TRƯỜNG ĐẠI HỌC NÔNG LÂM TP.HCM

KHOA: CÔNG NGHỆ THÔNG TIN



**Software High Level Design**

ĐỀ TÀI **: WEBSITE TOEIC**

MÔN: ĐỒ ÁN CHUYÊN NGÀNH

Thực hiện:

1. Nguyễn Công Vũ – 18130288
2. Phạm Tấn Tài - 17130204
3. Trần Ngô Tường Nguyên - 17130147

Link tài liệu : [SoftwareHighLevelDesign\_Webtoeic](https://docs.google.com/document/d/1V7XOozN_u-a4XWjeMFDtoeFrfpfZ2Ib7x9W9WKo_CuE/edit?usp=sharing)

**TABLE OF CONTENTS**

[**Giới thiệu:**](#_hgtdzrxqs7yd) **3**

[Mục đích:](#_1fob9te) 3

[Phạm vi:](#_lxm1cmka9swc) 3

[Definitions, Acronyms and Abbreviations:](#_u9dp0027qw31) 3

[**II. ARCHITECTURAL REPRESENTATION:**](#_csxhkdi0hr1x) **4**

[**III. LOGICAL VIEW:**](#_igwlyou5rkf5) **6**

[4+1 Architectural view model:](#_bbgsocc9e5ox) 6

[3.1. Logical view:](#_qsh70q) 7

[3.2. Process view:](#_5wzqun36ag7a) 7

[3.3. Development view:](#_u0f0pxhkto3h) 7

[3.4. Physical view:](#_gavtury3drpb) 7

[3.5. Scenarios:](#_ihr5e6l57htq) 8

[**IV. Spring Framework:**](#_cyvnsca5dadc) **8**

[Test](#_230xqfetaqrr) 8

[Spring Core Container](#_qqhiqlxhhopi) 9

[AOP, Aspects and Instrumentation](#_ulns0qcd8jr) 9

[Data Access / Integration](#_qm5zngi8pryp) 9

[Web](#_syclecz6oyda) 9

[3.1. Các lợi ích của Spring Framework:](#_hq9zr9xmwlm2) 9

[3.2. Một số dự án khác của Spring:](#_vnpbl13x6dk7) 10

[Spring MVC](#_g7i2zixyvtut) 10

[Spring Security](#_k3zmp44qzo72) 10

[Spring Boot](#_d58cpgrpwrlv) 10

# Giới thiệu:

## Mục đích:

Ở bài viết này mình sẽ giới thiệu một số nguyên tắc thiết kế **RESTful API** cho mọi người. Vì như các bạn thấy rằng để viết ra được các API thì là việc cũng khá đơn giản nhưng viết ra được 1 **RESTful API** sao cho hợp lý, những người làm FrontEnd có thể gọi nó một cách dễ dàng thì đây là việc mà không phải ai cũng làm được.

## Phạm vi:

**RESTful API** là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

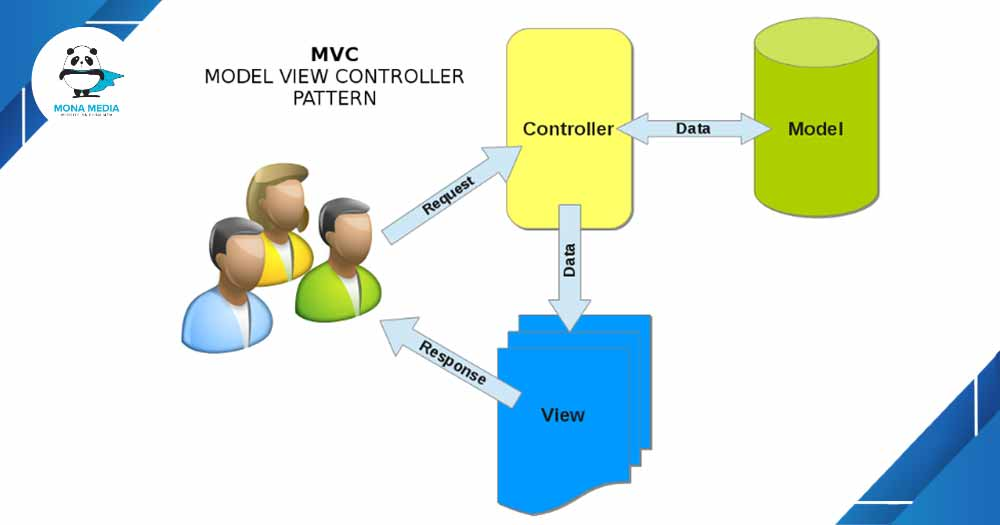
## Definitions, Acronyms and Abbreviations:

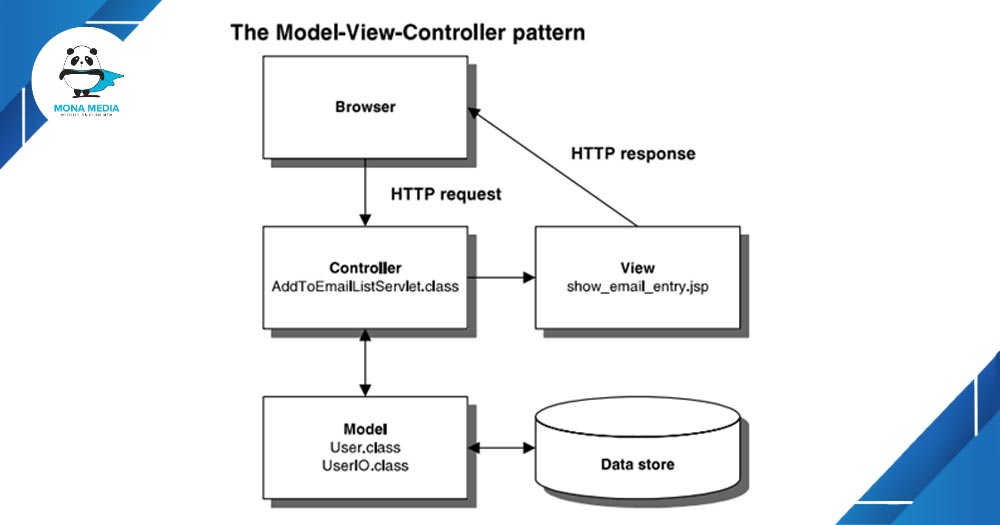
| **#** | **Item** | **Description** |
| --- | --- | --- |
| 1 | JSP | Java Server Page |
| 2 | JSTL | Java Standard Tag Library |
| 3 | API | Application Programming Interface |
| 4 | HTML | Hypertext Mark-up Language |
| 5 | WYSIWYG | What You See Is What You Get |
| 6 | JVM | Java Virtual Machine |
| 7 | HTTP | Hypertext-Transfer Protocol |
| 8 | MVC | Model – View – Control |
| 9 | J2EE | Java 2 Enterprise Edition - A set of standards for implementing scalable, reliable enterprise solutions, from reusable components. These components come with a set of services which handle the behavior of the application. |
| 11 | VO | Value Object |
| 12 | SPRING | Spring is a light weight framework to support the IoC dependency injection concept. It is becoming a widely adopted framework and has many books, articles and coding examples online.  Spring's main aim is to make J2EE easier to use and promote good programming practice. It does this by enabling a POJO (plain old java object)-based programming model that is applicable in a wide range of environments. |
| 14 | O-R | Object Relational |

# II. ARCHITECTURAL REPRESENTATION:

Kiến trúc MVC

**MVC** là từ viết tắt bởi 3 từ **Model – View – Controller**. Đây là mô hình thiết kế sử dụng trong kỹ thuật phần mềm. Mô hình source code thành 3 phần, tương ứng mỗi từ. Mỗi từ tương ứng với một hoạt động tách biệt trong một mô hình.





