# 14. Spring MVC

-Presentation layer is the front door into your app.

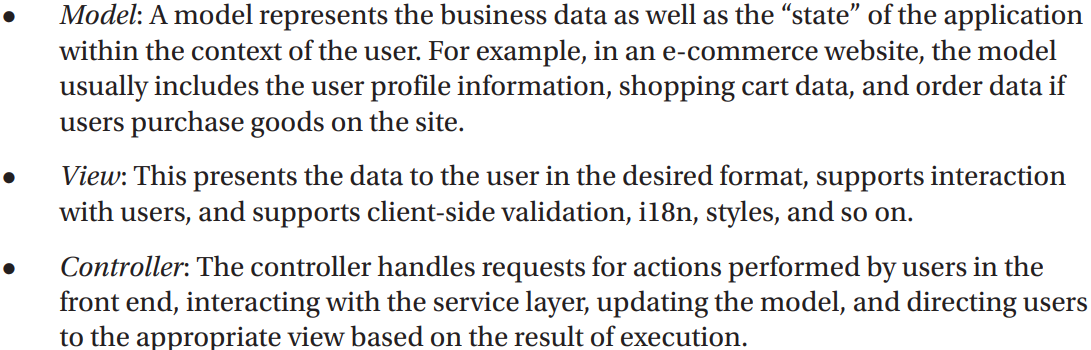
-Some major considerations when developing web applications: performance, user-friendliness, interactivity and richness, accessibility

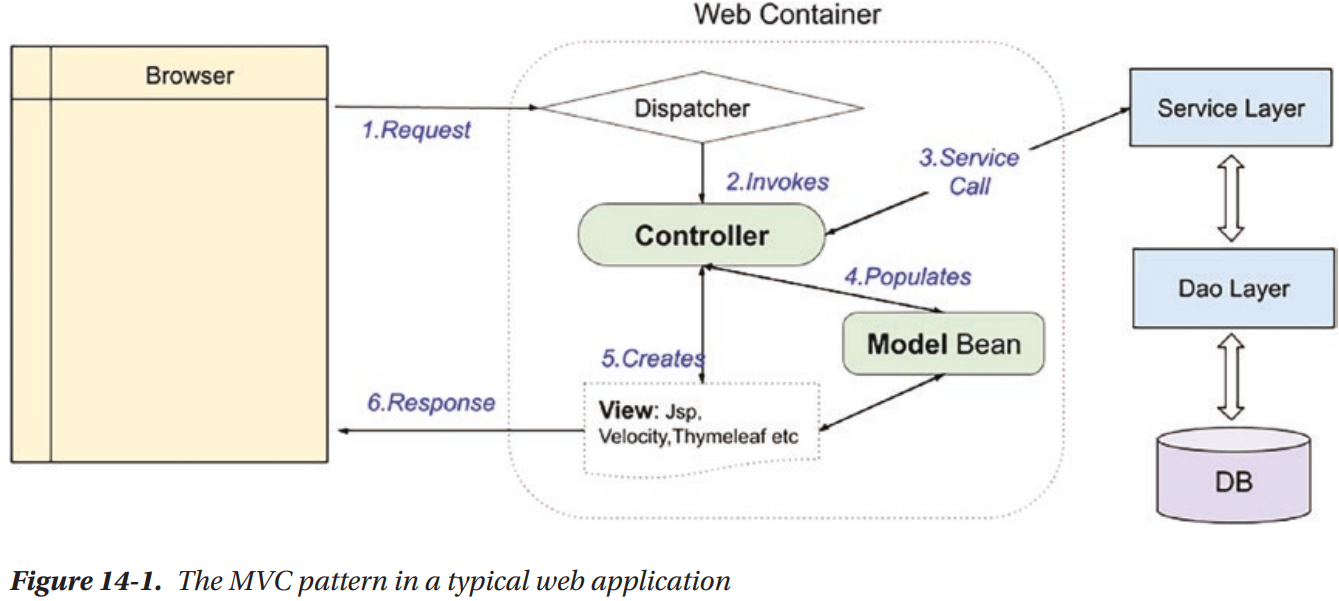
## 14.1 Setting up the Data and Lower-Level Layers

-(See in book)

