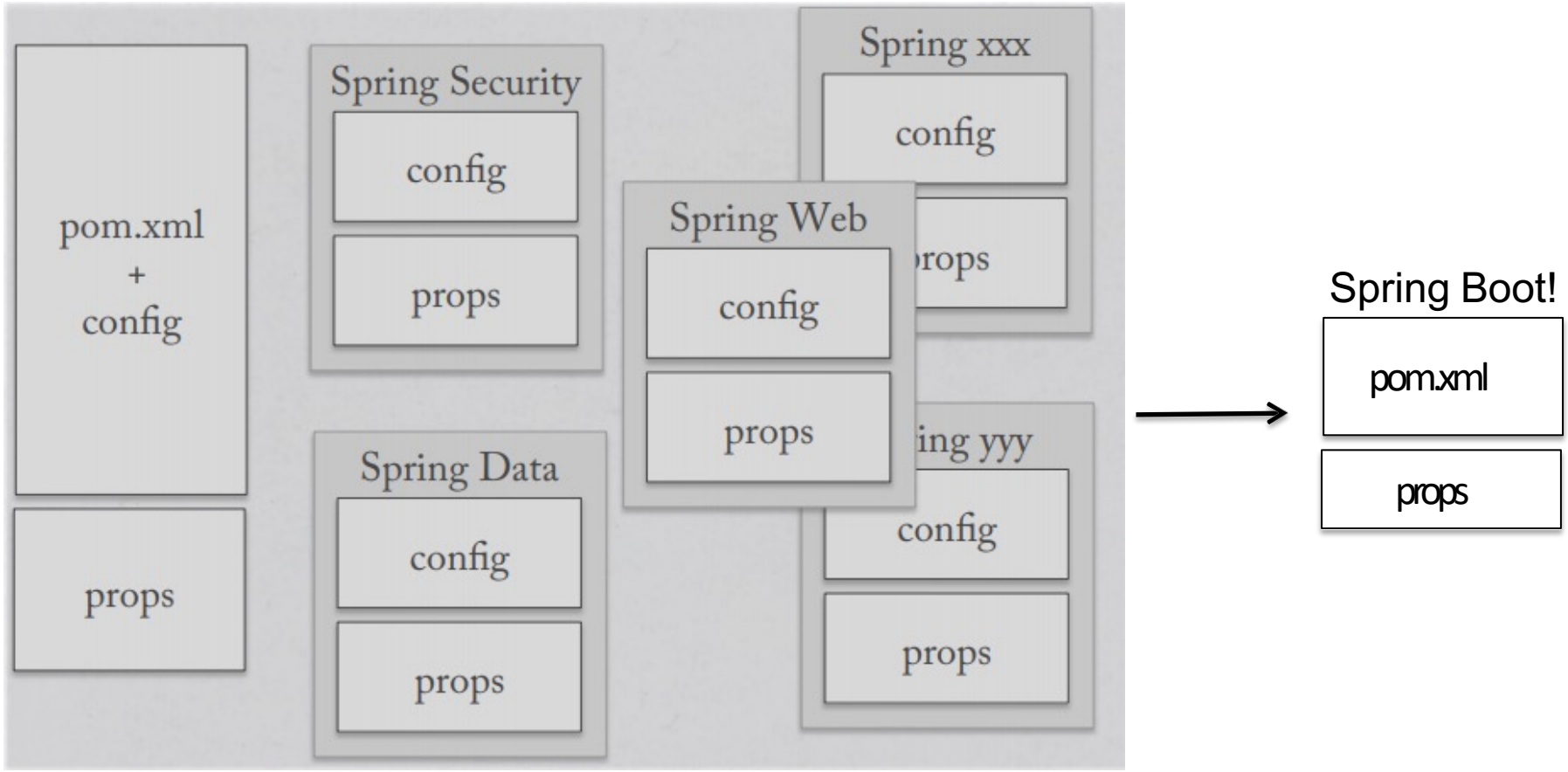
Introduction to Spring Boot & Thymeleaf

