



T3H

IT-Institute

# Lập trình Java

---

# Spring MVC

Ths. Vũ Duy Khương



1

**Giới thiệu cấu trúc MVC ?**

2

**Tạo Project Spring MVC**

3

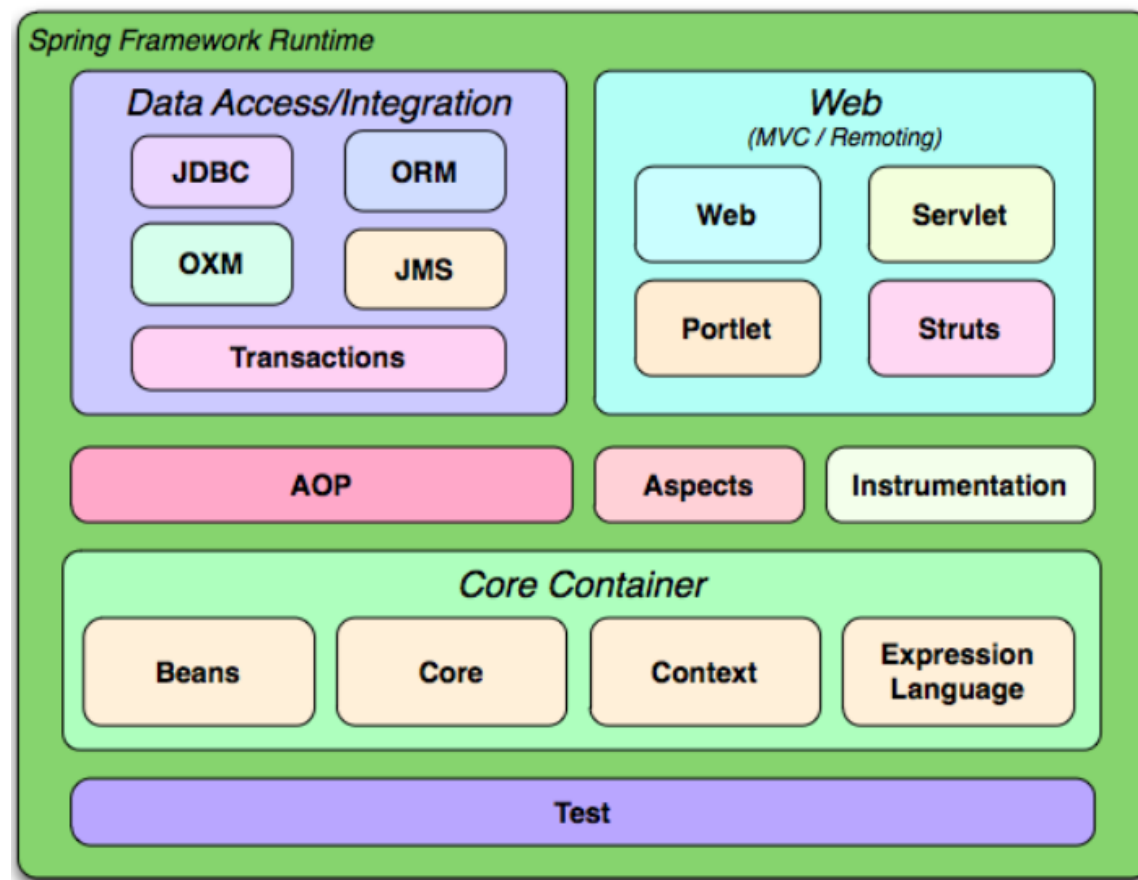
**Anotation trong Spring MVC**

4

**Tạo project Spring MVC Annotation**

# Giới thiệu về cấu trúc MVC?

## Module Spring MVC trong framework spring



# Giới thiệu về cấu trúc MVC?

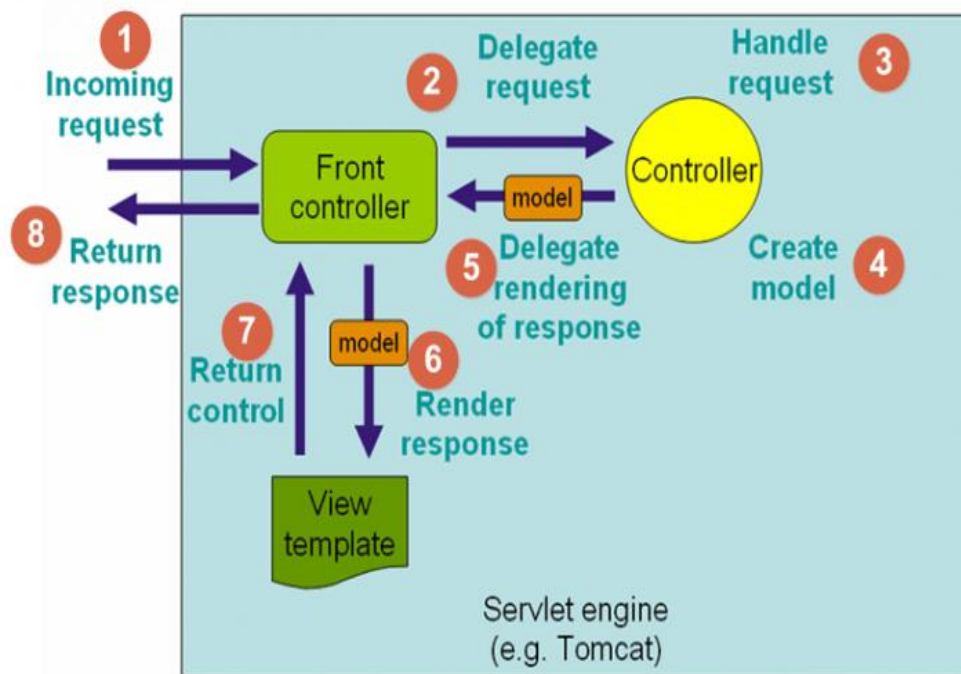
- ❖ Spring MVC là một Framework / 1 Project mã nguồn mở của Spring.
- ❖ Spring MVC Framework cung cấp kiến trúc MVC (Model-View-Controller) và các component được sử dụng để phát triển các ứng dụng web một cách linh hoạt.
- ❖ Được thiết kế xung quanh 1 DispatcherServlet để gửi yêu cầu tới các thành phần xử lý

# Giới thiệu về cấu trúc MVC?

- ❖ Những thành phần xử lý có thể cấu hình được
- ❖ Thành phần xử lý được dựa trên các annotation như @Controller, @RequestMapping

