Email Template
  - When student signs up for a course then send welcome email

```
Hi <<firstName>>,

Thanks for joining <<course>>!
```

Email

- CSV Template

Thymeleaf Use Cases (non-web)

- Email Template
  - When student signs up for a course then send welcome email

```
Hi <<firstName>>,

Thanks for joining <<course>>!
```

Email

- CSV Template
  - Generate a monthly report as CSV then upload to Google drive

```
Product,Quantity,Price,Total
…
…
```

CSV

Thymeleaf Use Cases (non-web)

- Email Template
  - When student signs up for a course then send welcome email

```
Hi <<firstName>>,

Thanks for joining <<course>>!
```

Email

- CSV Template
  - Generate a monthly report as CSV then upload to Google drive

```
Product,Quantity,Price,Total
…
…
```

CSV

- PDF Template

Thymeleaf Use Cases (non-web)

- Email Template
  - When student signs up for a course then send welcome email

```
Hi <<firstName>>,

Thanks for joining <<course>>!
```

Email

- CSV Template
  - Generate a monthly report as CSV then upload to Google drive

```
Product,Quantity,Price,Total
…
…
```

CSV

- PDF Template
  - Generate a travel confirmation PDF then send via email

```
Flight: <<flightNum>>
Depart: <<departAirport>>
Arrive: <<arrivalAirport>>
```

PDF

FAQ: Should I use JSP or Thymeleaf?

FAQ: Should I use JSP or Thymeleaf?

- Depends on your project requirements

FAQ: Should I use JSP or Thymeleaf?

- Depends on your project requirements
- If you only need web views you can go either way

FAQ: Should I use JSP or Thymeleaf?

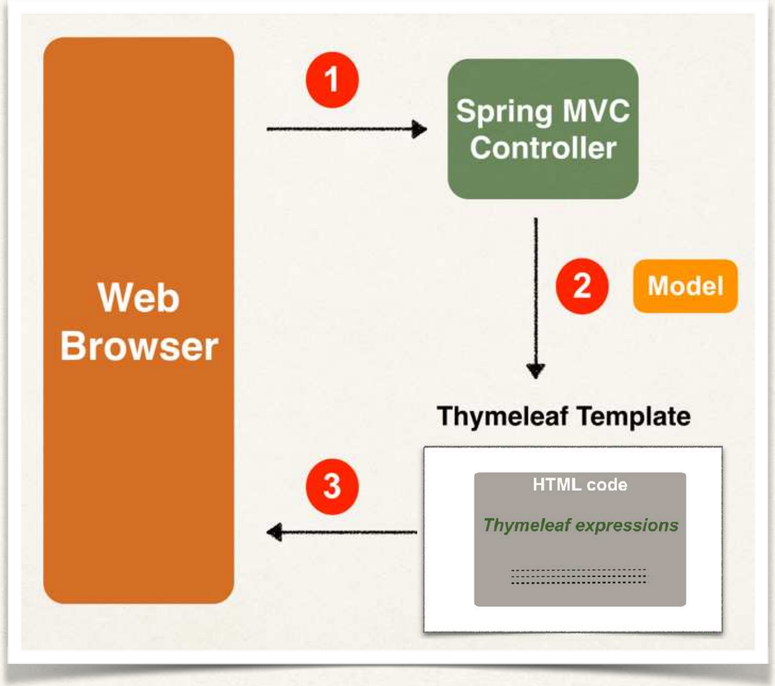
- Depends on your project requirements
- If you only need web views you can go either way
- If you need a general purpose template engine (non-web) use Thymeleaf

FAQ: Should I use JSP or Thymeleaf?

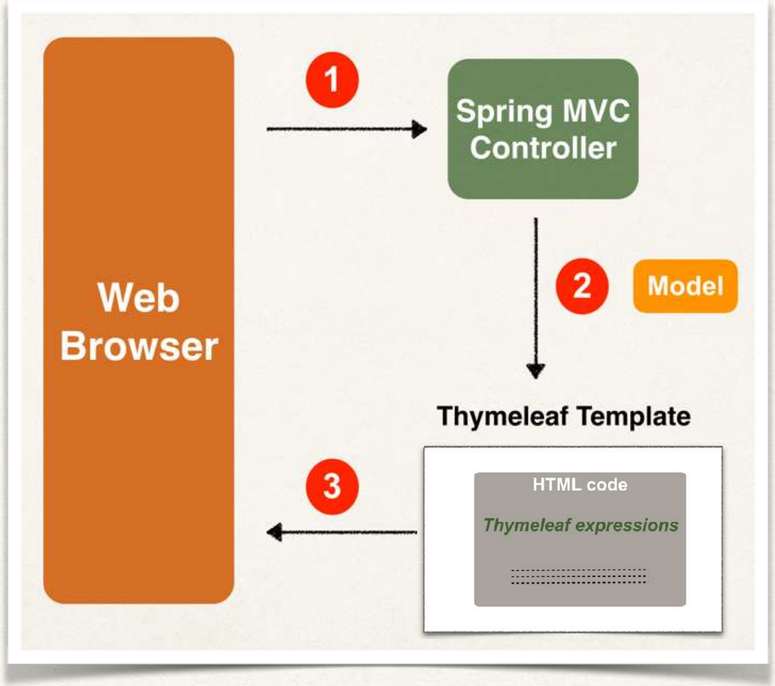
- Depends on your project requirements
- If you only need web views you can go either way
- If you need a general purpose template engine (non-web) use Thymeleaf
- If you want to **impress your interviewer**, mention Thymeleaf … LOL!

Thymeleaf Demo

Thymeleaf Demo



Thymeleaf Demo



Output

Time on the server is Sun Jan 06 17:00:40 EST 2019

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Add Thymeleaf to Maven POM file

Development Process

Step-By-Step

1. Add Thymeleaf to Maven POM file

2. Develop Spring MVC Controller

Development Process

Step-By-Step

1. Add Thymeleaf to Maven POM file

2. Develop Spring MVC Controller

3. Create Thymeleaf template

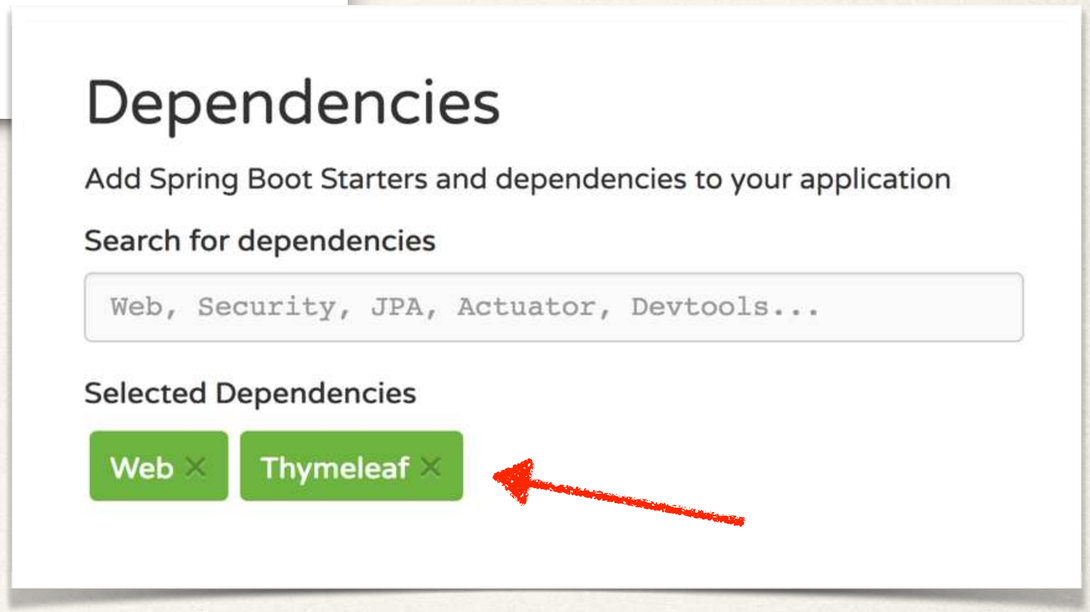
Step 1: Add Thymeleaf to Maven pom file

Step 1: Add Thymeleaf to Maven pom file

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

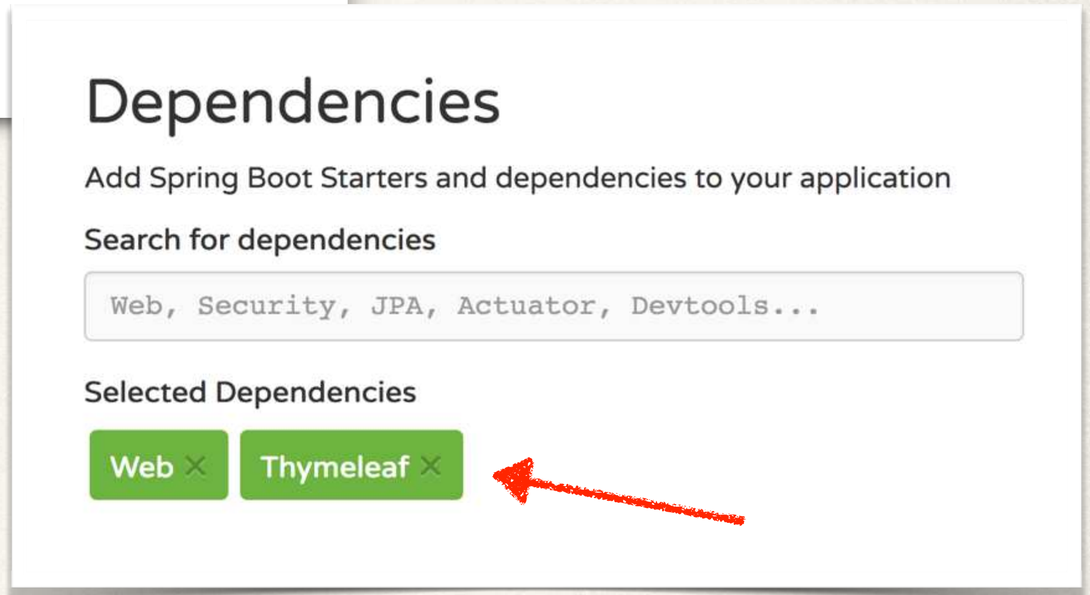
Step 1: Add Thymeleaf to Maven pom file

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```



Step 1: Add Thymeleaf to Maven pom file

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```



Based on this,
Spring Boot will auto configure to use Thymeleaf templates

Step 2: Develop Spring MVC Controller

Step 2: Develop Spring MVC Controller

File: DemoController.java

```
@Controller
public class DemoController {

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

    theModel.addAttribute("theDate", new java.util.Date());

    return "helloworld";
  }
}
```

Step 2: Develop Spring MVC Controller

File: DemoController.java

```
@Controller
public class DemoController {

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

    theModel.addAttribute("theDate", new java.util.Date());

    return "helloworld";
  }
}
```

```
src/main/resources/templates/helloworld.html
```

Where to place Thymeleaf template?

Where to place Thymeleaf template?

- In Spring Boot, your Thymeleaf template files go in

Where to place Thymeleaf template?

- In Spring Boot, your Thymeleaf template files go in
  - **src/main/resources/templates**

Where to place Thymeleaf template?

- In Spring Boot, your Thymeleaf template files go in
  - **src/main/resources/templates**

- For web apps, Thymeleaf templates have a **.html** extension

Step 3: Create Thymeleaf template

Step 3: Create Thymeleaf template

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

Step 3: Create Thymeleaf template

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

To use Thymeleaf expressions

Step 3: Create Thymeleaf template

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

To use Thymeleaf expressions

Thymeleaf expression

Step 3: Create Thymeleaf template

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

To use Thymeleaf expressions

Thymeleaf expression

Time on the server is Sun Jan 06 17:00:40 EST 2019

Step 3: Create Thymeleaf template

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

To use Thymeleaf expressions

Thymeleaf expression

Time on the server is Sun Jan 06 17:00:40 EST 2019

Step 3: Create Thymeleaf template

File: DemoController.java

```
@Controller
public class DemoController {

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

      theModel.addAttribute("theDate", new java.util.Date());

      return "helloworld";
   }
}
```

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

Thymeleaf expression

Time on the server is Sun Jan 06 17:00:40 EST 2019

Step 3: Create Thymeleaf template

File: DemoController.java

```
@Controller
public class DemoController {

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

      theModel.addAttribute("theDate", new java.util.Date());

      return "helloworld";
   }
}
```

1

File: src/main/resources/templates/helloworld.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

Thymeleaf expression

Time on the server is Sun Jan 06 17:00:40 EST 2019

Step 3: Create Thymeleaf template

File: DemoController.java

```
@Controller
public class DemoController {

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

      theModel.addAttribute("theDate", new java.util.Date());

      return "helloworld";
   }
}
```

1

Thymeleaf accesses "theDate" from the Spring MVC Model

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

Thymeleaf expression

Time on the server is Sun Jan 06 17:00:40 EST 2019

Step 3: Create Thymeleaf template

File: DemoController.java

```
@Controller
public class DemoController {

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

      theModel.addAttribute("theDate", new java.util.Date());

      return "helloworld";
   }
}
```

1

Thymeleaf accesses "theDate" from the Spring MVC Model

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head> … </head>

<body>
    <p th:text="'Time on the server is ' + ${theDate}" />
</body>

</html>
```

Thymeleaf expression

2

Time on the server is Sun Jan 06 17:00:40 EST 2019

Additional Features

Additional Features

- Looping and conditionals

Additional Features

- Looping and conditionals
- CSS and JavaScript integration

Additional Features

- Looping and conditionals
- CSS and JavaScript integration
- Template layouts and fragments

Additional Features

- Looping and conditionals
- CSS and JavaScript integration
- Template layouts and fragments

www.thymeleaf.org

CSS and Thymeleaf

Let's Apply CSS Styles to our Page

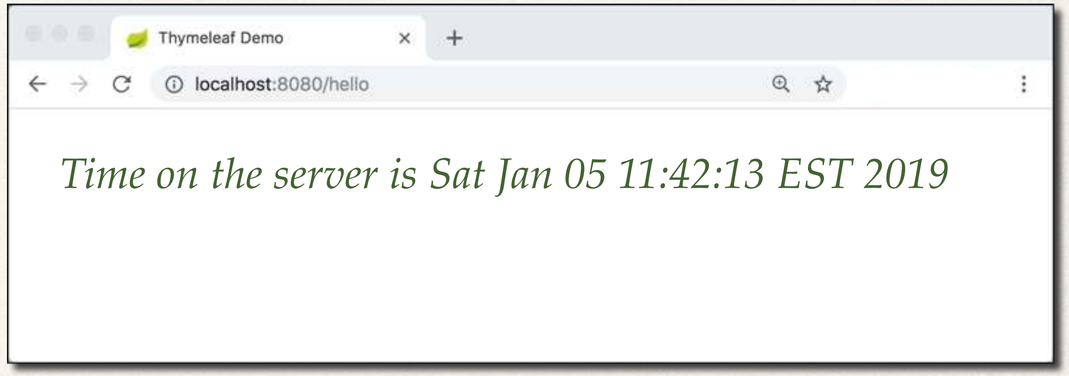
Let's Apply CSS Styles to our Page

Before

Let's Apply CSS Styles to our Page

Before

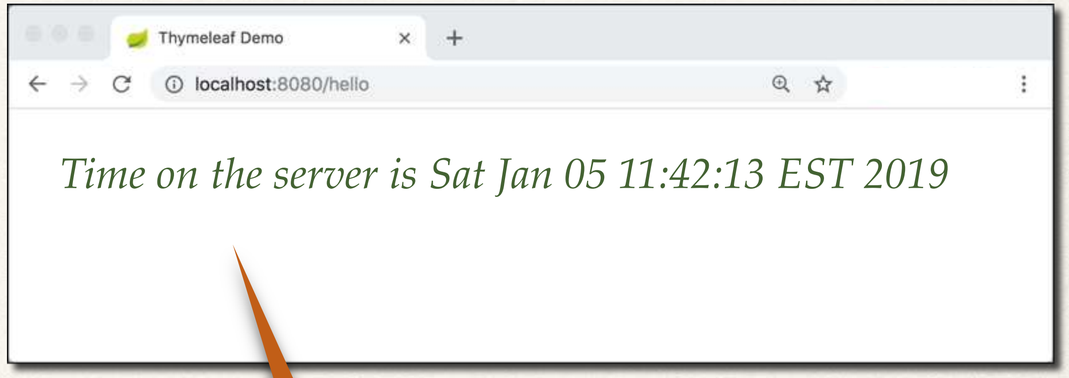
After



Let's Apply CSS Styles to our Page

Before

After




```
font-style: italic;
color: green;
```

Using CSS with Thymleaf Templates

Using CSS with Thymleaf Templates

- You have the option of using

Using CSS with Thymleaf Templates

- You have the option of using
  - Local CSS files as part of your project

Using CSS with Thymleaf Templates

- You have the option of using
  - Local CSS files as part of your project
  - Referencing remote CSS files

Using CSS with Thymleaf Templates

- You have the option of using
  - Local CSS files as part of your project
  - Referencing remote CSS files

- We'll cover both options in this video

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create CSS file

Development Process

Step-By-Step

1. Create CSS file

2. Reference CSS in Thymeleaf template

Development Process

Step-By-Step

1. Create CSS file

2. Reference CSS in Thymeleaf template

3. Apply CSS style

Step 1: Create CSS file

Step 1: Create CSS file

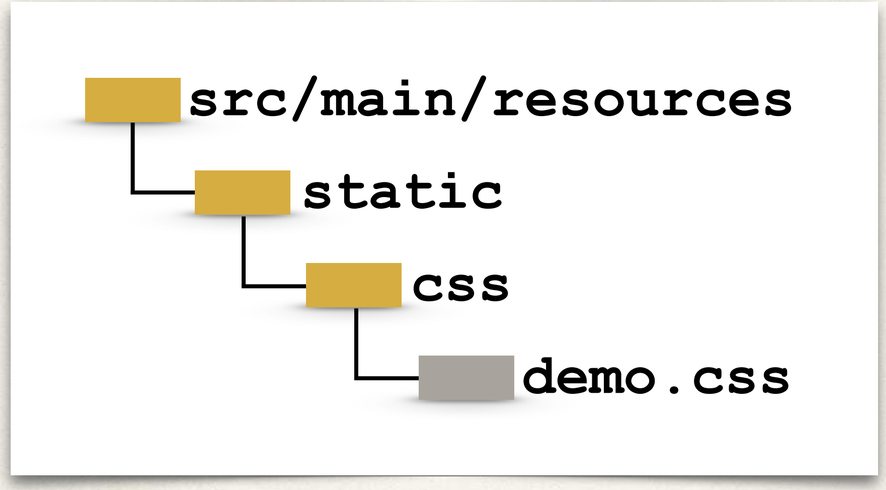
- Spring Boot will look for static resources in the directory

Step 1: Create CSS file

- Spring Boot will look for static resources in the directory
  - **`src/main/resources/static`**

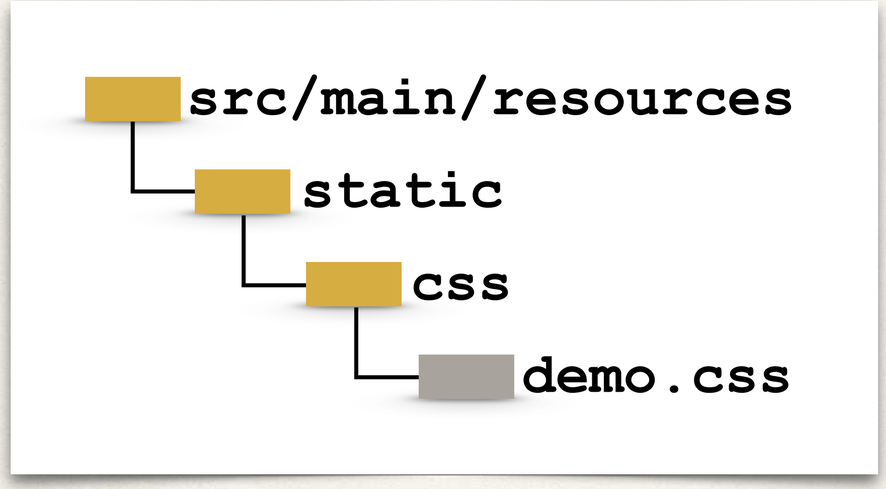
Step 1: Create CSS file

- Spring Boot will look for static resources in the directory
  - **`src/main/resources/static`**



Step 1: Create CSS file

- Spring Boot will look for static resources in the directory
  - **`src/main/resources/static`**

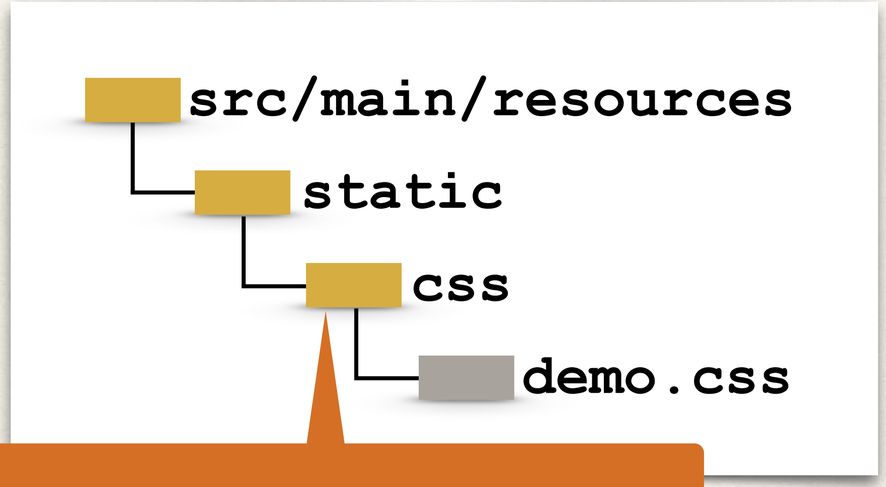


File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```

Step 1: Create CSS file

- Spring Boot will look for static resources in the directory
  - **`src/main/resources/static`**



Can be any sub-directory name

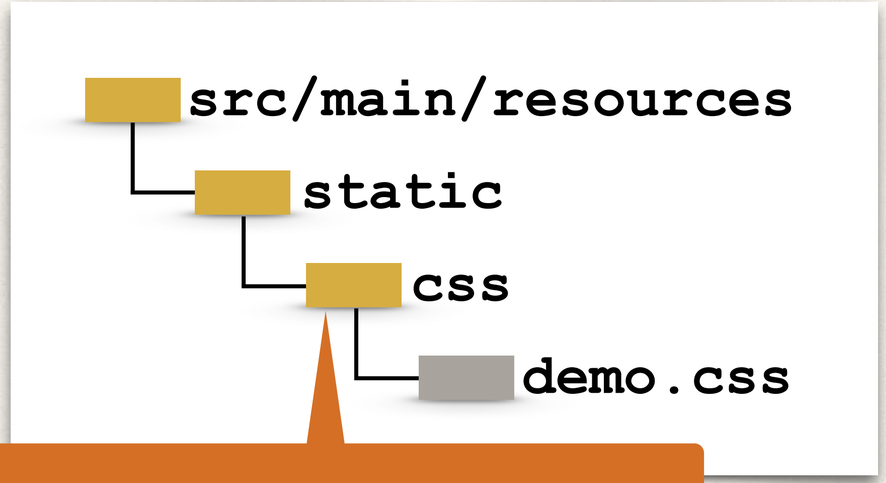
File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```

Step 1: Create CSS file

- Spring Boot will look for static resources in
  - **`src/main/resources/static`**

You can create your own custom sub-directories
`static/css`
`static/images`
`static/js`
etc …



Can be any sub-directory name

File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```

Step 2: Reference CSS in Thymeleaf template

Step 2: Reference CSS in Thymeleaf template

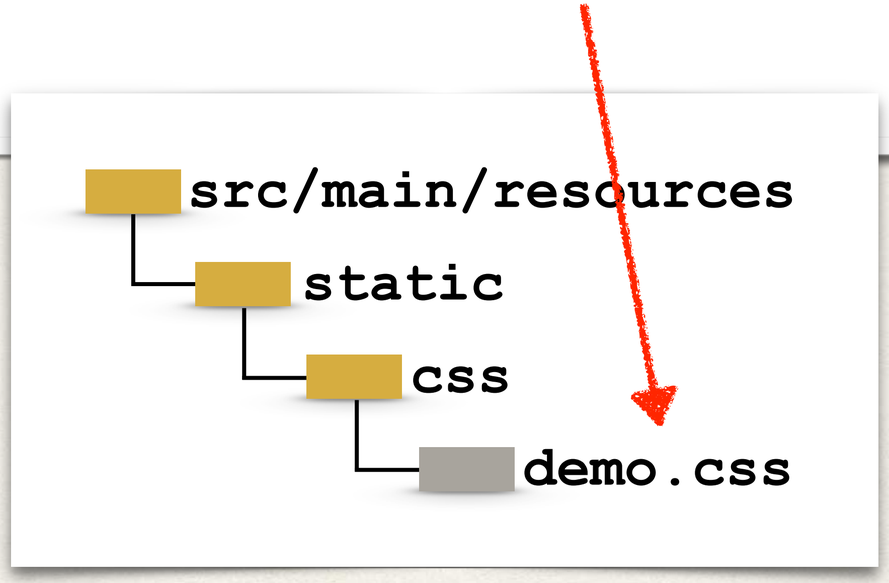
File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />

</head>
```

