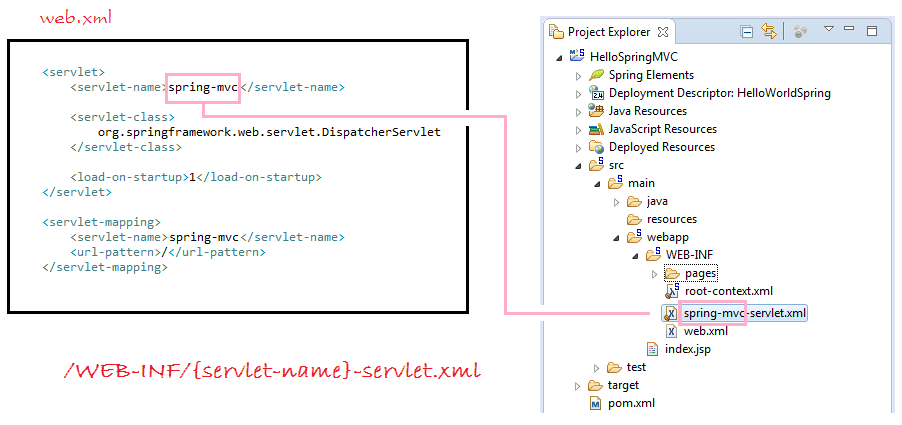
**2- 预览应用程序执行流程**

**Spring MVC** **DispatcherServlet**读取 xml 配置文件的原则：

* {servlet-name} ==> /WEB-INF/{servlet-name}-servlet.xml  
  

如果你不想用 SpringMVC 的使用原则，可以重新配置 SpringMVC  DispatcherServlet 在 web.xml 文件中：

<servlet>

<servlet-name>my-dispatcher-name</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<!-- override default name {servlet-name}-servlet.xml -->

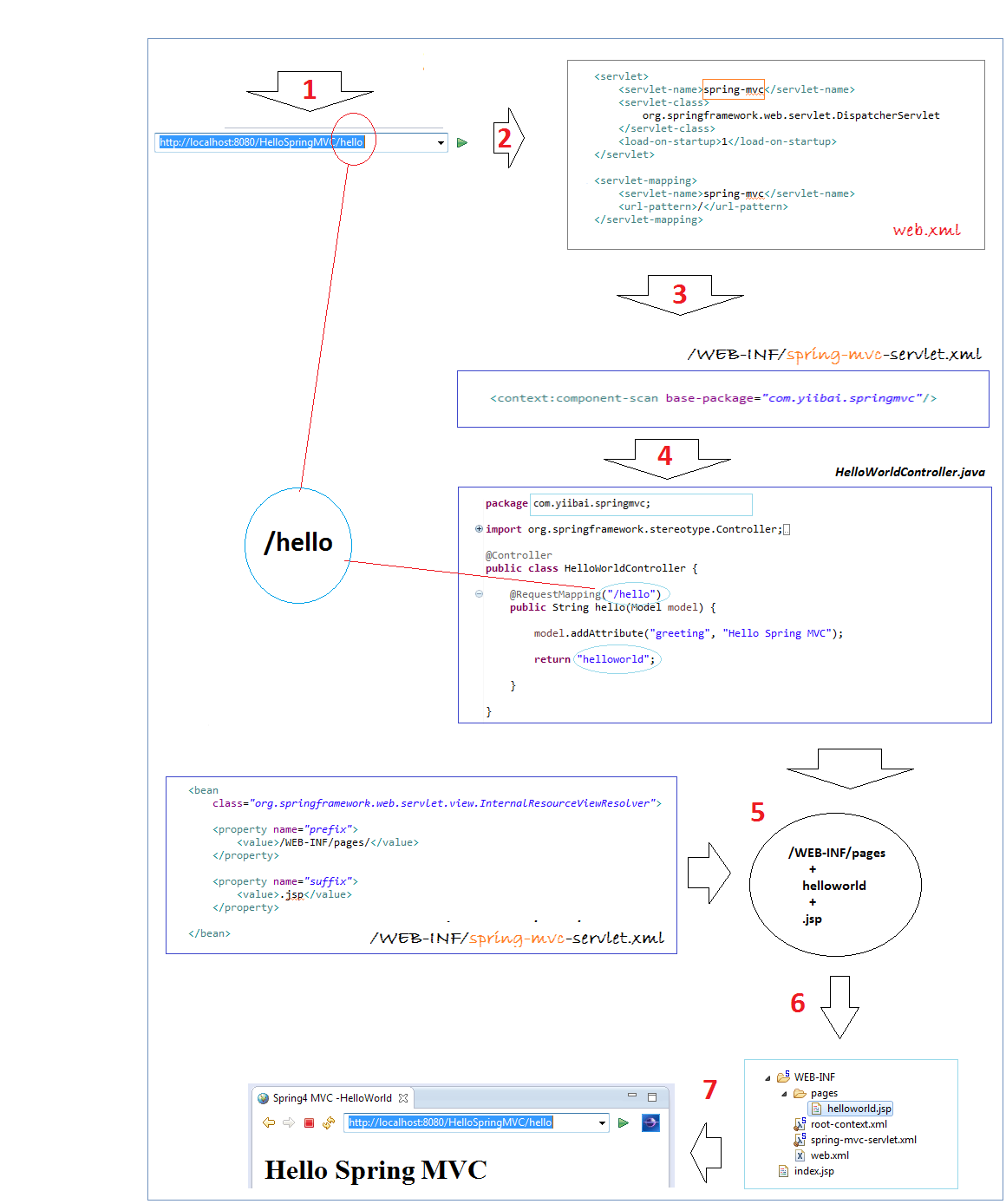
<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/springmvc-myconfig.xml</param-value>

