Spring MVC

[redacted]

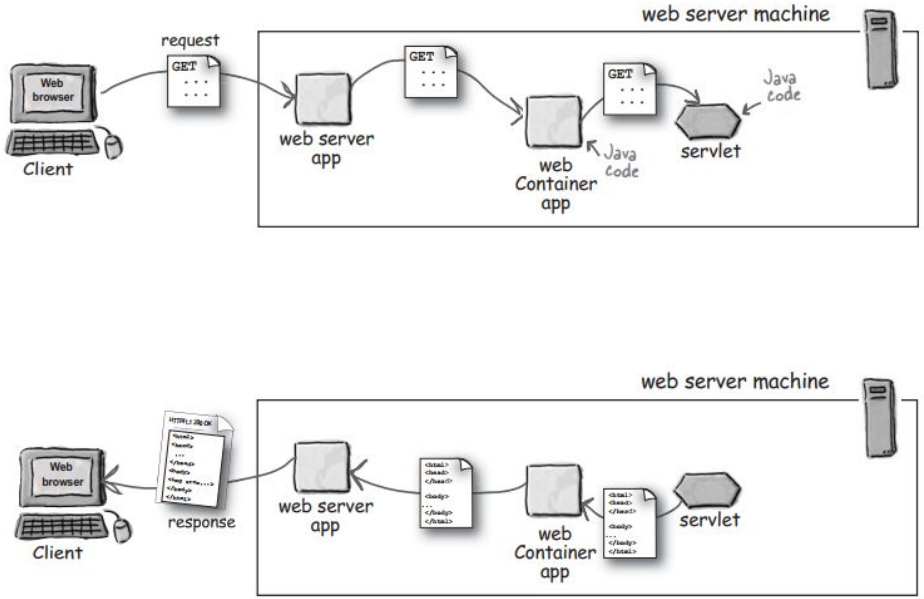
Content

1. HTTP Servlet (Review)
2. What is Spring MVC?
3. MVC Architecture
4. Request Processing Workflow in Spring MVC
5. Spring Web Application Context
6. Spring MVC Configuration
7. Important Annotations
8. Summary


HTTP Servlet (Review)
spring

HTTP Servlets

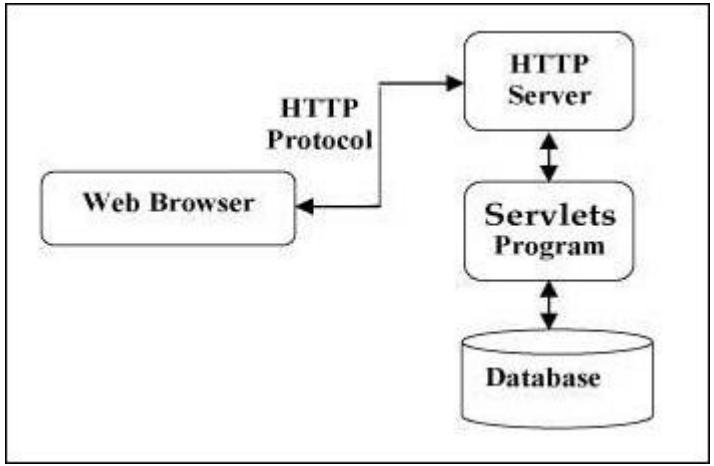
- Are Java classes that run on Web Servers to dynamically process HTTP requests and construct HTTP responses.
- Deployed inside a Servlet Container which run on a Web Server.
- Tomcat is a popular Servlet Container.



spring

Servlets Architecture

- Servlets act as a middle layer between a Web browser and databases or applications on the Web Server.
- The following diagram shows the position of Servlets in a Web Application



