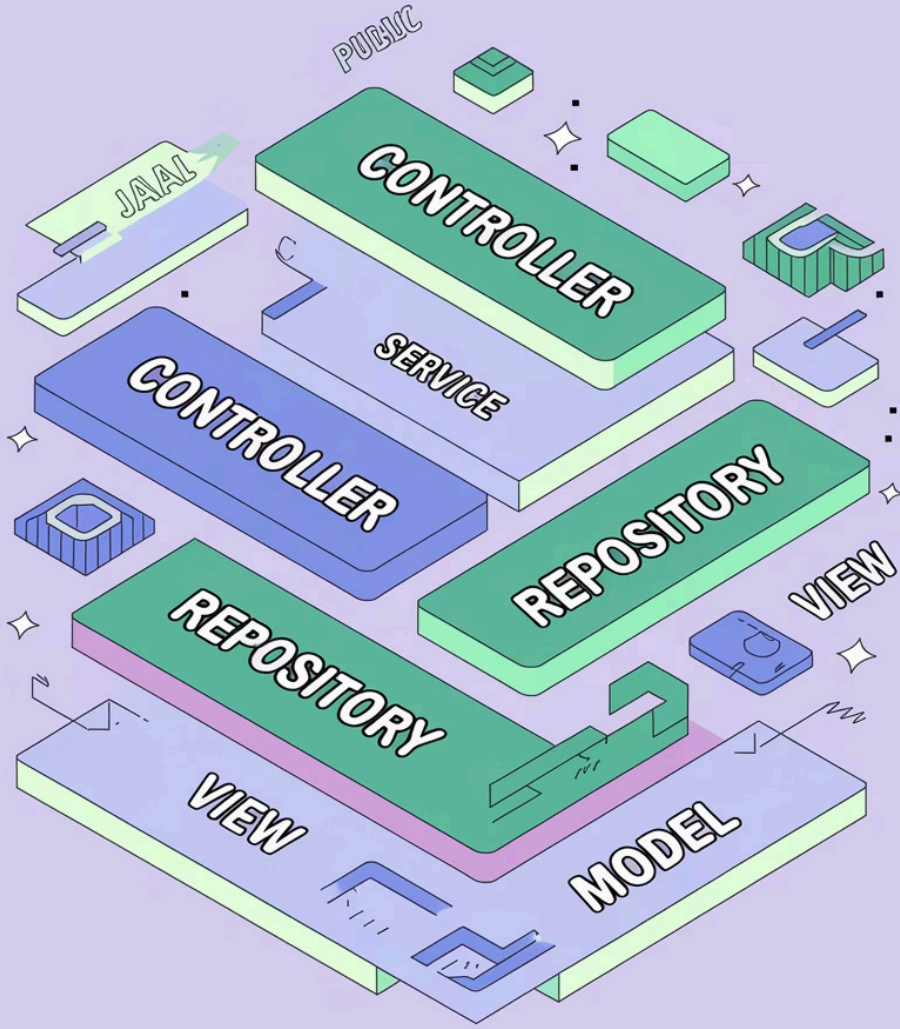
Spring MVC: Core Classes and Interfaces

Released in 2004, Spring MVC now powers 31% of enterprise Java web applications. The current version is 6.1.2 as of April 2025.

The framework offers a robust request processing pipeline with 7 key components. It builds on the Java Servlet API with enhanced capabilities.