</init-param>

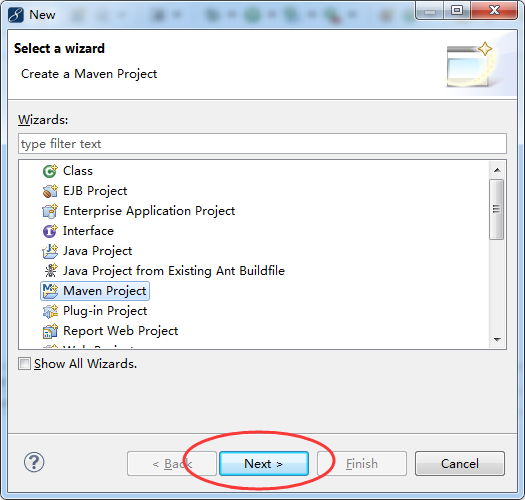
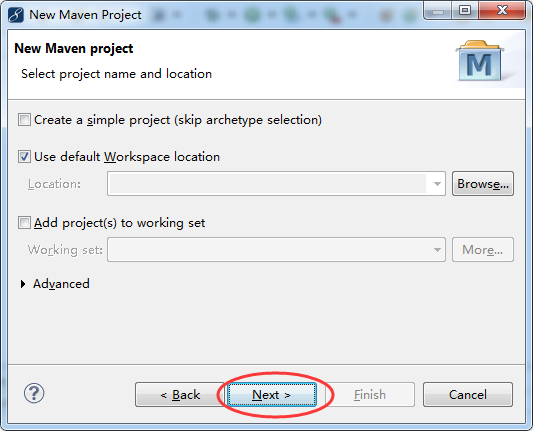
<load-on-startup>1</load-on-startup>

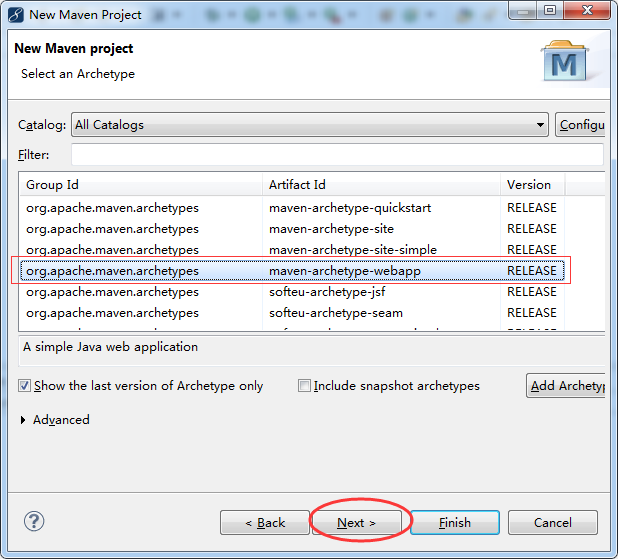
</servlet>

应用程序的流程：  


**3 - 创建Maven工程**

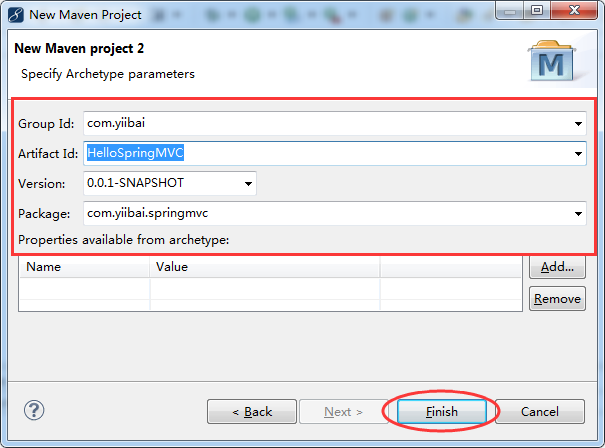
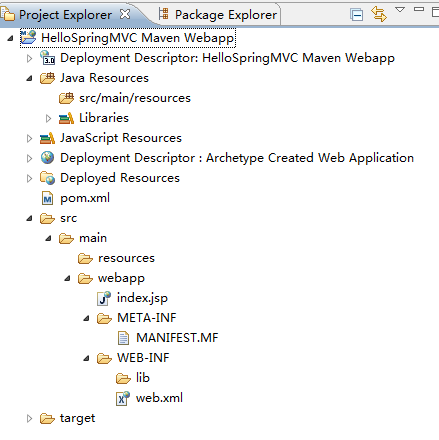
创建Maven项目类型。 Maven是帮助我们管理库的最好方式。

在 Eclipse, 选择 "File/New/Other..."  
  