Step 2: Reference CSS in Thymeleaf template

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />

</head>
```

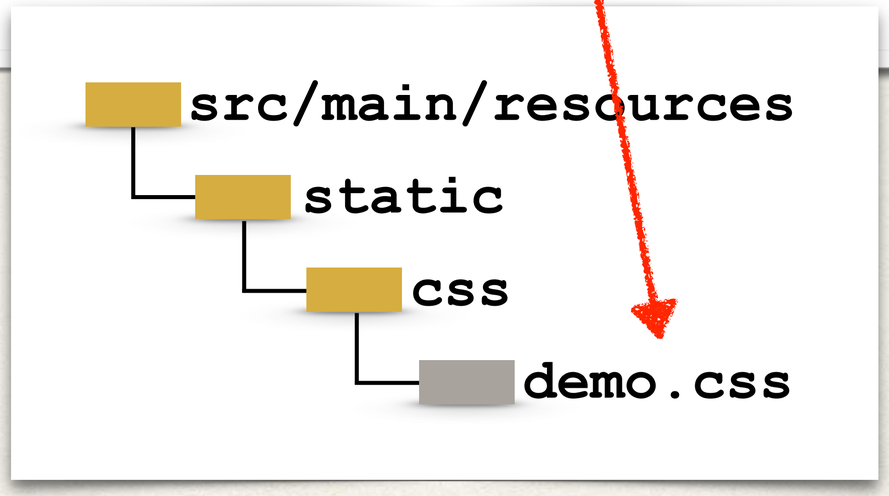


Step 2: Reference CSS in Thymeleaf template

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />

</head>
```

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



Step 3: Apply CSS

Step 3: Apply CSS

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />
</head>

<body>
  <p th:text="'Time on the server is ' + ${theDate}" class="funny" />
</body>
```

Step 3: Apply CSS

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />
</head>

<body>
  <p th:text="'Time on the server is ' + ${theDate}" class="funny" />
</body>
```

File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```

Step 3: Apply CSS

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />
</head>

<body>
  <p th:text="'Time on the server is ' + ${theDate}" class="funny" />
</body>
```

File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```

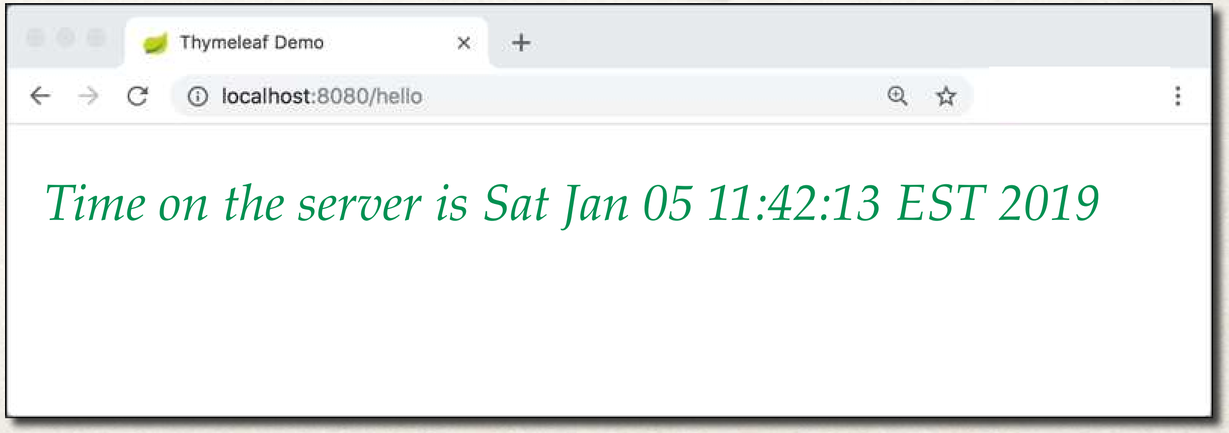
Step 3: Apply CSS

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />
</head>

<body>
  <p th:text="'Time on the server is ' + ${theDate}" class="funny" />
</body>
```

File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```



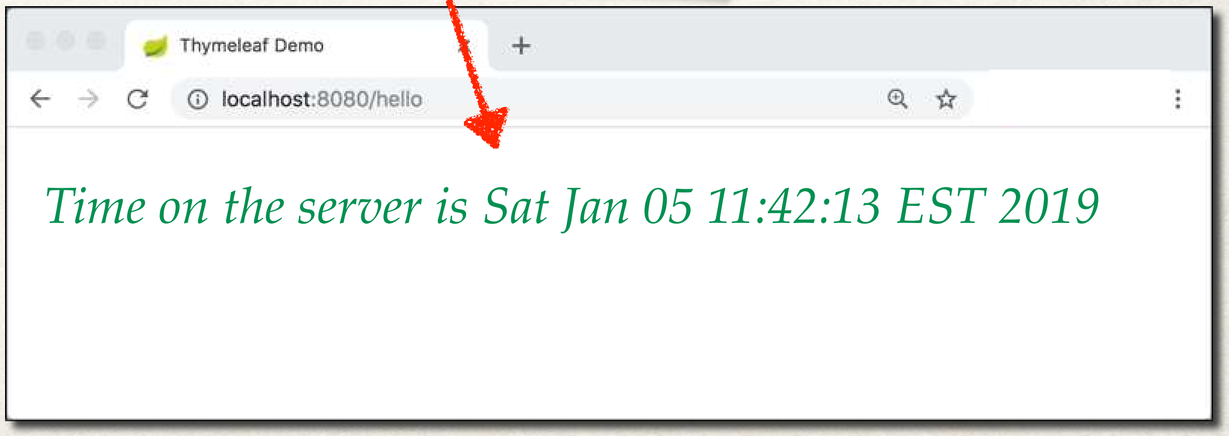
Step 3: Apply CSS

File: helloworld.html

```
<head>
  <title>Thymeleaf Demo</title>

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/demo.css}" />
</head>

<body>
  <p th:text="'Time on the server is ' + ${theDate}" class="funny" />
</body>
```

File: demo.css

```
.funny {
  font-style: italic;
  color: green;
}
```



Other search directories

Other search directories

Spring Boot will search following directories for static resources:

`/src/main/resources`

Other search directories

Spring Boot will search following directories for static resources:

`/src/main/resources`

1. **`/META-INF/resources`**

Other search directories

Spring Boot will search following directories for static resources:

`/src/main/resources`

1. **`/META-INF/resources`**
2. **`/resources`**

Other search directories

Spring Boot will search following directories for static resources:

`/src/main/resources`

1. **`/META-INF/resources`**
2. **`/resources`**
3. **`/static`**

Other search directories

Spring Boot will search following directories for static resources:

`/src/main/resources`

1. **`/META-INF/resources`**
2. **`/resources`**
3. **`/static`**
4. **`/public`**

Other search directories

Spring Boot will search following directories for static resources:

`/src/main/resources`

1. `/META-INF/resources`
2. `/resources`
3. `/static`
4. `/public`

Search order: top-down

3rd Party CSS Libraries - Bootstrap

3rd Party CSS Libraries - Bootstrap

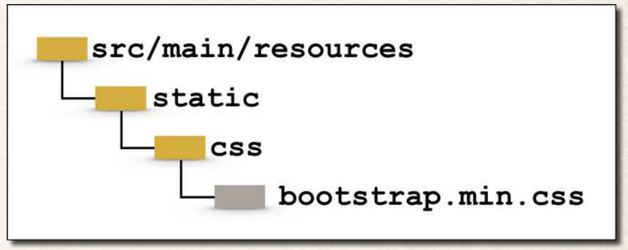
- Local Installation

3rd Party CSS Libraries - Bootstrap

- Local Installation
- Download Bootstrap file(s) and add to **`/static/css`** directory

3rd Party CSS Libraries - Bootstrap

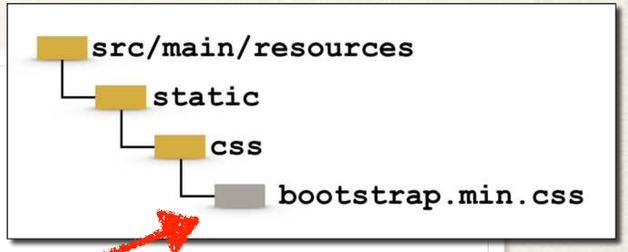
- Local Installation
- Download Bootstrap file(s) and add to **`/static/css`** directory



3rd Party CSS Libraries - Bootstrap

- Local Installation
- Download Bootstrap file(s) and add to **`/static/css`** directory

```
<head>
  … …

  <!-- reference CSS file -->
  <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />

</head>
```



3rd Party CSS Libraries - Bootstrap

3rd Party CSS Libraries - Bootstrap

- Remote Files

3rd Party CSS Libraries - Bootstrap

- Remote Files

```
<head>
  … …

  <!-- reference CSS file -->
  <link rel="stylesheet"
       href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css" />

  … …
</head>
```

Create HTML Tables with Thymeleaf

HTML Tables

Start with plain table
Will add CSS in later videos

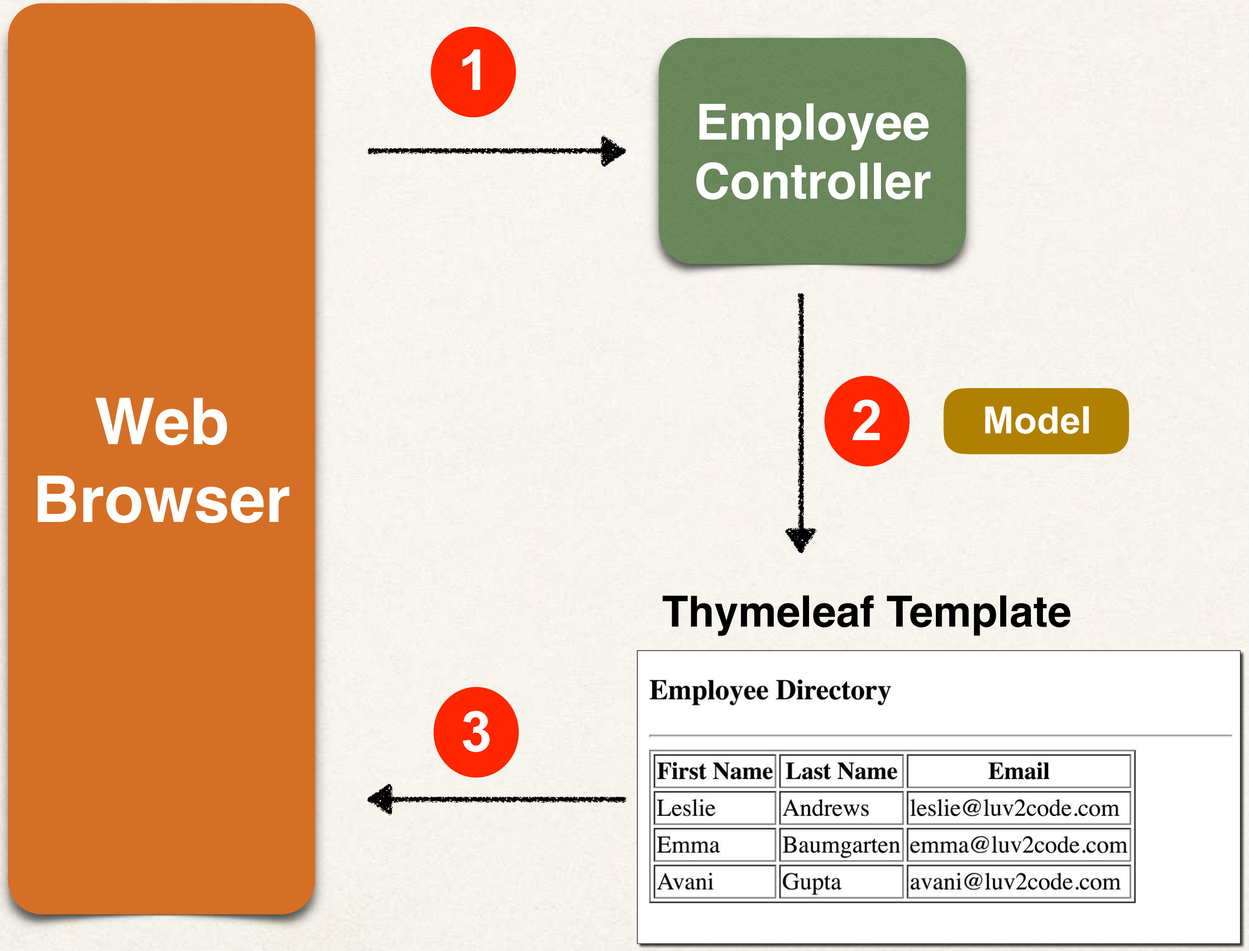
HTML Tables

Start with plain table
Will add CSS in later videos

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Big Picture


Web Browser
1
Employee Controller
2
Model
Thymeleaf Template
Employee Directory
First Name | Last Name | Email
Leslie | Andrews | leslie@luv2code.com
Emma | Baumgarten | emma@luv2code.com
Avani | Gupta | avani@luv2code.com
3

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create Employee.java

Development Process

Step-By-Step

1. Create Employee.java

2. Create EmployeeController.java

Development Process

Step-By-Step

1. Create Employee.java

2. Create EmployeeController.java

3. Create Thymeleaf template

Step 1: Create Employee.java

```java
public class Employee {

    private int id;
    private String firstName;
    private String lastName;
    private String email;

    public Employee() {

    }

    public Employee(int id, String firstName, String lastName, String email) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
```

Step 1: Create Employee.java

- Regular Java class: fields, constructors, getters / setters

```
public class Employee {

    private int id;
    private String firstName;
    private String lastName;
    private String email;

    public Employee() {

    }

    public Employee(int id, String firstName, String lastName, String email) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
```

Step 2: Create EmployeeController.java

Step 2: Create EmployeeController.java

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

Step 2: Create EmployeeController.java

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

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

Step 2: Create EmployeeController.java

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");
```

Will add database integration later

Step 2: Create EmployeeController.java

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();
```

Step 2: Create EmployeeController.java

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);
```

Step 2: Create EmployeeController.java

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);
```

Step 2: Create EmployeeController.java

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);
```

Our Thymleaf template will access this data

Step 2: Create EmployeeController.java

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Our Thymleaf template will access this data

Step 2: Create EmployeeController.java

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Our Thymleaf template will access this data

`src/main/resources/templates/list-employees.html`

Step 3: Create Thymeleaf template

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
```

To use Thymeleaf expressions

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
…
<body>

  <h3>Employee Directory</h3>
  <hr>
```

To use Thymeleaf expressions

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
…
<body>

  <h3>Employee Directory</h3>
  <hr>

  <table border="1">

    <!-- Build HTML table based on employees -->

  </table>
```

To use Thymeleaf expressions

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
…
<body>

  <h3>Employee Directory</h3>
  <hr>

  <table border="1">

    <!-- Build HTML table based on employees -->

  </table>
```

To use Thymeleaf expressions

To Do
Add code to loop over employees

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
…
<body>

  <h3>Employee Directory</h3>
  <hr>

  <table border="1">

    <!-- Build HTML table based on employees -->

  </table>

</body>

</html>
```

To use Thymeleaf expressions

To Do
Add code to loop over employees

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Loop over
list of employees

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      ail</th>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Loop parameter

Loop over list of employees

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      ail</th>


  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Loop parameter

Loop over list of employees

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      // create the list
      // add to the list
      … …

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      ail</th>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">

      <td th:text="${tempEmployee.firstName}" />
      <td th:text="${tempEmployee.lastName}" />
      <td th:text="${tempEmployee.email}" />

    </tr>
  </tbody>
</table>
```

Loop parameter

Loop over list of employees

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      // create the list
      // add to the list
      … …

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Thymeleaf and Bootstrap

Let's Make our Page Beautiful

Let's Make our Page Beautiful

Before

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Let's Make our Page Beautiful

Before

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

After

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Let's Make our Page Beautiful

Before

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

After

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Bootstrap

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Get links for remote Bootstrap files

Development Process

Step-By-Step

1. Get links for remote Bootstrap files

2. Add links in Thymeleaf template

Development Process

Step-By-Step

1. Get links for remote Bootstrap files

2. Add links in Thymeleaf template

3. Apply Bootstrap CSS styles

Step 1: Get links for remote Bootstrap files

Step 1: Get links for remote Bootstrap files

- Visit Bootstrap website: **`www.getbootstrap.com`**

Step 1: Get links for remote Bootstrap files

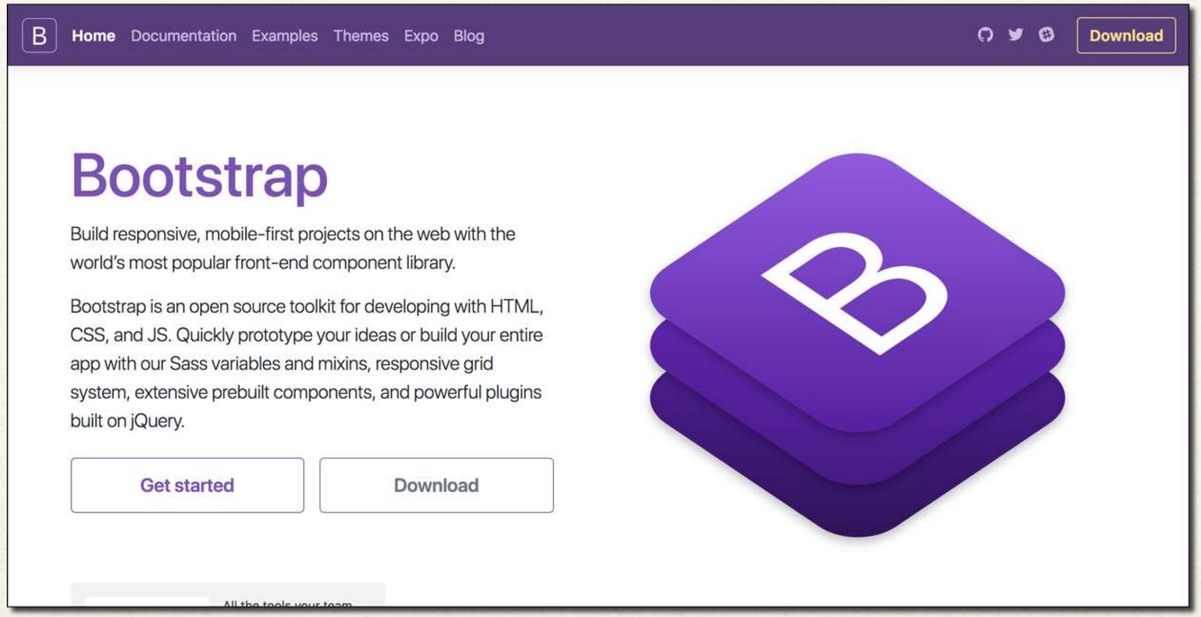
- Visit Bootstrap website: **`www.getbootstrap.com`**

- Website has instructions on how to **Get Started**

Step 1: Get links for remote Bootstrap files

- Visit Bootstrap website: **`www.getbootstrap.com`**

- Website has instructions on how to **Get Started**



Step 2: Add links in Thymeleaf template

Step 2: Add links in Thymeleaf template

```
<head>
    …
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/x.y.z/css/bootstrap.min.css" …>
    …
</head>
```

Step 2: Add links in Thymeleaf template