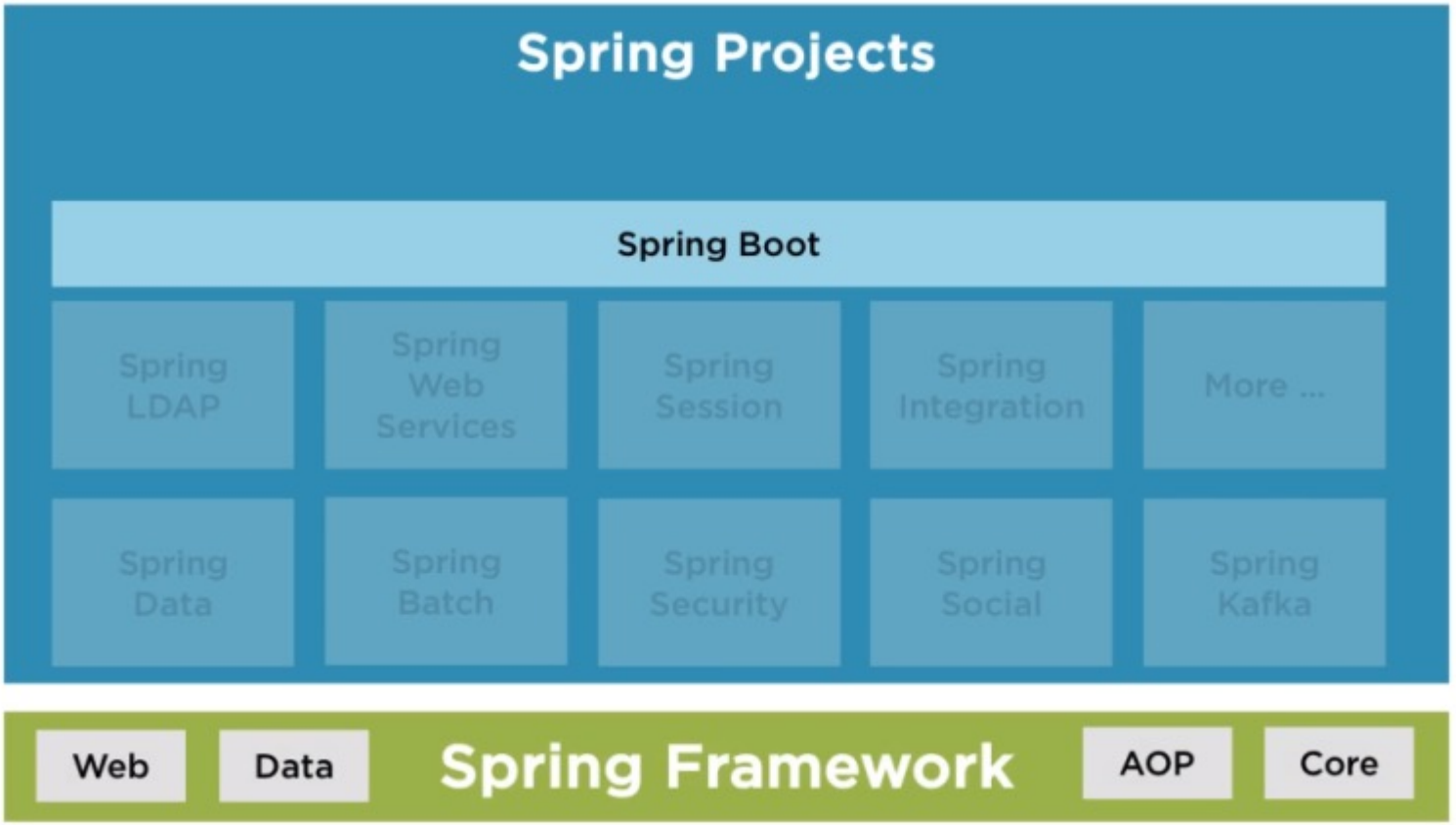
Home of All the Laws of Nature

Spring Boot Emerging



What is Spring Boot?