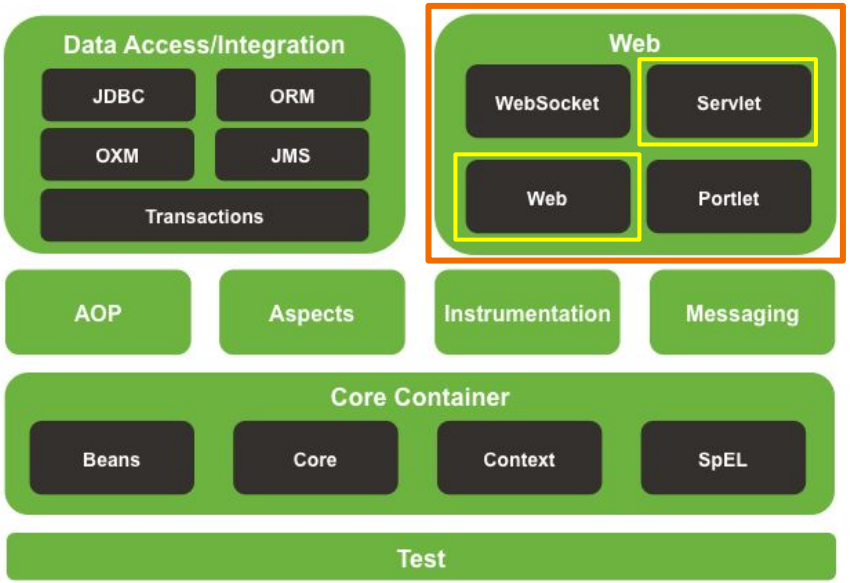
spring


What is Spring MVC ?
spring

What is Spring MVC?

Modules of Spring Framework on the **Web** layer:

- **Web** module provides basic web-oriented integration features and the initialization of the IoC container using servlet listeners and a web application context.
- **Servlet** module contains Spring MVC implementation for web applications.



spring

What is Spring MVC ? (cont.)

- A web framework built on the Servlet API.
- Provides Model-View-Controller (MVC) architecture and ready components that can be used to develop flexible and loosely coupled web applications.
- Request-driven, designed around a central Servlet that dispatches requests to controllers - the **DispatcherServlet**.


MVC
spring

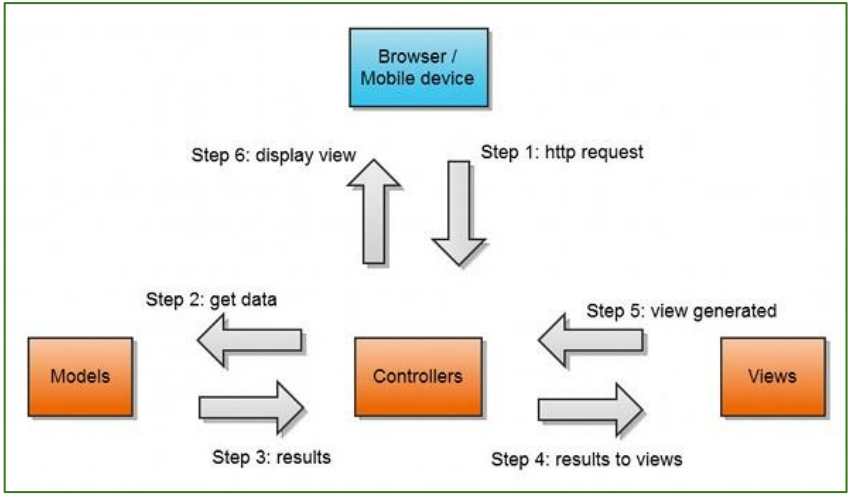
MVC (Model-View-Controller)

Introduced by **Trygve Reenskaug** at **Xerox Parc** in **1979**.

spring

MVC (Model-View-Controller)

