TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA CÔNG NGHỆ THÔNG TIN

# LẬP TRÌNH JAVA NÂNG CAO

## CHƯƠNG 3: Spring Core

Giảng Viên Giảng Dạy:
ThS. Nguyễn Minh Tân
ThS. Đặng Đình Hòa
ThS. Trần Công Thanh

HỌC KỲ III – NĂM HỌC 2023-2024

KHÓA 27T-IT

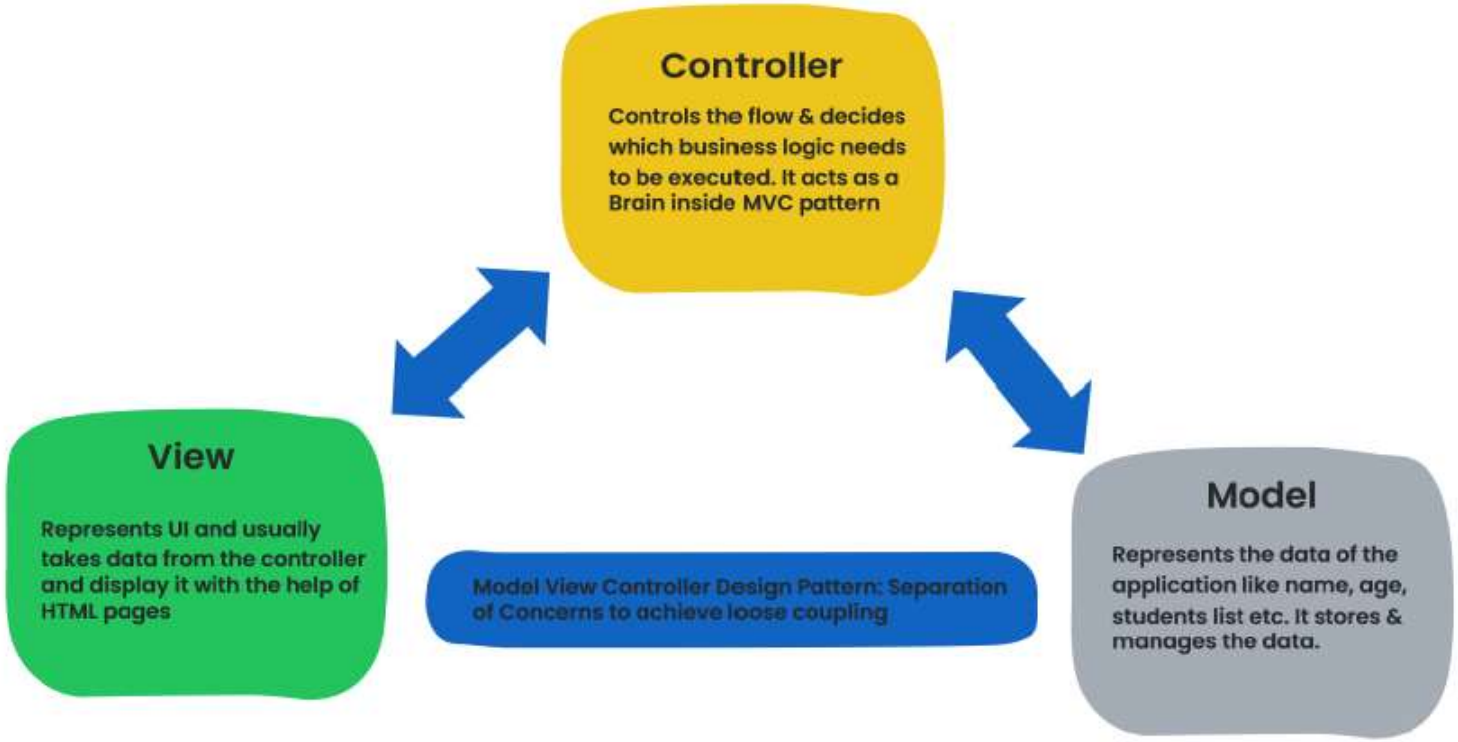
# Nội dung

1. Tổng quan về mô hình MVC
2. Lợi ích của mô hình MVC
3. Kết luận
4. Spring MVC
5. Q & A

# Mô hình MVC
# Model – View - Controller

# Mô hình MVC

- Mô hình MVC giúp chia nhỏ ứng dụng thành các thành phần riêng biệt và cung cấp một cách tổ chức hợp lý để phát triển ứng dụng hiệu quả.