- Spring Boot is a project built on the top of the Spring framework. It provides a simpler and faster way to set up, configure, and run both simple and web-based applications.

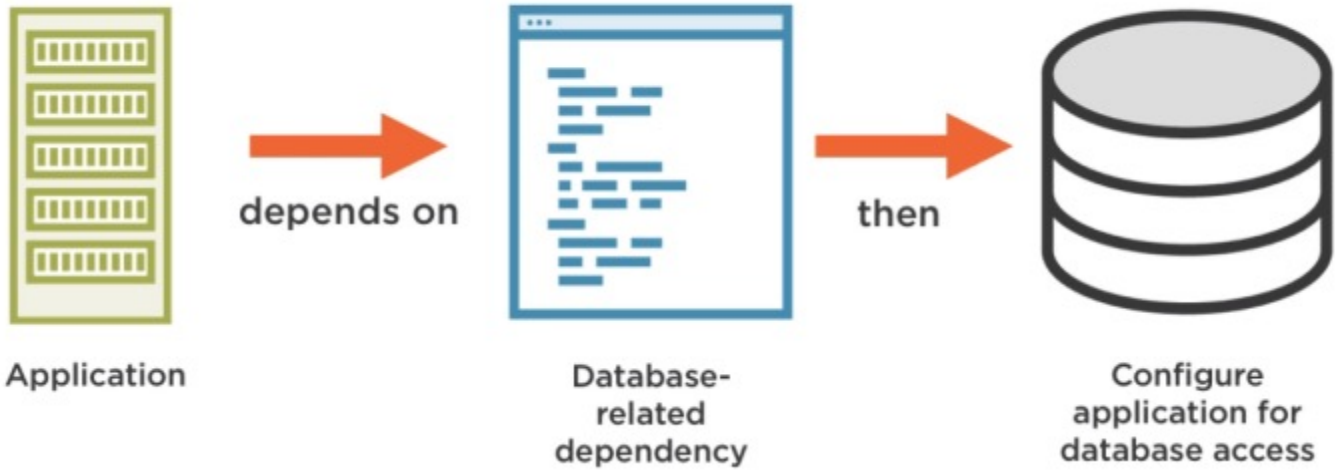


Features

- **Auto-configuration:** It sets up your application based on the surrounding environment, as well as hints what the developers provide.
- **Standalone:** Literally, it's completely standalone. Hence, you don't need to deploy your application to a web server or any special environment. Your only task is to click on the button or give out the run command, and it will start.
- **Opinionated:** This means that the framework chooses how to things for itself. In the most cases, the developers use the same most popular libraries. All that the Spring Boot does is that it loads and configures them in the most standard way. Hence, the developers don't need to spend a lot of time to configure up the same thing over and over again.

Auto Configuration
@EnableAutoConfiguration

- Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that you have added.
- Guess and configure beans that you are likely to need.