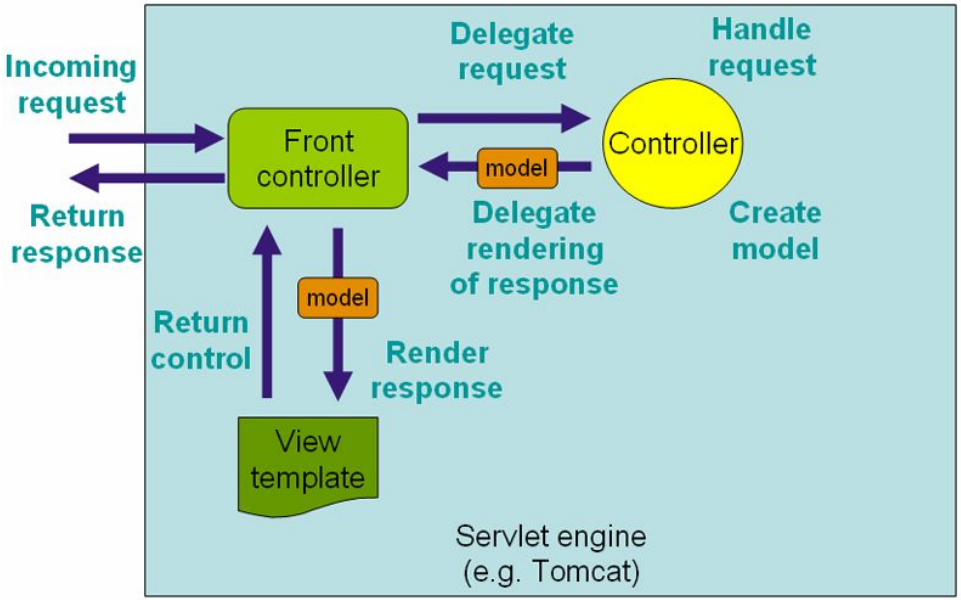
- An architectural pattern commonly used for developing user interfaces.
- An application is divided into 3 interconnected parts:
  - **Model** - Responsible for managing data of the application.
  - **View** - Responsible for displaying the model data to user.
  - **Controller** - Responsible for processing user requests and building an appropriate model and passes it to the view for rendering.



spring


Request Processing Workflow
in Spring MVC
spring

Request Processing Workflow (High Level)

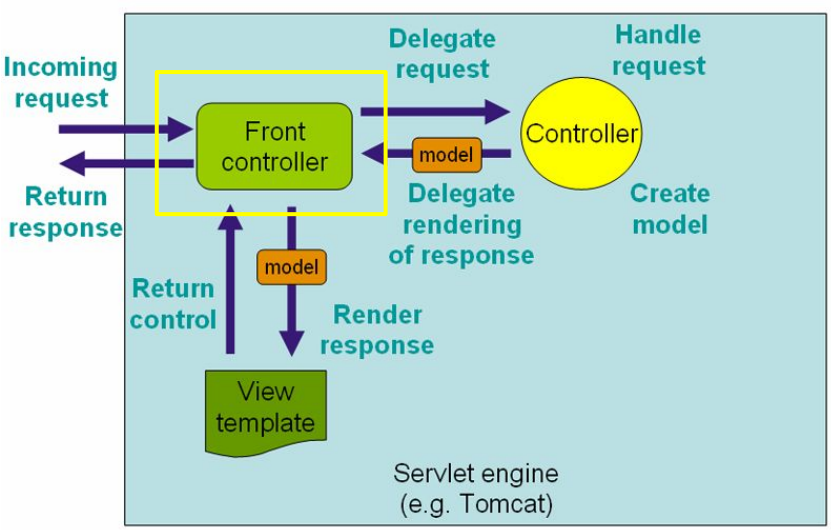


The request processing workflow in Spring Web MVC (high level)

spring

DispatcherServlet

- Spring MVC is designed around a central servlet named **DispatcherServlet**.
- DispatcherServlet acts as a central entry point to the Spring MVC application.
- Every request is handled by DispatcherServlet.
- DispatcherServlet is an expression of the **Front Controller** pattern.