# Model - M

- Đại diện cho dữ liệu và quản lý các logic liên quan đến dữ liệu. Model là nơi thao tác với cơ sở dữ liệu, xử lý và lưu trữ dữ liệu của ứng dụng.

```
public class Product {
    private Long id;
    private String name;
    private double price;

    // Constructors, getters, setters
}
```

# VIEW - V

- Là phần giao diện người dùng, hiển thị thông tin cho người dùng và tương tác với họ. View không xử lý dữ liệu, chỉ đơn thuần hiển thị thông tin được cung cấp từ Model.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Product List</title>
</head>
<body>
    <h1>Product List</h1>
    <table>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Price</th>
        </tr>
        <tr th:each="product : ${products}">
            <td th:text="${product.id}"></td>
            <td th:text="${product.name}"></td>
            <td th:text="${product.price}"></td>
        </tr>
    </table>
</body>
</html>
```

# Controller - C

- Là cầu nối giữa Model và View. Nó nhận các yêu cầu từ người dùng thông qua View, xử lý các yêu cầu này bằng cách tương tác với Model và sau đó cập nhật lại View dựa trên dữ liệu đã xử lý.

```java
@Controller
public class ProductController {
    private List<Product> products = new ArrayList<>();

    @GetMapping("/products")
    public String showProductList(Model model) {
        // Giả sử có một danh sách sản phẩm đã được lấy từ cơ sở dữ liệu
        products.add(new Product(1L, "Laptop", 1000.0));
        products.add(new Product(2L, "Điện thoại", 500.0));
        products.add(new Product(3L, "Tivi", 800.0));