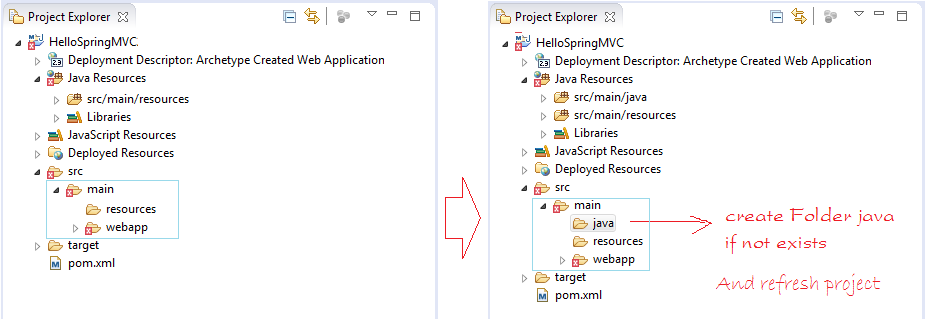
选择 archetype "maven-archetype-webapp"。  


输入：

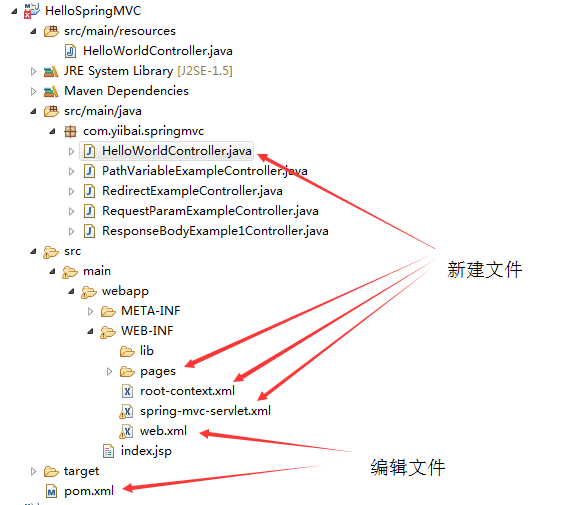
* Group Id: **com.yiibai**
* Artifact Id: **HelloSpringMVC**
* Package: **com.yiibai.springmvc**

  
这样将创建项目，结构如下图所示：  


不要担心项目在创建的时候出现错误信息。原因是，现在我们还没有声明 Servlet 库。

在 Eclipse 中创建 Maven 项目结构可能是错误的。需要我们去检查出来并完善。  


**4- 配置Spring**

这是项目建成后的文件结构图：  


配置 Maven 使用 Spring 库.

* pom.xml

<projectxmlns="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

http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.yiibai</groupId>

<artifactId>HelloSpringMVC</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>HelloSpringMVC Maven Webapp</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- Servlet Library -->

<!-- http://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

<!-- Spring dependencies -->

<!-- http://mvnrepository.com/artifact/org.springframework/spring-core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>4.1.4.RELEASE</version>

</dependency>

<!-- http://mvnrepository.com/artifact/org.springframework/spring-web -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.1.4.RELEASE</version>

</dependency>

<!-- http://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.1.4.RELEASE</version>

</dependency>

</dependencies>

<build>

<finalName>HelloSpringMVC</finalName>

<plugins>

<!-- Config: Maven Tomcat Plugin -->

<!-- http://mvnrepository.com/artifact/org.apache.tomcat.maven/tomcat7-maven-plugin -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<!-- Config: contextPath and Port (Default - /HelloSpringMVC : 8080) -->

<!--

<configuration>

<path>/</path>

<port>8899</port>

</configuration>

-->

</plugin>

</plugins>

</build>

</project>

配置 **web.xml：**

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaeehttp://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

id="WebApp\_ID" version="3.0">

<display-name>HelloWorldSpring</display-name>

<servlet>

<servlet-name>spring-mvc</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring-mvc</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<!-- Other XML Configuration -->

<!-- Load by Spring ContextLoaderListener -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/root-context.xml</param-value>

</context-param>

<!-- Spring ContextLoaderListener -->

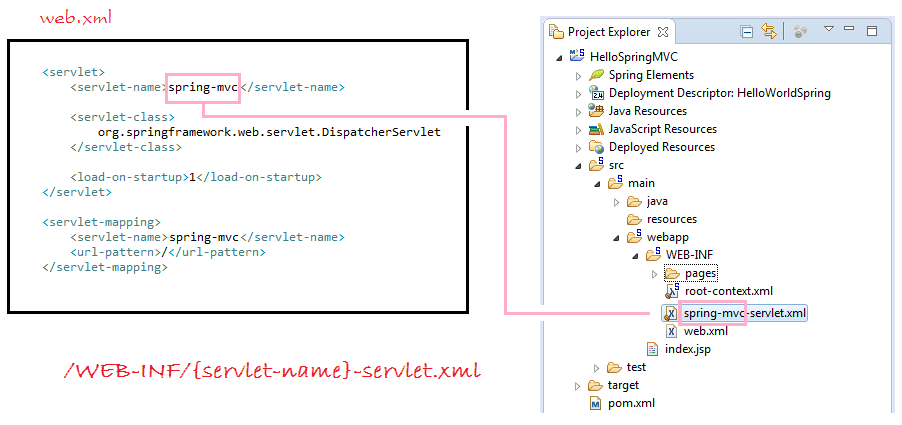
<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