```
<head>
    …
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/x.y.z/css/bootstrap.min.css" …>
    …
</head>
```

Version number

Step 3: Apply CSS

Step 3: Apply CSS

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Step 3: Apply CSS

Bootstrap CSS style

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Step 3: Apply CSS

Bootstrap CSS style

Bootstrap CSS styles

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Step 3: Apply CSS

Bootstrap CSS style

Bootstrap CSS styles

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Create HTML Tables with Thymeleaf

HTML Tables

HTML Tables

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

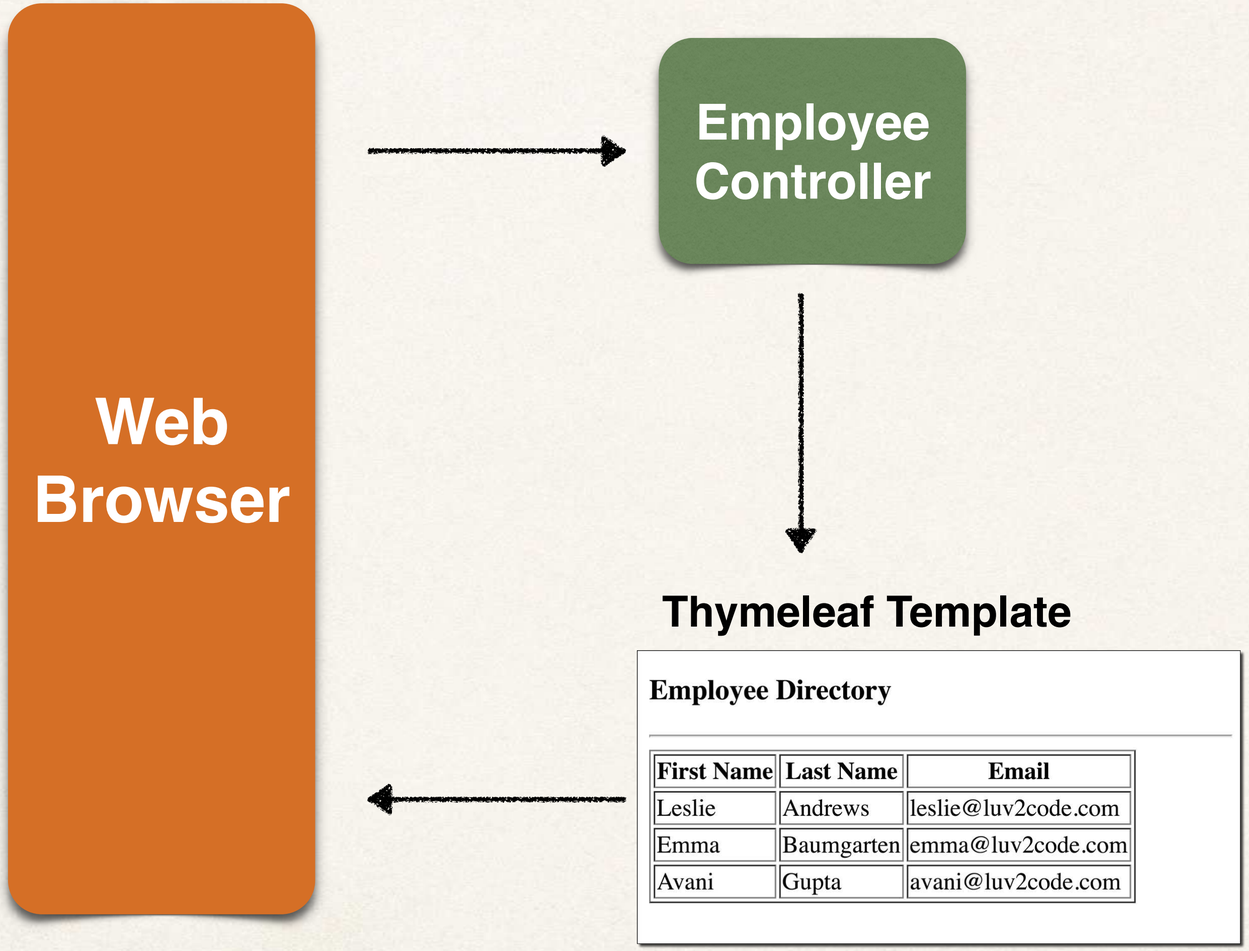
HTML Tables

Start with plain table
Will add CSS in later videos

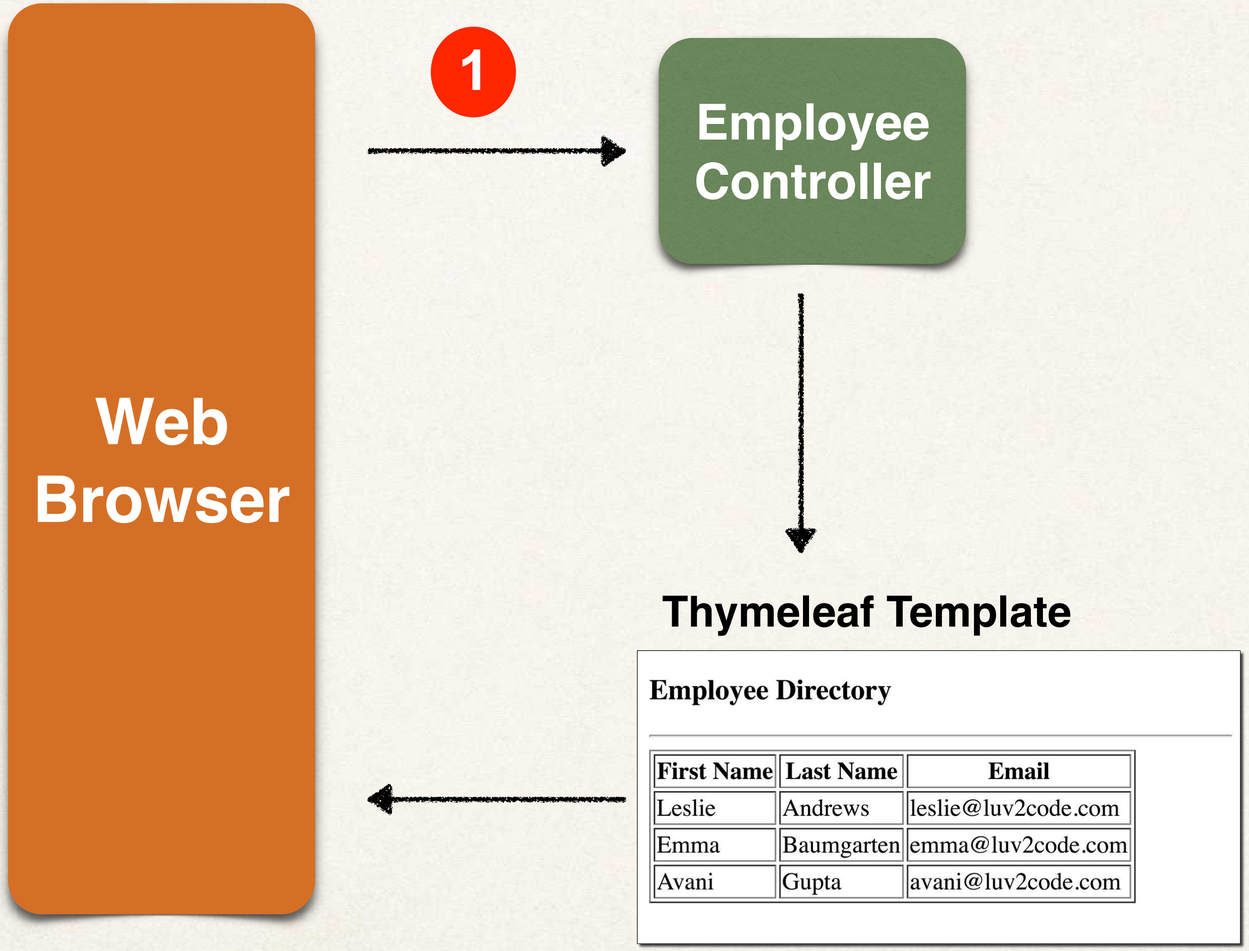
Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

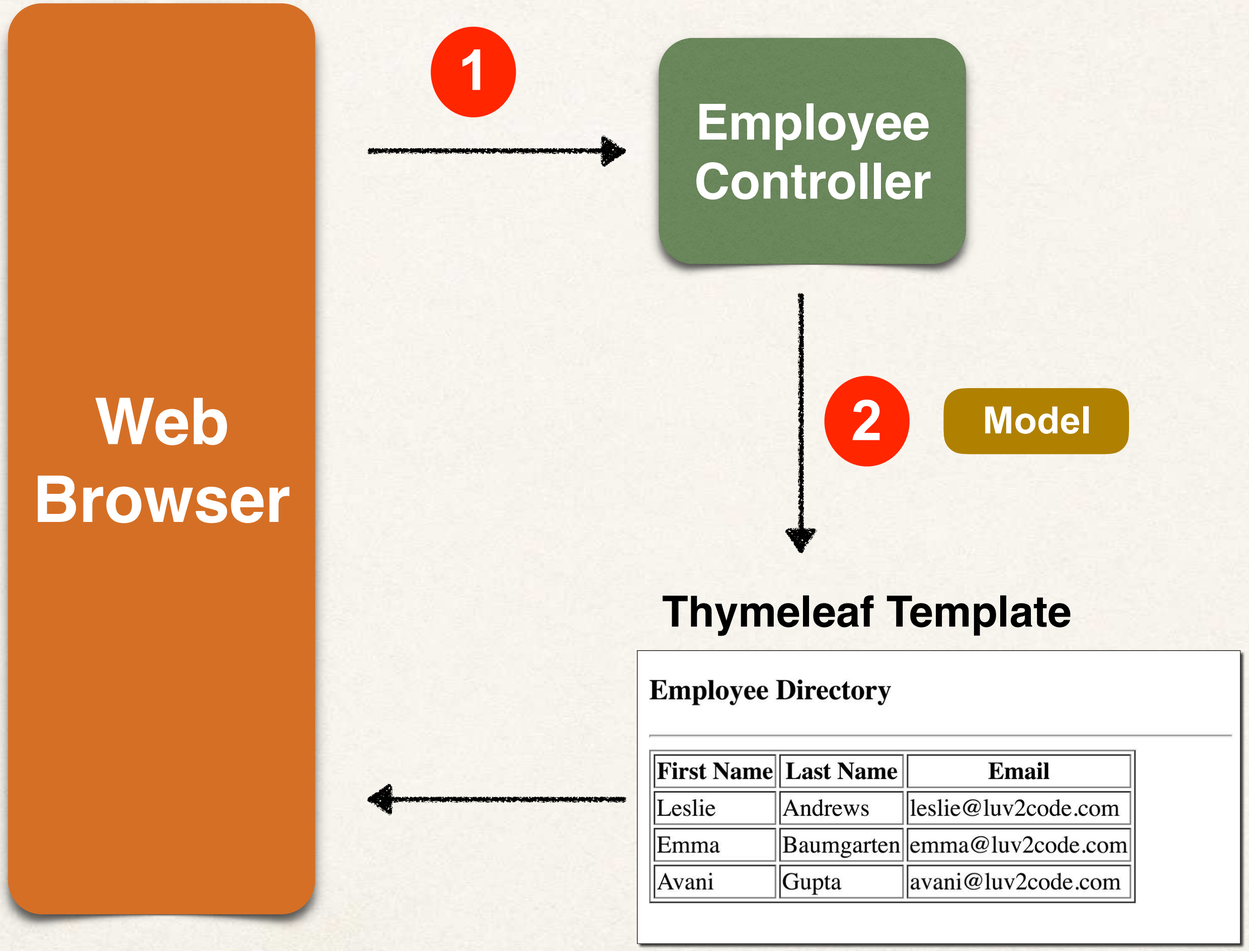
Big Picture


Web Browser
Employee Controller
Thymeleaf Template
Employee Directory
First Name | Last Name | Email
Leslie | Andrews | leslie@luv2code.com
Emma | Baumgarten | emma@luv2code.com
Avani | Gupta | avani@luv2code.com

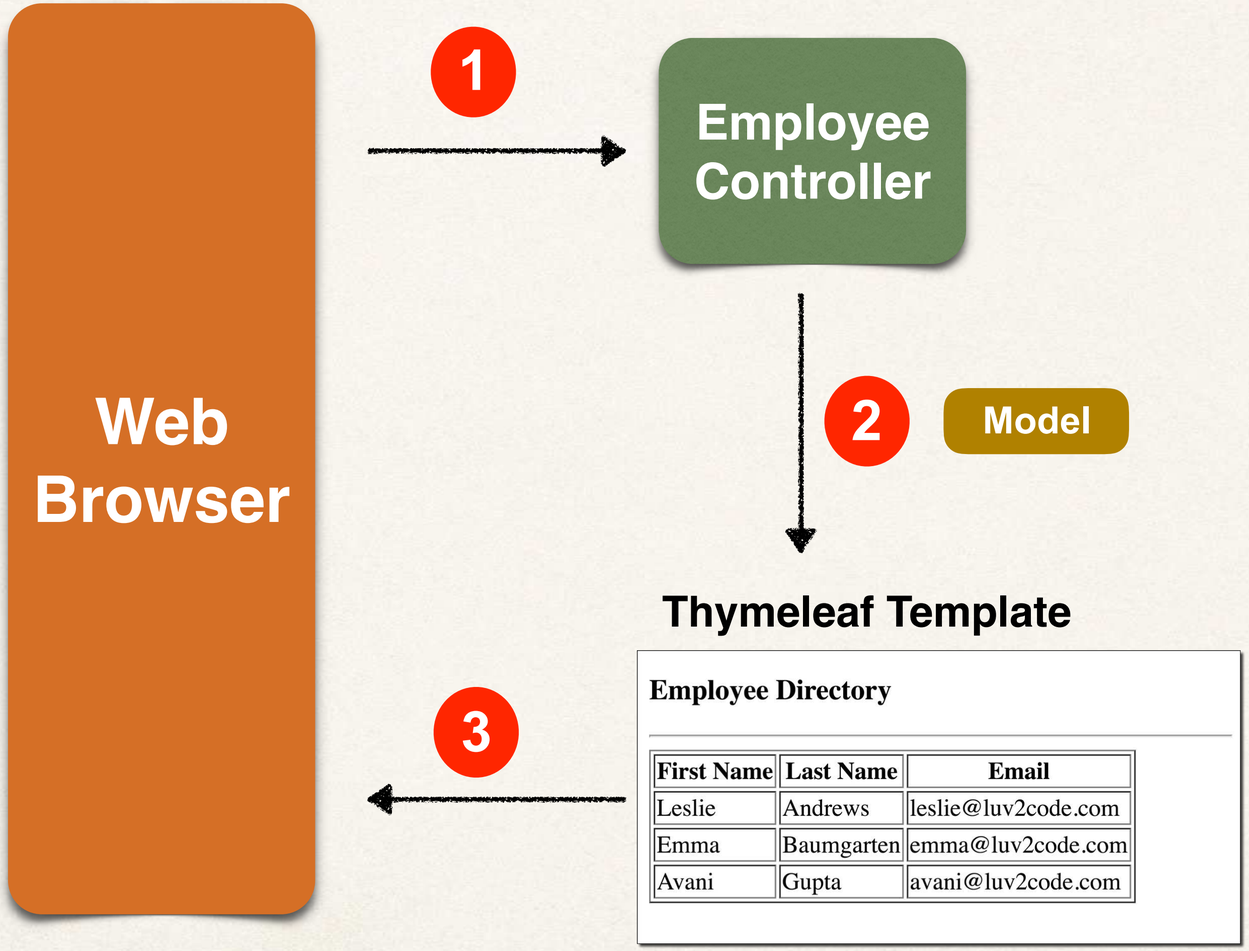
Big Picture


1
Web Browser
Employee Controller
Thymeleaf Template
Employee Directory
First Name | Last Name | Email
Leslie | Andrews | leslie@luv2code.com
Emma | Baumgarten | emma@luv2code.com
Avani | Gupta | avani@luv2code.com

Big Picture


Web Browser
1
Employee Controller
2
Model
Thymeleaf Template
Employee Directory
First Name | Last Name | Email
Leslie | Andrews | leslie@luv2code.com
Emma | Baumgarten | emma@luv2code.com
Avani | Gupta | avani@luv2code.com

Big Picture


Web Browser
1
Employee Controller
2
Model
Thymeleaf Template
Employee Directory
First Name | Last Name | Email
Leslie | Andrews | leslie@luv2code.com
Emma | Baumgarten | emma@luv2code.com
Avani | Gupta | avani@luv2code.com
3

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create Employee class

Development Process

Step-By-Step

1. Create Employee class

2. Create Employee Controller

Development Process

Step-By-Step

1. Create Employee class
2. Create Employee Controller
3. Create Thymeleaf template

Step 1: Create Employee class

Step 1: Create Employee class

- Regular Java class: fields, constructors, getters / setters

```
public class Employee {

    private int id;
    private String firstName;
    private String lastName;
    private String email;

    public Employee() {

    }

    public Employee(int id, String firstName, String lastName, String email) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
```

Step 2: Create Employee Controller

Step 2: Create Employee Controller

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

Step 2: Create Employee Controller

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

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

Step 2: Create Employee Controller

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");
```

Will add database integration later

Step 2: Create Employee Controller

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();
```

Step 2: Create Employee Controller

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);
```

Step 2: Create Employee Controller

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the Spring MVC model
      theModel.addAttribute("employees", theEmployees);
```

Step 2: Create Employee Controller

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the Spring MVC model
      theModel.addAttribute("employees", theEmployees);
```

Our Thymleaf template will access this data

Step 2: Create Employee Controller

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the Spring MVC model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Our Thymleaf template will access this data

Step 2: Create Employee Controller

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      Employee emp1 = new Employee(1, "Leslie", "Andrews", "leslie@luv2code.com");
      Employee emp2 = new Employee(2, "Emma", "Baumgarten", "emma@luv2code.com");
      Employee emp3 = new Employee(3, "Avani", "Gupta", "avani@luv2code.com");

      // create the list
      List<Employee> theEmployees = new ArrayList<>();

      // add to the list
      theEmployees.add(emp1);
      theEmployees.add(emp2);
      theEmployees.add(emp3);

      // add to the Spring MVC model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Our Thymleaf template will access this data

`src/main/resources/templates/list-employees.html`

Step 3: Create Thymeleaf template

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
```

To use Thymeleaf expressions

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
…
<body>

  <h3>Employee Directory</h3>
  <hr>
```

To use Thymeleaf expressions

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
…
<body>

  <h3>Employee Directory</h3>
  <hr>

  <table border="1">

    <!-- Build HTML table based on employees -->

  </table>

</body>

</html>
```

To use Thymeleaf expressions

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
…
<body>

  <h3>Employee Directory</h3>
  <hr>

  <table border="1">

    <!-- Build HTML table based on employees -->

  </table>

</body>

</html>
```

To use Thymeleaf expressions

To Do
Add code to loop over employees

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
    </tr>
  </thead>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Loop over
list of employees

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      ail</th>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Loop parameter

Loop over list of employees

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      ail</th>


  <tbody>
    <tr th:each="tempEmployee : ${employees}">
```

Loop parameter

Loop over list of employees

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      // create the list
      // add to the list
      … …

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Step 3: Create Thymeleaf template

File: list-employees.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      ail</th>

  <tbody>
    <tr th:each="tempEmployee : ${employees}">

      <td th:text="${tempEmployee.firstName}" />
      <td th:text="${tempEmployee.lastName}" />
      <td th:text="${tempEmployee.email}" />

    </tr>
  </tbody>
</table>
```

Loop parameter

Loop over list of employees

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

   @GetMapping("/list")
   public String listEmployees(Model theModel) {

      // create employees
      // create the list
      // add to the list
      … …

      // add to the spring model
      theModel.addAttribute("employees", theEmployees);

      return "list-employees";
   }
}
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Thymeleaf and Bootstrap

Let's Make our Page Beautiful

Let's Make our Page Beautiful

Before

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Let's Make our Page Beautiful

Before

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

After

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Let's Make our Page Beautiful

Before

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

After

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Bootstrap

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Get links for remote Bootstrap files

Development Process

Step-By-Step

1. Get links for remote Bootstrap files

2. Add links in Thymeleaf template

Development Process

Step-By-Step

1. Get links for remote Bootstrap files

2. Add links in Thymeleaf template

3. Apply Bootstrap CSS styles

Step 1: Get links for remote Bootstrap files

Step 1: Get links for remote Bootstrap files

- Visit Bootstrap website: **`www.getbootstrap.com`**

Step 1: Get links for remote Bootstrap files

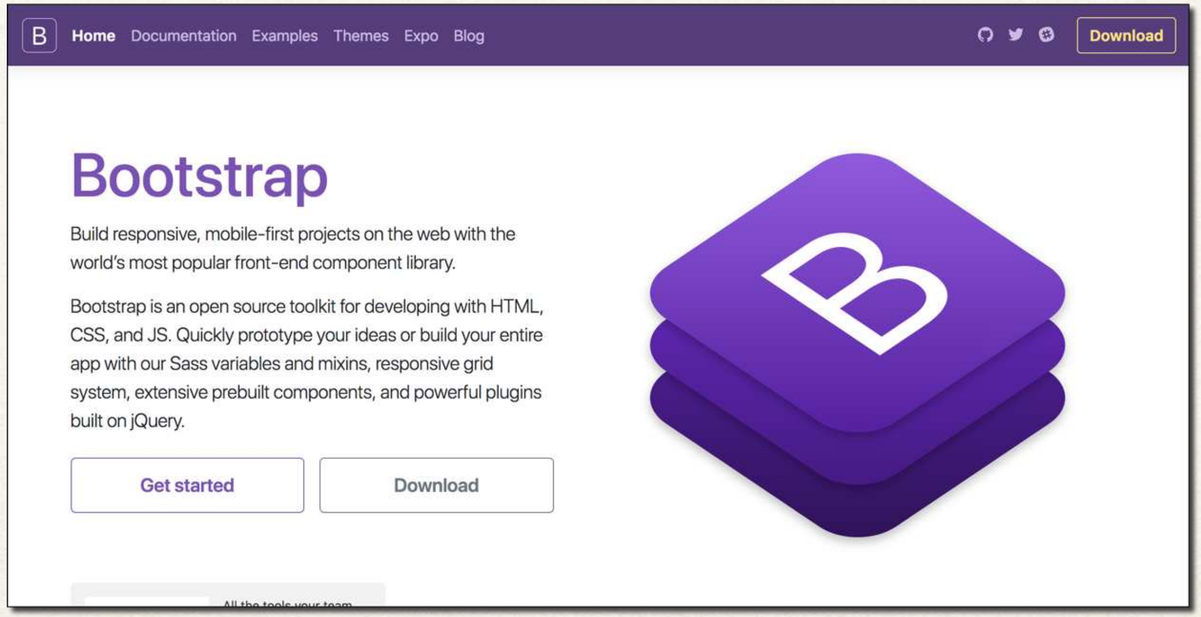
- Visit Bootstrap website: **`www.getbootstrap.com`**

- Website has instructions on how to **Get Started**

Step 1: Get links for remote Bootstrap files

- Visit Bootstrap website: **`www.getbootstrap.com`**

- Website has instructions on how to **Get Started**



Step 2: Add links in Thymeleaf template

Step 2: Add links in Thymeleaf template

```
<head>
    …
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/x.y.z/css/bootstrap.min.css" …>
    …
</head>
```

Step 2: Add links in Thymeleaf template

```
<head>
    …
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/x.y.z/css/bootstrap.min.css" …>
    …
</head>
```

Version number

Step 3: Apply CSS

Step 3: Apply CSS

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Step 3: Apply CSS

Bootstrap CSS style

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Step 3: Apply CSS

Bootstrap CSS style

Bootstrap CSS styles

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Step 3: Apply CSS

Bootstrap CSS style

Bootstrap CSS styles

```
<body>
  <div class="container">

    <h3>Employee Directory</h3>
    <hr>

    <table class="table table-bordered table-striped">
      <thead class="thead-dark">
        <tr>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
        </tr>
      </thead>
      … …
    </table>

  </div>
</body>
```