by Saratha Natarajan



Spring MVC Architecture

Model

Encapsulates application data and state

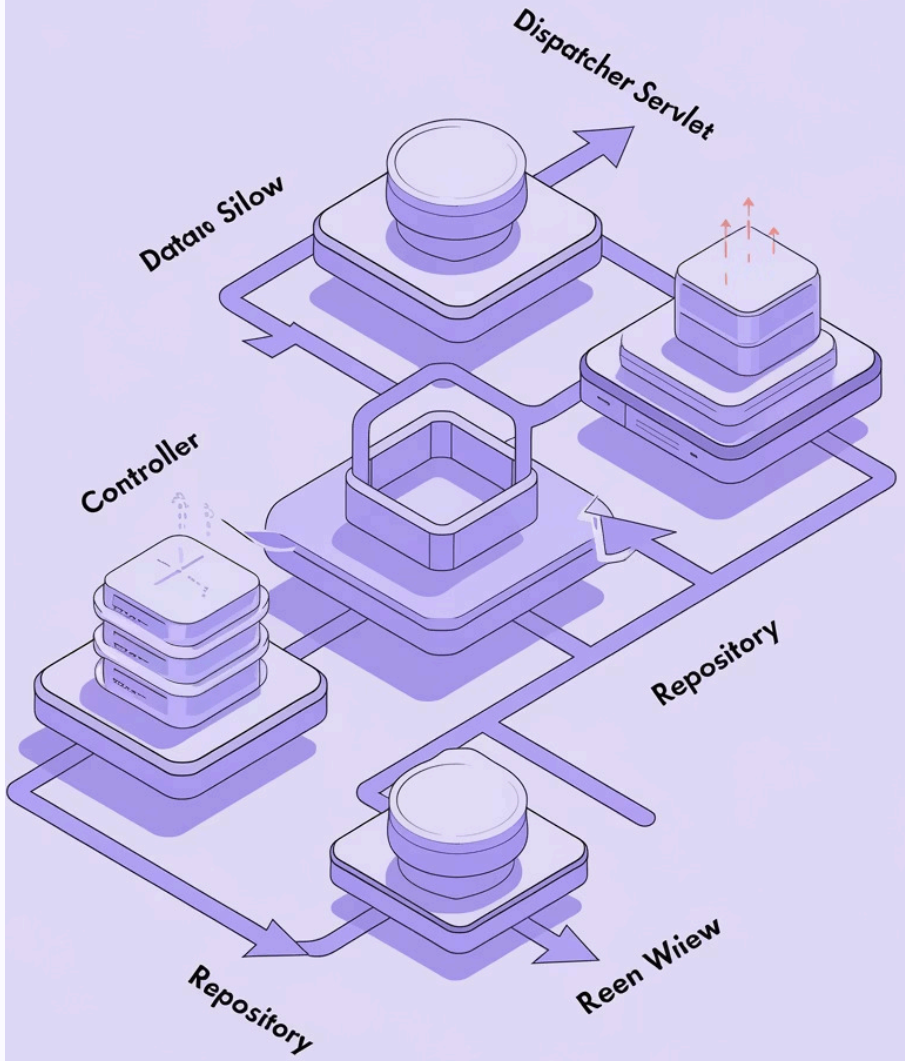
Controller

Processes incoming requests and business logic

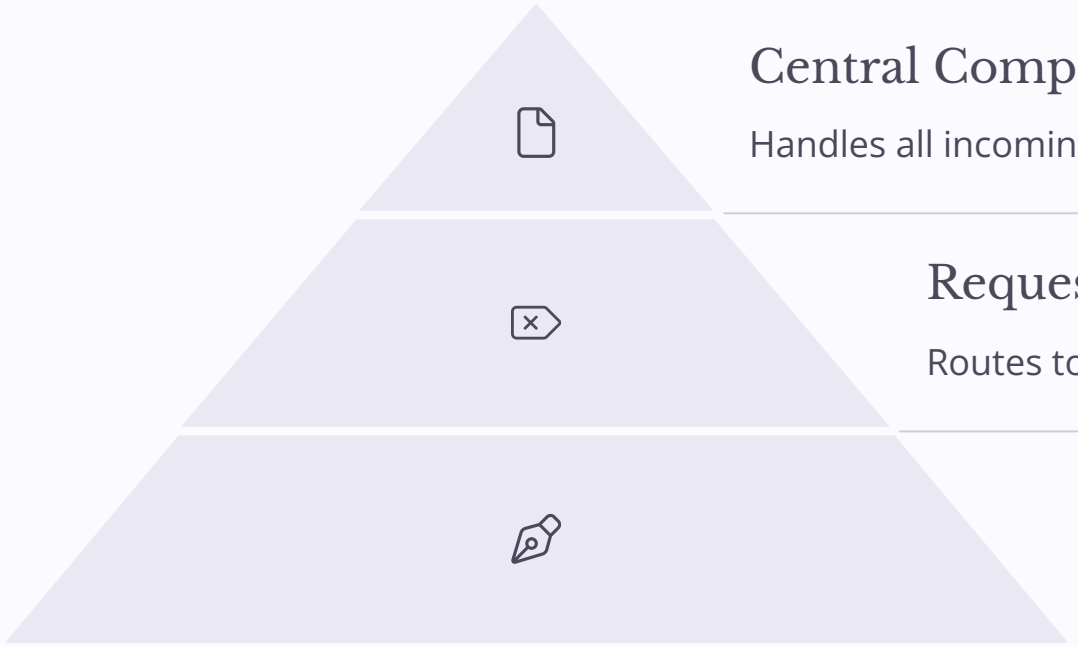
View

Renders data to clients in various formats

The architecture follows the Model-View-Controller design pattern. A central DispatcherServlet handles all HTTP requests.



