WebSystique

learn together

Spring 4 MVC HelloWorld Tutorial – Annotation/JavaConfig Example

August 3, 2014 websystiqueadmin

Spring MVC 4 HelloWorld Annotation/JavaConfig Example, step-by-step learning Spring MVC 4 annotations, project setup, code, deploy & Run, in simple way.

Other interesting posts you may like

- Spring 4 Caching Annotations Tutorial
- Spring 4 Cache Tutorial with EhCache
- Spring 4 MVC+JPA2+Hibernate Many-to-many Example
- Spring 4 Email Template Library Example
- Spring 4 Email With Attachment Tutorial
- Spring 4 Email Integration Tutorial
- Spring MVC 4+JMS+ActiveMQ Integration Example
- Spring 4+JMS+ActiveMQ @JmsLister @EnableJms Example
- Spring 4+JMS+ActiveMQ Integration Example
- Spring MVC 4+Apache Tiles 3 Integration Example
- Spring MVC 4+Spring Security 4 + Hibernate Integration Example
- Spring MVC 4+AngularJS Example
- Spring MVC 4+AngularJS Routing with UI-Router Example
- Spring 4 MVC HelloWorld – XML Example

WebSystiqu

G+ Follow

+ 234

Recent Posts

- Spring 4 Caching Annotations Tutorial
- Spring 4 Cache Tutorial with EhCache
- Spring 4 MVC+JPA2+Hibernate Many-to-many-Example
- Spring 4 Email using Velocity,Freemarker Template library
- Spring 4 Email With Attachment Tutorial

- Spring MVC 4+Hibernate 4+MySQL+Maven integration example
- Spring Security 4 Hello World Annotation+XML Example
- Hibernate MySQL Maven Hello World Example (Annotation)
- TestNG Hello World Example
- JAXB2 Helloworld Example
- Spring Batch- Read a CSV file and write to an XML file

In the previous Spring MVC 4 Hello World tutorial-XML example, we have developed a Hello world web application using XML configuration. However, XML is not the only way to configure spring application. Alternatively, we can configure the application using Java configuration.

If you look back on previous post, you will find that there are mainly two places where we have used XML configuration. First, in **spring-servlet.xml** where we have defined a view-resolver for identifying the real view , and location to search for beans via component-scanning. Second, in **web.xml**, we have defined the front-controller configuration and the url pattern it will be looking on.

In this tutorial, we will again create a Hello world example but using Java configuration this time. We will **REMOVE** both of above mentioned xml files and replace these xml configurations via their java counterparts.

Following technologies stack being used:

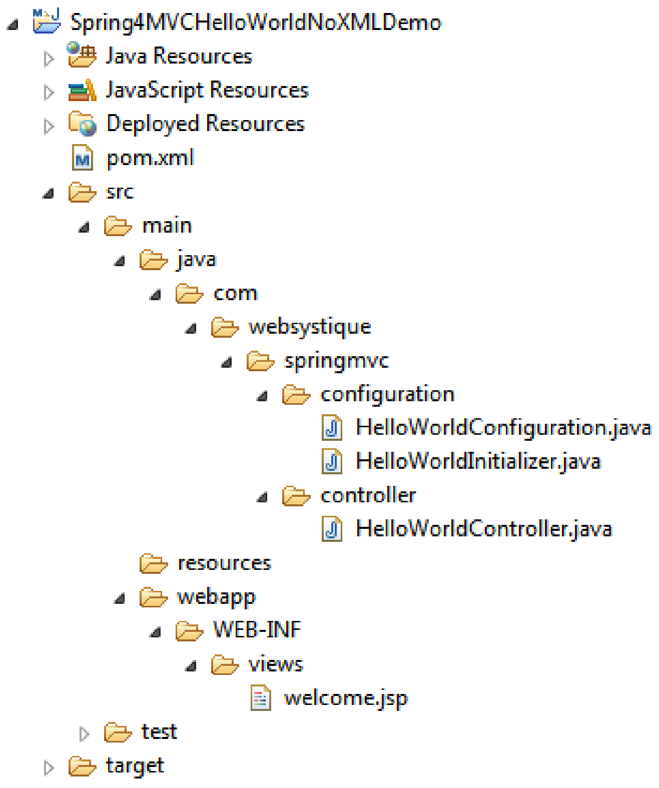
- Spring 4.0.6.RELEASE
- Maven 3
- JDK 1.6
- Tomcat 8.0.21
- Eclipse JUNO Service Release 2

Let’s begin.

Step 1: Create a project with required directory structure

Post Creating a maven web project with eclipse contains step-by-step instruction to create a `maven` project with eclipse.

Following will be the final Project structure.



```
Spring4MVCHelloWorldNoXMLDemo
    Java Resources
    JavaScript Resources
    Deployed Resources
    pom.xml
    src
        main
            java
                com
                    websystique
                        springmvc
                            configuration
                                HelloWorldConfiguration.java
                                HelloWorldInitializer.java
                            controller
                                HelloWorldController.java
            resources
            webapp
                WEB-INF
                    views
                        welcome.jsp
        test
    target
```

Now let’s add/update the content mentioned in above project structure discussing each in detail.