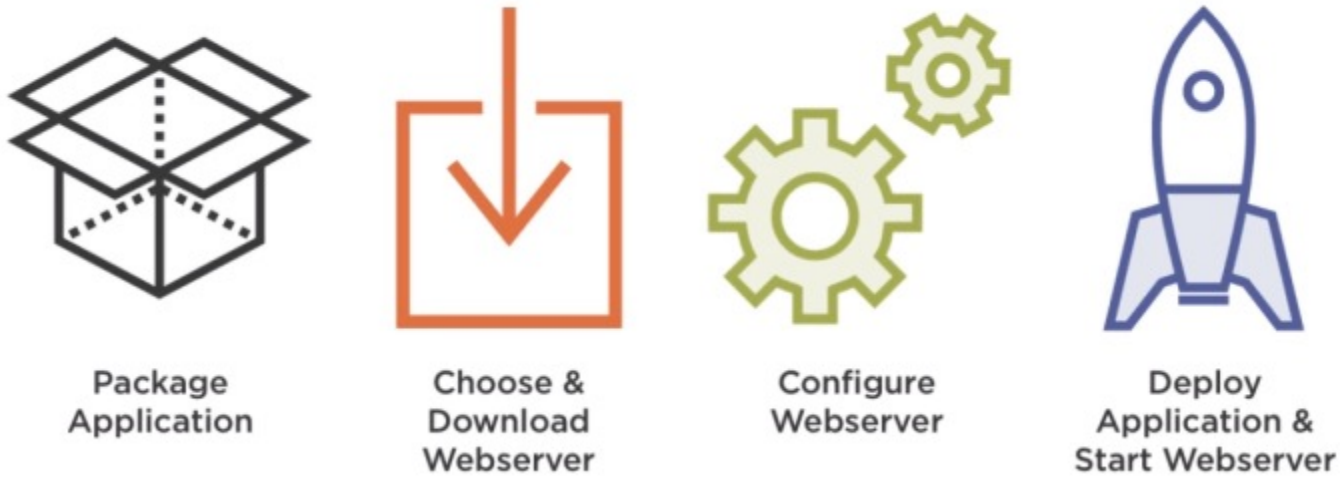


Features

- **Auto-configuration:** It sets up your application based on the surrounding environment, as well as hints what the developers provide.
- **Standalone:** Literally, it's completely standalone. Hence, you don't need to deploy your application to a web server or any special environment. Your only task is to click on the button or give out the run command, and it will start.
- **Opinionated:** This means that the framework chooses how to things for itself. In the most cases, the developers use the same most popular libraries. All that the Spring Boot does is that it loads and configures them in the most standard way. Hence, the developers don't need to spend a lot of time to configure up the same thing over and over again.

The Process of Starting a Java-Based Web Application

- **Package** your application.
- Choose **which type of web servers** you want to use and download it.
- **Configure** that specific web server.
- **Organize the deployment process** and start your web server.



Spring Boot Way

- Install JDK
- Package your application
- Run it with some simple command like
  java -jar my-application.jar
- *java -jar demo-0.0.1-SNAPSHOT.jar --server.port=8083*

Embedded Server

- With Embedded server, you can run a web application as a normal Java application.
  - Embedded server implies that our deployable unit contains the binaries for the server (example, tomcat.jar).

- spring-boot-starter-web: includes a dependency on starter tomcat.