DispatcherServlet: The Front Controller



Central Component

Handles all incoming HTTP requests

Request Delegation

Routes to specialized components

Lifecycle Management

Controls the complete request flow

DispatcherServlet extends the HttpServlet class. It can be configured via web.xml or WebApplicationInitializer.

HandlerMapping Interface

RequestMappingHandlerMapping

Processes @RequestMapping annotations to map requests to handlers

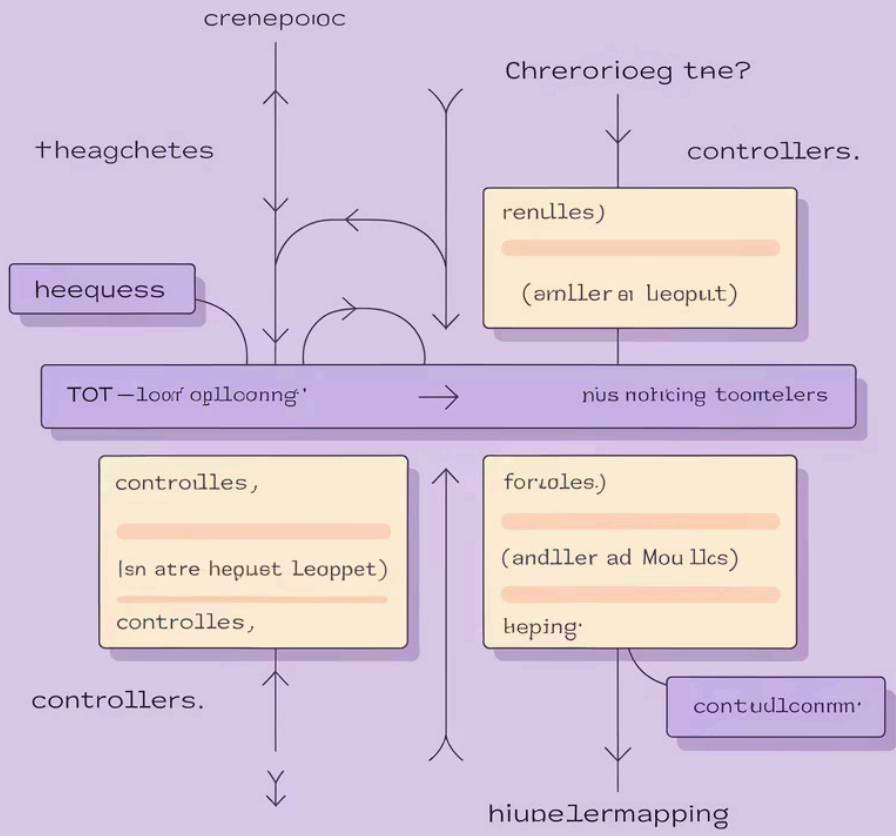
SimpleUrlHandlerMapping

Matches URL patterns directly to controller instances

BeanNameUrlHandlerMapping

Uses Spring bean names as URL paths

HandlerMapping maps incoming requests to appropriate handlers. Most Spring applications use 2-3 different mapping types.