Mô hình MVC và các thành phần bên trong của MVC.

**Model (M):**

Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Bộ phận này là một cầu nối giữa 2 thành phần bên dưới là View và Controller. Model thể hiện dưới hình thức là một cơ sở dữ liệu hoặc có khi chỉ đơn giản là một [file XML](https://monamedia.co/file-xml-la-gi/) bình thường. Model thể hiện rõ các thao tác với cơ sở dữ liệu như cho phép xem, truy xuất, xử lý dữ liệu,…

**View (V):**

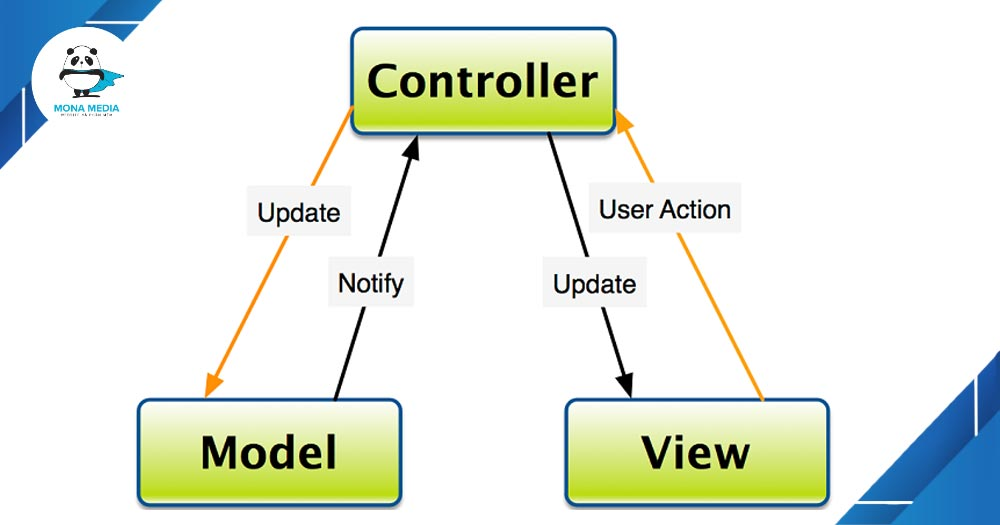
Đây là phần giao diện (theme) dành cho người sử dụng. Nơi mà người dùng có thể lấy được thông tin dữ liệu của MVC thông qua các thao tác truy vấn như tìm kiếm hoặc sử dụng thông qua các website.

Thông thường, các ứng dụng web sử dụng MVC View như một phần của hệ thống, nơi các thành phần [HTML](https://monamedia.co/html-va-css-la-gi/) được tạo ra. Bên cạnh đó, View cũng có chức năng ghi nhận hoạt động của người dùng để tương tác với Controller. Tuy nhiên, View không có mối quan hệ trực tiếp với Controller, cũng không được lấy dữ liệu từ Controller mà chỉ hiển thị yêu cầu chuyển cho Controller mà thôi.

Ví dụ: Nút “delete” được tạo bởi View khi người dùng nhấn vào nút đó sẽ có một hành động trong Controller.

**Controller (C):**

Bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua view. Từ đó, C đưa ra dữ liệu phù hợp với người dùng. Bên cạnh đó, Controller còn có chức năng kết nối với model.



Luồng tương tác giữa các thành phần trong MVC.

Ví dụ: Chức năng thêm bài viết mới trong trang quản trị website. Nơi đây có 2 trường nhập về tiêu đề và nội dung bài viết, thì trong đó:

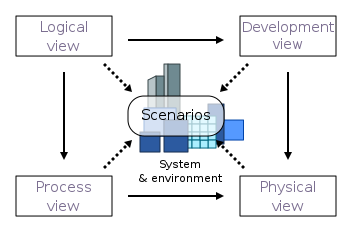
* View sẽ hiển thị ra phần nhập form tiêu đề và nội dung.
* Controller lấy dữ liệu từ 2 trường và gửi tới Model.
* Model lúc này sẽ nhận dữ liệu từ Controller để lưu trữ trong cơ sở dữ liệu.

