**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**ĐỘC LẬP - TỰ DO - HẠNH PHÚC**

.............................…………



BÁO CÁO

TASK\_DAILY

Người quản lý: **Nguyễn Ngọc Lâm**

Trưởng nhóm: **Nguyễn Hữu Úy**

Thành viên: Quách Diêm Phước

Thành phố Hồ Chí Minh, Ngày 6, Tháng 9, Năm 2021.

NỘI DUNG BÁO CÁO

[I. TÌM HIỂU VỀ SPRING VÀ RESTFUL 3](#_Toc18629)

[1. Spring là gì 3](#_Toc11024)

[ Spring Core Container 3](#_Toc23491)

[2. Các lợi ích của Spring Framework. 5](#_Toc13536)

[3. Một số dự án khác của Spring. 5](#_Toc21020)

[4. RESTful API là gì 7](#_Toc546)

[5. Diễn giải các thành phần 7](#_Toc5775)

[6. RESTful hoạt động như thế nào? 8](#_Toc30908)

[7. Status code 9](#_Toc3337)

[II. TÌM HIỂU VỀ JPA VÀ HIBERNATE 11](#_Toc5147)

[1. JPA là gì? 11](#_Toc5546)

[2. ORM là gì? 11](#_Toc25757)

[3. Hibernate là gì? 11](#_Toc29970)

[4. Sự khác nhau giữa JPA và Hibernate 12](#_Toc9014)

[III. TÌM HIỂU VỀ SPRING SECURITY 13](#_Toc13297)

[IV. TÌM HIỂU SPRING Boot sữ dụng SOAP Web Service 15](#_Toc17208)

# TÌM HIỂU VỀ SPRING VÀ RESTFUL

1. **Spring là gì**

* Spring là một Framework phát triển các ứng dụng Java, nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code

## 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](https://stackjava.com/design-pattern/dependency-injection-di-la-gi.html" \t "https://stackjava.com/spring/_blank).
* Spring Context hỗ trợ đa ngôn ngữ (internationalization), các tính năng Java EE như EJB, JMX.
* Expression Language được mở rộng từ Expresion 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.

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](https://stackjava.com/uncategorized/orm-la-gi-tong-quan-ve-orm-framework.html" \t "https://stackjava.com/spring/_blank), 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.
* …

1. **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

...

1. **RESTful API là gì**

* Có thể nói bản thân ****REST**** không phải là một loại công nghệ. Nó là phương thức tạo API với nguyên lý tổ chức nhất định. Những nguyên lý này nhằm hướng dẫn lập trình viên tạo môi trường xử lý API request được toàn diện.
* ****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.

1. **Diễn giải các thành phần**

* **API** (**A**pplication **P**rogramming **I**nterface) 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** hay **XML**.
* **REST** (**RE**presentational **S**tate **T**ransfer) 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**.

1. **RESTful hoạt động như thế nào?**

* **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)**: Xoá một Resource.
* Những phương thức hay hoạt động này thường được gọi là **CRUD** tương ứng với **Create**, **Read**, **Update**, **Delete** – Tạo, Đọc, Sửa, Xóa.

1. **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 **Resouce** 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
* **500 Internal Server Error** lỗi nội bộ của **Server**

# TÌM HIỂU VỀ JPA VÀ HIBERNATE

1. **JPA là gì?**

* JPA (Java Persistence API) là một mô tả đạt chuẩn của Java để làm việc với cơ sở dữ liệu quan hệ
* JPA là cầu nối giữa Java Object với Cơ sở dữ liệu. Là một tập hợp các đặc tả (interface) và cần có trình triển khai các đặc tả mà JPA định nghĩa
* Có những ORM Tools cung cấp trình triển khai cho JPA như Hibernate

1. **ORM là gì?**

* **ORM** viết tắt của **object-relational-mapping,** cho phép chuyển đổi từ các **object** trong ngôn ngữ hướng đối tượng sang **database quan hệ** và ngược lại.
* Hầu hết các ứng dụng đều cần lưu trữ, thao tác với cơ sở dữ liệu. Đối với công nghệ **JDBC** trước đây chúng ta phải thực hiện các công việc như kết nối cơ sở dữ liệu, viết câu truy vấn,..