Controller Interface & Annotations

@Controller

Marks classes as web request handlers

@RequestMapping

Defines URL paths and HTTP methods

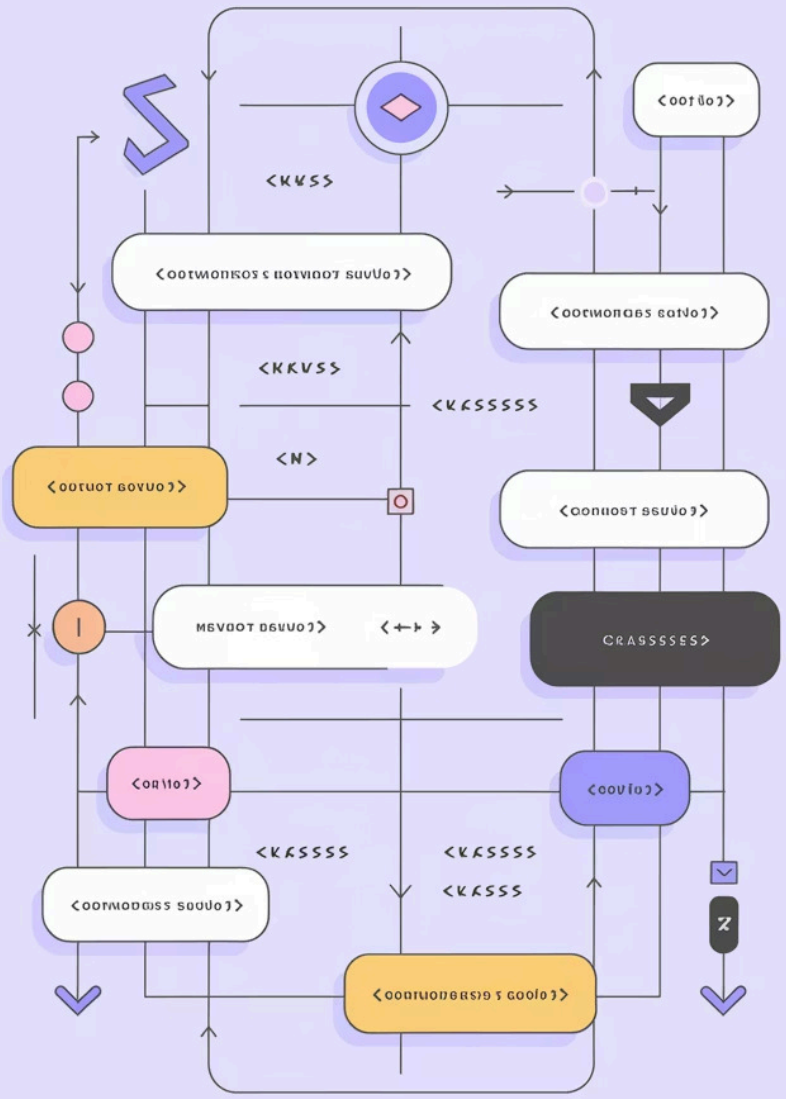
HTTP Method Annotations

@GetMapping, @PostMapping, @PutMapping, @DeleteMapping

Parameter Binding

@RequestParam, @PathVariable bind HTTP data to method parameters

Controllers handle requests and return ModelAndView objects. They can also use @ResponseBody for direct data returns.



HandlerAdapter Interface

Request Processing

Receives HTTP request from DispatcherServlet

Parameter Conversion

Transforms HTTP data to Java objects

Method Execution

Invokes controller method with appropriate parameters

Response Handling

Processes return value into HTTP response

HandlerAdapter executes controller methods with appropriate parameters. RequestMappingHandlerAdapter is most commonly used.

ModelAndView Class

Model Component

- Contains application data
- Key-value attribute pairs
- Accessible in view templates

View Component

- String view name
- Resolved by ViewResolver
- Determines presentation format

ModelAndView combines Model data and View references. It can be returned directly from controller methods.