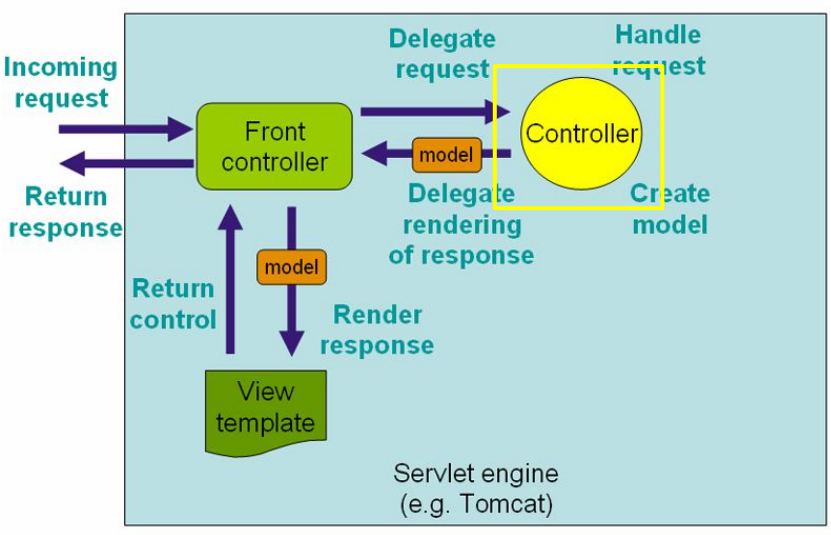


The request processing workflow in Spring Web MVC (high level)

spring

Controllers

- The Front Controller's job is to determine a suitable **handler** capable of performing the actual processing.
- **Handlers** are Spring MVC **Controllers**.
- The selected Controller interacts with the service layer; the relevant data are collected in a **model**.
- When the Controller has finished processing, the Front Controller determines which **view** to render.

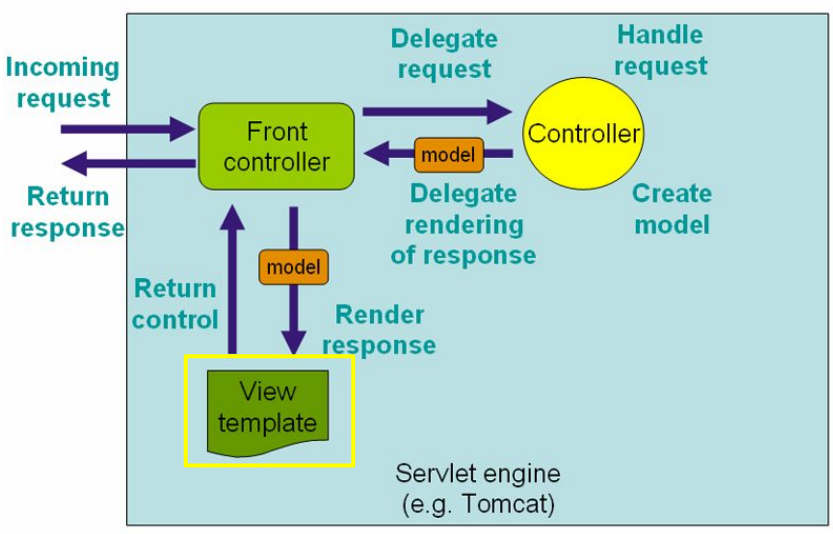


The request processing workflow in Spring Web MVC (high level)

spring

View

- When the Controller has finished processing, the Front Controller determines which **view** to render.
- The Front Controller passes the model to the view which is finally which is finally rendered on the browser.

