**SPRING VALIDATION**

In Spring, validation is a mechanism used to ensure that the data provided to your application adheres to certain rules or constraints.

This can prevent invalid or harmful data from being processed by your business logic.

Spring offers several ways to perform validation, both through built-in tools and custom approaches.

Spring provides several common validation annotations, which can be applied to Java beans (e.g., model classes).

These annotations are processed during runtime, and validation errors are thrown if any constraints are violated.

Common validation annotations:

- ***@NotNull:*** Ensures the field is not null.

- ***@NotEmpty:*** Ensures the field is not `null` or empty works for strings, collections, etc.

- ***@NotBlank:*** Ensures the field is not `null` or blank (only for strings, allows spaces).

- ***@Size(min=, max=):*** Validates that the length or size of the element is within a certain range.

- ***@Min(value):*** Validates that the field's value is not less than the specified minimum.

- ***@Max(value):*** Validates that the field's value is not greater than the specified maximum.

- ***@Pattern(regexp):*** Ensures the field matches a regular expression pattern.

- ***@Email:*** Ensures the field contains a valid email address.

**Error Handling:**

After validation, Spring provides tools to handle errors and return meaningful responses.

In a REST API, you can return an appropriate HTTP response like 400 Bad Request along with a detailed error message.

In the case of RESTful APIs, you can use @RestControllerAdvice to handle validation exceptions globally.

**SPRING SCOPES:**

In Spring, scopes define the lifecycle and visibility of beans.

When you declare a bean in Spring, you can define its scope, which determines how long the bean instance will live and how it will be shared.

***Singleton (default scope):***

- A single instance of the bean is created per Spring container, and the same instance is shared across the entire application.

- When you want a single, shared instance of the bean across the entire application.

- @Scope("singleton")

***Prototype:***

- A new bean instance is created every time it is requested from the Spring container.

- When you need multiple instances of the bean, for example, in scenarios where each user request or session should have a unique instance.

- @Scope("prototype")

***Request:***

- A new bean instance is created for each HTTP request. It is used in web applications and the bean is available only during the request's lifecycle.

- Useful for web applications where you need to store request-specific data.

- @Scope("request")

***Session:***

- A new bean instance is created for each HTTP session. The bean is stored until the session expires.

- Useful for storing session-specific data, such as user information during a session.

- @Scope("session")

***Application:***

- A single instance of the bean is created for the entire ServletContext. This scope is essentially a singleton for the web application context.

- Used when you need to share data across the entire application but only within a web application context.

- @Scope("application")

***WebSocket:***

- A new bean instance is created for each WebSocket session.

- Used in WebSocket-based applications where each WebSocket connection requires a separate instance.

- @Scope("websocket")

Singleton and Prototype are available in all Spring-based applications, including non-web apps.

Request, Session, Application, and WebSocket are typically used in web applications.

**WEATHER APPLICATION WITH SCOPE AND VALIDATION**

[SpringWeather/ at ScopeAndValidation · LalithaRavuri/SpringWeather](https://github.com/LalithaRavuri/SpringWeather/tree/ScopeAndValidation)