ViewResolver Interface

InternalResourceViewResolver

Resolves to JSP pages

ThymeleafViewResolver

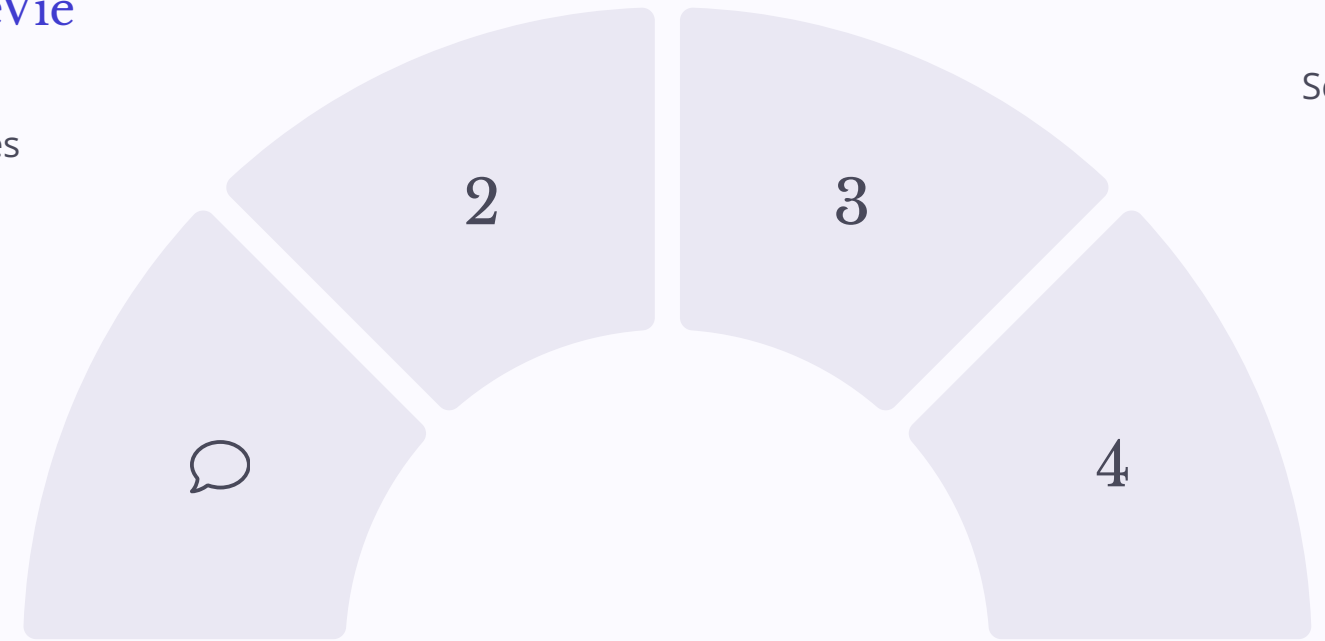
Resolves to Thymeleaf templates

FreeMarkerViewResolver

Resolves to FreeMarker templates

ContentNegotiatingViewResolver

Selects views based on content type

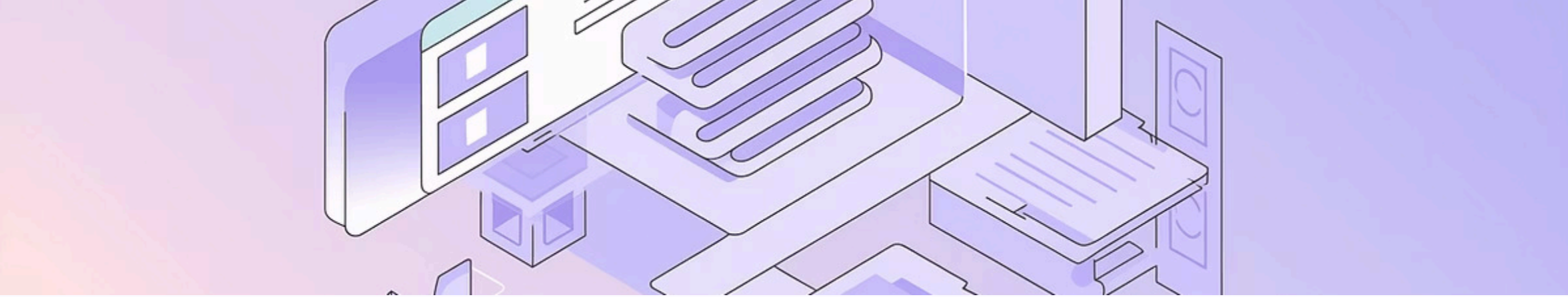


ViewResolver translates logical view names to actual View implementations. Multiple resolvers can be configured in a single application.

View Interface & Implementations

1	**JstlView** JSP pages with JSTL support
2	**ThymeleafView** Thymeleaf template engine
	**MappingJackson2JsonView** JSON responses via Jackson

The View interface renders model data to the client. It typically handles 10-30% of request processing time.