</web-app>

**Spring MVC 的 DispatcherServlet将根据原则读取XML配置文件：**

* {servlet-name} ==> /WEB-INF/{servlet-name}-servlet.xml  
  
* ***spring-mvc-servlet.xml***

<?xml version="1.0"encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.1.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-4.1.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">

<context:component-scan base-package="com.yiibai.tutorial.springmvc"/>

<context:annotation-config/>

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix">

<value>/WEB-INF/pages/</value>

</property>

<property name="suffix">

<value>.jsp</value>

</property>

</bean>

</beans>

**注**:  
在Spring应用程序 ContextLoaderListener 将读取其他 XML 配置文件(如下的 abc.xml 和 root-context.xml 两个文件)。 可能不需要配置 ContextLoaderListener，如果你的应用程序并不需要读取其他XML配置文件。

<!-- web.xml -->

<!-- Spring ContextLoaderListener -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- Load by Spring ContextLoaderListener -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/root-context.xml,

/WEB-INF/abc.xml

</param-value>

</context-param>

* **/WEB-INF/root-context.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Empty -->

</beans>

* **HelloWorldController.java**

package com.yiibai.springmvc;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller

public class HelloWorldController {

@RequestMapping("/hello")

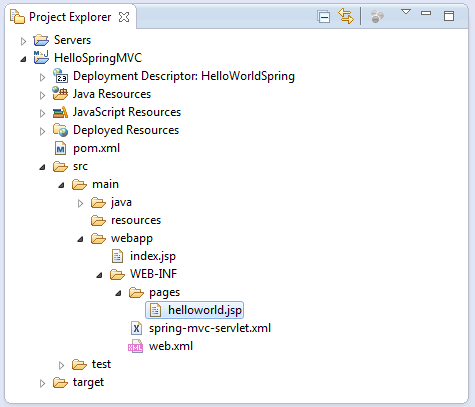
public String hello(Model model) {

model.addAttribute("greeting", "Hello Spring MVC");

return"helloworld";

}

}



* ***helloworld.jsp***

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Spring4 MVC -HelloWorld</title>

</head>

<body>

<h1>${greeting}</h1>

</body>

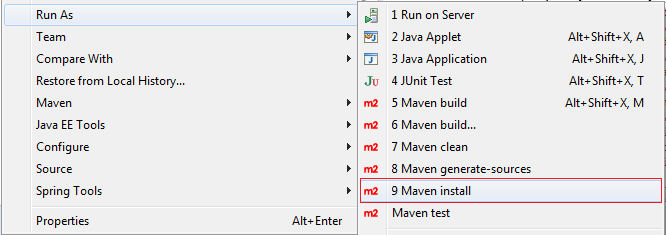
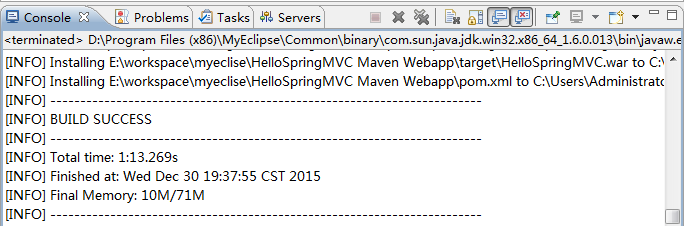
</html>

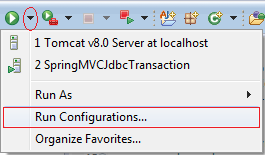
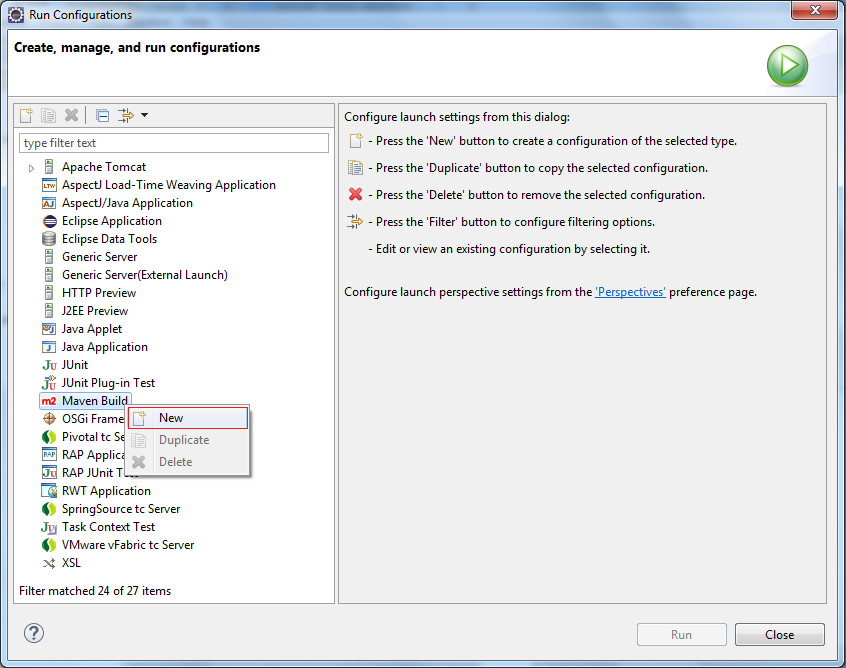
**5- 运行Spring应用程序**