Mỗi bộ phận thực hiện chức năng nhất định, nhưng chúng có sự thống nhất, liên kết với nhau tạo nên **mô hình MVC**. Mô hình này tương đối nhẹ. Nó có thể tích hợp được nhiều tính năng có trong ASP.NET hiện giờ. Ví dụ như authentication (quá trình xác thực).

# III. LOGICAL VIEW:

## 4+1 Architectural view model:

Được sử dụng để "mô tả kiến ​​trúc của các hệ thống sử dụng phần mềm chuyên sâu, dựa trên việc sử dụng nhiều khung nhìn đồng thời". Các khung nhìn được sử dụng để mô tả hệ thống từ quan điểm của các bên liên quan khác nhau, chẳng hạn như người dùng cuối, nhà phát triển, kỹ sư hệ thống và người quản lý dự án. Bốn góc nhìn của mô hình là logic, phát triển, quá trình và vật lý. Ngoài ra, các trường hợp sử dụng hoặc kịch bản đã chọn được sử dụng để minh họa kiến ​​trúc đóng vai trò là chế độ xem 'cộng một'.



**Hình 3.1 Hình minh họa mô hình Architectural view model**

## 3.1. Logical view:

Chế độ xem logic liên quan đến chức năng mà hệ thống cung cấp cho người dùng cuối. UML diagram được sử dụng để biểu diễn chế độ xem logic và bao gồmClass diagram vàState diagram.

## 3.2. Process view:

Chế độ xem quy trình đề cập đến các khía cạnh động của hệ thống, giải thích các quy trình hệ thống và cách chúng giao tiếp và tập trung vào hành vi thời gian chạy của hệ thống. Chế độ xem quy trình đề cập đến sự đồng thời, phân phối, tích hợp, hiệu suất và khả năng mở rộng, v.v. Biểu đồ UML để biểu diễn chế độ xem quy trình bao gồm sequence diagram, communication diagram, activity diagram.

## 3.3. Development view:

Chế độ xem phát triển minh họa một hệ thống từ quan điểm của một lập trình viên và có liên quan đến việc quản lý phần mềm. Dạng xem này còn được gọi là dạng xem thực thi. Biểu đồ UML được sử dụng để biểu diễn dạng xem phát triển bao gồm Package diagram và Component diagram.

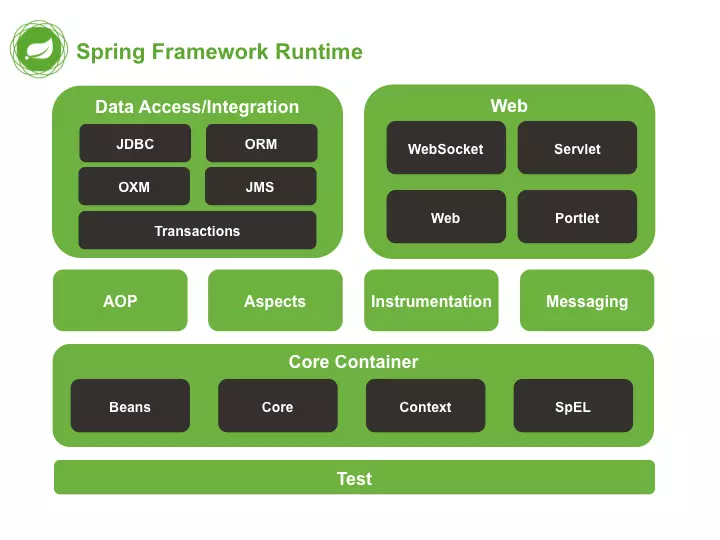
## 3.4. Physical view:

Chế độ xem vật lý (còn gọi là *chế độ xem triển khai* ) mô tả hệ thống từ quan điểm của kỹ sư hệ thống. Nó liên quan đến cấu trúc liên kết của các thành phần phần mềm trên lớp vật lý cũng như các kết nối vật lý giữa các thành phần này. Biểu đồ UML được sử dụng để biểu diễn chế độ xem vật lý bao gồm deployment diagram.