1. **Hibernate là gì?**

* **Hibernate** là 1 **ORM** framework cho phép người lập trình thao tác với database một cách hoàn toàn tự nhiên thông qua các đối tượng. Lập trình viên hoàn toàn không cần quan tâm đến loại database sử dụng, SQL…
* Hay nói cách khác, **Hibernate** chính là cài đặt của **JPA** (**JPA** là 1 tập các interface, còn **Hibernate** implements các interface ấy 1 cách chi tiết).

1. Sự khác nhau giữa **JPA** và **Hibernate**

* **JPA** là tập các giao diện (interfaces), qui chuẩn, định nghĩa. còn **Hibernate** thực hiện cài đặt (implements) các giao diện đó, định nghĩa đó.

# TÌM HIỂU VỀ SPRING SECURITY

1. **Spring Security là gì?**

* **Spring Security** là một **Framework** cung cấp các **Authentication**, **Authorization** và **Protection** để bảo mật và tránh các sự xâm nhập từ bên ngoài.
* **Spring Security** hỗ trợ cho cả **Imperative** và **Reactive Applications**,và là tiêu chuẩn thực tế để bảo vệ các ứng dụng dựa trên **Spring**.
* **Spring Security** là một lớp phần quyền bảo mật trang web

1. **Điều kiện tiên quyết**

* **Spring Security** yêu cầu chạy môi trường **Runtime** phải từ **Java8** trở lên.
* **Spring Security** chạy dựa trên **hoạt động khép kính** nên **không cần phải cài đặt thêm** bất kì **cấu hình** nào trong môi trường **Runtime**

1. **Các tính năng mới trong Spring Security**
2. **Servlet**

* OAuth 2.0 Client
* Hỗ trợ thêm **Jwt Client Authentication** **private\_key\_jwt** và **client\_secret\_jwt**
* Triển khai thêm **R2DBC** implementation của **ReactiveOAuth2AuthorizedClientService**
* OAuth 2.0 Resource Server
* Bộ giải mã **JWT** nâng cao các dẫn xuất của các **thuật toán**
* Cải thiện các chức năng phủ định nội dung
* Cải thiện việc hỗ trợ cho nhiều người sử dụng
* Configuration
* Hỗ trợ công cụ phù hợp với **DispatcherType**
* Hỗ trợ **AuthorizationManager** để bảo mật bộ lọc

1. **WebFlux**

* Thêm hỗ trợ quy trình **Kotlin** cho **EnableReactiveMethodSecurity**

1. **Build**

* Tất cả các ứng dụng mẫu đều được chuyển sang một dự án riêng biệt
* Bản build đầy đủ hiện đang hoạt động với **JDK 11**

1. **Spring Boot với Maven**

* **Spring Boot** hỗ trợ **spring-boot-starter-security** để khởi động các **depdency** liên quan đến **Spring Security** lại với nhau.

<dependencies> *<!-- ... other dependency elements ... -->* <dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency> </dependencies>

# TÌM HIỂU SPRING Boot sỬ dụng SOAP Web Service

1. **Những gì bạn cần có khi sử dụng SOAP Web Service**

* Hiểu được **SOAP Web Service** là gì?
* Một trình soạn văn bản hoặc một **IDE** yêu thích
* JDK 1.8 trở lên
* Gradle 4 hoặc Maven 3.2 trở lên
* Bạn có thể nhập mã trực tiếp vào **IDE** mình đang sử dụng
* **Spring Tool Suite (STS)**
* **IntelliJ IDEA**

1. **Hướng dẫn phát triển bằng SOAP Web Service**
2. Để bắt đầu, Thêm **Dependency** với **Spring Initializr.**

<?xml version="1.0" encoding="UTF-8"?><project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.5.2</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>consuming-web-service-initial</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>consuming-web-service-initial</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

1. **Import Dependency SOAP Web Service** vào **Maven**

<plugin>

<groupId>org.jvnet.jaxb2.maven2</groupId>

<artifactId>maven-jaxb2-plugin</artifactId>

<version>0.14.0</version>

<executions>

<execution>

<goals>

<goal>generate</goal>

</goals>

</execution>

</executions>

<configuration>

<schemaLanguage>WSDL</schemaLanguage>

<generatePackage>com.example.consumingwebservice.wsdl</generatePackage>

<schemas>

<schema> <url>http://localhost:8080/ws/countries.wsdl</url>

</schema>

</schemas>

</configuration>

</plugin>

*//Note: Ví dụ trong bài tập*

*Vì Object có giá trị = null nên nó ko thể gọi được các hàm của nó*

*=> NullPointerException*