        model.addAttribute("products", products);
        return "productList"; // Trả về tên của View, ở đây là "productList"
    }
}
```

# Lợi ích của mô hình MVC

- **Tách biệt trách nhiệm**: Mô hình MVC giúp tách biệt các thành phần chính của ứng dụng, làm cho mã nguồn trở nên dễ đọc, dễ hiểu và dễ bảo trì hơn.
- **Tính mở rộng và tái sử dụng**: Nhờ có việc tách biệt trách nhiệm, bạn có thể dễ dàng mở rộng chức năng của ứng dụng mà không ảnh hưởng đến các phần khác. Điều này cũng giúp tái sử dụng mã nguồn một cách hiệu quả.
- **Phát triển song song**: Các thành viên trong nhóm phát triển có thể làm việc đồng thời trên các phần khác nhau của ứng dụng, như Model, View và Controller, giúp tăng hiệu suất và tiết kiệm thời gian.
- **Kiểm thử dễ dàng**: Việc tách biệt các thành phần trong mô hình MVC giúp đơn giản hóa việc kiểm thử, do đó giảm thiểu khả năng gây lỗi cho toàn bộ ứng dụng.

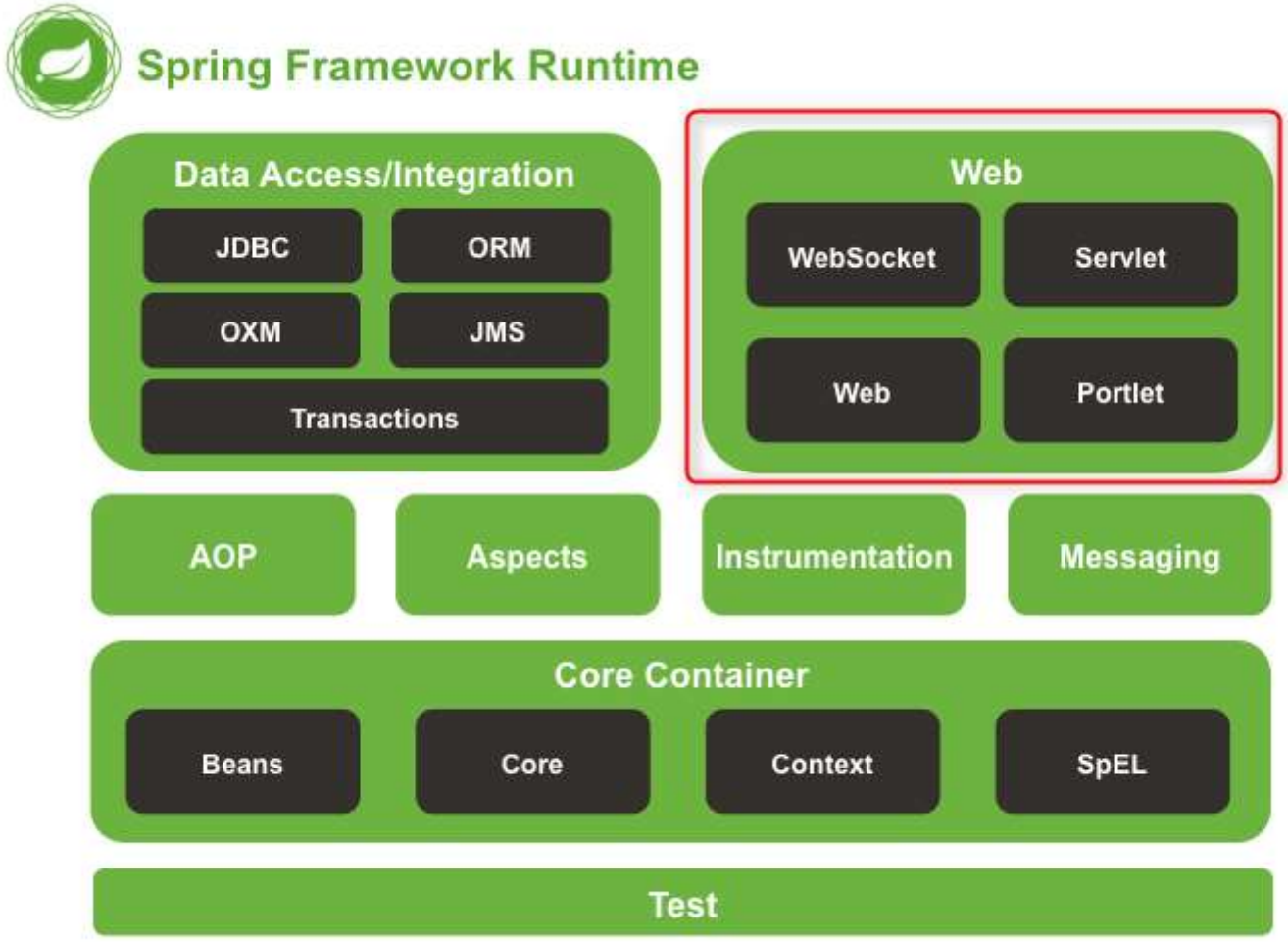
# Kết luận

- Mô hình MVC là một trong những mô hình phát triển phần mềm phổ biến và hữu ích. Nó giúp tách biệt trách nhiệm giữa dữ liệu, giao diện người dùng và logic xử lý, giúp chúng ta dễ dàng phát triển và duy trì ứng dụng một cách hiệu quả.

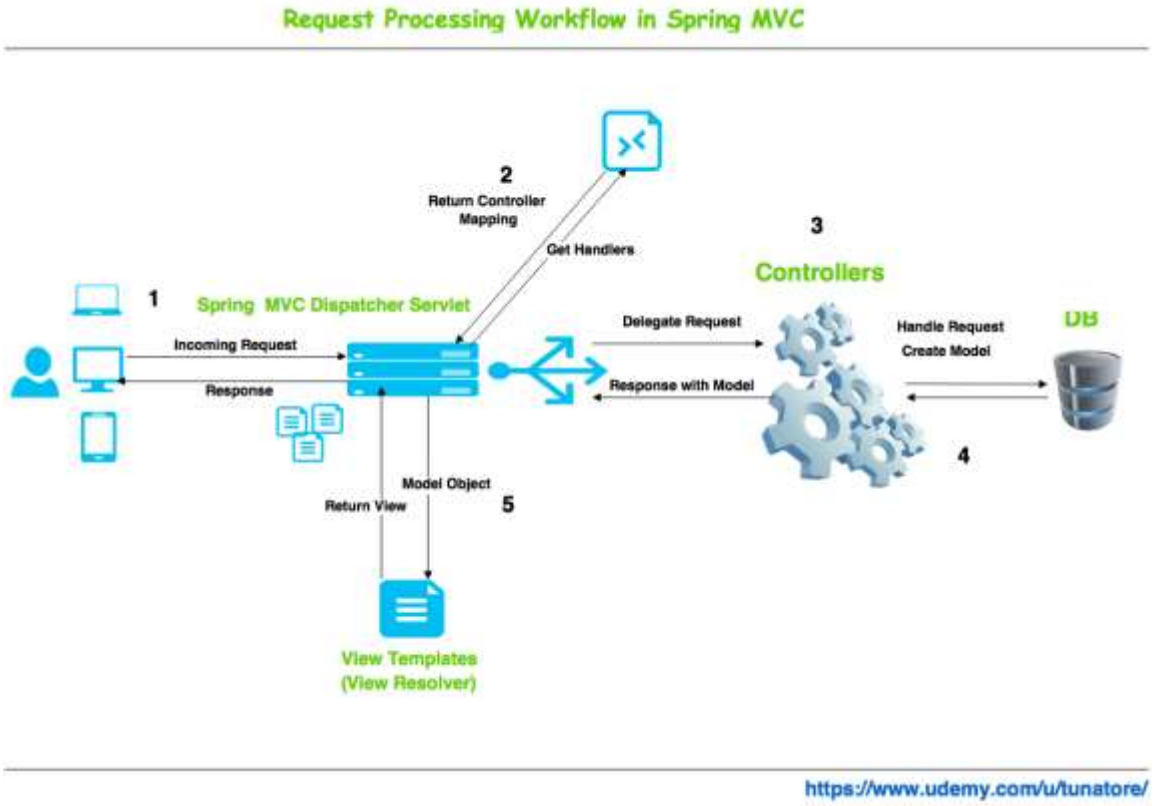
# Một số mô hình MVC khác

- Mô hình MVVM (Model-View-ViewModel)
- Mô hình MVP (Model-View-Presenter)