## 3.5. Scenarios:

Mô tả kiến ​​trúc được minh họa bằng cách sử dụng một tập hợp nhỏ các [trường hợp sử dụng](https://en.wikipedia.org/wiki/Use_case) hoặc kịch bản, chúng trở thành dạng xem thứ năm. Các kịch bản mô tả chuỗi tương tác giữa các đối tượng và giữa các quá trình. Chúng được sử dụng để xác định các yếu tố kiến ​​trúc và để minh họa và xác nhận thiết kế kiến ​​trúc. Chúng cũng đóng vai trò là điểm khởi đầu cho các thử nghiệm của một nguyên mẫu kiến ​​trúc. Dạng xem này còn được gọi là dạng xem use case view.

# IV. Spring Framework:



**Hình 4.1. Kiến trúc tổng thể Spring framework**

## Test

Tầng này cung cấp khả năng hỗ trợ kiểm thử với JUnit và TestNG.

## Spring Core Container

Bao gồm các module spring core, beans, context và expression languate (EL)

* Spring core, bean cung cấp tính năng IOC và Dependency Injection.
* Spring Context hỗ trợ đa ngôn ngữ (internationalization), các tính năng Java EE như EJB, JMX.
* Expression Language được mở rộng từ Expression Language trong JSP. Nó cung cấp hỗ trợ việc setting/getting giá trị, các method cải tiến cho phép truy cập collections, index, các toán tử logic…

## AOP, Aspects and Instrumentation

Những module này hỗ trợ cài đặt lập trình hướng khía cạnh (Aspect Oriented Programming), hỗ trợ tích hợp với AspectJ.

## Data Access / Integration

Nhóm này bao gồm JDBC, ORM, OXM, JMS và module Transaction. Những module này cung cấp khả năng giao tiếp với database

## Web

Hay còn gọi là Spring MVC Nhóm này gồm Web, Web-Servlet… hỗ trợ việc tạo ứng dụng web.

## 3.1. Các lợi ích của Spring Framework:

* Spring cho phép lập trình viên sử dụng POJOs. Việc sử dụng POJOs giúp bạn không phải làm việc với EJB, ứng dụng, các luồng chạy, cấu hình… đơn giản hơn rất nhiều.
* Spring được tổ chức theo kiểu mô đun. Số lượng các gói và các lớp khá nhiều, nhưng bạn chỉ cần quan tâm đến những gì bạn cần và không cần quan tâm đến phần còn lại.
* Spring hỗ trợ sử dụng khá nhiều công nghệ như ORM Framework, các logging framework, JEE, các thư viện tạo lịch trình (Quartz và JDK timer)…
* Module Web của Spring được thiết kế theo mô hình MVC nên nó cung cấp đầy đủ các tính năng giúp thay thế các web framework khác như Struts.

## 3.2. Một số dự án khác của Spring:

Cũng dựa trên các nguyên tắc thiết kế cơ bản của spring core. Spring còn phát triển nhiều project con như:

### **Spring MVC**

Spring MVC được thiết kế dành cho việc xây dựng các ứng dụng nền tảng web.

### **Spring Security**

Cung cấp các cơ chế xác thực (authentication) và phân quyền (authorization) cho ứng dụng của bạn.

### **Spring Boot**

Spring Boot là một framework giúp chúng ta phát triển cũng như chạy ứng dụng một cách nhanh chóng.

### 

**~ End ~**