Employee Directory

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com

Thymeleaf CRUD - Real Time Project

Application Requirements

Application Requirements

From the Boss

Create a Web UI for the Employee Directory

Application Requirements

From the Boss

Create a Web UI for the Employee Directory

Users should be able to

- Get a list of employees

Application Requirements

From the Boss

Create a Web UI for the Employee Directory

Users should be able to
- Get a list of employees
- Add a new employee

Application Requirements

From the Boss

Create a Web UI for the Employee Directory

Users should be able to

- Get a list of employees
- Add a new employee
- Update an employee

Application Requirements

From the Boss

Create a Web UI for the Employee Directory

Users should be able to

- Get a list of employees
- Add a new employee
- Update an employee
- Delete an employee

Application Requirements

From the Boss

Create a Web UI for the Employee Directory

Users should be able to

- Get a list of employees
- Add a new employee
- Update an employee
- Delete an employee

Thymeleaf + Spring Boot

Real-Time Project

Thymeleaf + Spring Boot

Real-Time Project

Thymeleaf + Spring Boot

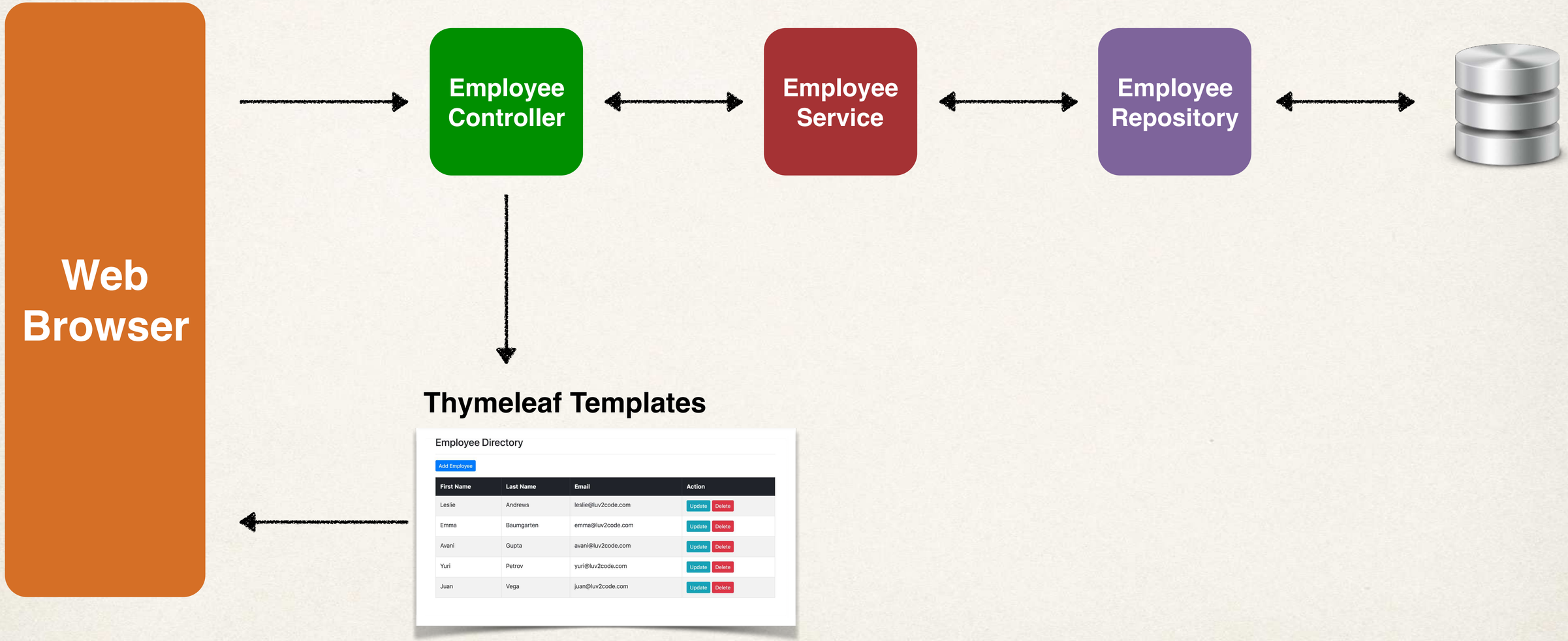
Employee Directory

Add Employee

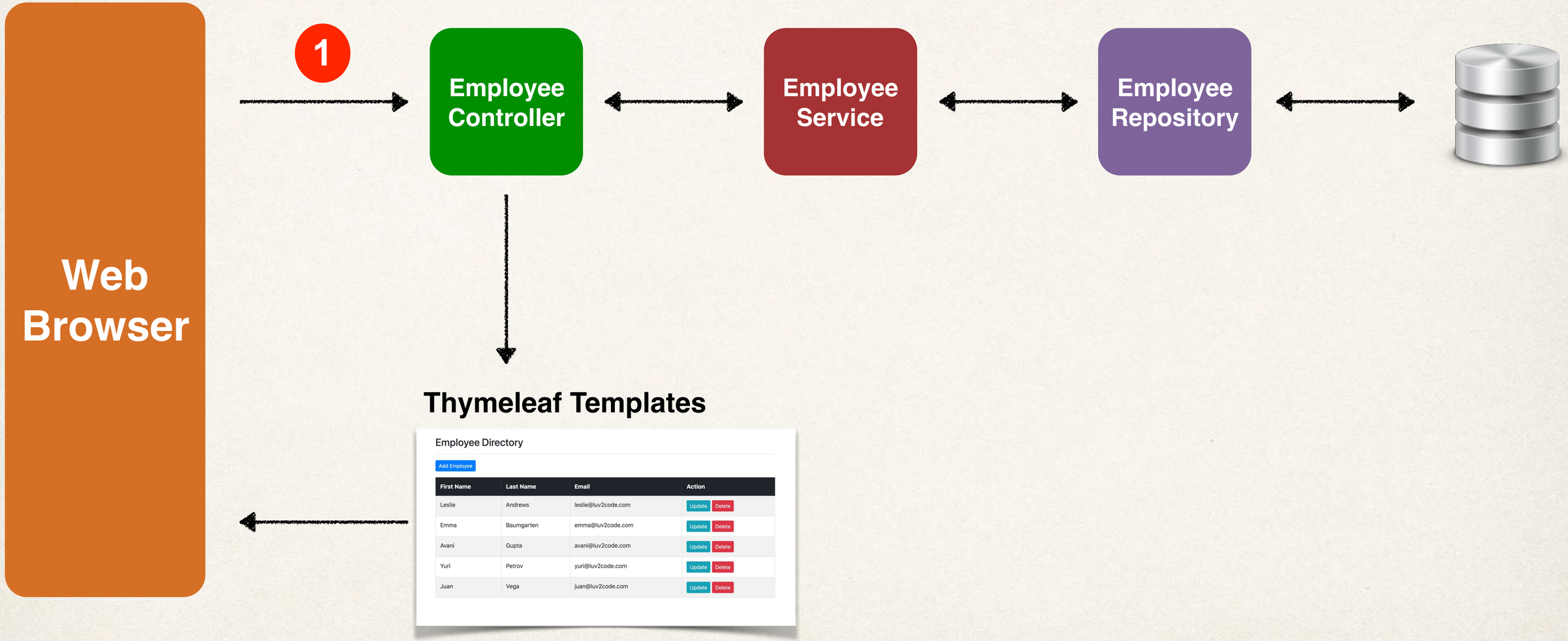
First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete
Emma	Baumgarten	emma@luv2code.com	Update Delete
Avani	Gupta	avani@luv2code.com	Update Delete
Yuri	Petrov	yuri@luv2code.com	Update Delete
Juan	Vega	juan@luv2code.com	Update Delete

Big Picture

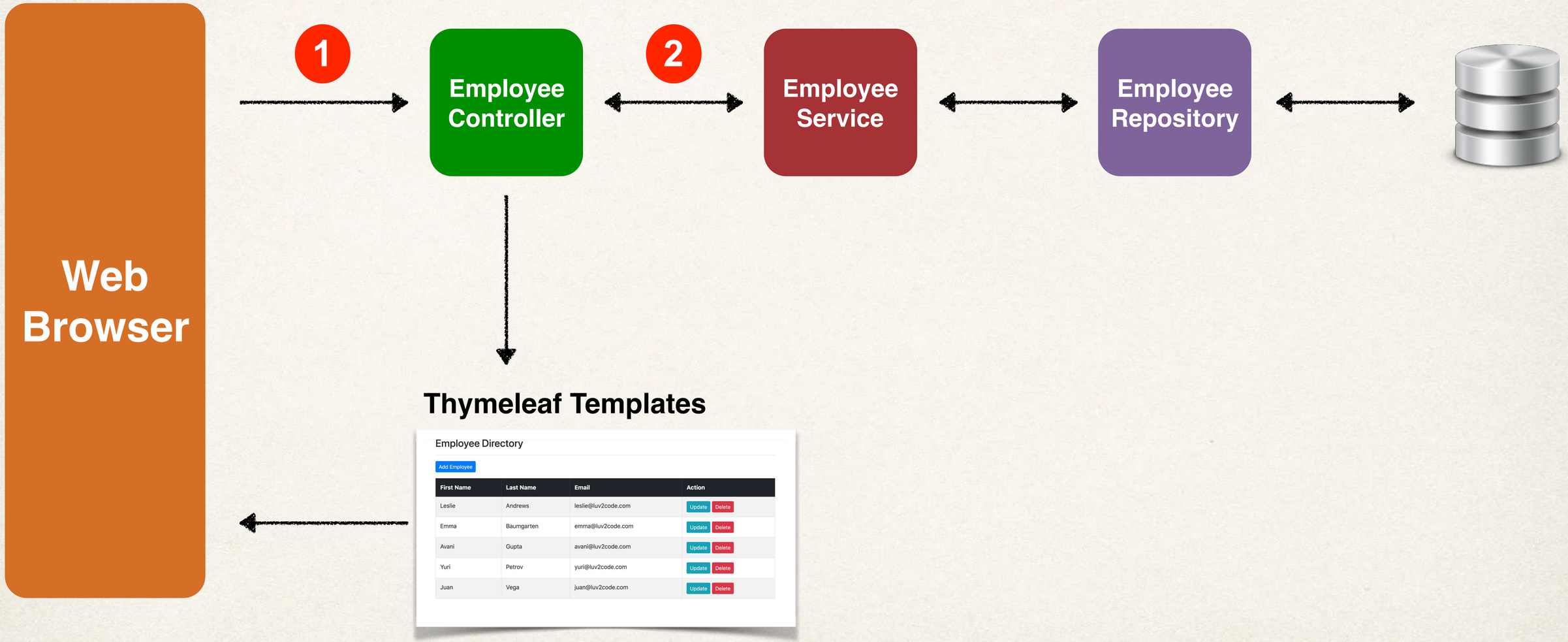
Big Picture


Web Browser
Employee Controller
Employee Service
Employee Repository
Thymeleaf Templates
Employee Directory
Add Employee
First Name
Last Name
Email
Action
Leslie
Andrews
leslie@luv2code.com
Update
Delete
Emma
Baumgarten
emma@luv2code.com
Update
Delete
Avani
Gupta
avani@luv2code.com
Update
Delete
Yuri
Petrov
yuri@luv2code.com
Update
Delete
Juan
Vega
juan@luv2code.com
Update
Delete

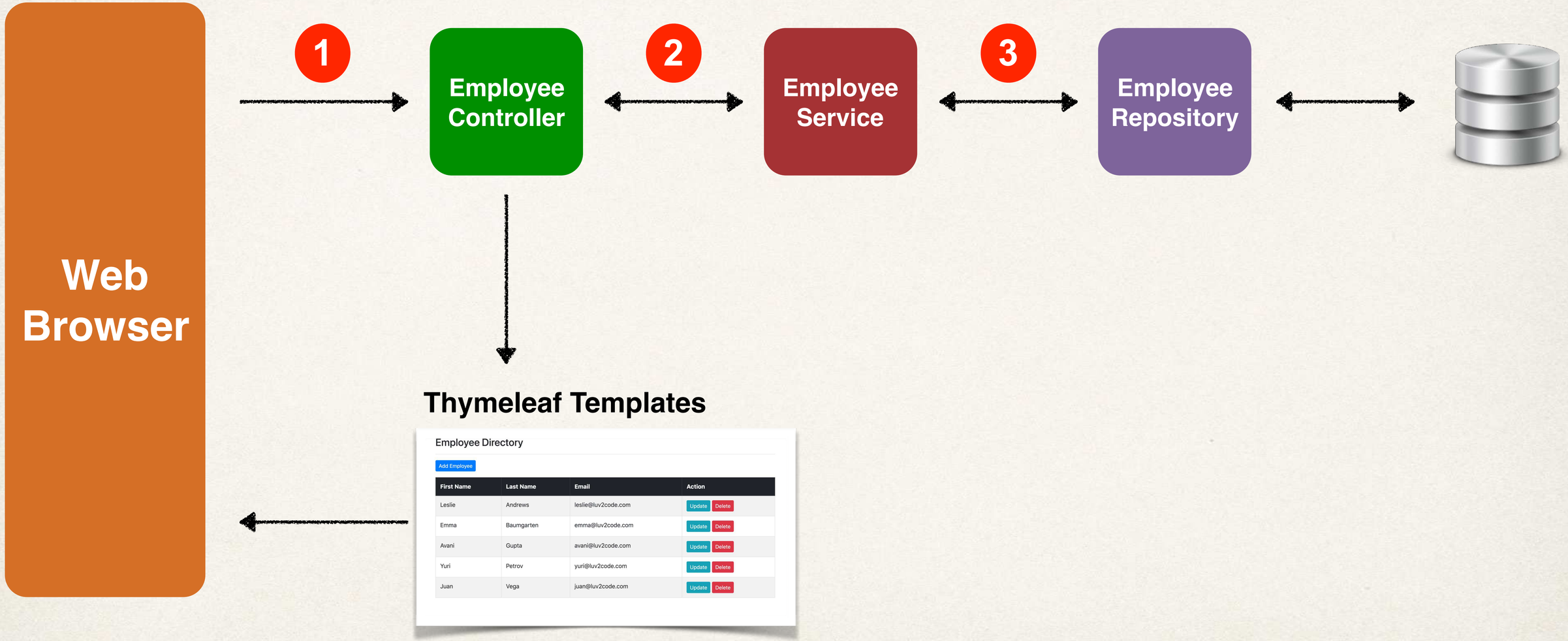
Big Picture


Web Browser
1
Employee Controller
Employee Service
Employee Repository
Thymeleaf Templates
Employee Directory
Add Employee
First Name
Last Name
Email
Action
Leslie
Andrews
leslie@luv2code.com
Update
Delete
Emma
Baumgarten
emma@luv2code.com
Update
Delete
Avani
Gupta
avani@luv2code.com
Update
Delete
Yuri
Petrov
yuri@luv2code.com
Update
Delete
Juan
Vega
juan@luv2code.com
Update
Delete

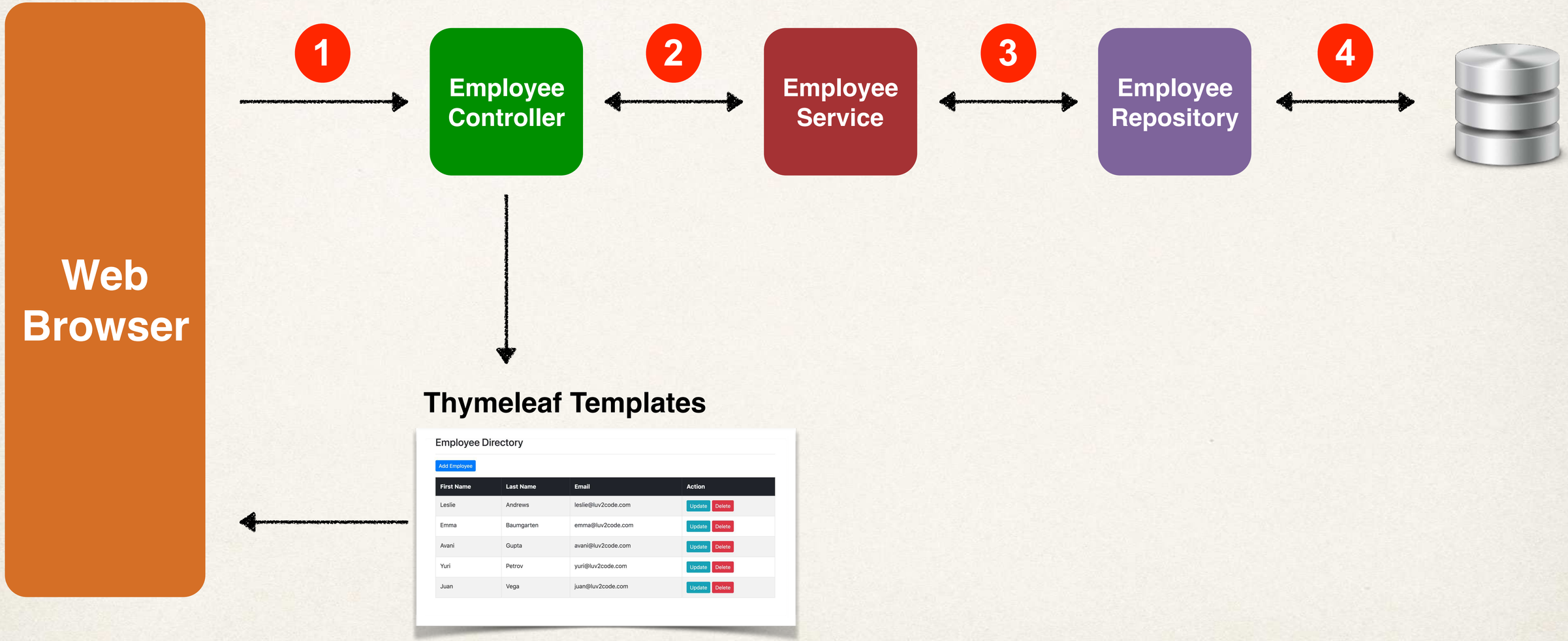
Big Picture


Web Browser
1
Employee Controller
2
Employee Service
Employee Repository
Thymeleaf Templates
Employee Directory
Add Employee
First Name
Last Name
Email
Action
Leslie
Andrews
leslie@luv2code.com
Update
Delete
Emma
Baumgarten
emma@luv2code.com
Update
Delete
Avani
Gupta
avani@luv2code.com
Update
Delete
Yuri
Petrov
yuri@luv2code.com
Update
Delete
Juan
Vega
juan@luv2code.com
Update
Delete

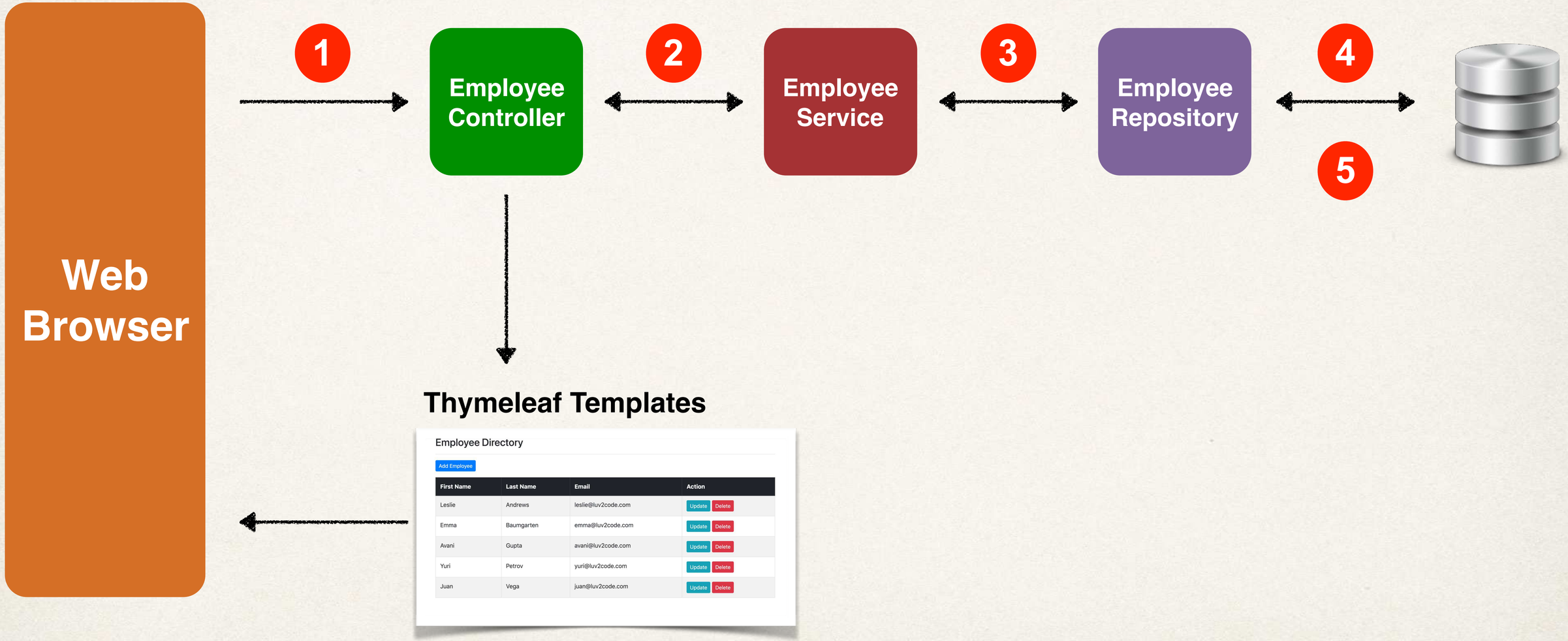
Big Picture


Web Browser
1
Employee Controller
2
Employee Service
3
Employee Repository
Thymeleaf Templates
Employee Directory
Add Employee
First Name | Last Name | Email | Action
Leslie | Andrews | leslie@luv2code.com | Update Delete
Emma | Baumgarten | emma@luv2code.com | Update Delete
Avani | Gupta | avani@luv2code.com | Update Delete
Yuri | Petrov | yuri@luv2code.com | Update Delete
Juan | Vega | juan@luv2code.com | Update Delete