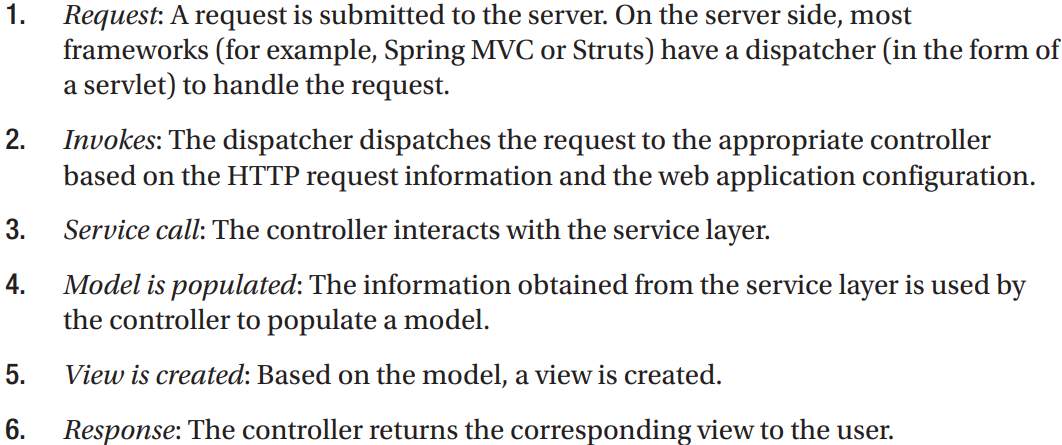
## 14.2 Introduce MVC

-MVC in pattern in implementing the presentation layer of app. Main principle of MVC: define an architecture with clear responsibilities for different components.





-A normal view request is handled:



## 14.3 Introduce Spring MVC

-In Spring Framework, Spring MVC module provides support for MVC pattern with support features (theming, i18n, validation, type conversion, formatting)

### 14.3.1 Spring MVC WebApplicationContext Hierarchy

-**DispatcherServlet**: the central servlet that receives requests and dispatches them to appropriate controllers. There can be any number of DispatcherServlet instances for various purposes (UI requests, RESTful-WS requests). Each DispatcherServlet has its own WebApplicationContext configuration which defines servlet-level characteristics.

-Spring MVC maintains a root WebApplicationContext which includes application-level configurations (back-end data source, security, service, persistence layer configuration). The root WebApplication will be available to all servlet-level WebApplicationContexts

-Example: 2 DispatcherServlet instances: application servlet and RESTful servlet.

A diagram of a data processing process

AI-generated content may be incorrect.

### 14.3.2 Spring MVC Request Life Cycle

-Main components involved in handling a request in Spring MVC:

A diagram of a service

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.

A close up of a text

AI-generated content may be incorrect.

A close up of a text

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.

### 14.3.3 Spring MVC Configuration

-To **configure Spring MVC support** for web applications, perform configurations for web deployment descriptor:

A close up of words

AI-generated content may be incorrect.

-The **easiest way** is to extend **AbstractAnnotationConfigDispatcherServletInitializer**

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

A close up of a computer code

AI-generated content may be incorrect.

-AbstractAnnotationConfigDispatcherServletInitializer implement **WebApplicationInitializer**: in order to configure **ServletContext**.

A screenshot of a computer

AI-generated content may be incorrect.

-All classes implement **WebApplicationInitializer** are automatically detected by **SpringServletContainerInitializer** (implement **ServletContainerInitializer**, which bootstrap automatically in Servlet 3.0 containers)

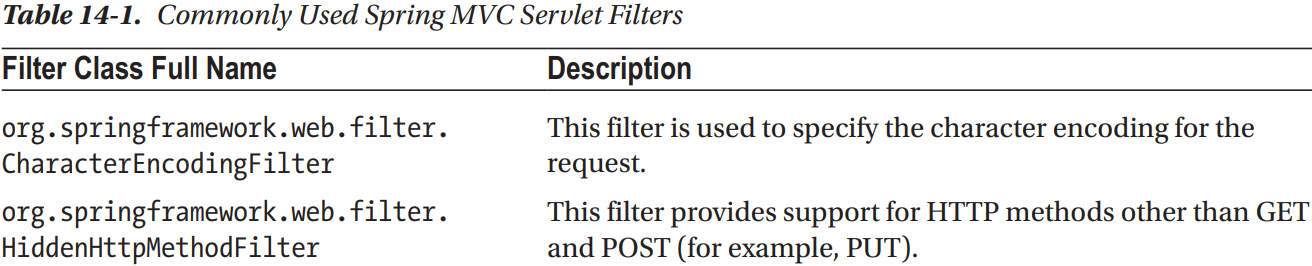
-Overridden methods:

A screenshot of a computer program

AI-generated content may be incorrect.