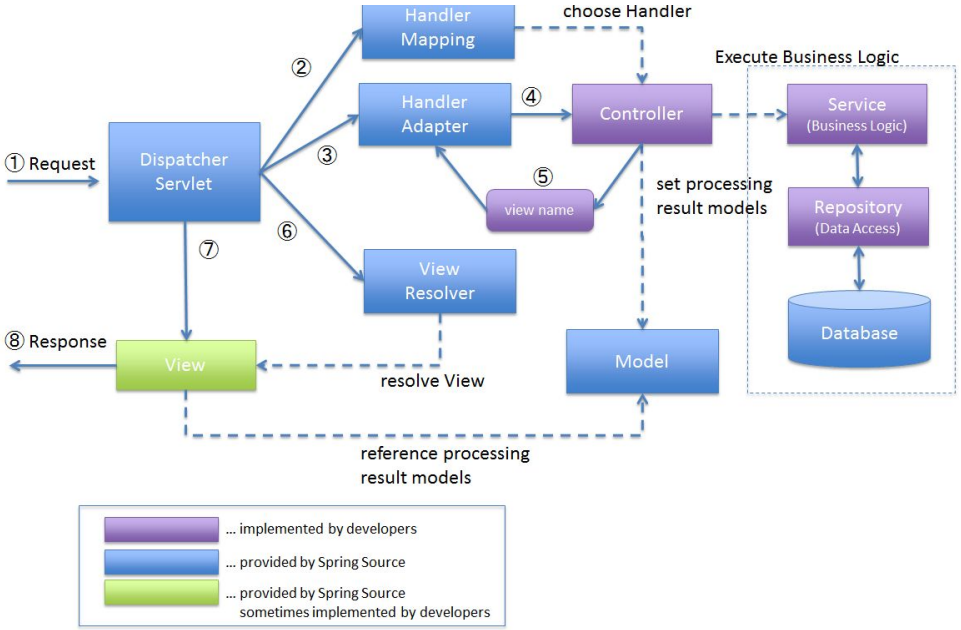


The request processing workflow in Spring Web MVC (high level)

spring

Request Processing Workflow (Details Level)

1. **DispatcherServlet** receives the request.
2. **DispatcherServlet** dispatches the task of selecting an appropriate controller to **HandlerMapping**. **HandlerMapping** selects the Controller which is mapped to the incoming request URL and returns the (selected **Handler**) and **Controller** to DispatcherServlet.
3. **DispatcherServlet** dispatches the task of executing of business logic of **Controller** to **HandlerAdapter**.
4. **HandlerAdapter** calls the business logic process of **Controller**.
5. **Controller** executes the business logic, sets the processing result in **Model** and returns the logical name of view to **HandlerAdapter**.
6. **DispatcherServlet** dispatches the task of resolving the **View** corresponding to the View name to **ViewResolver**. **ViewResolver** returns the **View** mapped to View name.
7. **DispatcherServlet** dispatches the rendering process to returned **View**.
8. **View** renders **Model** data and returns the response.

