在这一节中，我们以 Spring4 MVC HelloWorld 注释/JavaConfig为示例，一步一步以简单的方式学习Spring4 MVC 的注解，项目设置，代码，部署和运行。

在先前的 [Spring MVC 4 Hello World XML教程示例中](http://www.yiibai.com/spring_mvc/spring-mvc-tutorial-for-beginners.html), 我们已经使用XML配置开发了一个Hello World Web应用程序。但是，XML不是配置Spring应用程序的唯一途径。或者，我们可以使用Java配置来配置应用程序。

如果回头看看之前的教程，你会发现我们已经使用XML配置在两个地方。第一个是 spring-servlet.xml 在这里我们定义的视图解析程序识别真正的视图，位置搜索，通过组件扫描Bean。第二个是 web.xml, 我们定义前端控制器配置和URL模式将被寻找匹配。

在本教程中，我们将再次创建一个Hello world的例子，但这个时候我们使用Java配置。 我们将删除上面提到的XML文件，并通过它们对应的Java替换这些XML配置。

以下技术堆栈需要使用到：

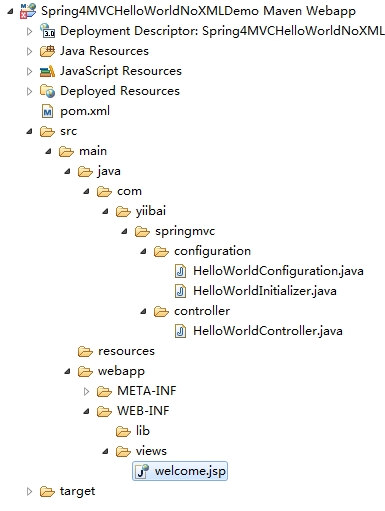
* Spring 4.0.6.RELEASE
* Maven 3
* JDK 1.6
* Tomcat 8.0.21
* Eclipse JUNO Service Release 2

现在我们开始！

**第1步：创建所需的目录结构项目**

文章 [使用Eclipse创建一个Maven web工程](http://www.yiibai.com/maven/create-a-maven-web-project-with-eclipse.html) 包含使用Eclipse一步一步的向导来创建一个Maven项目。

以下将是最后的项目结构。



现在，让我们来添加/更新上面讨论项目结构中每一个细节提到的内容。

**第2步：使用Spring和Servlet依赖更新pom.xml**

我们要讨论以 Spring Java为基础的配置取决于Servlet 3.0 的API， 因此，我们需要包含的依赖在 pom.xml 中。

<?xml version="1.0"?>

<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"

xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<modelVersion>4.0.0</modelVersion>

<groupId>com.yiibai.springmvc</groupId>

<artifactId>Spring4MVCHelloWorldNoXMLDemo</artifactId>

<packaging>war</packaging>

<version>1.0.0</version>

<name>Spring4MVCHelloWorldNoXMLDemo</name>

<properties>

<springframework.version>4.0.6.RELEASE</springframework.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${springframework.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

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

<version>3.1.0</version>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

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

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

<build>

<pluginManagement>

<plugins>

<plugin>

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

<artifactId>maven-war-plugin</artifactId>

<version>2.4</version>

<configuration>

<warSourceDirectory>src/main/webapp</warSourceDirectory>

<warName>Spring4MVCHelloWorldNoXMLDemo</warName>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</pluginManagement>

<finalName>Spring4MVCHelloWorldNoXMLDemo</finalName>

</build>

</project>

首先要注意这里maven-war-plugin 插件的声明。正如我们将完全删除web.xml ，我们需要配置这个插件，以避免Maven构建war包失败。第二个变化是加入了JSP/Servlet/Jstl 的依赖关系，这些我们可能需要，因为我们将要使用 servlet API和JSTL视图在我们的代码中。在一般情况下，容器已经包含这些库，从而在pom.xml中为他们提供了，我们可以设置作用范围。

**第3步：添加控制器**

在src/main/java下添加一个控制器类，如下所示：

com.yiibai.springmvc.controller.HelloWorldController

package com.yiibai.springmvc.controller;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

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

@Controller

@RequestMapping("/")

public class HelloWorldController {

@RequestMapping(method = RequestMethod.GET)

public String sayHello(ModelMap model) {

model.addAttribute("greeting", "Hello World from Spring 4 MVC");

return "welcome";