首先，运行应用程序之前，需要构建整个项目。

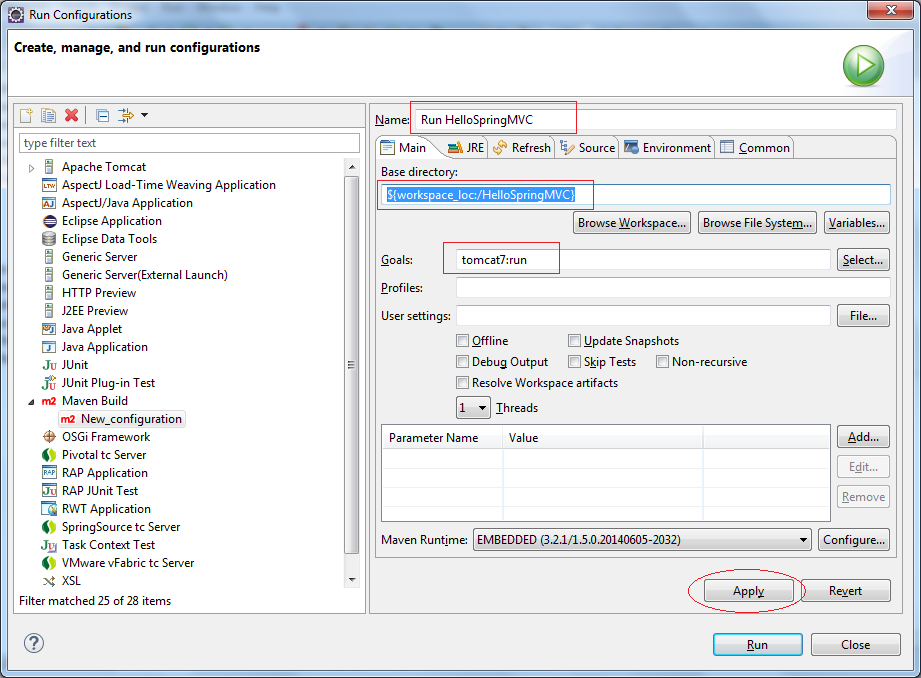
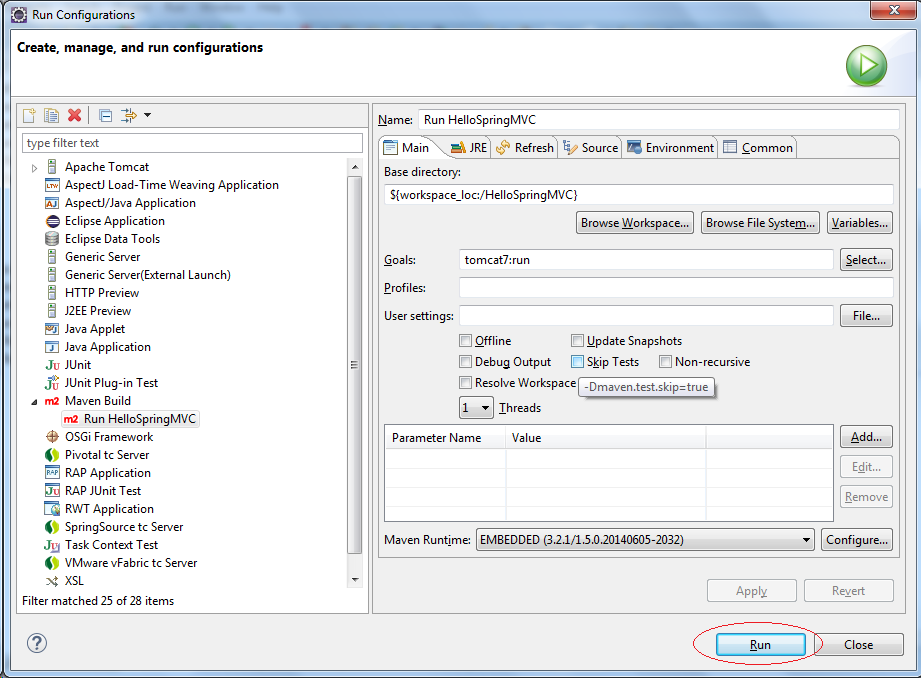
右键单击该项目并选择：

