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 : *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**

[2.1. Restful API:](#_glc1dviz9n8h) 4

[2.2. Diễn giải các thành phần:](#_7a1evpapkkq3) 4

[2.3. RESTful hoạt động như thế nào?](#_ab4mlrkrwkkv) 5

[2.4. Status code:](#_w4eicq9jibx7) 6

[2.5. API Document:](#_wx85gc25wezr) 6

[**III. LOGICAL VIEW:**](#_igwlyou5rkf5) **7**

[4+1 Architectural view model:](#_bbgsocc9e5ox) 7

[3.1. Logical view:](#_qsh70q) 8

[3.2. Process view:](#_5wzqun36ag7a) 8

[3.3. Development view:](#_u0f0pxhkto3h) 8

[3.4. Physical view:](#_gavtury3drpb) 8

[3.5. Scenarios:](#_ihr5e6l57htq) 9

[**IV. Spring Framework:**](#_cyvnsca5dadc) **9**

[Test](#_230xqfetaqrr) 9

[Spring Core Container](#_qqhiqlxhhopi) 10

[AOP, Aspects and Instrumentation](#_ulns0qcd8jr) 10

[Data Access / Integration](#_qm5zngi8pryp) 10

[Web](#_syclecz6oyda) 10

[3.1. Các lợi ích của Spring Framework:](#_hq9zr9xmwlm2) 10

[3.2. Một số dự án khác của Spring:](#_vnpbl13x6dk7) 11

[Spring MVC](#_g7i2zixyvtut) 11

[Spring Security](#_k3zmp44qzo72) 11

[Spring Boot](#_d58cpgrpwrlv) 11

[Spring Batch](#_8ftrfhijs64m) 11

[Spring Social](#_h72auyfq5yba) 11

[Spring IO](#_u4um3vz6l2jz) 11

[Spring Cloud](#_7wt2186uraut) 11

[Spring Mobile](#_jisxzmbnp6tg) 12

[Spring for Android](#_uki6zzeuibrd) 12

[Spring Session](#_u4gwg6agpqss) 12

# 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.

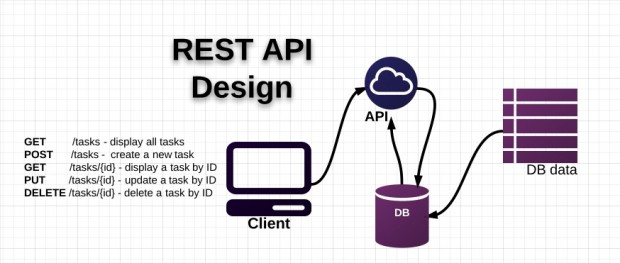
## Definitions, Acronyms and Abbreviations:

| **#** | **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:

## 2.1. Restful API:

**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.



**Hình 2.1.1: Rest API design**

## 2.2. Diễn giải các thành phần:

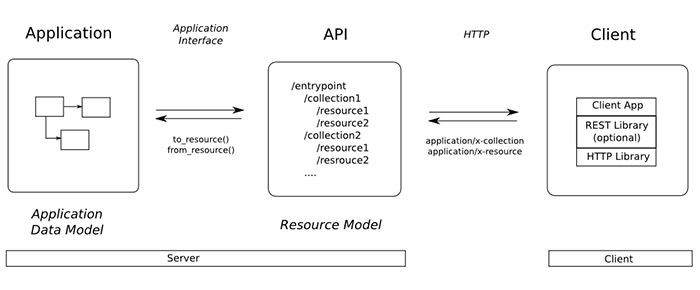
API (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như [JSON](https://topdev.vn/blog/json-la-gi/) hay XML.

REST (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.

RESTful API là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

## 2.3. RESTful hoạt động như thế nào?



**Hình 2.3.1: Hoạt động của restful**

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

* GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
* POST (CREATE): Tạo mới một Resource.
* PUT (UPDATE): Cập nhật thông tin cho Resource.
* DELETE (DELETE): Xóa một Resource.