A close up of black text

AI-generated content may be incorrect.



### 14.3.4 Create the 1st View in Spring MVC

-Thymeleaf: server-side Java template engine for both web and stand-alone environments, it integrates smoothly with Spring applications since it was created specially for them.

-Dependencies: Jakarta.servlet-api, spring-webmvc, thymeleaf-spring

### 14.3.5 Configure DispatcherServlet

-Configuration class that defines all infrastructure beans needed for Spring web application:

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

-**WebMvcConfigurer**: defines callback methods to customize configuration for Spring MVC enabled by @EnableWebMvc.

A close up of a text

AI-generated content may be incorrect.

A screenshot of a computer

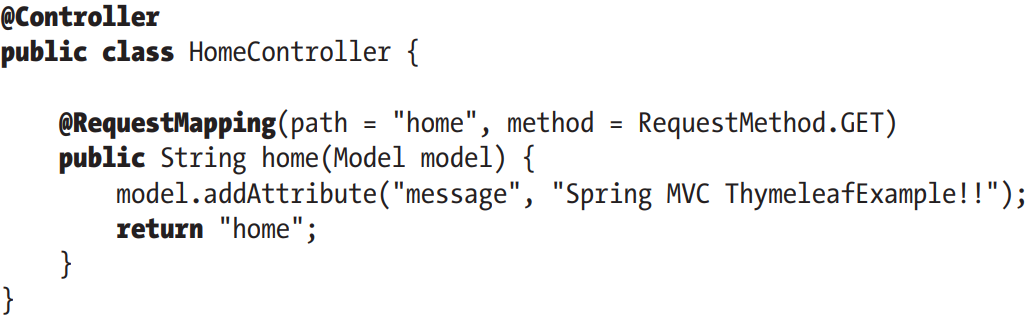
AI-generated content may be incorrect.

A close up of text

AI-generated content may be incorrect.

### 14.3.6 Implement Spring Controllers

-HomeController:

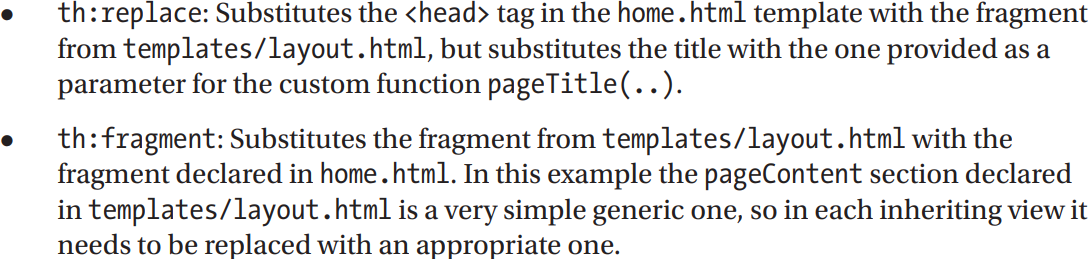


-The controller class is sometimes called a handler, the methods are handler methods.

### 14.3.7 Implement the View

A screenshot of a computer program

AI-generated content may be incorrect.