MultipartResolver Interface

File Uploads

Handles multipart form data containing files

StandardServletMultipartResolver

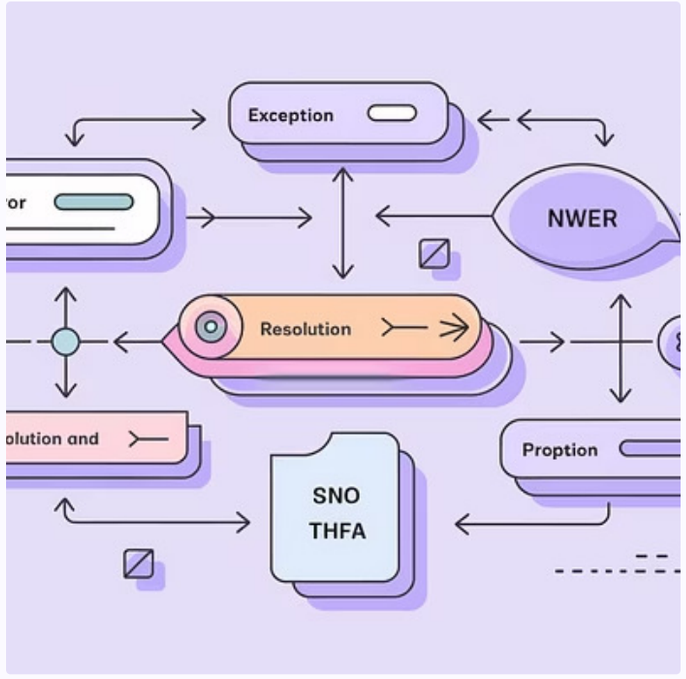
Uses Servlet 3.0+ capabilities

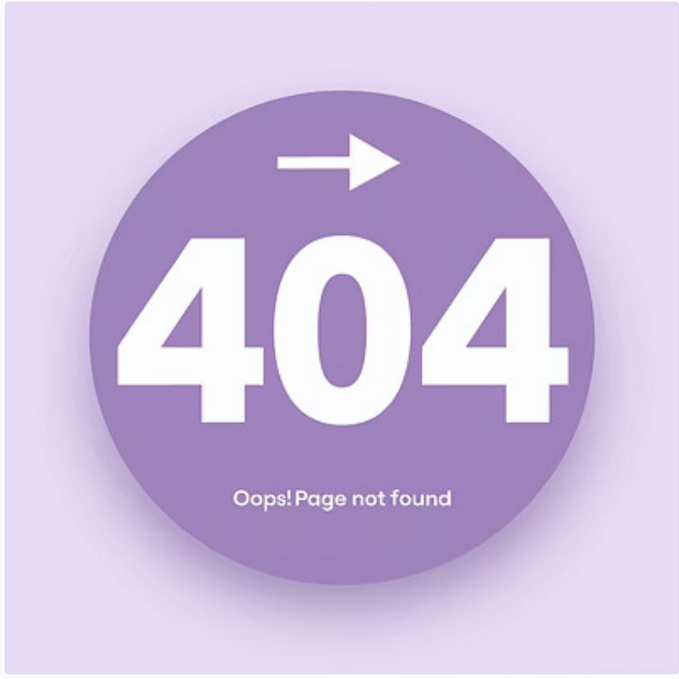
CommonsMultipartResolver

Uses Apache Commons FileUpload library

MultipartResolver handles file uploads in multipart requests. It has a default max file size of 1MB and is used in 62% of Spring MVC applications.