# Giới thiệu về cấu trúc MVC?

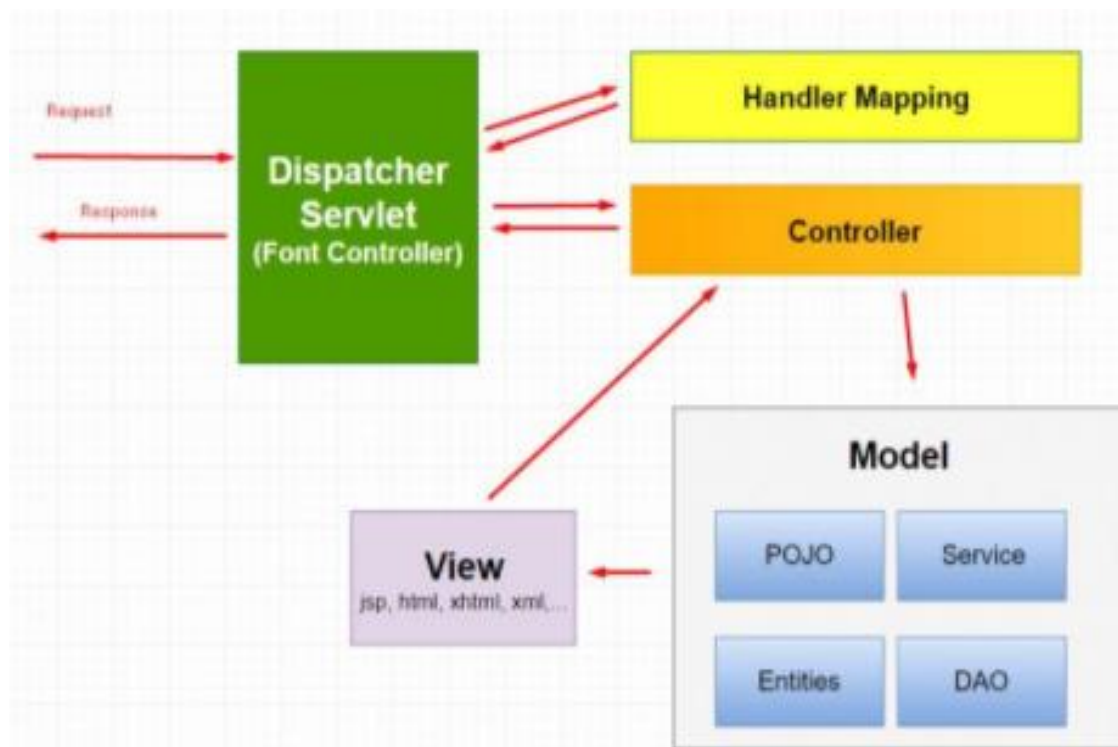
## - FLOW trong SPRING MVC



- Bất kỳ request nào tới ứng dụng web đều sẽ được gửi tới Front Controller (Dispatcher Servlet)
- Front Controller sẽ sử dụng Handler Mapping để biết được controller nào sẽ xử lý request đó
- Controller nhận request, gọi tới các class service thích hợp để xử lý yêu cầu.
- Sau khi xử lý xong, Controller sẽ nhận được model từ tầng Service hoặc tầng DAO.
- Controller gửi model vừa nhận được tới Front Controller (Dispatcher Servlet)
- Dispatcher Servlet sẽ tìm các mẫu view, sử dụng view resolver và truyền model vào nó.
- View template, model, view page được build và gửi trả lại Front Controller
- Front Controller gửi một page view tới trình duyệt để hiển thị nó cho người dùng.

# Giới thiệu về cấu trúc MVC?

## - FLOW trong SPRING MVC



- Model: là các file POJO, Service, DAO thực hiện truy cập database, xử lý business
- View: là các file JSP, html...
- Control: là Dispatcher Controller, Handler Mapping, Controller – thực hiện điều hướng các request.

# Giới thiệu về cấu trúc MVC?

## Các ưu điểm của SPRING MVC

- Các tầng trong Spring MVC độc lập nên việc unit test dễ dàng hơn.
- Phần view có thể tích hợp với nhiều Framework về UI như JSF, Freemarker, Themeleaf...
- Spring MVC base trên các POJO class nên các hành động của nó khá đơn giản