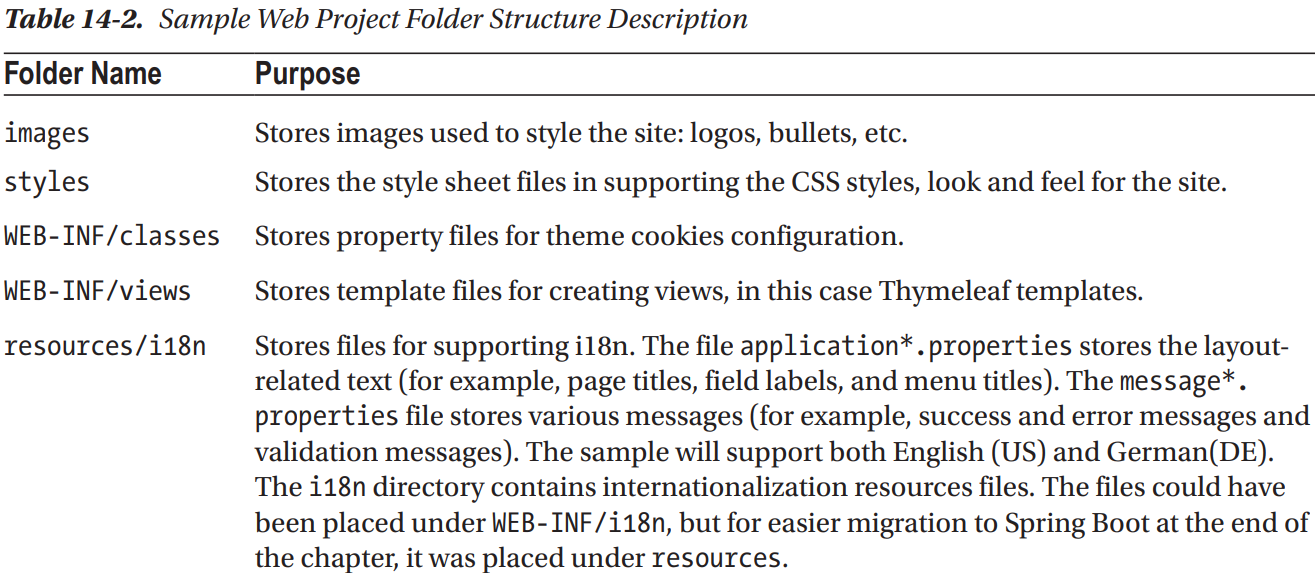
### 14.3.8 Testing the Home View

A screen shot of a computer program

AI-generated content may be incorrect.

### 14.3.9 Understand the Spring MVC Project Structure

-In web application, lots of files are required to support various features. It’s good practice to store files that serve different purposes in well-structured folder hierarchy:



## 14.4 Enabling Internationalization (i18n)

-Enabling i18n: externalize language-related UI settings into various properties files in /resources/i18n

### 14.4.1 Configure i18n in DispatcherServlet Configuration

### 14.4.2 Modify Views for i18n Support

## 14.5 Use Theming and Templating

## 14.6 Implement more Complex Views

## 14.7 Enabling JSR-349 (Bean Validation)

## 14.8 Exception Handling

## 14.9 Switch to Spring Boot

# 15. Spring REST Support

## 15.1 Introduce RESTful Web Services

### 15.1.1 Use Spring MVC to Expose RESTful Web Services

### 15.1.2 Implement SingerController

### 15.1.3 Test the RESTful-WS Application

### 15.1.4 REST Exception Handling Using ResponeEntity<t>

### 15.1.5 REST Exception Handling use @RestControllerAdvice

## 15.2 RESTful-WS with Spring Boot

# 16 Spring Native and Other Goodies

## 16.1 Spring Native Images

### 16.1.1 The Application

### 16.1.2 The Configuration

## 16.2 Spring for GraphQL

## 16.3 Spring Kotlin Applications

### 16.3.1 The Configuration

### 16.3.2 The Code

# 17. Securing Spring Web Applications

## 17.1 Configure Spring Security: The Classic Way

### 17.1.1 JDBC Authentication

### 17.1.2 Testing Secured Web Applications

## 17.2 Configure Spring Security: The Spring Boot way

# 18. Monitoring Spring Applications

## 18.1 JMX Support in Spring

### 18.1.1 Exporting a Spring Bean to JMX

### 18.1.2 Using VisualVM for JMX Monitoring

### 18.1.3 Monitoring Hibernate Statistics

## 18.2 JMX with Spring Boot

## 18.3 Monitoring Applications with Spring Boot Actuator

### 18.3.1 Spring Boot Actuator Endpoints

### 18.3.2 Use Spring Boot Actuator with Micrometer

# 19. Spring WebSocket Support

## 19.1 Introduce WebSocket

## 19.2 Use WebSocket with Spring

## 19.3 Use WebSocket API

## 19.4 Use SockJS

## 19.5 Send Messages with STOMP

### 19.5.1 Spring Boot Equivalent Application