}

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

public String sayHelloAgain(ModelMap model) {

model.addAttribute("greeting", "Hello World Again, from Spring 4 MVC");

return "welcome";

}

}

在类名@Controller注解声明这个类的Spring bean 以及 @RequestMapping注解声明了这个类是默认处理程序键入“/”的所有请求。第一种方法没有声明因此任何映射，它将继承映射的映射声明是在类级别上，默认处理GET请求。方法二(由于额外的映射声明使用value属性)形式 /hello 将再次请求。属性方法说哪种类型的HTTP请求这种方法可以服务。

方法说哪种类型的HTTP请求这种方法可以服务。 ModelMap是一个Map实现，在这里作为替代[request.getAttribute()/request.setAttribute()] 设定值作为请求属性。请注意，我们从这个方法返回“welcome”字符串。此字符串将后缀和前缀后缀，在视图解析器定义的前缀(见上面的 spring-servlet.xml)，形成真正的视图文件名。

**第4步：添加视图**

创建一个新的文件夹命名为views在WEB-INF目录下，并添加一个简单的JSP页面welcome.jsp (WEB-INF/views/welcome.jsp)从控制器到简单的访问模式值。

<%@ 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>HelloWorld page</title>

</head>

<body>

Greeting : ${greeting}

</body>

</html>

**第5步：添加配置类**

在src/main/java下添加下面提到的类指定的包，如下图所示。这种构造类可以被看作是一个替代 spring-servlet.xml，因为它包含了所有必需的组件的扫描和视图解析器的信息。

com.yiibai.springmvc.configuration.HelloWorldConfiguration

package com.yiibai.springmvc.configuration;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.ViewResolver;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.view.InternalResourceViewResolver;

import org.springframework.web.servlet.view.JstlView;

@Configuration

@EnableWebMvc

@ComponentScan(basePackages = "com.yiibai.springmvc")

public class HelloWorldConfiguration {

@Bean

public ViewResolver viewResolver() {

InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();

viewResolver.setViewClass(JstlView.class);

viewResolver.setPrefix("/WEB-INF/views/");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

}

@Configuration指明该类包含注解为@Bean 生产 bean管理是由Spring容器的一个或多个bean方法。 以上配置类对应等同于以下XML：

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

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

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

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-4.0.xsd

http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd">

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

<mvc:annotation-driven />

<bean

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

<property name="prefix">

<value>/WEB-INF/views/</value>

</property>

<property name="suffix">

<value>.jsp</value>

</property>

</bean>

</beans>

@EnableWebMvc 等同于 mvc:annotation-driven 在XML中. 它能够为使用@RequestMapping向特定的方法传入的请求映射@Controller-annotated 类。

@ComponentScan 等同于 context:component-scan base-package="..." 提供 spring 在哪里寻找 管理 beans/classes.

**第6步：添加初始化类**

添加一个初始化类实现 WebApplicationInitializer 在src/main/java 中使用如下图所示指定包(在这种情况下，作为替代在 web.xml 中定义的任何 Spring 配置)。在Servlet 3.0的容器启动时，这个类将被加载并初始化，并在启动由servlet容器调用方法。

com.yiibai.springmvc.configuration.HelloWorldInitializer

package com.yiibai.springmvc.configuration;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.ServletRegistration;

import org.springframework.web.WebApplicationInitializer;

import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;

import org.springframework.web.servlet.DispatcherServlet;

public class HelloWorldInitializer implements WebApplicationInitializer {

public void onStartup(ServletContext container) throws ServletException {

AnnotationConfigWebApplicationContext ctx = new AnnotationConfigWebApplicationContext();

ctx.register(HelloWorldConfiguration.class);

ctx.setServletContext(container);

ServletRegistration.Dynamic servlet = container.addServlet("dispatcher", new DispatcherServlet(ctx));

servlet.setLoadOnStartup(1);

servlet.addMapping("/");

}

}

内容上面类似 web.xml 在之前的教程中的内容，因为我们使用的是前端控制器 DispatcherServlet，分配映射(URL模式的XML)和而不是提供给 Spring 配置文件(spring-servlet.xml)的路径，在这里，我们正在注册的配置类。总体而言，我们都在做同样的事情，只是方式有所不同。