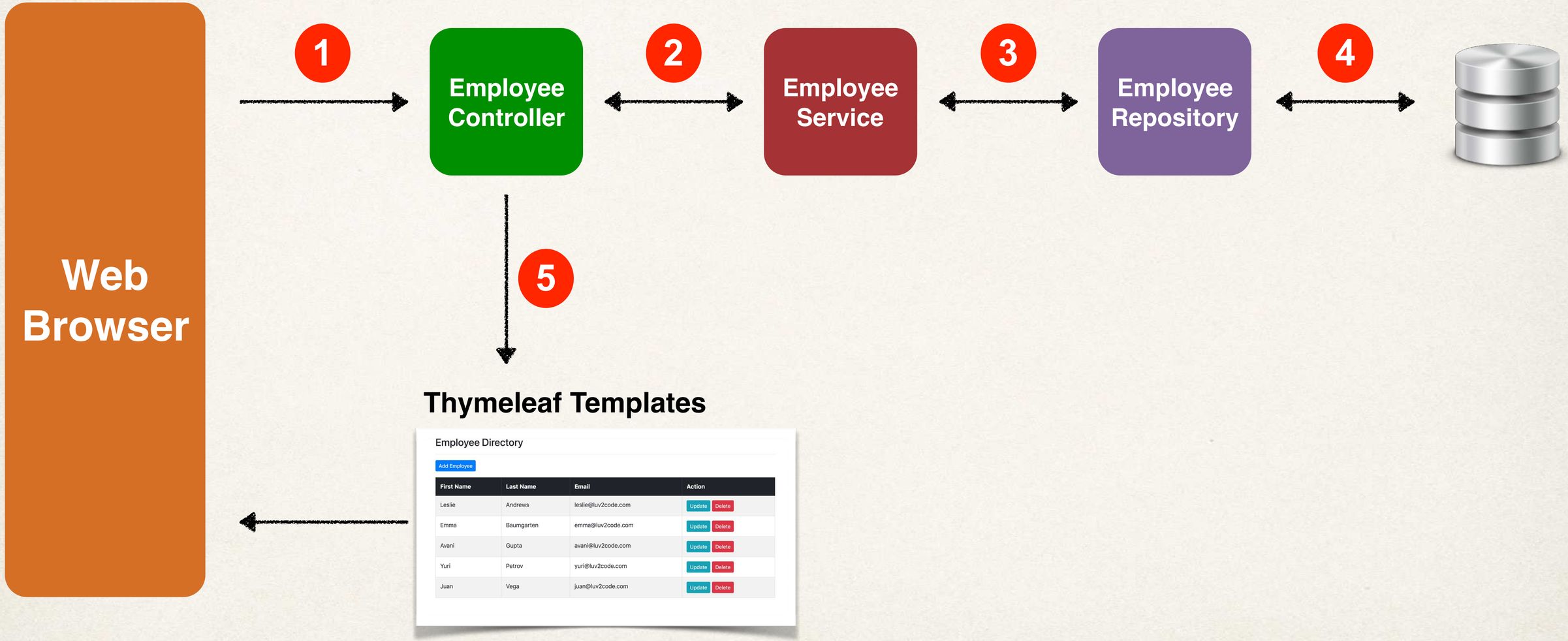
Big Picture


Web Browser
1
Employee Controller
2
Employee Service
3
Employee Repository
4
Thymeleaf Templates
Employee Directory
Add Employee
First Name
Last Name
Email
Action
Leslie
Andrews
leslie@luv2code.com
Update
Delete
Emma
Baumgarten
emma@luv2code.com
Update
Delete
Avani
Gupta
avani@luv2code.com
Update
Delete
Yuri
Petrov
yuri@luv2code.com
Update
Delete
Juan
Vega
juan@luv2code.com
Update
Delete

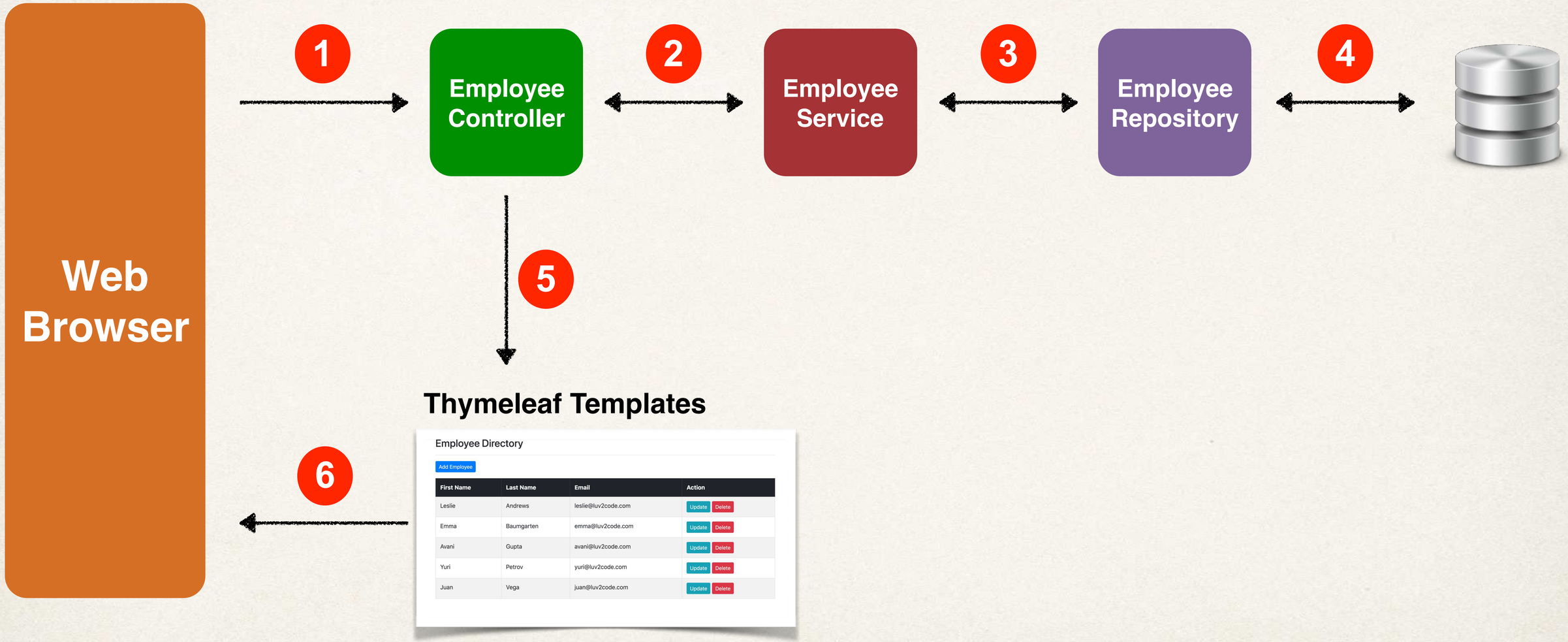
Big Picture


Web Browser
1
Employee Controller
2
Employee Service
3
Employee Repository
4
5
Thymeleaf Templates
Employee Directory
Add Employee
First Name
Last Name
Email
Action
Leslie
Andrews
leslie@luv2code.com
Update
Delete
Emma
Baumgarten
emma@luv2code.com
Update
Delete
Avani
Gupta
avani@luv2code.com
Update
Delete
Yuri
Petrov
yuri@luv2code.com
Update
Delete
Juan
Vega
juan@luv2code.com
Update
Delete

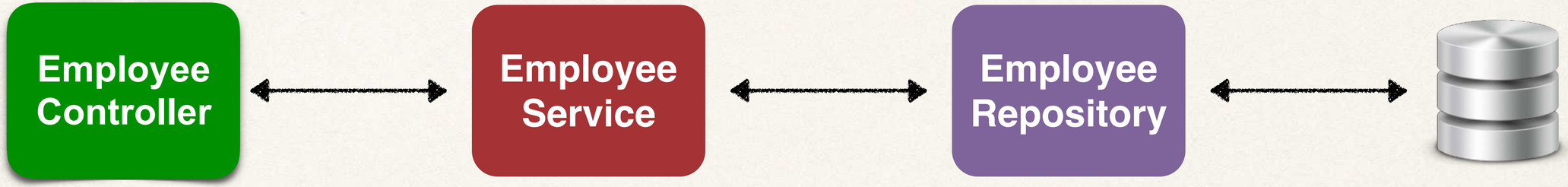
Big Picture


Web Browser
1
Employee Controller
2
Employee Service
3
Employee Repository
4
5
Thymeleaf Templates
Employee Directory
Add Employee
First Name
Last Name
Email
Action
Leslie
Andrews
leslie@luv2code.com
Update
Delete
Emma
Baumgarten
emma@luv2code.com
Update
Delete
Avani
Gupta
avani@luv2code.com
Update
Delete
Yuri
Petrov
yuri@luv2code.com
Update
Delete
Juan
Vega
juan@luv2code.com
Update
Delete

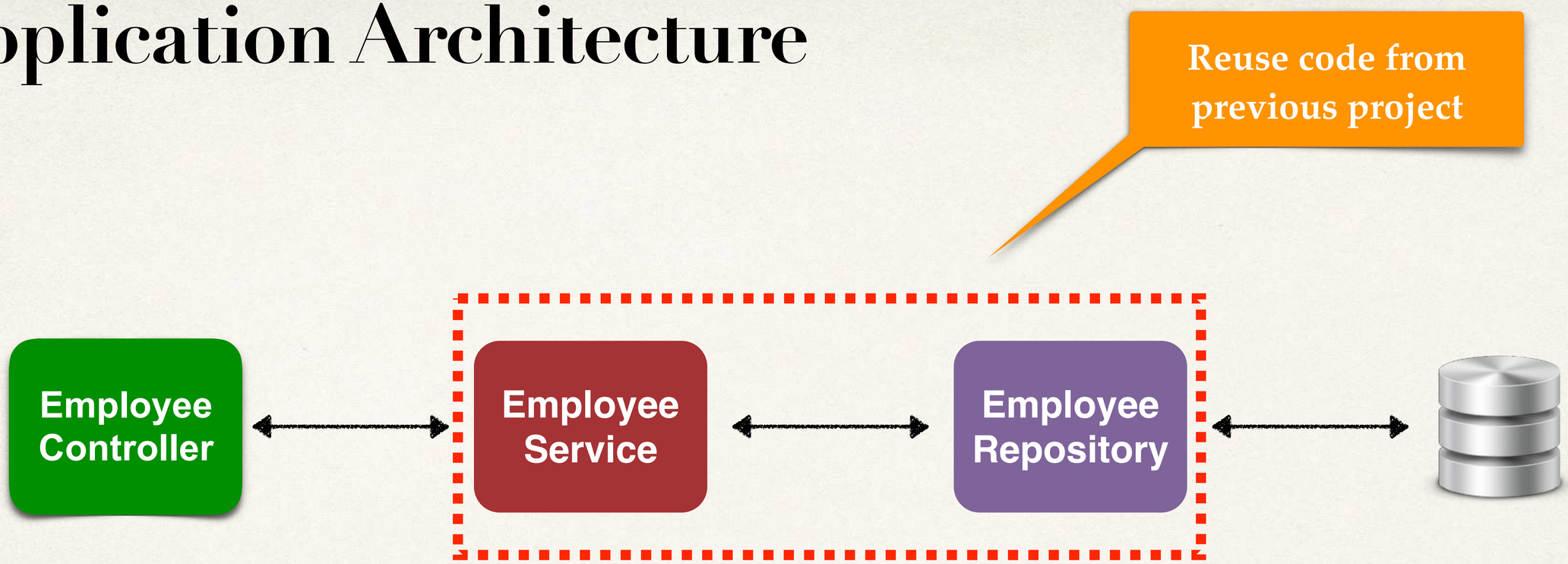
Big Picture


Web Browser
1
Employee Controller
2
Employee Service
3
Employee Repository
4
5
Thymeleaf Templates
6
Employee Directory
Add Employee
First Name
Last Name
Email
Action
Leslie
Andrews
leslie@luv2code.com
Update
Delete
Emma
Baumgarten
emma@luv2code.com
Update
Delete
Avani
Gupta
avani@luv2code.com
Update
Delete
Yuri
Petrov
yuri@luv2code.com
Update
Delete
Juan
Vega
juan@luv2code.com
Update
Delete

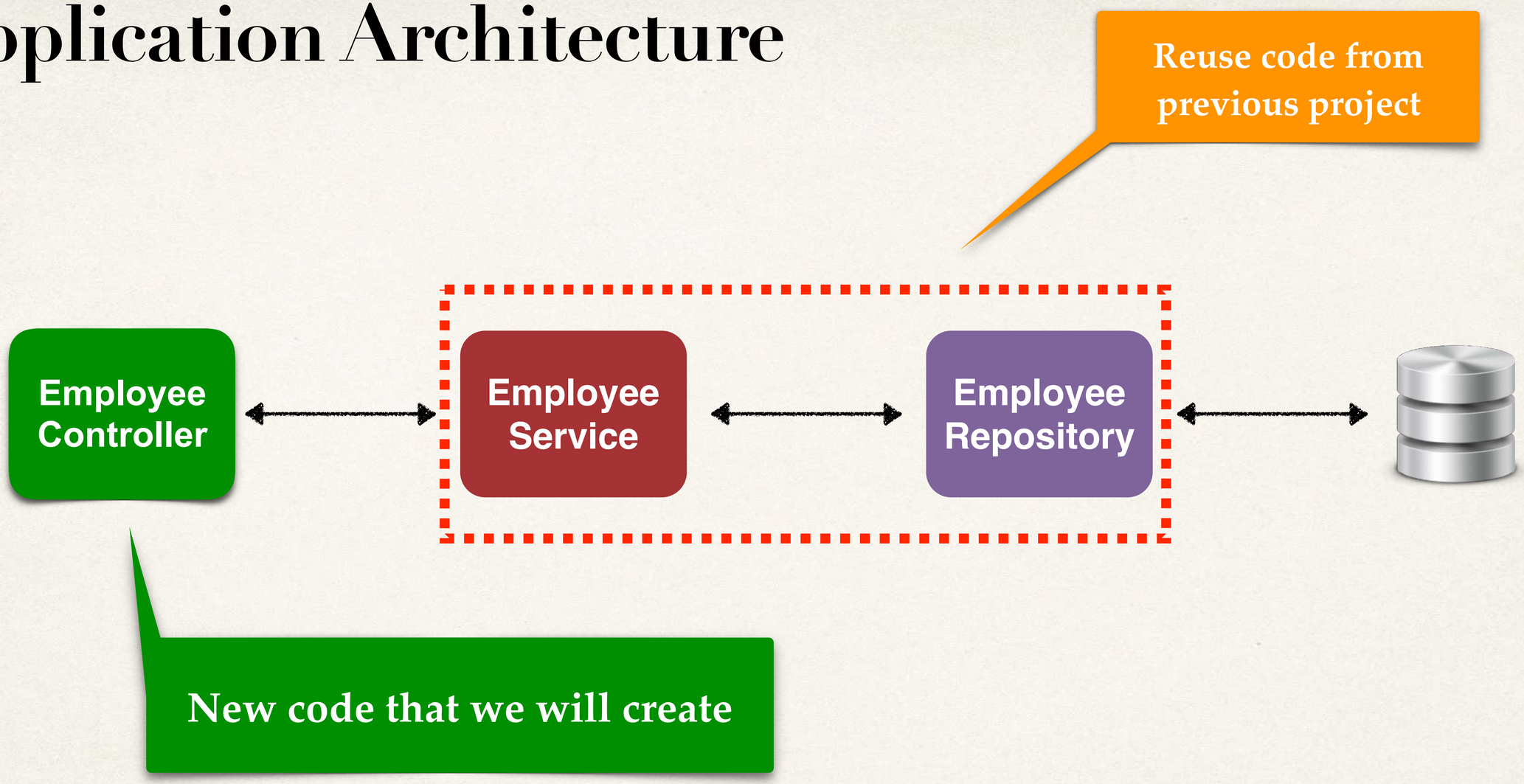
Application Architecture



Application Architecture



Application Architecture



Project Set Up

Project Set Up

- We will extend our existing Employee project and add DB integration

Project Set Up

- We will extend our existing Employee project and add DB integration

- Add **`EmployeeService`**, **`EmployeeRepository`** and **`Employee`** entity

Project Set Up

- We will extend our existing Employee project and add DB integration

- Add **`EmployeeService`**, **`EmployeeRepository`** and **`Employee`** entity
  - Available in one of our previous projects

Project Set Up

- We will extend our existing Employee project and add DB integration

- Add **`EmployeeService`**, **`EmployeeRepository`** and **`Employee`** entity
  - Available in one of our previous projects
  - We created all of this code already from scratch … so we'll just copy/paste it

Project Set Up

- We will extend our existing Employee project and add DB integration

- Add **`EmployeeService`**, **`EmployeeRepository`** and **`Employee`** entity
  - Available in one of our previous projects
  - We created all of this code already from scratch … so we'll just copy/paste it

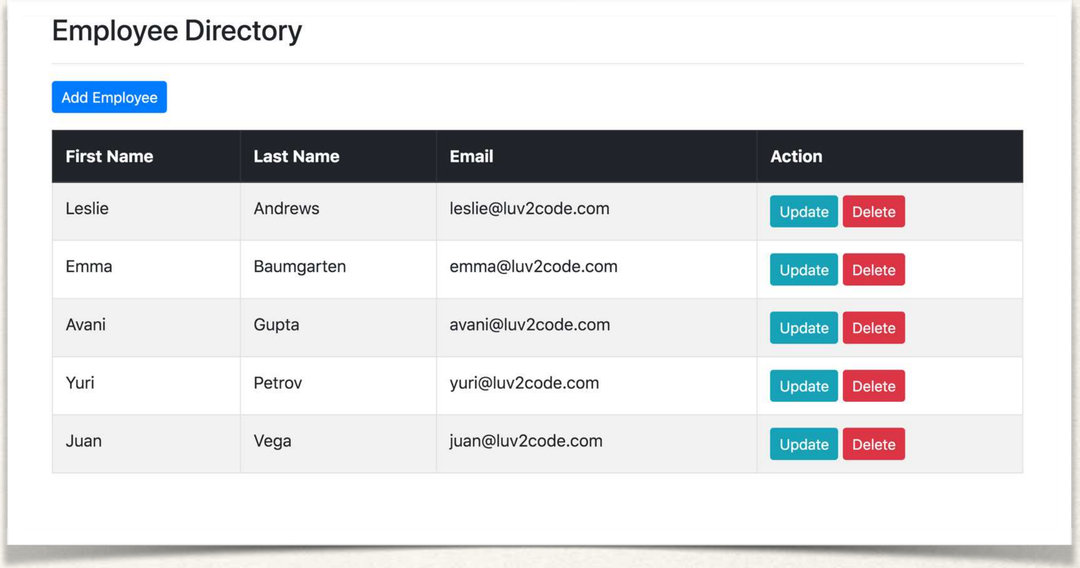
- Allows us to focus on creating **`EmployeeController`** and Thymeleaf templates

Development Process - Big Picture

Step-By-Step

Development Process - Big Picture

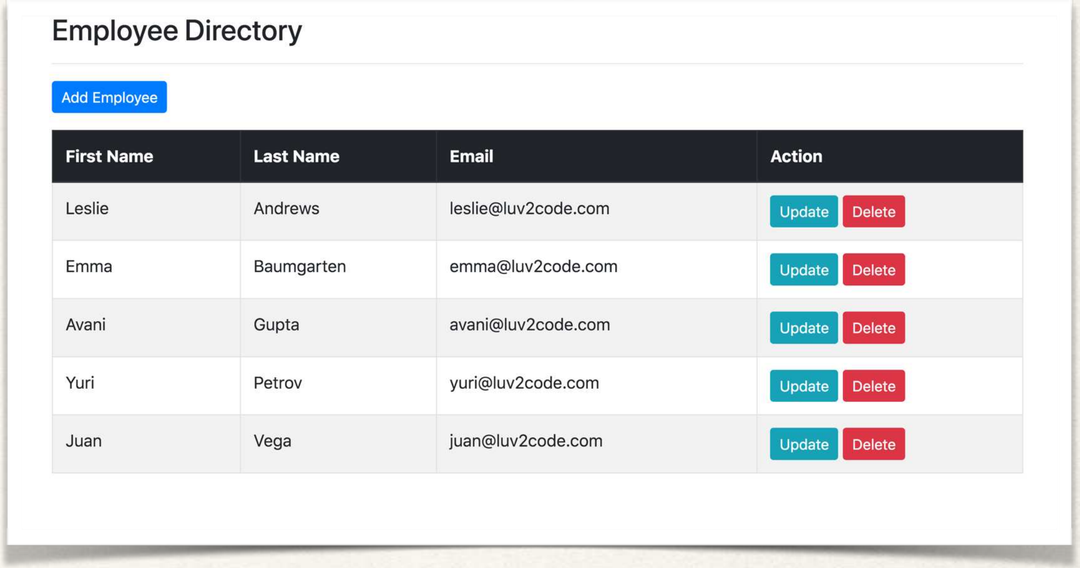
Step-By-Step



Development Process - Big Picture

Step-By-Step

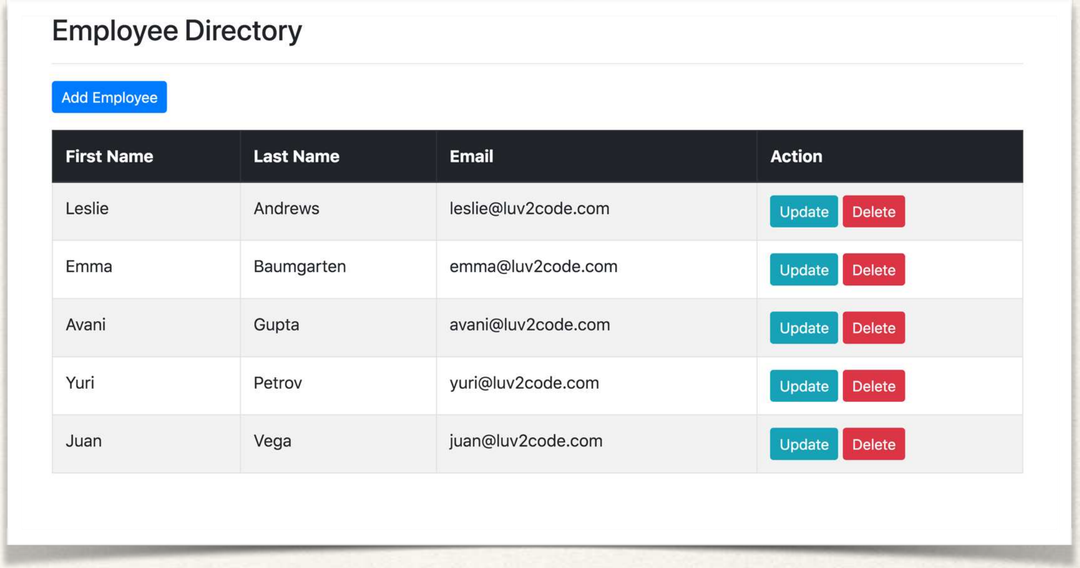
1. Get list of employees



Development Process - Big Picture

Step-By-Step

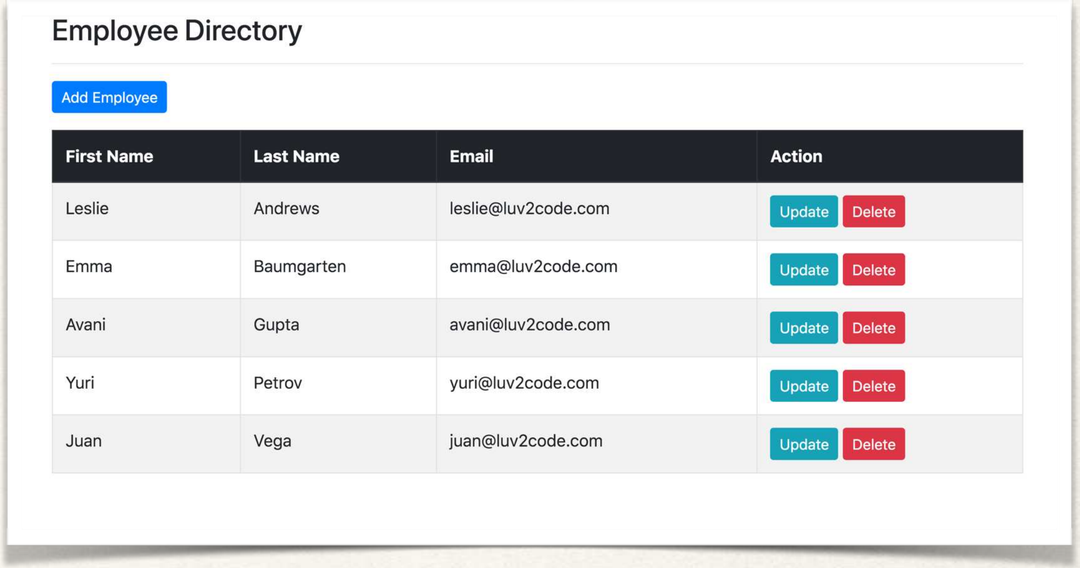
1. Get list of employees
2. Add a new employee