* Run As/Maven install

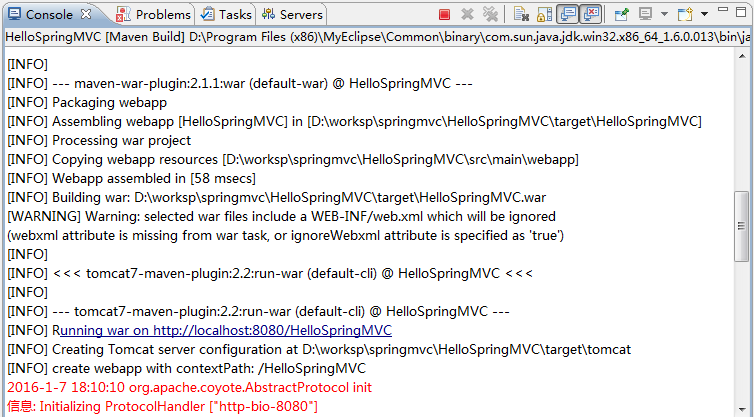
**  
  
运行配置：**

  
  
输入:

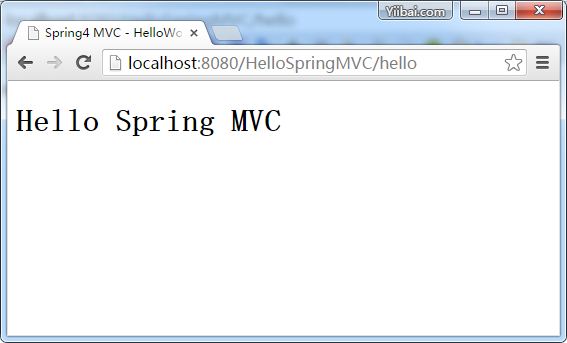
* Name: Run HelloSpringMVC
* Base directory: ${workspace\_loc:/HelloSpringMVC} =>${workspace\_loc:/HelloSpringMVC Maven Webapp}
* Goals: tomcat7:run

  
点击Run:  


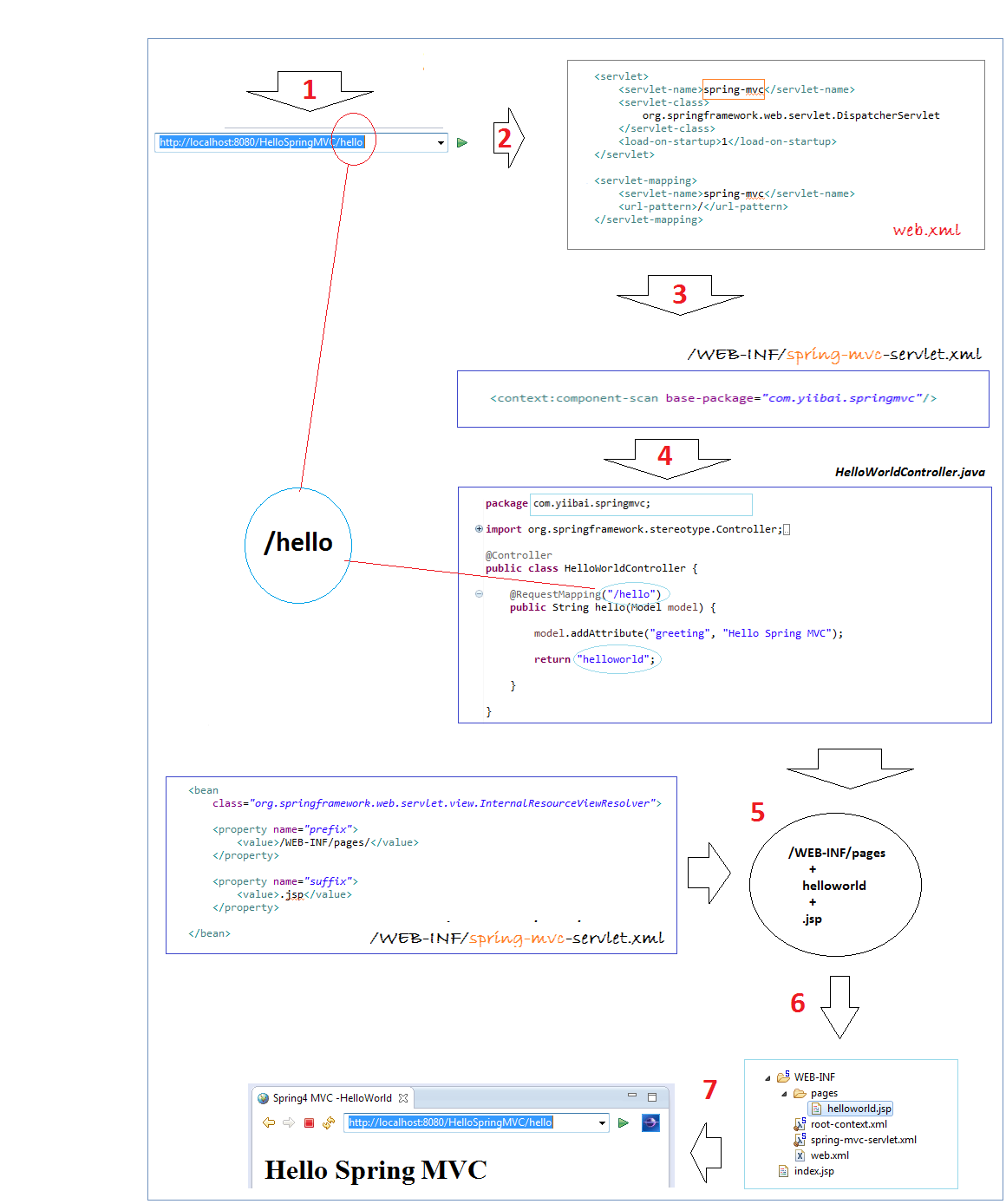
第一次运行该程序将需要几分钟(看你的网速)，它需要下载 Tomcat 插件库才能运行。

