**Primefaces Spring Boot Stack vs. Thymeleaf Spring Boot Stack**

**Coding Paradigm**

Primefaces Spring Boot:

1. Built on JSF framework and has its own MVC controllers.
2. Web pages are mainly xhtml pages that can directly invoke controllers’ public methods, such as getters, setters, and other methods that contain business logics. For example:

#{controller.property} or #{controller.method()}

1. Controllers (JSF Backing Beans) have 4 scopes: Application, Session, View, and Request scopes. So variables in a session life cycle can be accessed through a controller in session scope instead of a session DTO we manually define.
2. Unlike Spring MVC or other Restful service coding pattern, controllers (JSF Backing Beans) and their methods are independent of HTTP request URL mappings. So it can be good or bad. Good: Those controllers are less coupled with HTTP request related API. Bad: Controllers do not follow Restful service design pattern.

Thymeleaf Spring Boot:

1. Respect Spring MVC framework.
2. Respect Restful service coding patterns. Controllers and their methods are tightly associated with HTTP request URL mappings.
3. Thymeleaf is a template engine and supports HTML and XML processing and rendering. Coding web pages of Thymeleaf is similar to JSP, but it is much neater and more readable. It has the following expressions to access objects:

Variable Expressions: ${...}

Selection Variable Expressions: \*{...}

Message Expressions: #{...}

Link URL Expressions: @{...}

Fragment Expressions: ~{...}

1. The controllers in Spring MVC are mainly configured as singletons by default but can be also explicitly configured in other scopes: prototype, request, session, application, and websocket.

**GUI Richness**

Primefaces:

1. More than 100 UI components have been prebuilt. They have built-in CSS and javascripts. There are many Primefaces specific component tags plus JSF standard tags.
2. There are prebuilt themes. Community themes are free to use, but more advanced themes have some cost.
3. Developers can overwrite or customize CSS associated with those GUI components.

Thymeleaf:

1. Similar to JSP, Thymeleaf provides basic tags for constructing pages.
2. Developers may need to apply CSS and JavaScript libraries, such as Metronics to craft rich GUI components.

**Flexibility**

Primefaces:

1. Primefaces or JSF, in general, does not follow Restful web service pattern, so there may be some degree of coupling between backing beans (controllers) and xhtml pages.
2. Front end pages depend on JSF and Primefaces components tags, so there could be a vendor lock.

Thymeleaf Spring Boot:

1. Follow Restful web service pattern. Front end can be highly decoupled from the back end because controllers can send data in XML or Json format to the front end for consumption. HTTP request URL and HTTP Method mappings provide a layer of business abstraction.
2. Other modern UI framework (Angular, React.js, Vue.js…) may be easily applied or swapped if they only communicate with the back end using HTTP Request and Json objects.

**Support**

Primefaces Spring Boot:

There are some useful online tips and projects for integrating Primefaces or JSF with Spring or Spring Boot. JSF is part of Java EE standard. Primefaces provides good documentation and demo code.

Thymeleaf Spring Boot:

Strong support from Spring community. Thymeleaf is well integrated with Spring and Spring Boot, and it is the default front end page solution in Spring ecosystem.

**Development Effort**

Primefaces Spring Boot:

1. Developers may have initial learning curve since the coding paradigm is a little different from Spring MVC.
2. All front-end pages and controller layers may need to be rewritten. All control flows may need to be rewired. However, DAO and service layers may not be heavily modified.
3. Since themes and UI components are prebuilt, developers do not need to spend excessive time on using CSS and JavaScript to craft UI components. In many cases, a UI component can be quickly added to a web page, sort of plug-in-and-play.

Thymeleaf Spring Boot:

1. Since Thymeleaf + Spring Boot highly respect the same Restful and MVC patterns that Spring MVC follows, there is less learning curve.
2. Modifications on DAO, service, and controller layers can be minimized.
3. However, developers need to spend time on learning another CSS and Javascript library (Metronics, Bootstrap…) in order to craft UI components, so there is a learning curve.