- Mô hình MVW (Model-View-Whatever)
- Mô hình FLUX
- Mô hình Clean Architecture
- Mô hình HMVC (Hierarchical Model-View-Controller)

# Spring MVC


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
WebSocket
Servlet
Web
Portlet
AOP
Aspects
Instrumentation
Messaging
Core Container
Beans
Core
Context
SpEL
Test

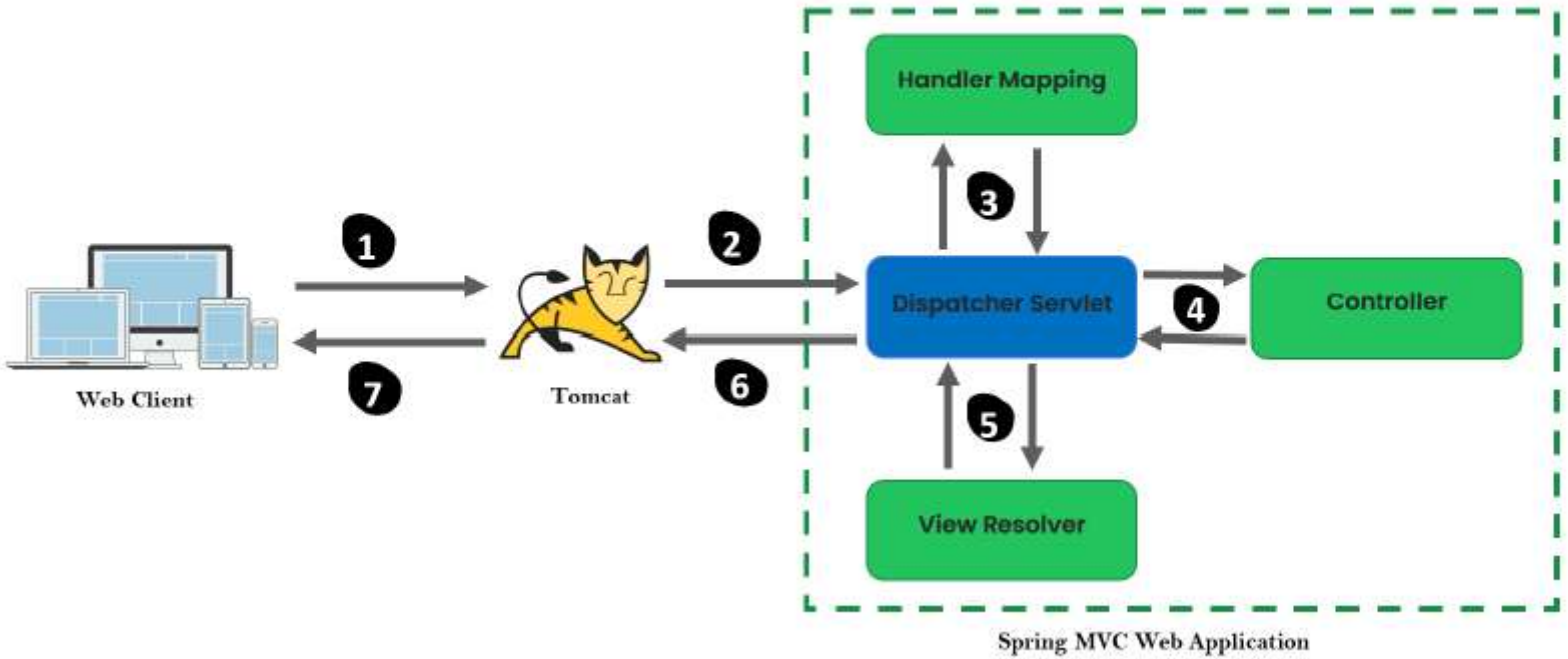
# Spring MVC là gì




- Spring MVC sử dụng mô hình MVC (Model-View-Controller) giúp chia nhỏ ứng dụng thành các thành phần riêng biệt và cung cấp một cách tổ chức hợp lý để phát triển ứng dụng web.

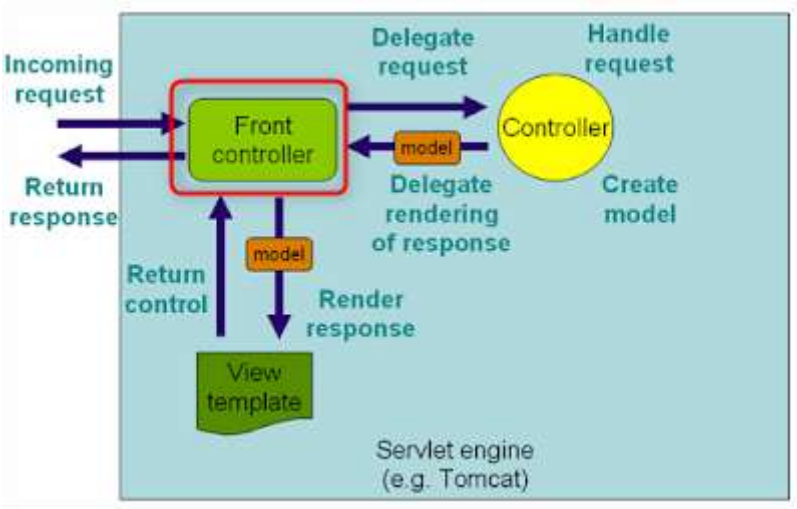
## SPRING MVC ARCHITECTURE & INTERNAL FLOW

eazy bytes