一切准备就绪：  


运行URL，如下图：

* <http://localhost:8080/HelloSpringMVC/hello>  
  

**6 - 应用程序的流程**

完成您的项目后，并成功地在上一步中运行。现在，我们来看一看程序的运行方式。  


**7- 控制器和方法**

**7.1- HttpServletRequest & HttpServletResponse**

可以使用 HttpServletRequest, HttpServletResponse 在控制器的方法中。

* OtherExampleController.java

package com.yiibai.springmvc;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class OtherExampleController {

......

@RequestMapping("/somePath")

public String requestResponseExample(HttpServletRequest request,

HttpServletResponse reponses, Model model) {

// Todo something here

return "someView";

}

......

}

**7.2- 控制器中的重定向**

使用前缀 "redirect:" ，该方法返回字符串，可以重定向到另一页面。参见图：

* RedirectExampleController.java

package com.yiibai.springmvc;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class RedirectExampleController {

@RequestMapping(value = "/redirect", method = RequestMethod.GET)

public String authorInfo(Model model) {

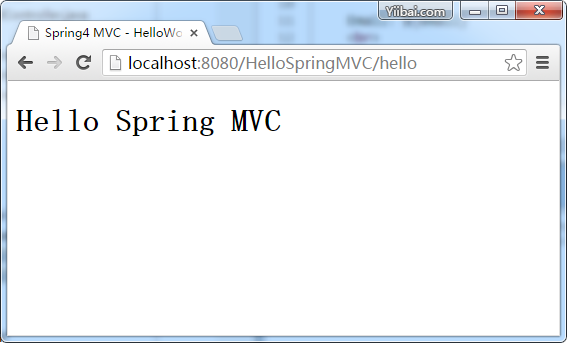
// Do somethong here

return "redirect:/hello";

}

}

运行URL:

* [http://localhost:8080/HelloSpringMVC/redirect](http://localhost:8080/HelloSpringMVC/redirect" \t "_blank)  
  

**7.3- @RequestParam示例**

使用@RequestParam 注解将请求参数绑定到你的控制器方法参数。

下面的代码片段显示了这个用法：

* RequestParamExampleController.java

package com.yiibai.springmvc;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class RequestParamExampleController {

@RequestMapping("/user")

public String userInfo(Model model,

@RequestParam(value = "name", defaultValue = "Guest") String name) {

model.addAttribute("name", name);

if("admin".equals(name)) {

model.addAttribute("email", "admin@yiibai.com");

} else{

model.addAttribute("email", "Not set");

}

return "userInfo";

}

}

* **/WEB-INF/pages/userInfo.jsp**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>User Info</title>

</head>

<body>

<h2>${name}</h2>

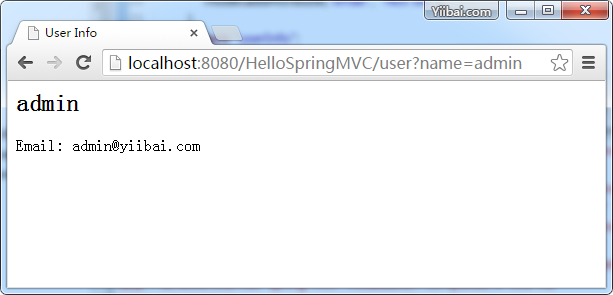
Email: ${email}

<br>

</body>

</html>

运行 URL:

* [http://localhost:8080/HelloSpringMVC/user?name=admin](http://localhost:8080/HelloSpringMVC/user?name=admin" \t "_blank)  
  