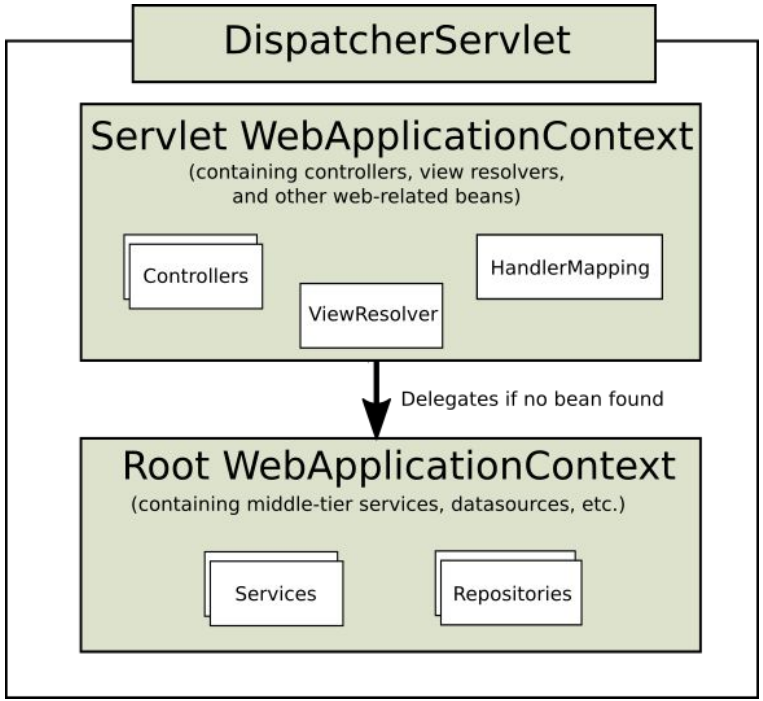


spring

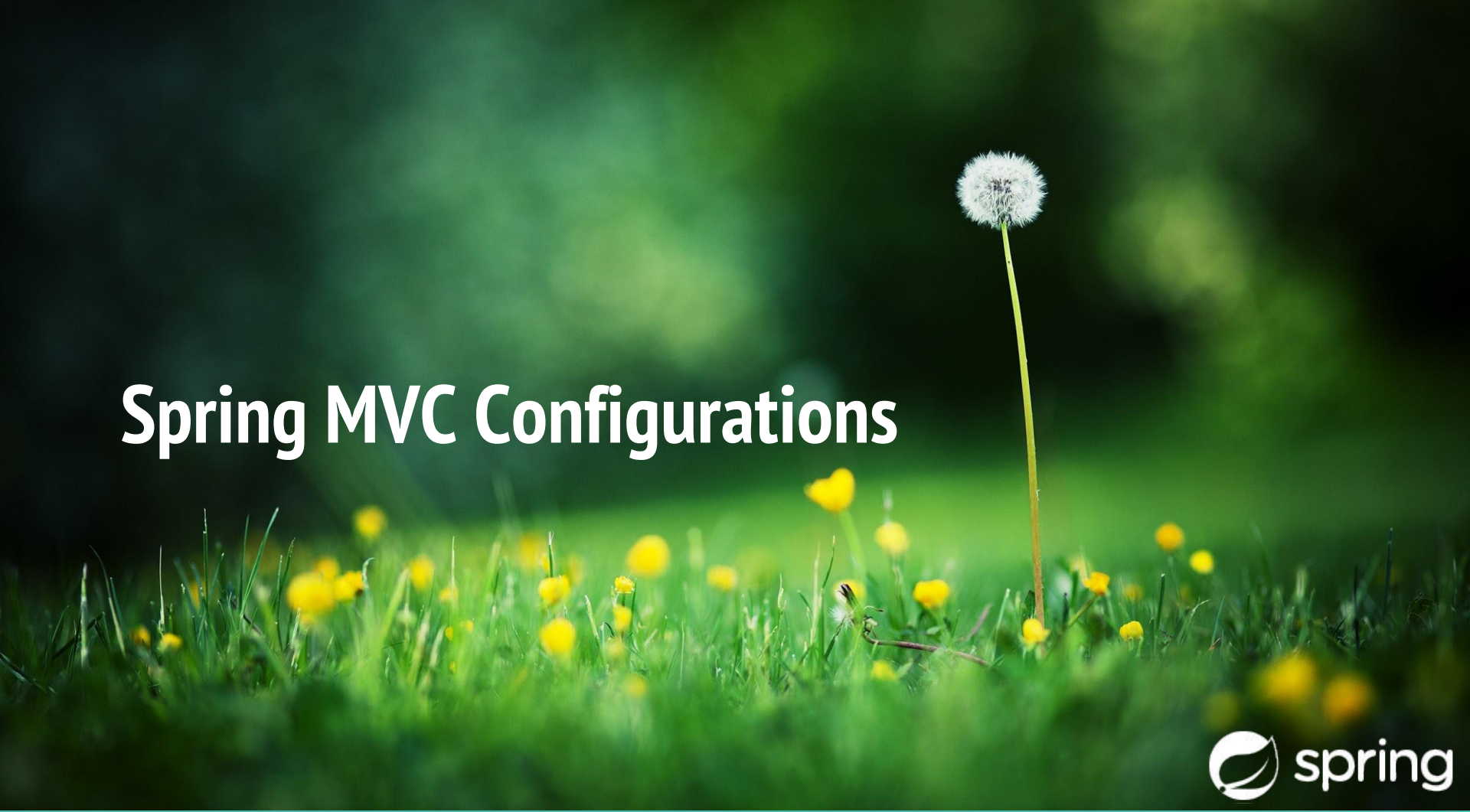

Web Application Context
spring

Web Application Context

- **DispatcherServlet** expects a **WebApplicationContext**, an extension of a plain **ApplicationContext**, for its own configuration.
- **WebApplicationContext** has a link to the **ServletContext** and **Servlet** it is associated with.
- The **root WebApplicationContext** typically contains infrastructure beans such as data **repositories** and business **services** which are inherited and could be overridden in the **Servlet WebApplicationContext**.
- Servlet WebApplicationContext contains web-related beans: such as controllers, handler mappings,...



spring


Spring MVC Configurations
spring

Spring MVC Configurations

Spring MVC supports 2 type of configurations:

- *XML Configuration*
- *Java-based Configuration*

Important annotations

Important annotations

Some important annotations which are used in a Spring MVC application.

- *@Controller* and *@RestController*
- *@RequestMapping*
  - @GetMapping, @PostMapping, @PutMapping and @DeleteMapping
- *@RequestParam*, *@PathVariable*
- *@RequestBody*
- *@ResponseBody*

Summary