Features

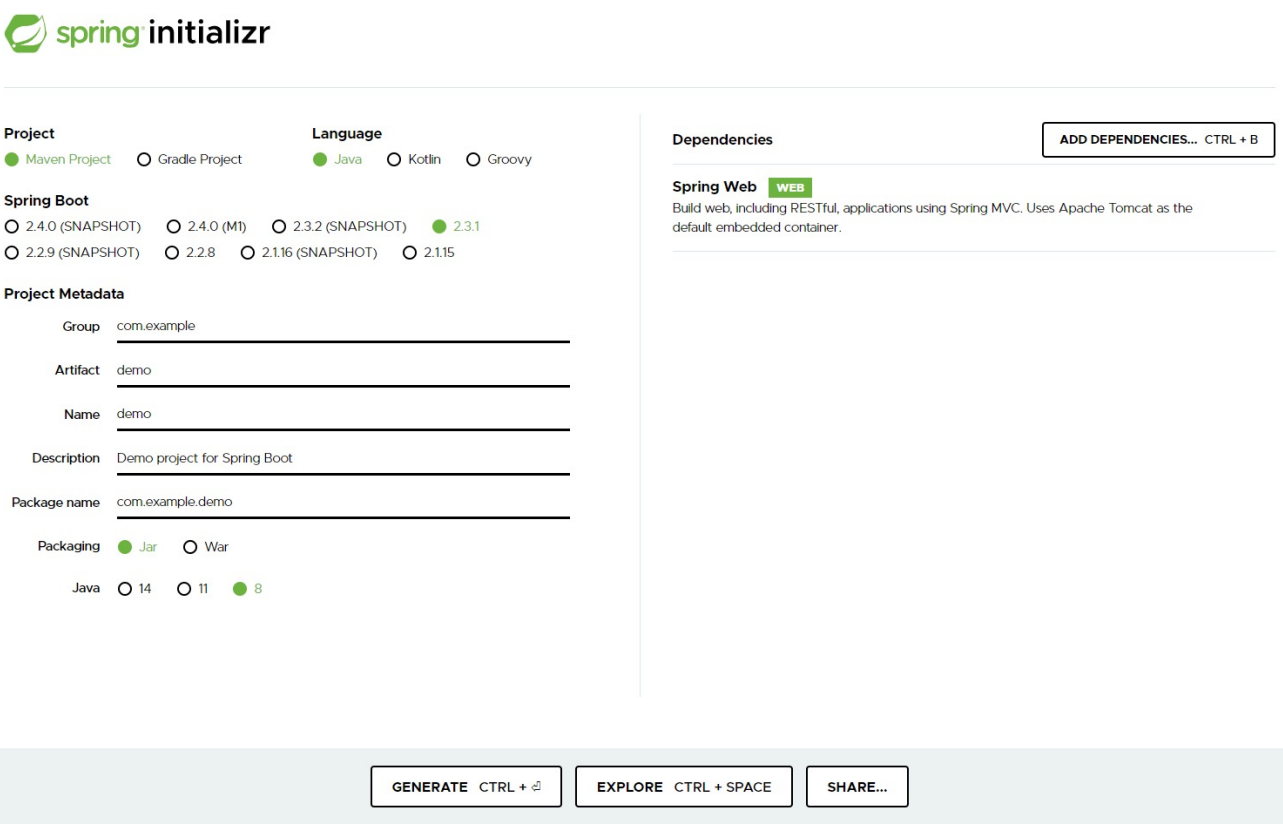
- **Auto-configuration:** It sets up your application based on the surrounding environment, as well as hints what the developers provide.
- **Standalone:** Literally, it's completely standalone. Hence, you don't need to deploy your application to a web server or any special environment. Your only task is to click on the button or give out the run command, and it will start.
- Opinionated: This means that the framework chooses how to configure things for itself. In the most cases, the developers use the same most popular libraries. All that the Spring Boot does is that it loads and configures them in the most standard way. Hence, the developers don't need to spend a lot of time to configure up the same thing over and over again.

pom.xml

- Spring Boot provides a number of "Starters" that let you add jars to your classpath.
- spring-boot-starter-parent
  - inherits dependency management from spring-boot-dependencies
  - Override property e.g. java.version
  - default configuration for plugins e.g.maven-jar-plugin
- Other starters
  - spring-boot-starter-web
  - spring-boot-starter-thymeleaf
  - spring-boot-starter-test

Spring MVC With Spring Boot

1. Create a Spring Starter Project → http://start.spring.io
2. Specify Project Name, artifact Id, etc
3. Select dependency
4. Generate



Main Application Class

```
@SpringBootApplication
public class SpringBootWebJspApplication {

   public static void main(String[] args) {
      SpringApplication.run(SpringBootWebJspApplication.class,
        args);
   }
}
```

`@SpringBootApplication` annotation is equivalent to using

- `@Configuration` : enable Java-based configuration,
- `@EnableAutoConfiguration` and
- `@ComponentScan`: enable component scanning

with their default attributes

Controller & JSP

```
@Controller
public class WelcomeController {
  @GetMapping("/")
  public String index() {
    return "index";
  }
}
```

- index.jsp

```
<body>
  Welcome...
</body>
```

Spring Boot JSPs

- Spring boot has limitations in JSP support
- It works if you use war packaging. A jar will not work because of a hard coded file pattern in Tomcat.
- Add dependency to be able to compile JSP – MUST