**7.4- @PathVariable示例**

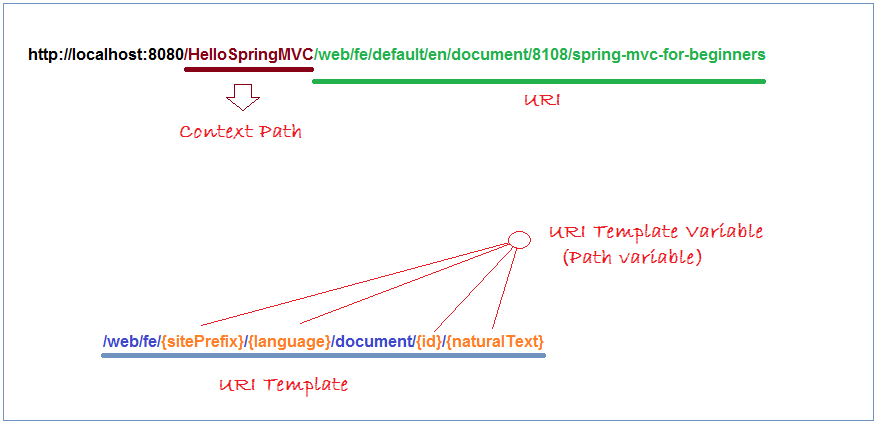
在Spring MVC中，可以使用@PathVariable注释将一个方法参数绑定到一个URI模板变量的值：

例如，这是一个模板的URI：

* /web/fe/{sitePrefix}/{language}/document/{id}/{naturalText}

而下面的 URI 模板匹配上面：

1. /web/fe/default/en/document/8108/spring-mvc-for-beginners
2. /web/fe/default/vi/document/8108/spring-mvc-cho-nguoi-moi-bat-dau
3. .....

  
下面的代码片段显示了用法：

* PathVariableExampleController.java

package com.yiibai.springmvc;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller

public class PathVariableExampleController {

/\*\*

\* @PathVariable Example:

\*

\*/

@RequestMapping("/web/fe/{sitePrefix}/{language}/document/{id}/{naturalText}")

public String documentView(Model model,

@PathVariable(value = "sitePrefix") String sitePrefix,

@PathVariable(value = "language") String language,

@PathVariable(value = "id") Long id,

@PathVariable(value = "naturalText") String naturalText) {

model.addAttribute("sitePrefix", sitePrefix);

model.addAttribute("language", language);

model.addAttribute("id", id);

model.addAttribute("naturalText", naturalText);

String documentName = "Java tutorial for Beginners";

if(id == 8108) {

documentName = "Spring MVC for Beginners";

}

model.addAttribute("documentName", documentName);

return "documentView";

}

}

* /WEB-INF/pages/documentView.jsp

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>${documentName}</title>

</head>

<body>

<h3>${documentName}</h3>

Site Prefix: ${sitePrefix}

<br> Language: ${language}

<br> ID: ${id}

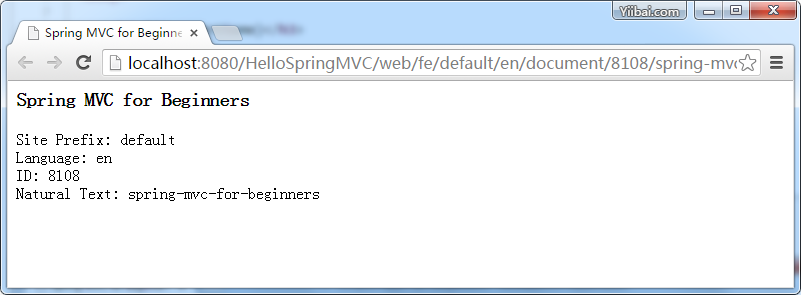
<br> Natural Text: ${naturalText}

<br>

</body>

</html>

运行 URL:

* [http://localhost:8080/HelloSpringMVC/web/fe/default/en/document/8108/spring-mvc-for-beginners](http://localhost:8080/HelloSpringMVC/web/fe/default/en/document/8108/spring-mvc-for-beginners" \t "_blank)  
  

**7.5- @ResponseBody示例**

如果您使用 @ResponseBody 注释到方法， spring 将尝试转换它的返回值，并自动写入到HTTP响应。在这种情况下，并不需要一个特定的视图。

*注：方法不一定需要返回字符串类型。*

**使用@ResponseBody和方法返回字符串的简单例子。**

* ResponseBodyExample1Controller.java

package com.yiibai.springmvc;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

@Controller

public class ResponseBodyExample1Controller {

// Simple example, method returns String.

@RequestMapping(value = "/saveResult")

@ResponseBody

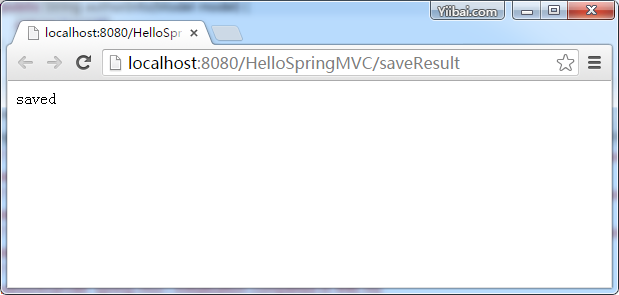
publicString authorInfo(Model model) {

return "saved";

}

}

运行示例的结果：

* [http://localhost:8080/HelloSpringMVC/saveResult](http://localhost:8080/HelloSpringMVC/saveResult" \t "_blank)  
  

使用@ResponseBody 的一个例子，方法返回一个对象。