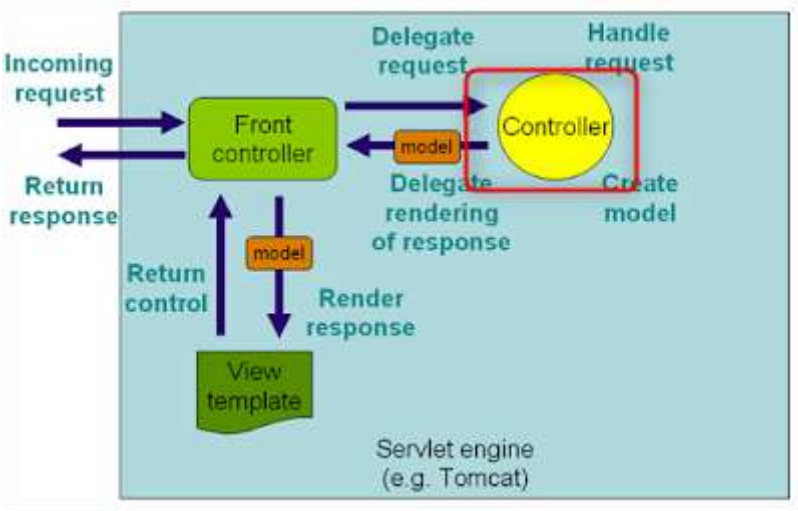
# Các thành phần chính của mô hình MVC

- DispatcherServlet: Là một Servlet đặc biệt trong Spring MVC, nó là trung tâm điều khiển của toàn bộ ứng dụng. DispatcherServlet nhận yêu cầu từ người dùng và gửi chúng tới Controller phù hợp để xử lý.



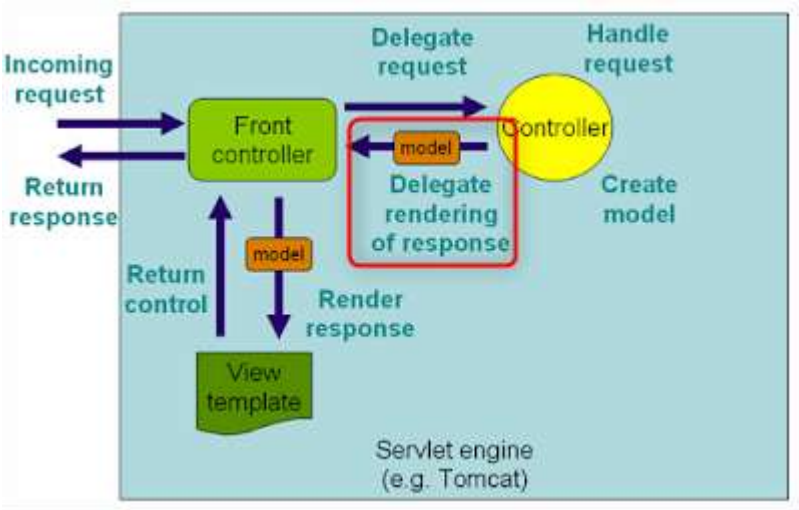
# Các thành phần chính của mô hình MVC

- Controller: Là thành phần xử lý logic trong ứng dụng, nhận yêu cầu từ DispatcherServlet, tương tác với Model để lấy hoặc cập nhật dữ liệu, sau đó trả về dữ liệu cho DispatcherServlet để hiển thị thông tin lên View.



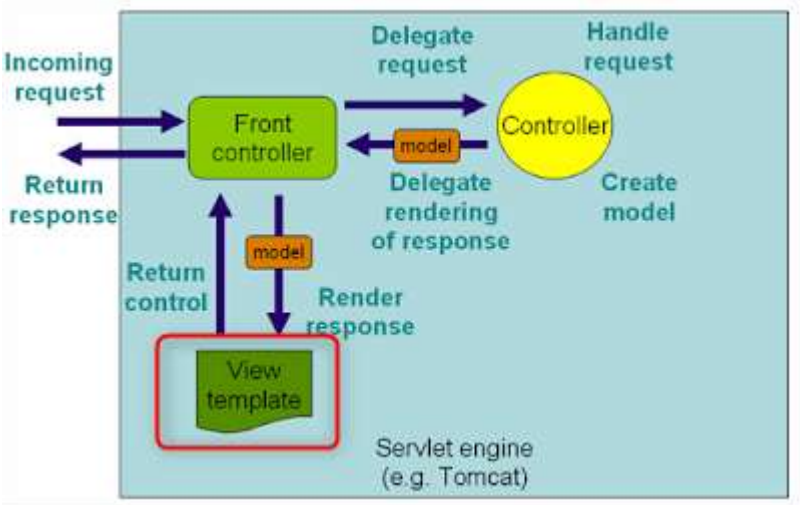