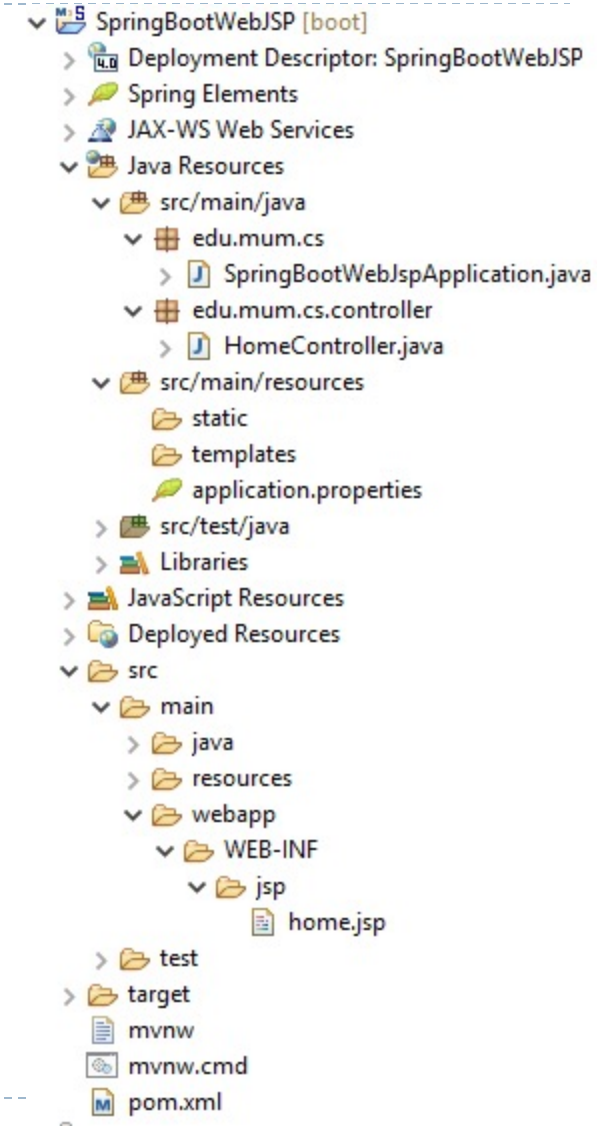
```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
```

- Config view resolver in application.properties

```
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

How to Run

- Run As -> Maven Install – It generates jar/war file under target folder
- Open cmd, java –jar SpringBootWebJSP-0.0.1-SNAPSHOT.war
- MANIFEST.MF
  - Main-Class: org.springframework.boot.loader.WarLauncher
  - Start-Class: edu.mum.cs.SpringBootWebJspApplication



Developer Tools

- Automatic restart
  - Code changes require restart of application
  - Two classloaders in Spring boot
- LiveReload
  - html, javascript, templates, images, stylesheets, …

Main Point

- Spring Boot simplifies dependency management AND application configuration by adhering to a set of rules that apply to all applications.

- Adhering to the fundamental Laws of Nature allows life to be lived simply and easily.

What is Thymeleaf?

- A template engine for Java
  - Open Source
  - First Version 2011 – Currently ***3.0.11***
- It can process six kinds of templates: HTML, XML, TEXT, etc.
- Usually view layer in Spring MVC
- What does it look like?