## 2.4. Status code:

Khi chúng ta request một API nào đó thường thì sẽ có vài status code để nhận biết sau:

* 200 OK – Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.
* 201 Created – Trả về khi một Resource vừa được tạo thành công.
* 204 No Content – Trả về khi Resource xoá thành công.
* 304 Not Modified – Client có thể sử dụng dữ liệu cache.
* 400 Bad Request – Request không hợp lệ
* 401 Unauthorized – Request cần có auth.
* 403 Forbidden – bị từ chối không cho phép.
* 404 Not Found – Không tìm thấy resource từ URI
* 405 Method Not Allowed – Phương thức không cho phép với user hiện tại.
* 410 Gone – Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.
* 415 Unsupported Media Type – Không hỗ trợ kiểu Resource này.
* 422 Unprocessable Entity – Dữ liệu không được xác thực
* 429 Too Many Requests – Request bị từ chối do bị giới hạn

## 2.5. API Document:

Ai cũng biết việc viết API docs là cần thiết, tuy nhiên để có một API docs hoàn chỉnh cũng tiêu tốn khá nhiều thời gian. Nhất là trong lúc dự án gấp rút thì mọi người thường chỉ để API docs ở mức siêu cơ bản. Tham khảo thêm cách viết [API Document](https://topdev.vn/blog/viet-api-document-cho-du-an-nhu-the-nao/).

**API document** là một phần tương tự như Unit Test vậy – lấy ngắn để nuôi dài.

Nếu không được chăm sóc kỹ, thì đến lúc maintain hoặc thay đổi spec thì hậu quả sẽ rất thảm khốc, dưới đây là một số lưu ý lúc viết docs:

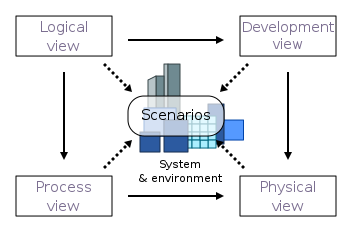
* Mô tả đầy đủ về params request: gồm những params nào, datatype, require hay optional.
* Nên đưa ra các ví dụ về HTTP requests và responses với data chuẩn.
* Cập nhật Docs thường xuyên, để sát nhất với API có bất cứ thay đổi gì.
* Format, cú pháp cần phải nhất quán, mô tả rõ ràng, chính xác.

## 

# 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.

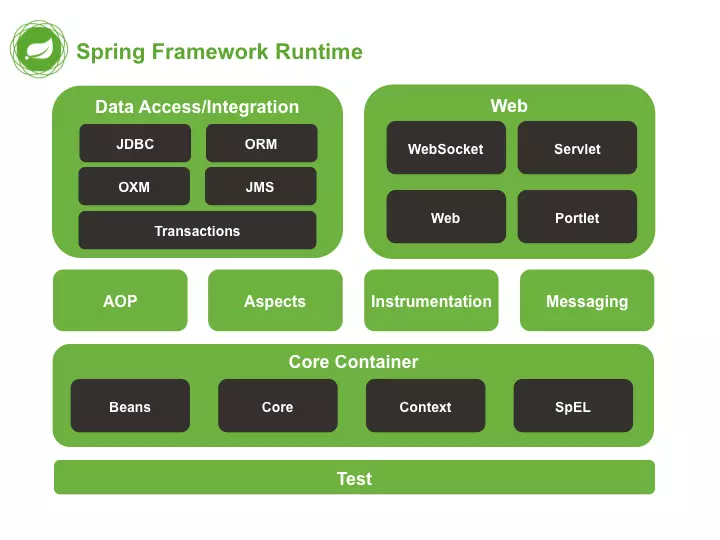
## 3.4. Physical view:

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.

### **Spring Batch**

Dự án này giúp chúng ta dễ dàng tạo các lịch trình (scheduling) và tiến trình (processing) cho các công việc xử lý theo mẻ (batch job).

### **Spring Social**

Dự án này sẽ kết nối ứng dụng của bạn với các API bên thứ ba của Facebook, Twitter, Linkedin … (ví dụ đăng nhập bằng facebook, google+ …)

### **Spring IO**

### **Spring Cloud**

### **Spring Mobile**

### **Spring for Android**

### **Spring Session**

**~ End ~**