Development Process - Big Picture

Step-By-Step

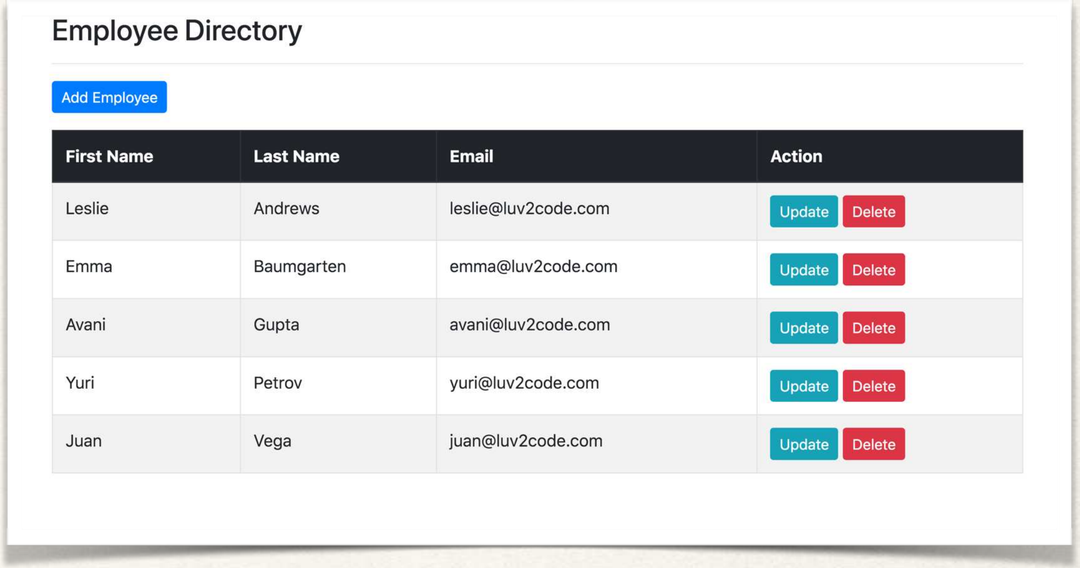
1. Get list of employees
2. Add a new employee
3. Update an existing employee



Development Process - Big Picture

Step-By-Step

1. Get list of employees
2. Add a new employee
3. Update an existing employee
4. Delete an existing employee



Thymeleaf - Add Employee

Add Employee - DEMO

Employee Directory

Add Employee

First Name	Last Name	Email
Leslie	Andrews	leslie@luv2code.com
Emma	Baumgarten	emma@luv2code.com
Avani	Gupta	avani@luv2code.com
Yuri	Petrov	yuri@luv2code.com
Juan	Vega	juan@luv2code.com

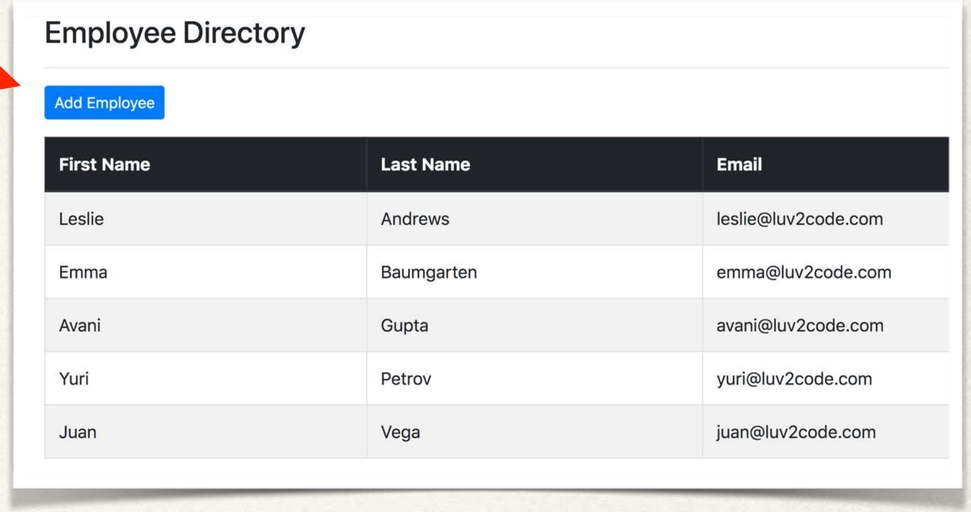
Add Employee

Step-By-Step

Add Employee

Step-By-Step

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

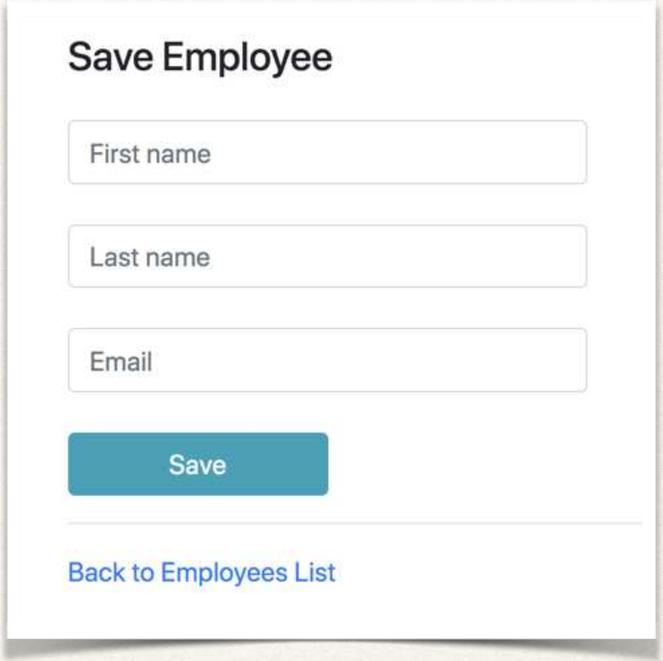


Add Employee

Step-By-Step

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

2. Create HTML form for new employee



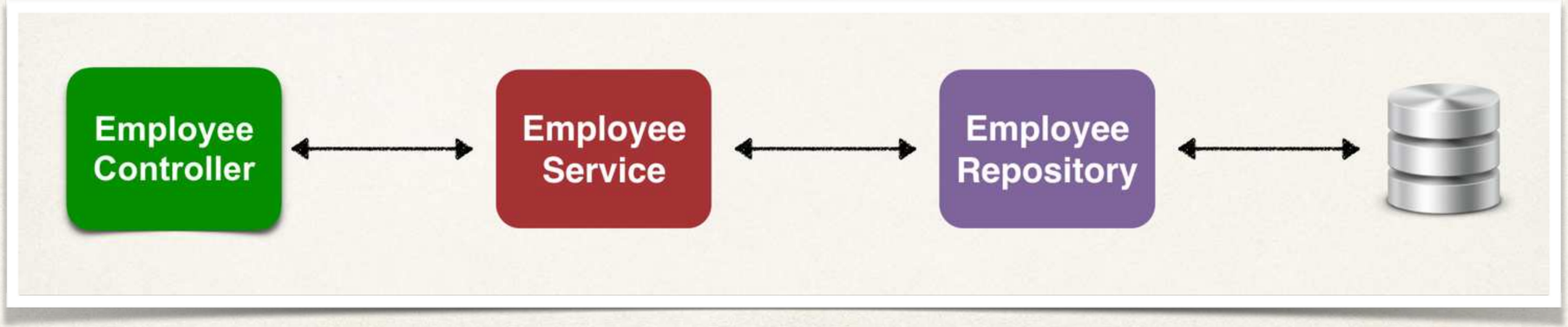
Add Employee

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>
```

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>
```

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
  - 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>
```

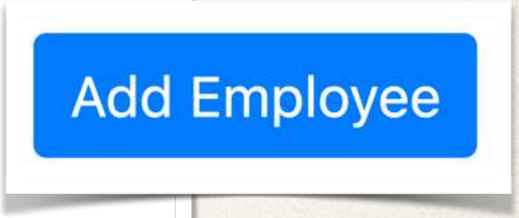
Add Employee

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

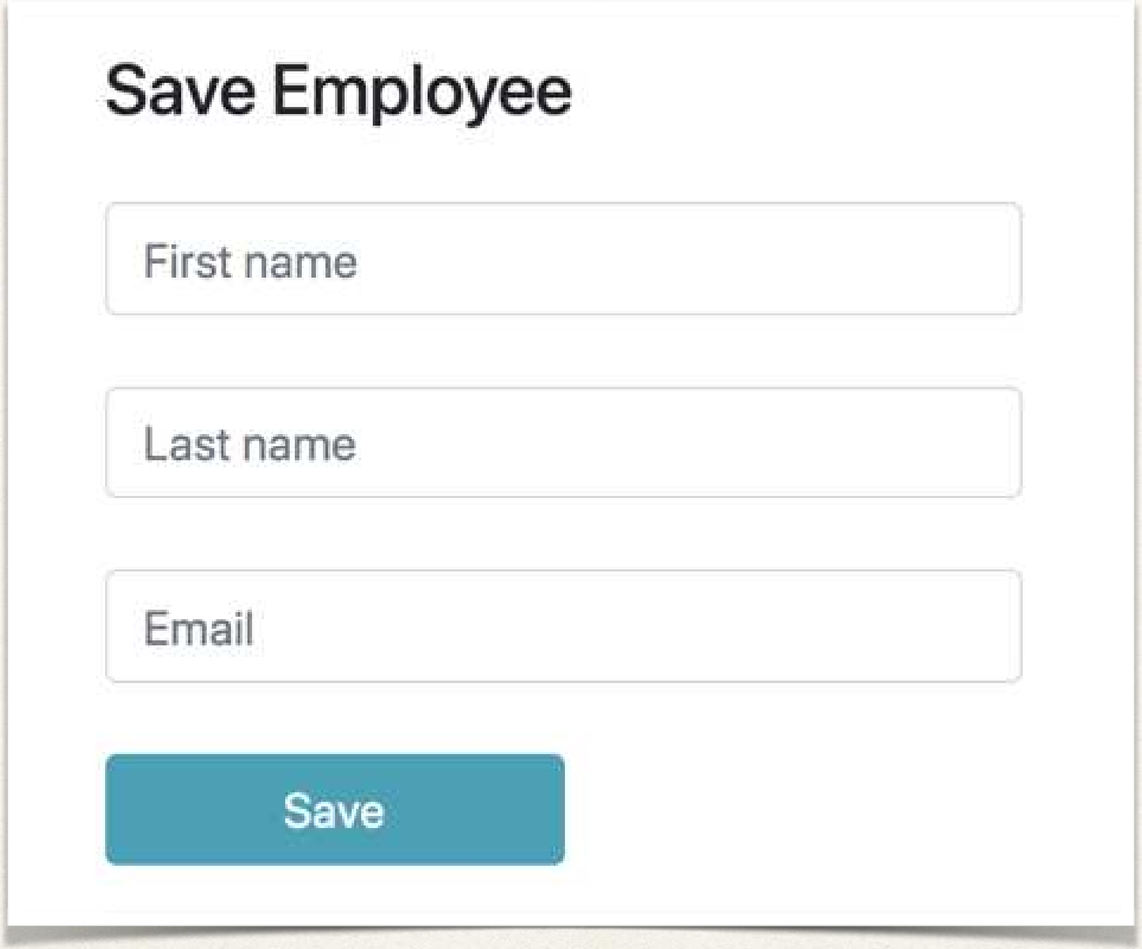
```
<form action="#" th:action="@{/employees/save}"
      th:object="${employee}" method="POST">


</form>
```

Our model attribute

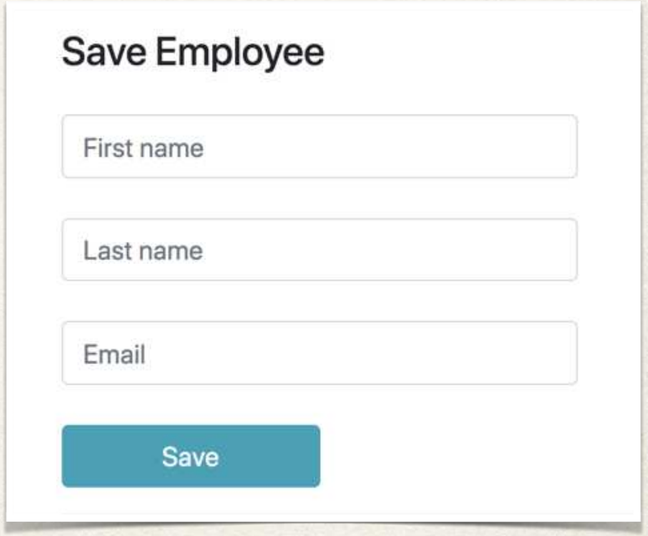
```
theModel.addAttribute("employee", theEmployee);
```

Step 2: Create HTML form for new employee

Step 2: Create HTML form for new employee

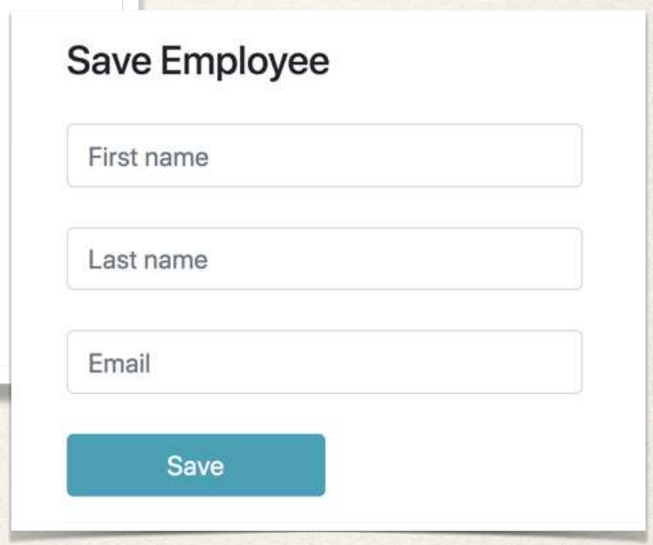


Step 2: Create HTML form for new employee



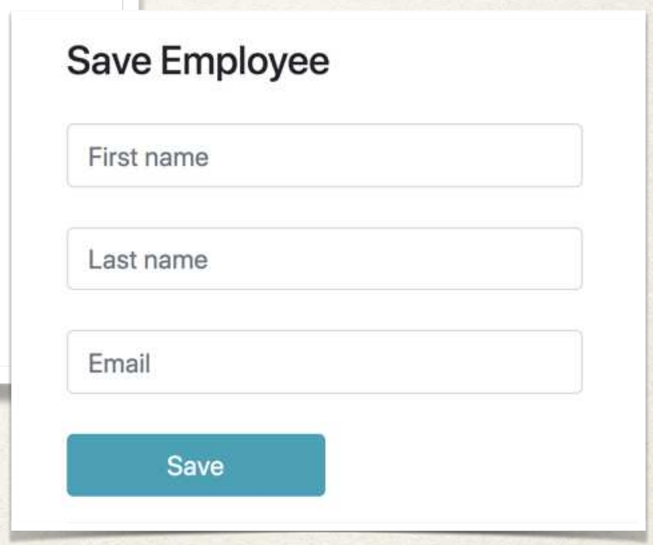
Step 2: Create HTML form for new employee

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



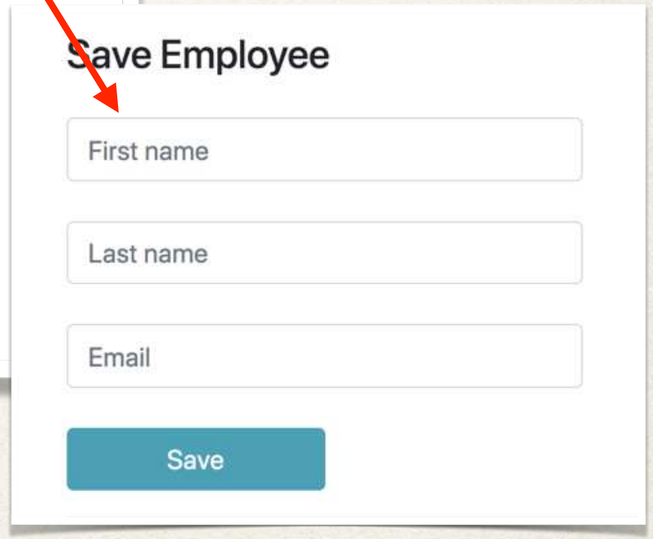
Step 2: Create HTML form for new employee

```
<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

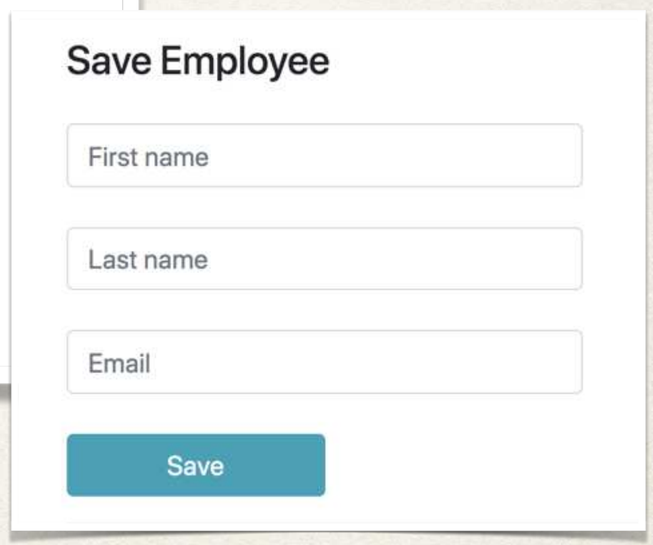
```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <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">
```

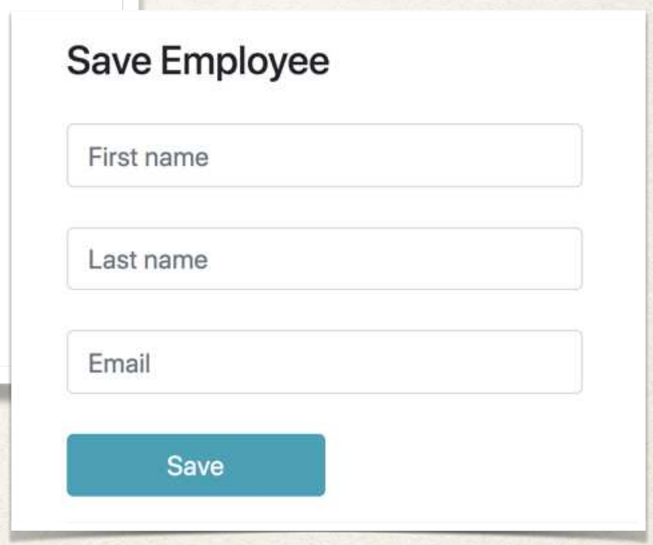
***{...}**
Selects property on referenced
th:object



Step 2: Create HTML form for new employee

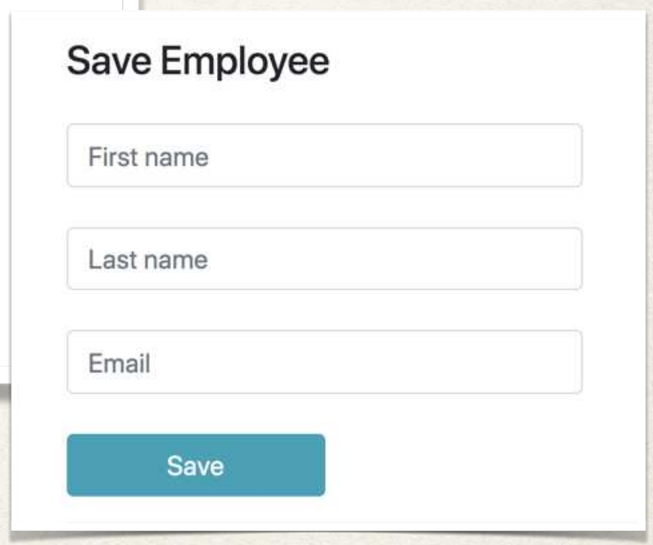
```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

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

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



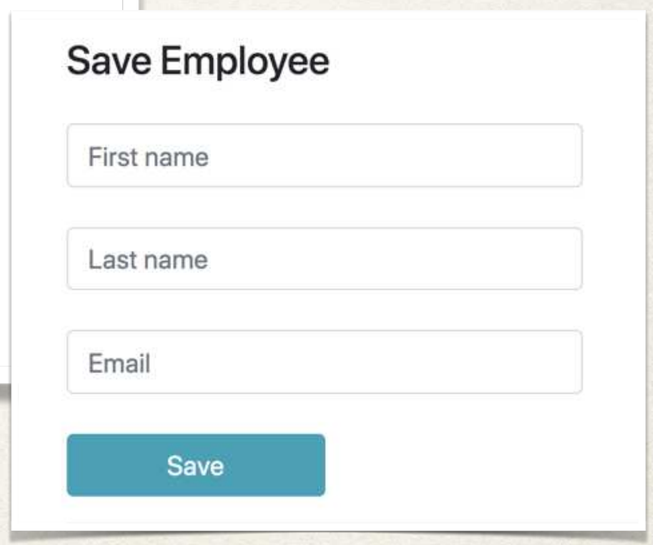
Step 2: Create HTML form for new employee

```
<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:

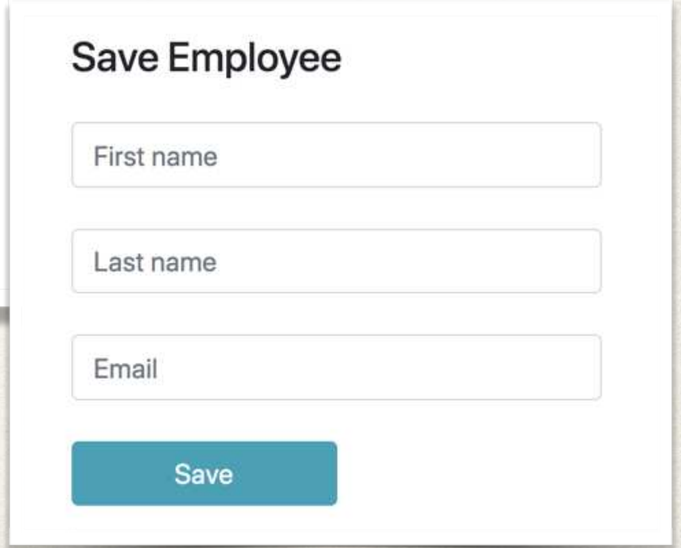
employee.setFirstName(...)
...
employee.setLastName(...)