更新：请注意，现在你可以更简洁写上面的类[和它的最佳方法] 来扩展 AbstractAnnotationConfigDispatcherServletInitializer 类，如下所示：

package com.yiibai.springmvc.configuration;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class HelloWorldInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

@Override

protected Class<?>[] getRootConfigClasses() {

return new Class[] { HelloWorldConfiguration.class };

}

@Override

protected Class<?>[] getServletConfigClasses() {

return null;

}

@Override

protected String[] getServletMappings() {

return new String[] { "/" };

}

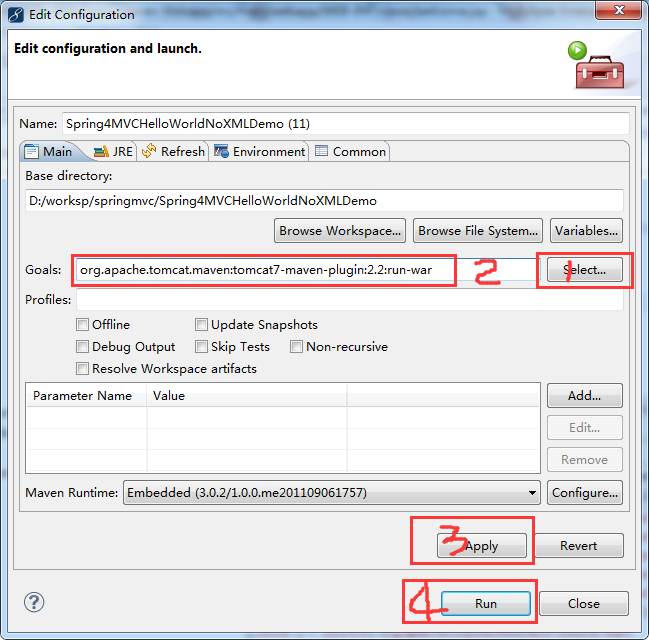
}

**第7步：构建和部署应用程序**

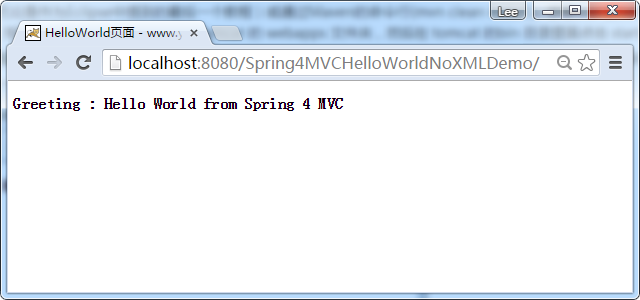
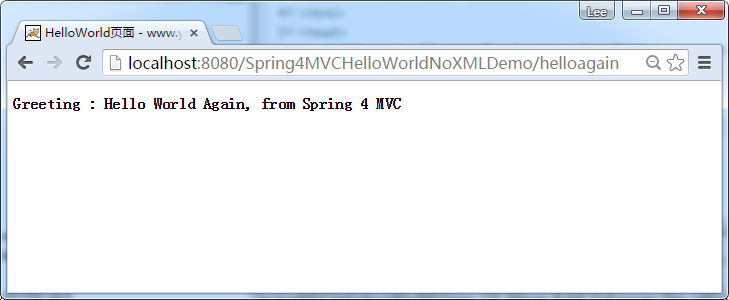
有一点要记住，像WebApplicationInitializer，Spring 是基于Java 的配置API依赖于 Servlet3.0容器。确保你没有使用Servlet声明任何小于3.0。对于我们的情况，我们将从应用程序中删除 web.xml 文件。

现在构建war (无论是作为Eclipse中提到的最后一个教程)或通过Maven的命令行(mvn clean install)。部署war 到Servlet3.0容器。由于我在这里使用Tomcat，我就干脆把这个 war 文件放到 Tomcat 的 webapps 文件夹，然后在 tomcat 的bin 目录里面点击 start.bat 运行。

或者 右键工程 =>Run As => Maven install 完成后，再次 右键工程 =>Run As => Maven build，弹出选择：



运行应用程序，现在我们来访问URL: http://localhost:8080/Spring4MVCHelloWorldNoXMLDemo，显示结果如下所示：

就这样，完成！