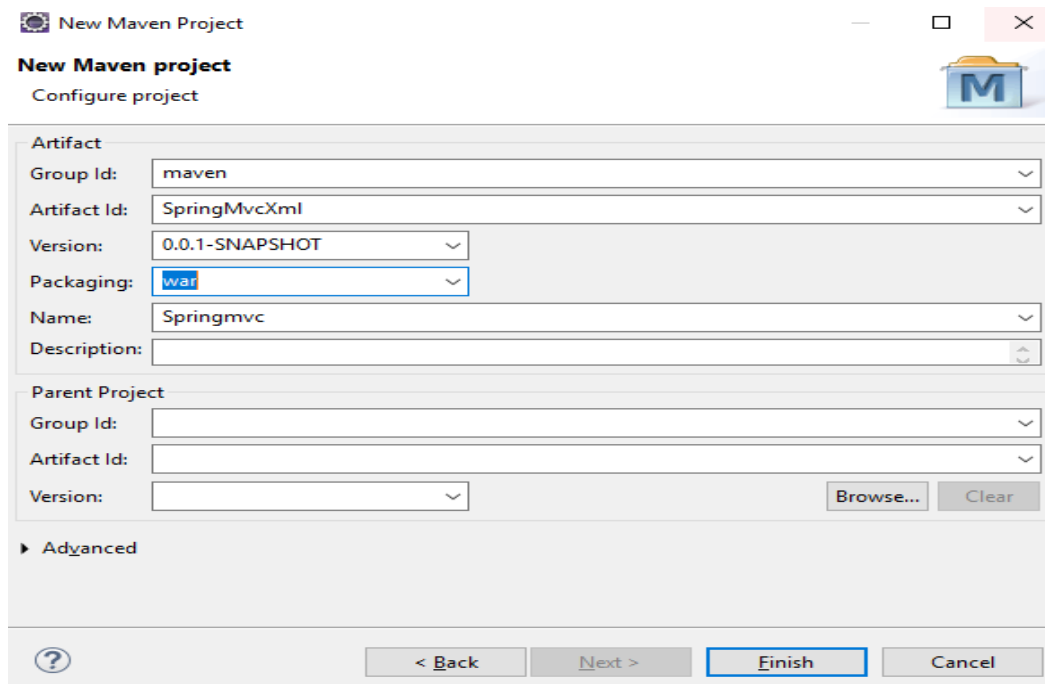
# Giới thiệu về cấu trúc MVC?

## Các ưu điểm của SPRING MVC

- Hỗ trợ cả Annotation và XML config giúp việc phát triển nhanh hơn và sạch hơn.
- Cung cấp việc phân chia một cách rõ ràng, linh hoạt giữa controller, service, data acces layer.

# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B1. Tạo Maven Project**



New Maven Project

**New Maven project**  
Configure project

Artifact

Group Id: maven

Artifact Id: SpringMvcXml

Version: 0.0.1-SNAPSHOT

Packaging: war

Name: Springmvc

Description:

Parent Project

Group Id:

Artifact Id:

Version:

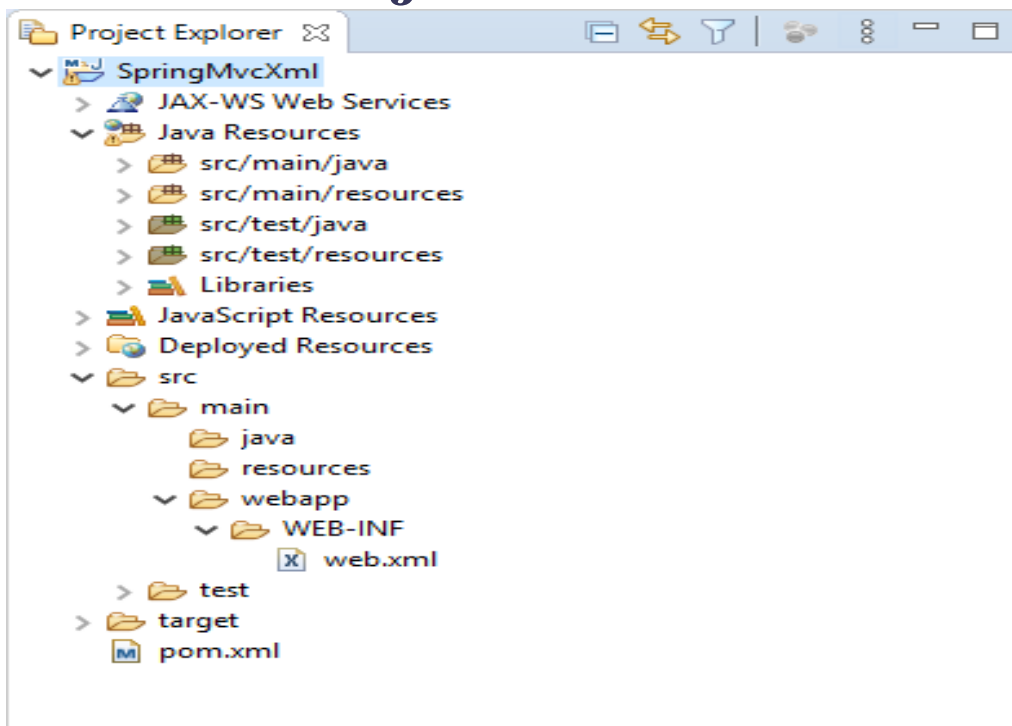
Browse... Clear

Advanced

< Back Next > Finish Cancel

# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B1. Tạo Maven Project**



# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo Pom.xml**

```

SpringMvcXml/pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.i
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>maven</groupId>
4   <artifactId>SpringMvcXml</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <packaging>war</packaging>
7   <name>Springmvc</name>
8
9   <properties>
10    <spring.version>5.0.2.RELEASE</spring.version>
11  </properties>
12  <dependencies>
13    <!-- Dependency thu vien Sping MVC -->
14    <dependency>
15      <groupId>org.springframework</groupId>
16      <artifactId>spring-webmvc</artifactId>
17      <version>${spring.version}</version>
18    </dependency>
19    <!-- Dependency thu vien ServletC -->
20    <dependency>
21      <groupId>javax.servlet</groupId>
22      <artifactId>jsp-api</artifactId>
23      <version>2.0</version>
24      <scope>provided</scope>
25    </dependency>
26  </dependencies>
27 </project>
  
```

# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo web.xml**

```

web.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xmlns="http://java.sun.com/xml/ns/javaee"
4      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/we
5      version="2.5">
6      <display-name>SpringMVCHello</display-name>
7      <welcome-file-list>
8          <welcome-file>index.jsp</welcome-file>
9      </welcome-file-list>
10     <!-- Servlet -->
11     <servlet>
12         <servlet-name>spring-mvc</servlet-name>
13         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
14         <load-on-startup>1</load-on-startup>
15     </servlet>
16     <servlet-mapping>
17         <servlet-name>spring-mvc</servlet-name>
18         <!-- url-pattern là '/' tức là tất cả các request sẽ đều đi qua dispatcher servlet -->
19         <url-pattern>/</url-pattern>
20     </servlet-mapping>
21
22     <!-- file web.xml không chỉ rõ file config cho Spring (Dispatcher Servlet)
23     thì mặc định Spring sẽ tìm file WEB-INF/{servlet-name}-servlet.xml -->
24     <!-- Spring sẽ tìm file spring-mvc-servlet.xml trong folder WEB-INF. -->
25 </web-app>
    
```

# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo spring-mvc-servlet.xml ( file config Spring )**

```

spring-mvc-servlet.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springf
6     http://www.springframework.org/schema/context http://www.springframework.org/schem
7
8     <!-- Thực hiện auto scan component với package stackjava.com.springmvchello -->
9     <context:component-scan
10         base-package="controller" />
11     <!-- Bean InternalResourceViewResolver -->
12     <bean
13         class="org.springframework.web.servlet.view.InternalResourceViewResolver">
14         <property name="prefix">
15             <value>/WEB-INF/views/jsp</value>
16         </property>
17         <property name="suffix">
18             <value>.jsp</value>
19         </property>
20     </bean>
21     <!-- InternalResourceViewResolver: thực hiện mapping các file view tương
22         ứng, TH này nó sẽ map các file trong folder WEB-INF/views/jsp có đuôi là
23         .jsp -->
24 </beans>
    
```

# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo Controller**

```
@Controller
public class HelloController {

    @RequestMapping("/")
    public String index() {
        return "index"; // Tên file JSP
    }
    @RequestMapping(value = "/hellolink", method = RequestMethod.GET)
    public String hello() {
        return "hello"; // Tên file JSP
    }
    @RequestMapping(value = "/googlelink", method = RequestMethod.GET)
    public String google() {
        return "google"; // Tên file JSP
    }
}
```

# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo File View**

```
index.jsp ✕
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>INDEX</title>
8 </head>
9 <body>
10 <h1>Spring MVC JAVA WEB!</h1>
11 <h1>INDEX</h1>
12 <a href="hellolink">Click sang link hello</a>
13 </body>
14 </html>
```



# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo File View**

```
hello.jsp
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="ISO-8859-1">
7   <title>HELLO</title>
8 </head>
9 <body>
10   <h1>Spring MVC HELLO!</h1>
11   <h1>HELLO</h1>
12   <a href="googleLink">Click sang link google</a>
13
14 </body>
15 </html>
```

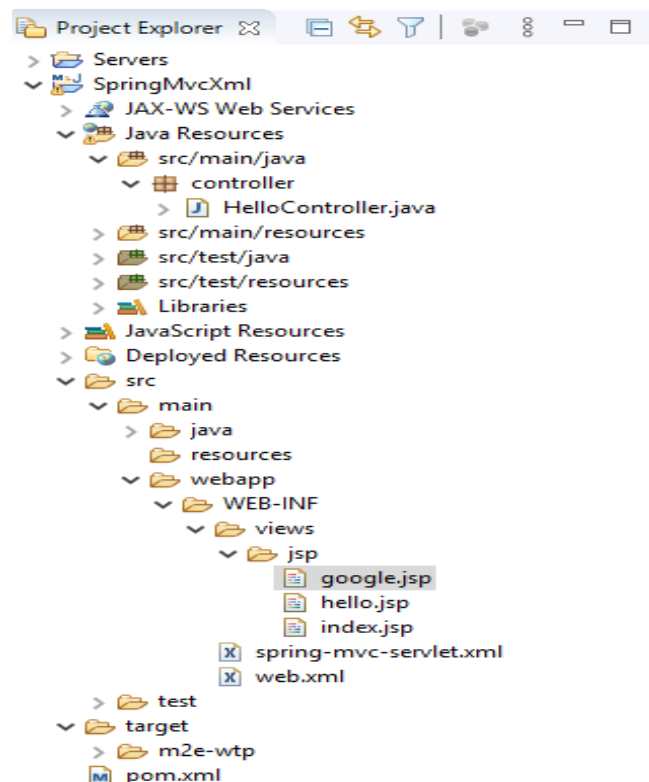
# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo File View**

```
google.jsp ✕  
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
2   pageEncoding="ISO-8859-1"%>  
3 <!DOCTYPE html>  
4 <html>  
5 <head>  
6   <meta charset="ISO-8859-1">  
7   <title>HELLO</title>  
8 </head>  
9 <body>  
10   <h1>Spring MVC GOOGLE!</h1>  
11   <h1>GOOGLE</h1>  
12   <a href="">Click sang link index</a>  
13  
14 </body>  
15 </html>
```

# Tạo Project Spring MVC

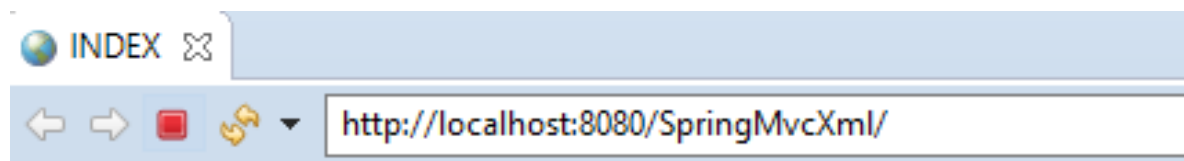
- Tạo Project SpringMVC sử dụng xml config
- **B2. Tạo File View**



# Tạo Project Spring MVC

- Tạo Project SpringMVC sử dụng xml config

Browser : Link “/”



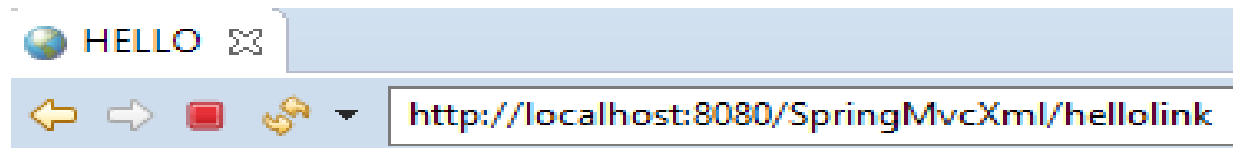
## Spring MVC JAVA WEB!