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

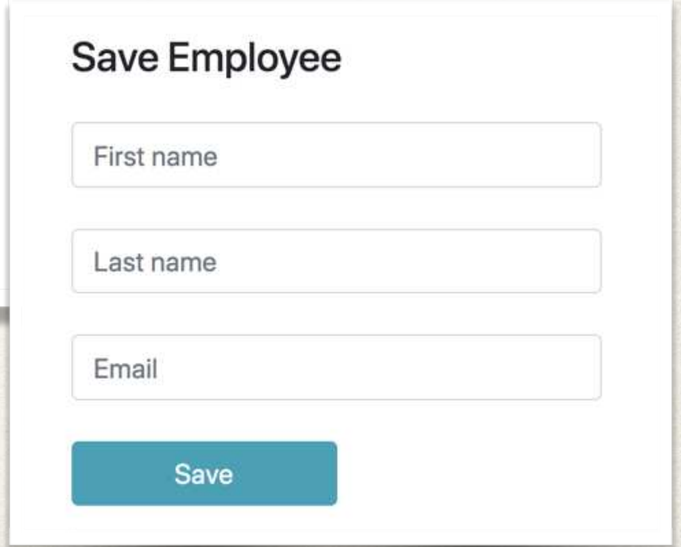
```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <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">


  <                                {email}" placeholder="Email">


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

</form>
```

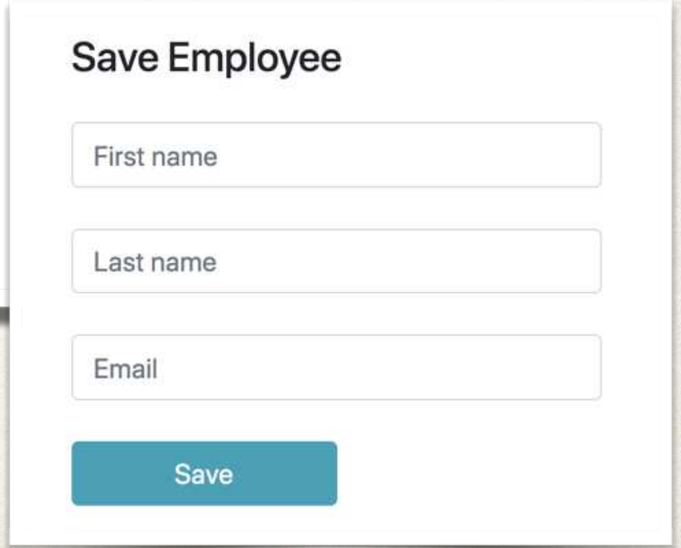
Apply Bootstrap styles



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>
```

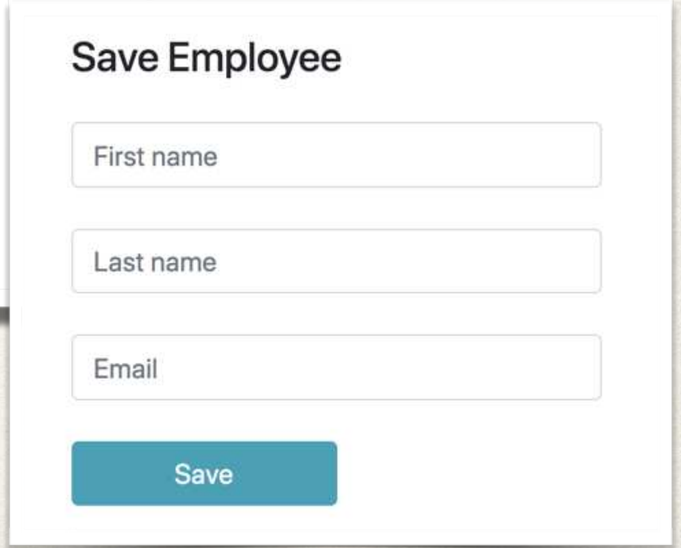
Apply Bootstrap styles



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

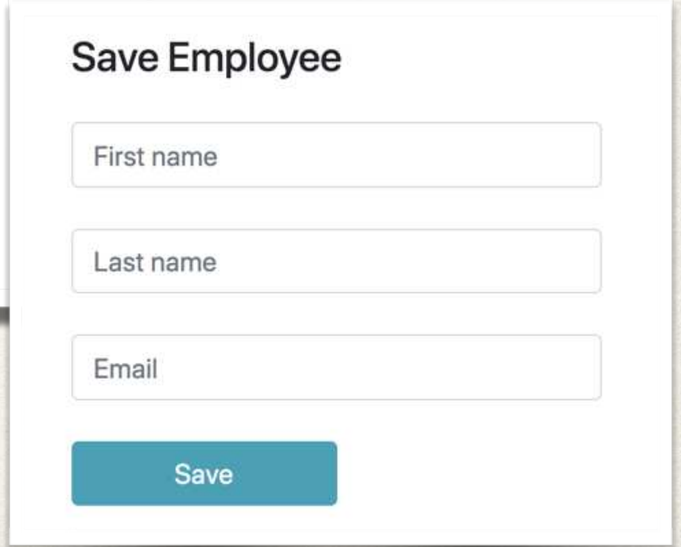


Step 2: Create HTML form for new employee

```
<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>
```

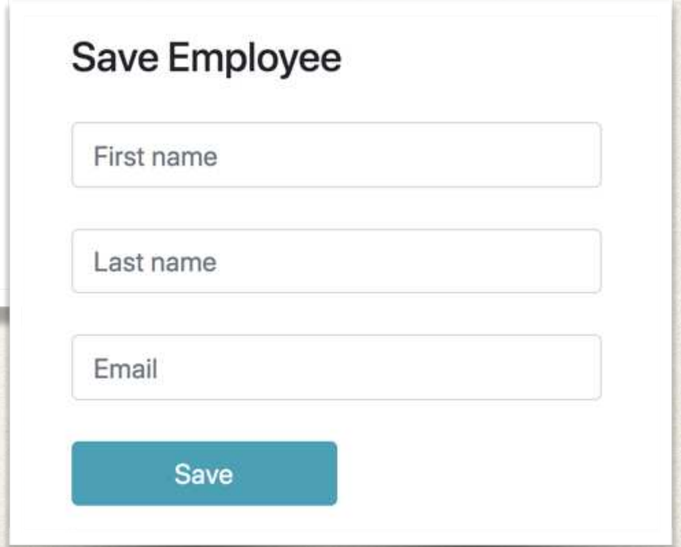
Apply Bootstrap styles

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>
```

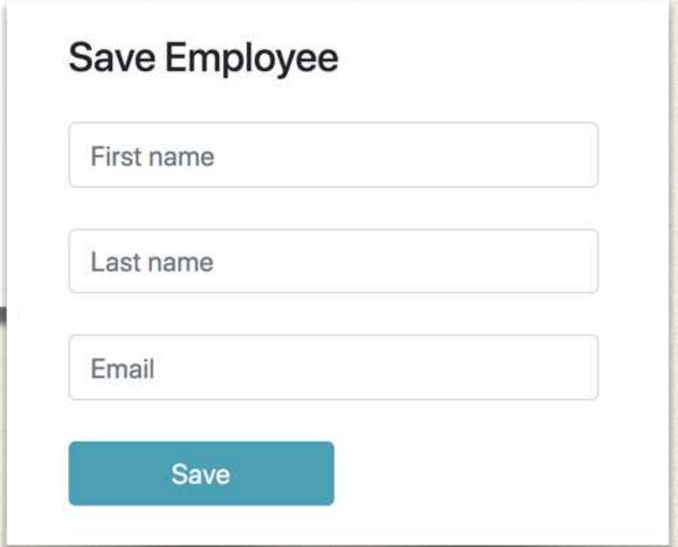


Step 2: Create HTML form for new employee

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

  <input type="text" th:field="*{firstName}" placehol
         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>
```

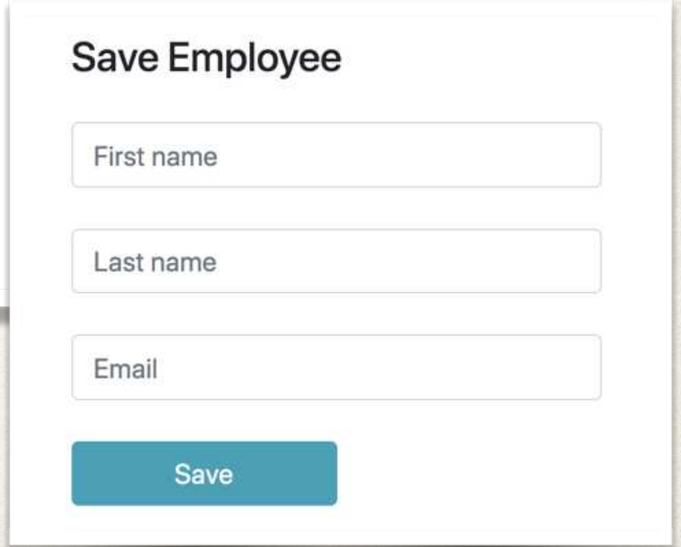
Button
Button Info
Column Span 2

Apply Bootstrap styles



Step 2: Create HTML form for new employee

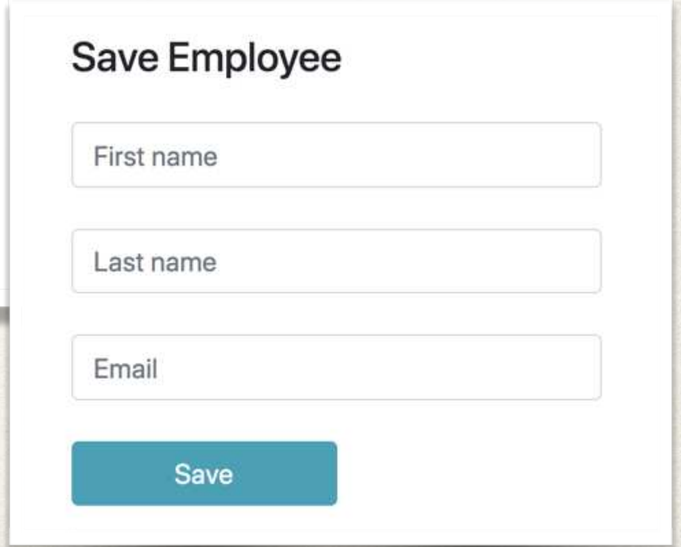
```
<form action="#" th:action="@{/employees/save}"
          th:object="${employee}" method="POST">

  <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) {
```

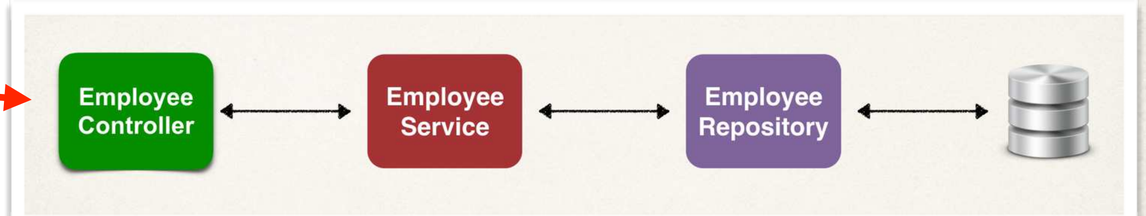
Step 3: Process form data to save employee

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

   private EmployeeService emp

   public EmployeeController(E
      employeeService = theEmpl
   }

   @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

Thymeleaf - Update Employee

Update Employee - Demo

Employee Directory

Add Employee

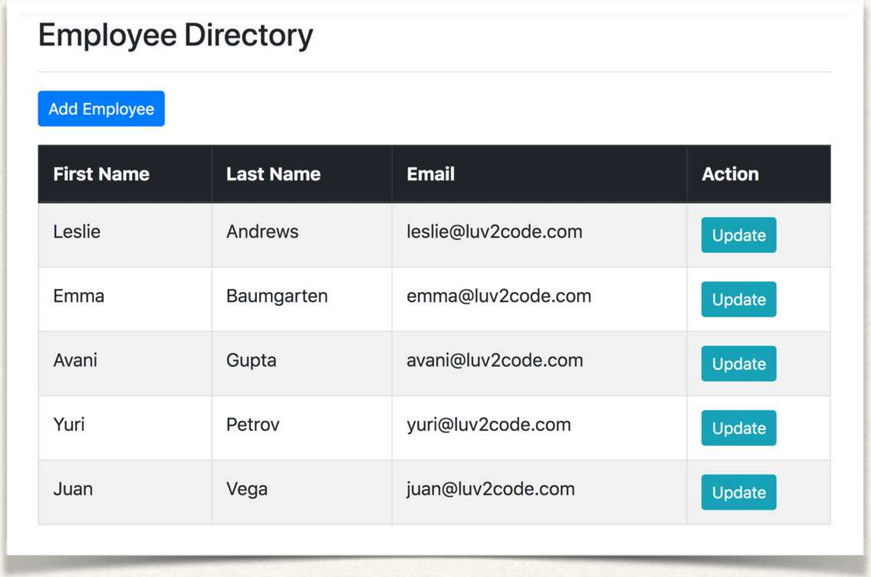
First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update
Emma	Baumgarten	emma@luv2code.com	Update
Avani	Gupta	avani@luv2code.com	Update
Yuri	Petrov	yuri@luv2code.com	Update
Juan	Vega	juan@luv2code.com	Update

Update Employee

Step-By-Step

Update Employee

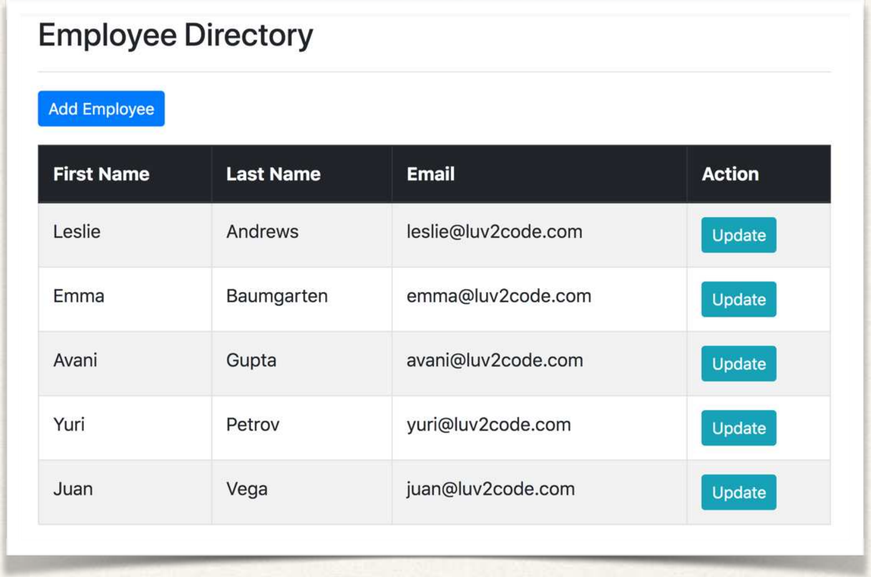
Step-By-Step



Update Employee

Step-By-Step

1. "Update" button

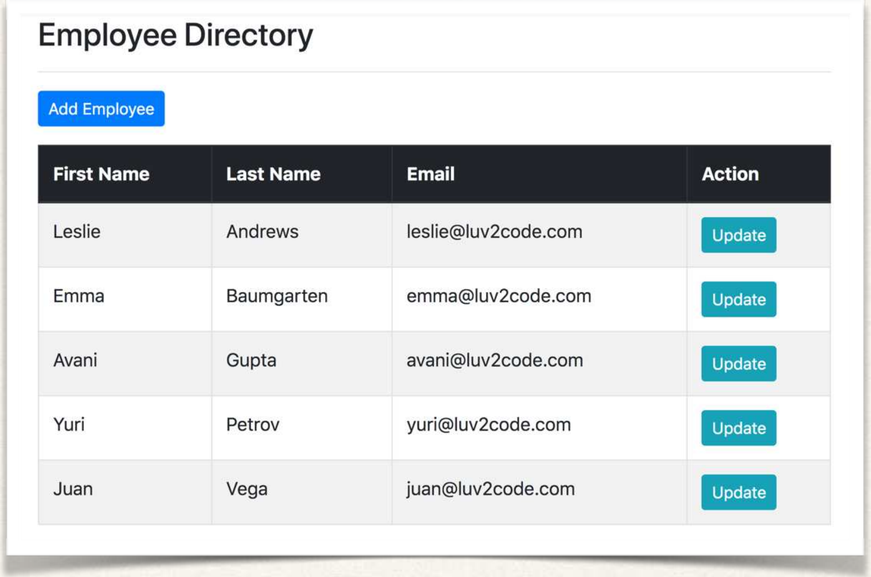


Update Employee

Step-By-Step

1. "Update" button

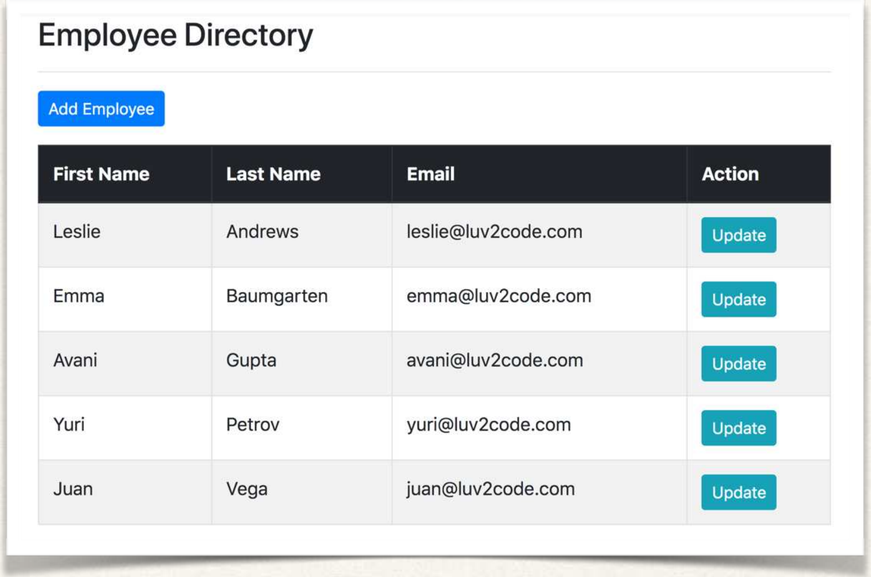
2. Pre-populate the form



Update Employee

Step-By-Step

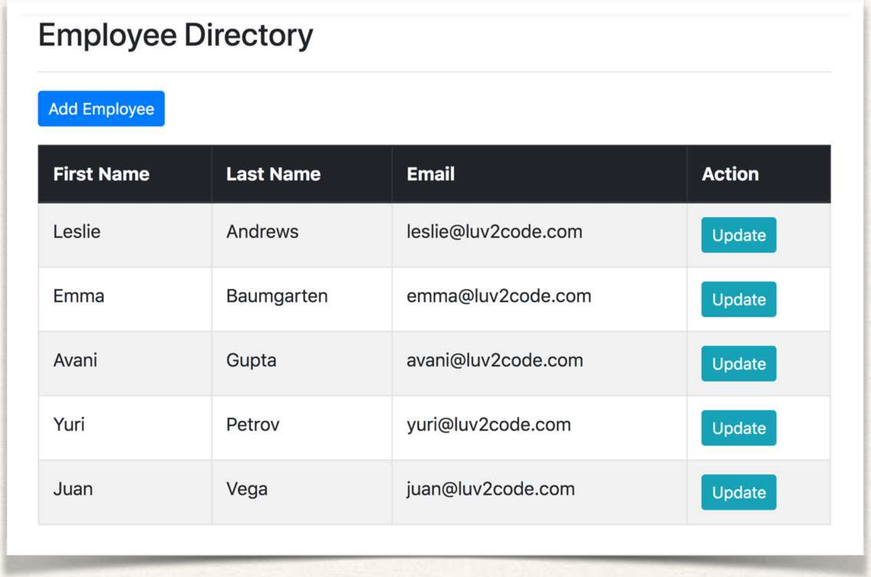
1. "Update" button
2. Pre-populate the form
3. Process form data

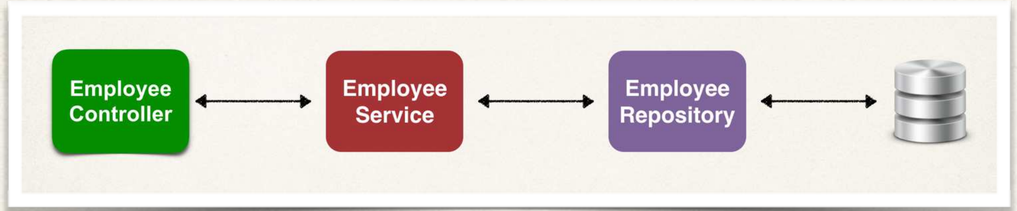


Update Employee

Step-By-Step

1. "Update" button
2. Pre-populate the form
3. Process form data





Step 1: "Update" Button

Step 1: "Update" Button

Employee Directory

Add Employee

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update
Emma	Baumgarten	emma@luv2code.com	Update
Avani	Gupta	avani@luv2code.com	Update
Yuri	Petrov	yuri@luv2code.com	Update
Juan	Vega	juan@luv2code.com	Update

Step 1: "Update" Button

Employee Directory

Add Employee

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update
Emma	Baumgarten	emma@luv2code.com	Update
Avani	Gupta	avani@luv2code.com	Update
Yuri	Petrov	yuri@luv2code.com	Update
Juan	Vega	juan@luv2code.com	Update

Each row has an **Update** link

- current employee id embedded in link

When **clicked**

- will load the employee from database
- prepopulate the form

Step 1: "Update" button

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update
Emma	Baumgarten	emma@luv2code.com	Update

Step 1: "Update" button

- **Update** button includes employee id

Step 1: "Update" button

- **Update** button includes employee id

