Spring Boot with Spring MVC refers to leveraging the Model-View-Controller (MVC) architectural pattern within a Spring Boot application for building web applications and RESTful APIs.

Key Concepts:

* **Spring MVC:**

This is a framework within the broader Spring ecosystem that implements the MVC design pattern.

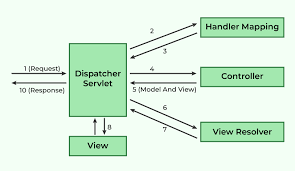
* + **Model:** Represents the application's data and business logic.
  + **View:** Responsible for presenting the data from the model to the user (e.g., HTML, JSP, Thymeleaf).
  + **Controller:** Handles incoming requests, interacts with the model, and selects the appropriate view to render the response.
  + Annotations like @Controller and @RestController are used to define controller classes and handle HTTP requests.
* **Spring Boot:**

This is a framework that simplifies the development of Spring-based applications.

* + **Auto-configuration:** Spring Boot automatically configures many aspects of a Spring application based on the dependencies present in the classpath, significantly reducing boilerplate configuration.
  + **Starter dependencies:** Provides convenient "starters" (e.g., spring-boot-starter-web) that bundle common dependencies, including Spring MVC, embedded servlet containers (like Tomcat or Jetty), and other necessary components.
  + **Embedded Servers:** Allows running Spring applications as standalone executable JARs with embedded servlet containers, eliminating the need for external server deployments.

How they work together:

When you create a Spring Boot application and include the spring-boot-starter-web dependency, Spring Boot automatically configures Spring MVC for you. This means:



**The Spring Boot MVC flow describes how a web request is processed within a Spring Boot application following the Model-View-Controller (MVC) architectural pattern.**

* **Request Initiation:**

**A client (e.g., web browser) sends an HTTP request to the Spring Boot application's URL.**

* **DispatcherServlet:**

**The DispatcherServlet acts as the front controller in Spring MVC. It intercepts all incoming requests and is responsible for routing them to the appropriate handler.**

* **Handler Mapping:**

**The DispatcherServlet consults HandlerMapping to determine which controller and method are responsible for handling the incoming request based on the URL and other request parameters.**

* **Controller Execution:**

**The DispatcherServlet dispatches the request to the identified controller. The controller processes the request, potentially interacting with service layers and data access layers (Model) to perform business logic and retrieve or update data.** **. In Spring Boot, controllers are typically annotated with @Controller or @RestController (for RESTful services).**

* **Model and View Name:**

**After processing, the controller returns a ModelAndView object or simply a logical view name, along with any data (Model) that needs to be displayed in the view.**

* **View Resolution:**

**The DispatcherServlet then consults a ViewResolver to translate the logical view name into a concrete view implementation (e.g., a Thymeleaf template, JSP page, or HTML file).**

* **View Rendering:**

**The resolved view, along with the data from the Model, is used to render the final response, typically in the form of HTML.**

* **Response to Client:**

**The DispatcherServlet sends the rendered response back to the client, completing the request-response cycle.**

This integration allows developers to quickly build web applications using the familiar MVC pattern, while benefiting from Spring Boot's conventions, auto-configuration, and ease of deployment.

Key advantages of this architecture in Spring Boot:

* **Separation of Concerns:**

Promotes modularity and maintainability by separating data, presentation, and control logic.

* **Testability:**

Each component can be tested independently.

* **Flexibility:**

Allows for easy integration with various view technologies and data sources.

* **Rapid Development:**

Spring Boot's conventions and auto-configuration significantly reduce boilerplate code, enabling faster development of MVC applications.