```
<table>
  <thead>
    <tr>
      <th th:text="#{msgs.headers.name}">Name</th>
      <th th:text="#{msgs.headers.price}">Price</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="prod : ${allProducts}">
      <td th:text="${prod.name}">Oranges</td>
      <td th:text="${#numbers.formatDecimal(prod.price,1,2}">0.99</td>
    </tr>
  </tbody>
</table>
```

Why Thymeleaf?

- The design-development roundtrip issue

- "**Natural Templates:** templates can be documents as valid as the final result, the engine syntax doesn't break the document structure".
- *Comparison of web template engines [Wikipedia]*

```
<form:inputText name="userName" value="${user.name}" />
```

```
<input type="text" name="userName" value="James Carrot" th:value="${user.name}" />
```

The trick

- Use non-standard attributes

```
<ul>
  <li th:each="f : ${fruits}" th:text="${f.name}">Apricot</li>
</ul>
```

- Browsers ignore them
- HTML5 allows custom attributes

```
<ul>
  <li data-th-each="f : ${fruits}" data-th-text="${f.name}">Apricot</li>
</ul>
```

- They don't have influence on the DOM!
- XML namspace
  - which has no influence at all in template processing, but works as an incantation that prevents our IDE from complaining about the lack of a namespace definition for all those th:* attributes.

```
<html xmlns:th="http://www.thymeleaf.org">
```

Thymeleaf Standard Expression syntax

- Simple expressions:
  - Variable Expressions: ${...}
  - Selection Variable Expressions: *{...}
  - Message Expressions: #{...}
  - Link URL Expressions: @{...}
- Conditionals
  - If-then: (if) ? (then)
  - If-then-else: (if) ? (then) : (else)
  - Default(Elvis Operator): (value) ?: (defaultvalue)
  - If – Unless
- Iteration
  - th:each
  - iteration status

Demo: SpringBootWebThymeleaf

Thymeleaf Layout

- layout.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head><title layout:title-pattern="$CONTENT_TITLE - $LAYOUT_TITLE">MUM</title></head>
<body>
    <nav th:replace="fragments/dashboard_nav :: nav">This nav content
    is going to be replaced.</nav>
    <div id="main-content" layout:fragment="main-content">
            <p>This content will be replaced if the page using the layout
            also defines a layout:fragment="content" segment.</p>
    </div>
    <footer th:replace="fragments/dashboard_footer :: footer"> </footer>
</body></html>
```

- dashboard_footer.html

```
<footer class="sticky-footer" th:fragment="footer">
    <div class="container my-auto">
        <div class="copyright text-center my-auto">
            <span>Copyright MUM. All rights reserved.</span>
        </div>
    </div>
</footer>
```

Thymeleaf Layout (cont)

- Title: `layout:title-pattern="$CONTENT_TITLE - $LAYOUT_TITLE"`
- `th:insert` : insert the specified fragment as the body of its host tag.
- `th:replace` : *replaces* its host tag with the specified fragment.
- `th:include` : similar to th:insert, but instead of inserting the fragment it only inserts the *contents* of this fragment.

```
<footer th:fragment="copy">
  &copy; 2011 The Good Thymes Virtual Grocery
</footer>
```

```
<body>

  ...

  <div th:insert="footer :: copy"></div>

  <div th:replace="footer :: copy"></div>

  <div th:include="footer :: copy"></div>

</body>
```

```
<body>

  ...

  <div>
    <footer>
      &copy; 2011 The Good Thymes Virtual Grocery
    </footer>
  </div>

  <footer>
    &copy; 2011 The Good Thymes Virtual Grocery
  </footer>

  <div>
    &copy; 2011 The Good Thymes Virtual Grocery
  </div>

</body>
```

layout:decorate

- *layout:decorate* : tells which layout template to decorate using this content template.
- *layout:fragment* : replace the content of placeholder in the template

```
<!DOCTYPE html>
<html xmlns:layout="http://www.w3.org/1999/xhtml"
   layout:decorate="~{layouts/dashboard_layout}"
       xmlns:th="http://www.thymeleaf.org">

<head><title>Home Page</title></head>
<body>

<div class="col" layout:fragment="main-content">
  <h1>Welcome to CS545, <span th:text="${firstName}"
   th:remove="tag"></span></h1>
</div>

</body>
</html>
```

Main Point

- Thymeleaf is the natural template engine which bridge the gap between designer and developer.

- By doing TM twice a day, we enjoy greater efficiency and accomplish more.