```
<tr th:each="tempEmployee : ${employees}">
  ...
```

Step 1: "Update" button

- **Update** button includes employee id

```
<tr th:each="tempEmployee : ${employees}">
  …
  <td>

    <a th:href="@{/employees/showFormForUpdate(employeeId=${tempEmployee.id})}"
       class="btn btn-info btn-sm">
      Update
    </a>

  </td>

</tr>
```

Step 1: "Update" button

- **Update** button includes employee id

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update
Emma	Baumgarten	emma@luv2code.com	Update

```
<tr th:each="tempEmployee : ${employees}">
  …
  <td>

    <a th:href="@{/employees/showFormForUpdate(employeeId=${tempEmployee.id})}"
       class="btn btn-info btn-sm">
      Update
    </a>

  </td>

</tr>
```

Appends to URL

?employeeId=xxx

Step 2: Pre-populate Form

Step 2: Pre-populate Form

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

Step 2: Pre-populate Form

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

  @GetMapping("/showFormForUpdate")
  public String showFormForUpdate(@RequestParam("employeeId") int theId,
                                  Model theModel) {
```

Step 2: Pre-populate Form

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

  @GetMapping("/showFormForUpdate")
  public String showFormForUpdate(@RequestParam("employeeId") int theId,
                                  Model theModel) {
```

```
<a th:href="@{/employees/showFormForUpdate(employeeId=${tempEmployee.id})}"
```

Step 2: Pre-populate Form

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

  @GetMapping("/showFormForUpdate")
  public String showFormForUpdate(@RequestParam("employeeId") int theId,
                                  Model theModel) {

    // get the employee from the service
    Employee theEmployee = employeeService.findById(theId);
```

```
<a th:href="@{/employees/showFormForUpdate(employeeId=${tempEmployee.id})}"
```

Step 2: Pre-populate Form

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

  @GetMapping("/showFormForUpdate")
  public String showFormForUpdate(@RequestParam("employeeId") int theId,
                                  Model theModel) {

    // get the employee from the service
    Employee theEmployee = employeeService.findById(theId);

    // set employee as a model attribute to pre-populate the form
    theModel.addAttribute("employee", theEmployee);
```

```
<a th:href="@{/employees/showFormForUpdate(employeeId=${tempEmployee.id})}"
```

Step 2: Pre-populate Form

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

  @GetMapping("/showFormForUpdate")
  public String showFormForUpdate(@RequestParam("employeeId") int theId,
                                  Model theModel) {

    // get the employee from the service
    Employee theEmployee = employeeService.findById(theId);

    // set employee as a model attribute to pre-populate the form
    theModel.addAttribute("employee", theEmployee);

    // send over to our form
    return "employees/employee-form";
  }

}
```

```
<a th:href="@{/employees/showFormForUpdate(employeeId=${tempEmployee.id})}"
```

Step 2: Pre-populate Form

Step 2: Pre-populate Form

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

  <!-- Add hidden form field to handle update -->
  <input type="hidden" th:field="*{id}" />

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

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

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

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

</form>
```

Step 2: Pre-populate Form

1

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

employee.getFirstName()
...
employee.getLastName

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

  <!-- Add hidden form field to handle update -->
  <input type="hidden" th:field="*{id}" />

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

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

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

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

</form>
```

Step 2: Pre-populate Form

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

  <!-- Add hidden form field to handle update -->
  <input type="hidden" th:field="*{id}" />

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

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

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

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

</form>
```

1

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

employee.getFirstName()
...
employee.getLastName

This is how form is pre-populated
Thanks to calls to getters

Step 2: Pre-populate Form

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

  <!-- Add hidden form field to handle update -->
  <input type="hidden" th:field="*{id}" />

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

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

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

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

</form>
```

Step 2: Pre-populate Form

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

  <!-- Add hidden form field to handle update -->
  <input type="hidden" th:field="*{id}" />

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

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

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

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

</form>
```

Hidden form field required for updates

Step 2: Pre-populate Form

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

  <!-- Add hidden form field to handle update -->
  <input type="hidden" th:field="*{id}" />

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

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

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

  <button type="submit" class="btn btn-info c

</form>
```

Hidden form field required for updates

This binds to the model attribute

Tells your app which employee to update

Step 3: Process form data to save employee

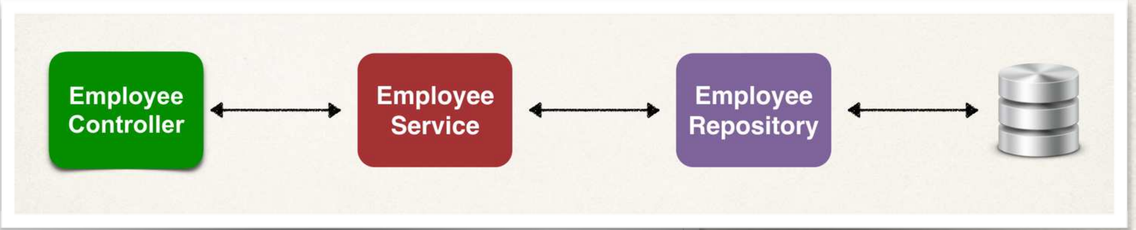
Step 3: Process form data to save employee

- No need for new code … we can reuse our existing code

Step 3: Process form data to save employee

- No need for new code … we can reuse our existing code
- Works the same for add or update :-)

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

   @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";
   }
   …
}
```



Thymeleaf - Delete Employee

Delete Employee - DEMO

Delete Employee - DEMO

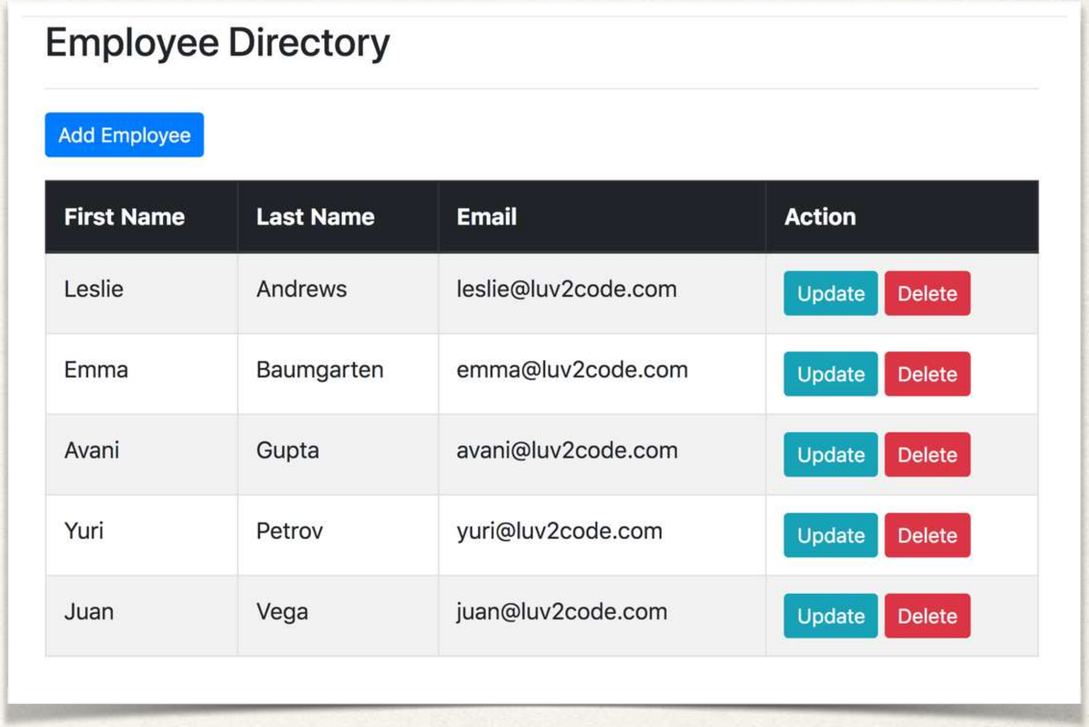
Employee Directory

Add Employee

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete
Emma	Baumgarten	emma@luv2code.com	Update Delete
Avani	Gupta	avani@luv2code.com	Update Delete
Yuri	Petrov	yuri@luv2code.com	Update Delete
Juan	Vega	juan@luv2code.com	Update Delete

Delete Employee

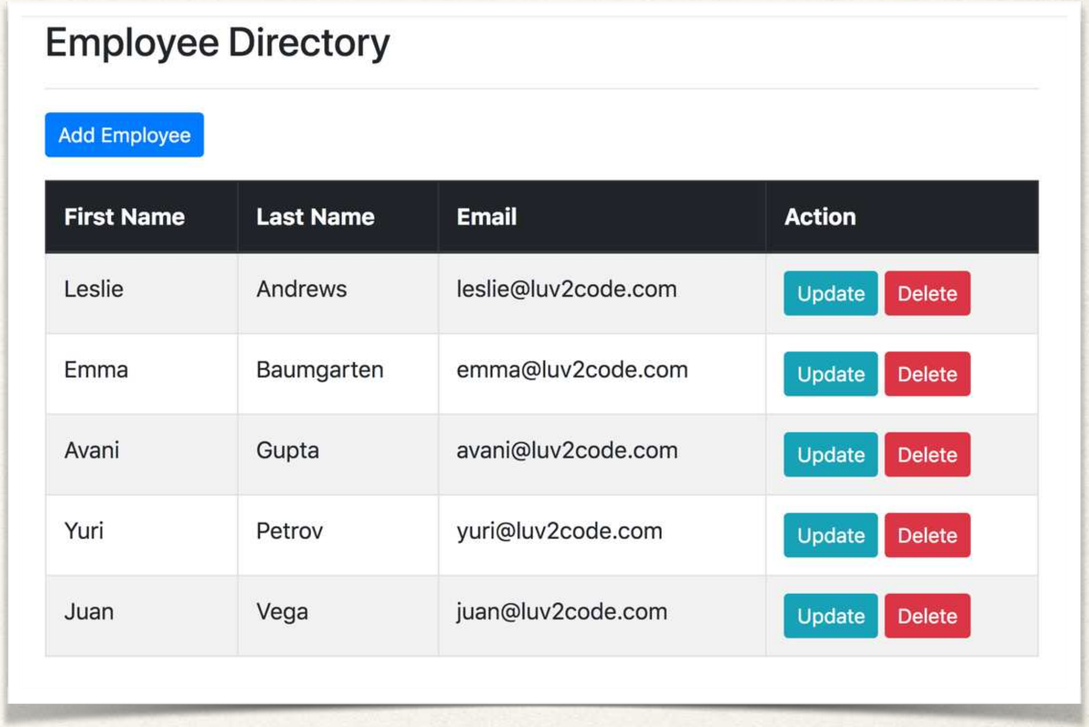
Step-By-Step



Delete Employee

Step-By-Step

1. Add “Delete” button/link on page

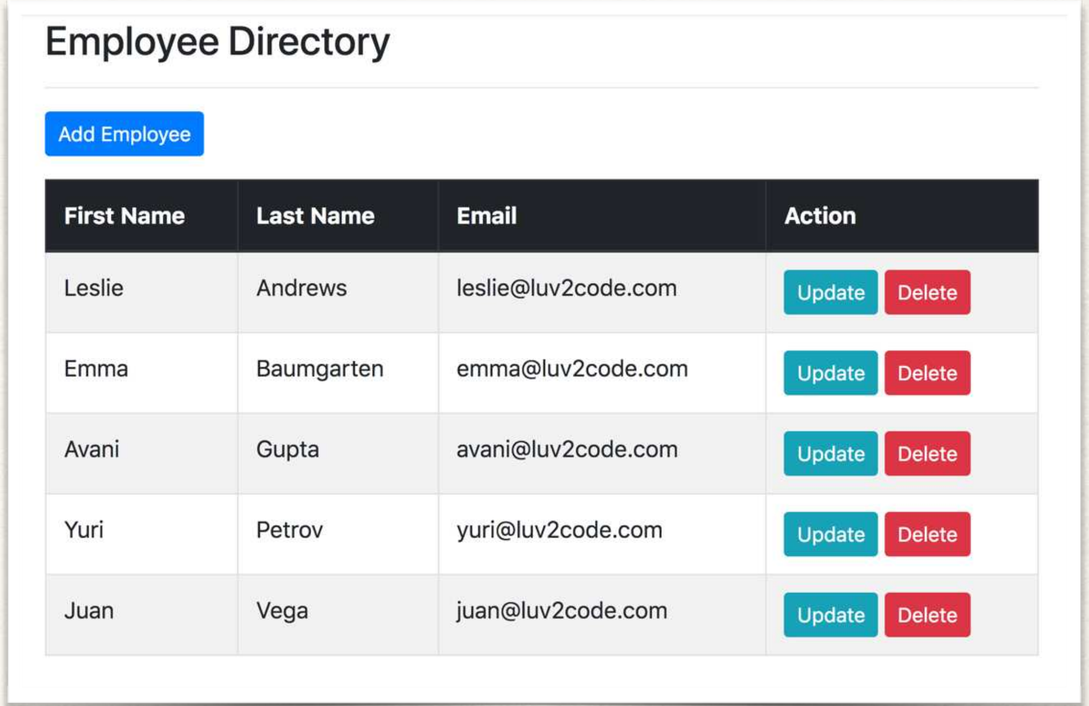


Delete Employee

Step-By-Step

1. Add “Delete” button/link on page

2. Add controller code for “Delete”

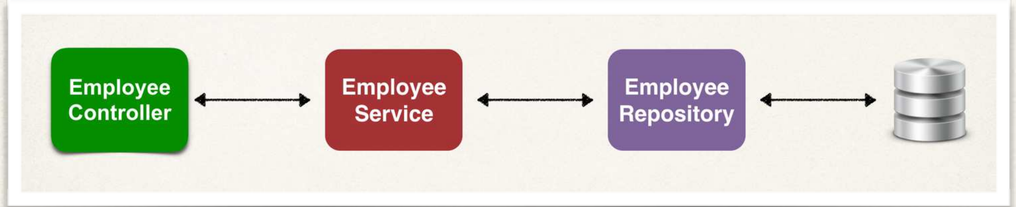


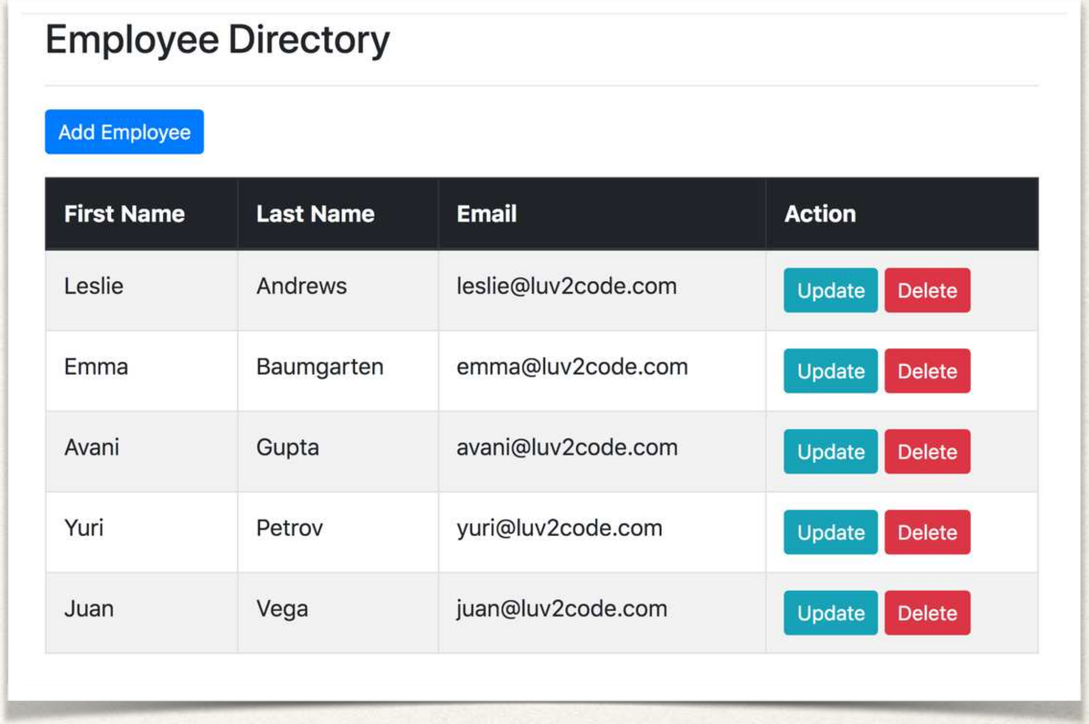
Delete Employee

Step-By-Step

1. Add “Delete” button/link on page

2. Add controller code for “Delete”





Step 1: "Delete" button

Employee Directory

Add Employee

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete
Emma	Baumgarten	emma@luv2code.com	Update Delete
Avani	Gupta	avani@luv2code.com	Update Delete
Yuri	Petrov	yuri@luv2code.com	Update Delete
Juan	Vega	juan@luv2code.com	Update Delete

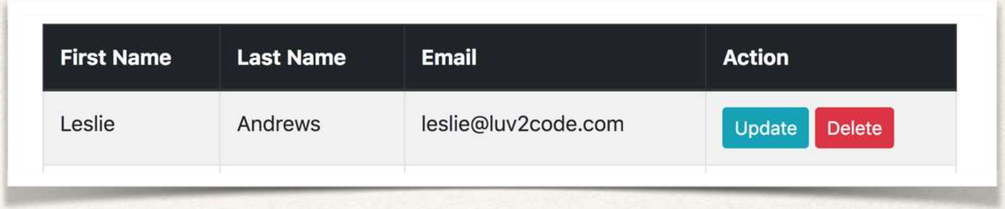
Each row has a **Delete** button/link

- current employee id embedded in link

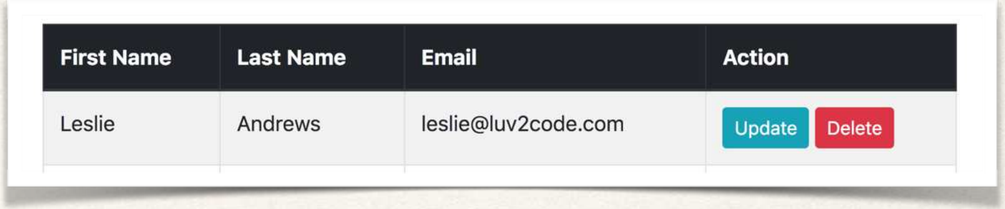
When **clicked**

- prompt user
- will delete the employee from database

Step 1: "Delete" button



Step 1: "Delete" button



- **Delete** button includes employee id

Step 1: "Delete" button

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete

- **Delete** button includes employee id

```
<tr th:each="tempEmployee : ${employees}">
  ...
```

Step 1: "Delete" button

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete

- **Delete** button includes employee id

```
<tr th:each="tempEmployee : ${employees}">
  …
  <td>

    <a th:href="@{/employees/delete(employeeId=${tempEmployee.id})}"
       class="btn btn-danger btn-sm"
       onclick="if (!(confirm('Are you sure you want to delete this employee?'))) return false">
      Delete
    </a>

 </td>

</tr>
```

Step 1: "Delete" button

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete

- **Delete** button includes employee id

Appends to URL

?employeeId=xxx

```
<tr th:each="tempEmployee : ${employees}">
  …
  <td>

    <a th:href="@{/employees/delete(employeeId=${tempEmployee.id})}"
       class="btn btn-danger btn-sm"
       onclick="if (!(confirm('Are you sure you want to delete this employee?'))) return false">
      Delete
    </a>

 </td>

</tr>
```

Step 1: "Delete" button

First Name	Last Name	Email	Action
Leslie	Andrews	leslie@luv2code.com	Update Delete

- **Delete** button includes employee id

```
<tr th:each="tempEmployee : ${employees}">
  …
  <td>

    <a th:href="@{/employees/delete(employeeId=${tempEmployee.id})}"
       class="btn btn-danger btn-sm"
       onclick="if (!(confirm('Are you sure you want to delete this employee?'))) return false">
      Delete
    </a>

 </td>

</tr>
```

Appends to URL

?employeeId=xxx

JavaScript to prompt user before deleting

Step 2: Add controller code for delete

Step 2: Add controller code for delete

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

Step 2: Add controller code for delete

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

  @GetMapping("/delete")
  public String delete(@RequestParam("employeeId") int theId) {
```

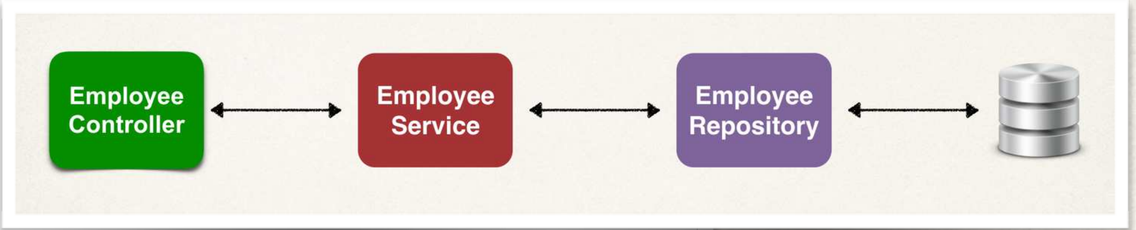
Step 2: Add controller code for delete

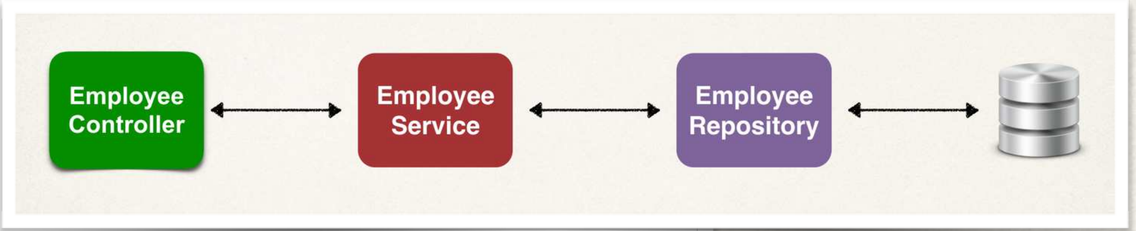
```
@Controller
@RequestMapping("/employees")
public class EmployeeController {
  …

  @GetMapping("/delete")
  public String delete(@RequestParam("employeeId") int theId) {
```

```
<a th:href="@{/employees/delete(employeeId=${tempEmployee.id})}"
```

Step 2: Add controller code for delete

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

  @GetMapping("/delete")
  public String delete(@RequestParam("employeeId") int theId) {

    // delete the employee
    employeeService.deleteById(theId);
```



Step 2: Add controller code for delete

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

  @GetMapping("/delete")
  public String delete(@RequestParam("employeeId") int theId) {

    // delete the employee
    employeeService.deleteById(theId);

    // redirect to /employees/list
    return "redirect:/employees/list";
  }
  …
}
```