HandlerExceptionResolver Interface





ExceptionHandlerExceptionResolver

Processes @ExceptionHandler annotations

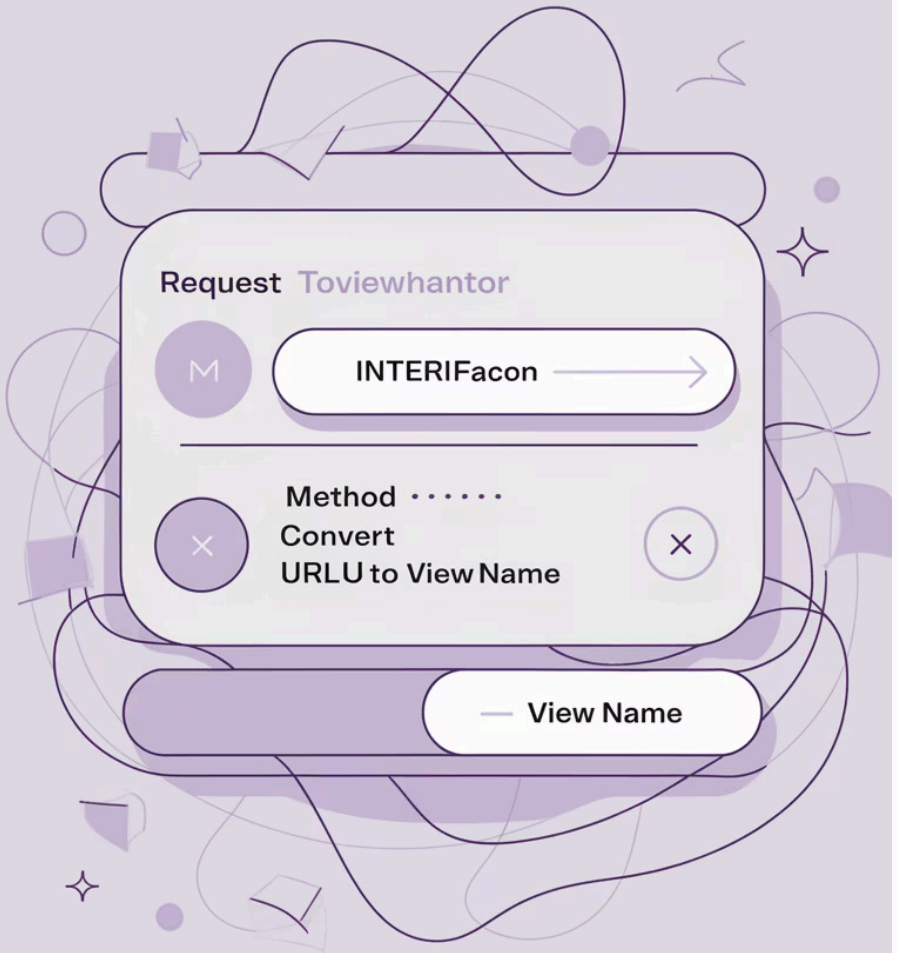
ResponseStatusExceptionResolver

Handles @ResponseStatus annotations

DefaultHandlerExceptionResolver

Manages standard Spring MVC exceptions

HandlerExceptionResolver processes exceptions during request handling. It provides centralized exception management across the application.



RequestToViewNameTranslator Interface

Request Processing

Controller returns null or void without view specification

URL Translation

URL path converted to logical view name

View Resolution

Generated view name passed to ViewResolver

DefaultRequestToViewNameTranslator is the standard implementation. It converts URL paths to view names when controllers return null.

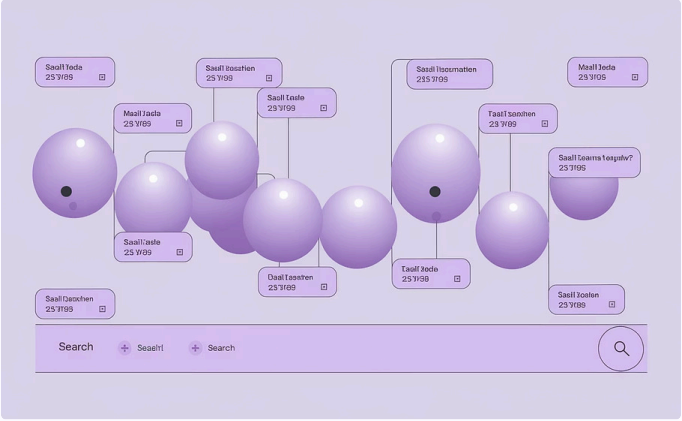
LocaleResolver Interface

AcceptHeaderLocaleResolver

Uses browser's Accept-Language header to determine locale

CookieLocaleResolver

Stores locale preference in browser cookies



SessionLocaleResolver

Maintains locale selection within user's session

LocaleResolver determines the locale for internationalization. Fixed and default locales are supported with configuration.

ThemeResolver Interface

8%

Usage Rate

Percentage of Spring MVC applications using themes

3

Implementation Types

Common ThemeResolver implementations

100+

Theme Resources

Typical number of assets in a complex theme

ThemeResolver determines themes for UI customization. FixedThemeResolver provides a single theme. Cookie and session-based resolvers store user preferences.

Spring MVC Request Processing Flow

Request Reception

DispatcherServlet receives HTTP request

View Resolution

ViewResolver translates view name to View

Handler Mapping

HandlerMapping identifies appropriate controller

Handler Execution

HandlerAdapter executes controller method

Model Processing

ModelAndView returned with data and view name

The Spring MVC request flow follows a predictable pattern. Each component has a specific responsibility in the processing pipeline.

Best Practices & Resources

Favor annotation-based configuration over XML	Reduces boilerplate and improves readability
Keep controllers lightweight	Move business logic to service layer
Follow RESTful design principles	URI structure and HTTP methods
Use Spring Boot	Auto-configuration reduces setup time

Key resources include docs.spring.io and baeldung.com. GitHub samples provide practical implementation examples for common patterns.