### INDEX

[Click sang link hello](#)

# Tạo Project Spring MVC

Tạo Project SpringMVC sử dụng xml config  
Browser : Link “hellolink”



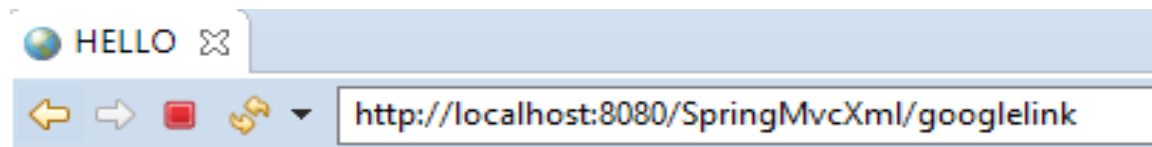
**Spring MVC HELLO!**

**HELLO**

[Click sang link google](#)

# Tạo Project Spring MVC

Tạo Project SpringMVC xử dụng xml config  
Browser : Link “googlelink”



**Spring MVC GOOGLE!**

**GOOGLE**

[Click sang link index](#)

# Anotation trong Spring

## Annotation @RequestMapping

được sử dụng để map request với class hoặc method xử lý request đó.

@RequestMapping có thể được áp dụng với controller class hoặc method trong controller class.

# Anotation trong Spring

## Annotation @RequestMapping

```
@Controller
@RequestMapping("/index")
public class BaseController {

    public String index() {
        return "index";
    }

}
```

```
@Controller
public class HomeController {

    @RequestMapping("/method0")
    public String method0() {
        return "page0";
    }

    @RequestMapping("/method1")
    public String method1() {
        return "page1";
    }

}
```



## Annotation @PathVariable

được sử dụng để xử lý những URI động, có một hoặc nhiều paramter bên trong URI.

```
@RequestMapping("/test1/{id}")
public String test1(@PathVariable("id") int id, Model model) {
    model.addAttribute("id", id);
    return "test1";
}

@RequestMapping("/test2/{id}/{name}")
public String test2(@PathVariable("id") int id, @PathVariable("name") String name, Model model) {
    model.addAttribute("id", id);
    model.addAttribute("name", name);
    return "test2";
}
```

# Anotation trong Spring

## Annotation @RequestParam

Khi submit method GET, trên URL **sẽ không chứa** các giá trị của các ô input được submit. Sử dụng Annotation RequestParam giúp chúng ta lấy được giá trị đó.

```
@RequestMapping("/test3")
public String test3(@RequestParam("name") String name, @RequestParam("id") int id, Model model) {
    model.addAttribute("id", id);
    model.addAttribute("name", name);
    return "test3";
}
```

# Anotation trong Spring

## Annotation @ResponseBody

Được thêm vào trước các method của các controller để chỉ dẫn rằng method này sẽ trả về text thay vì trả về view.

```
@RequestMapping("/test1")
@ResponseBody
public String test1() {
    return "test1";
}
```

# Anotation trong Spring

## Annotation @RestController

Tương đương với @Controller + @ResponseBody

Được dùng trước các class, các method trong class này sẽ trả về text thay vì trả về view.

```
@RestController
public class APIController {

    @RequestMapping("/test2")
    public String test2() {
        return "test1";
    }

}
```

# Anotation trong Spring

## Annotation @RestController

Tương đương với @Controller + @ResponseBody

Được dùng trước các class, các method trong class này sẽ trả về text thay vì trả về view.

```
@RestController
public class APIController {

    @RequestMapping("/test2")
    public String test2() {
        return "test1";
    }

}
```





T3H

*IT-Institute*

**THANK YOU**