# Các thành phần chính của mô hình MVC

- Model: Đại diện cho dữ liệu của ứng dụng, chứa các lớp POJO (Plain Old Java Object) để lưu trữ thông tin. Model cũng chịu trách nhiệm xử lý logic liên quan đến dữ liệu.

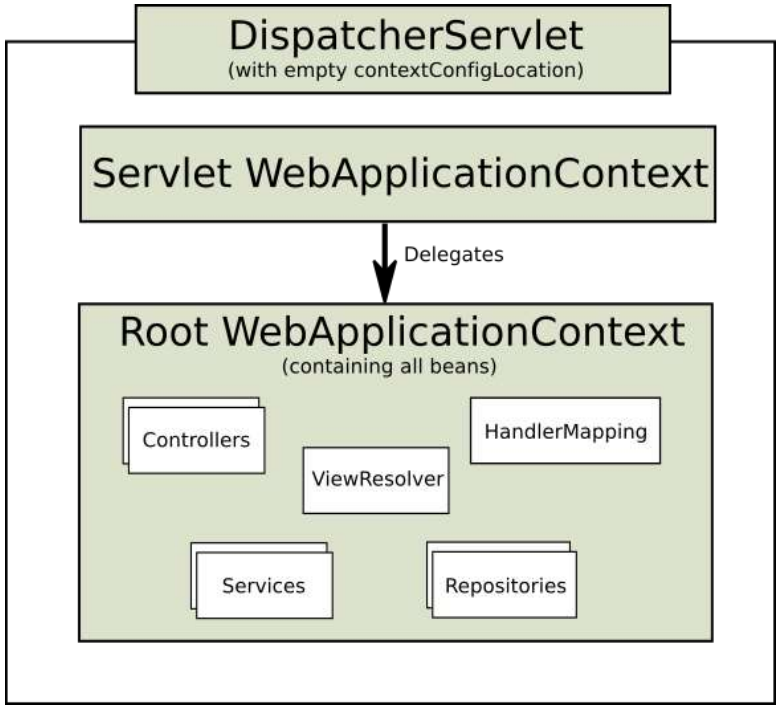


# Các thành phần chính của mô hình MVC

- View: Là giao diện người dùng, hiển thị dữ liệu từ Model cho người dùng. Spring hỗ trợ nhiều công nghệ View khác nhau như JSP, Thymeleaf, Freemarker, ...



# Web Application Context



# Spring MVC Configurations

Spring MVC hỗ trợ 2 loại cấu hình là:

- *XML Configuration*
- Java-based Configuration

# Import 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*

spring

TRƯỜNG ĐẠI HỌC
VĂN LANG

KHOA CÔNG NGHỆ THÔNG TIN