Step 2: Update pom.xml with Spring and Servlet dependency

The Spring java-based configuration we are going to discuss depends on Servlet 3.0 api, so we need to include that as a dependency in `pom.xml`

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://mav
    xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.or

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.websystique.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-compiler-plugin</artifactId>
                    <version>3.2</version>
                    <configuration>
                        <source>1.6</source>
                        <target>1.6</target>
                    </configuration>
                </plugin>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-war-plugin</artifactId>
                    <version>2.4</version>
                    <configuration>
                        <warSourceDirectory>src/main/webapp</warSourceDir
                        <warName>Spring4MVCHelloWorldNoXMLDemo</warName>
                        <failOnMissingWebXml>false</failOnMissingWebXml>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
        <finalName>Spring4MVCHelloWorldNoXMLDemo</finalName>
    </build>
</project>
```

First thing to notice here is the `maven-war-plugin` declaration. As we will be completely removing web.xml, we will need to configure this plugin in order to avoid maven failure to build war package. Second change is the inclusion of JSP/Servlet/Jstl dependencies which we might be needing as we are going to use servlet api's and jstl view in our code.In general, containers already contains these libraries, so we can set the scope as provided for them in pom.xml.

Additionally, `maven-compiler-plugin` has been added here to explicitly specify the jdk-version we are going to use. Do note that it also forces eclipse to respect the jdk-version being used for the project. if it is not present, and you perform mvn-update from within your eclipse, eclipse switches jdk-version back to default jdk-version [1.5] which is annoying. So do add it in your project pom as well.

Step 3: Add Controller

Add a controller class under `src/main/java` as shown below:

`com.websystique.springmvc.controller.HelloWorldController`

```
package com.websystique.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
        return "welcome";
    }

}
```

`@Controller` annotation on class name declares this class as spring bean and `@RequestMapping` annotation declares that this class is default handler for all requests of type '/'. First method does not have any mapping declared so, it will inherit the mapping from mapping declared on class level, acting as default handler for GET requests. Second method (due to additional mapping declaration with `value` attribute) will serve the request of form /helloagain. Attribute `method` says which type of HTTP request this method can serve. `ModelMap` is a Map implementation, which here acting as replacement of [request.getAttribute()/request.setAttribute()]setting values as request attribute. Note that we are returning "welcome" string form this method. This string will be suffixed and prefixed with suffix and prefix defined in `view resolver`(see spring-servlet.xml above) to form the real view file name.

Step 4: Add View

Create a new folder named views under WEB-INF and add in a Simple JSP page welcome.jsp (`WEB-INF/views/welcome.jsp`) to simply access the model value sent from controller.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://ww
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>HelloWorld page</title>
</head>
<body>
    Greeting : ${greeting}
</body>
</html>
```

Step 5: Add Configuration Class

Add the below mentioned class under `src/main/java` with specified package as shown below. This configuration class can be treated as a replacement of spring-servlet.xml as it contains all the information required for component-scanning and view resolver.

`com.websystique.springmvc.configuration.HelloWorldConfiguration`

```
package com.websystique.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.websystique.springmvc")
public class HelloWorldConfiguration {
    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceV
        viewResolver.setViewClass(JstlView.class);
        viewResolver.setPrefix("/WEB-INF/views/");
        viewResolver.setSuffix(".jsp");

        return viewResolver;
    }

}
```

`@Configuration` indicates that this class contains one or more bean methods annotated with `@Bean` producing bean manageable by spring container. Above Configuration class is equivalent to following XML counterpart:

```
<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:
    http://www.springframework.org/schema/mvc http://www.springframework.
    http://www.springframework.org/schema/context http://www.springframew

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

    <mvc:annotation-driven />

    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewR
        <property name="prefix">
            <value>/WEB-INF/views/</value>
        </property>
        <property name="suffix">
```

```
            <value>.jsp</value>
        </property>
    </bean>

</beans>
```

`@EnableWebMvc` is equivalent to `mvc:annotation-driven` in XML. It enables support for @Controller-annotated classes that use `@RequestMapping` to map incoming requests to specific method.

`@ComponentScan` is equivalent to `context:component-scan base-package="..."` providing with where to look for spring managed beans/classes.

Step 6: Add Initialization class

Add an initializer class implementing `WebApplicationInitializer` under src/main/java with specified package as shown below(which in this case acts as replacement of any spring configuration defined in web.xml). During Servlet 3.0 Container startup, this class will be loaded and instantiated and its onStartup method will be called by servlet container.

`com.websystique.springmvc.configuration.HelloWorldInitializer`

```
package com.websystique.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.AnnotationConfigWebApplica
import org.springframework.web.servlet.DispatcherServlet;

public class HelloWorldInitializer implements WebApplicationInitializer {

    public void onStartup(ServletContext container) throws ServletExcepti

        AnnotationConfigWebApplicationContext ctx = new AnnotationConfigW
        ctx.register(HelloWorldConfiguration.class);
        ctx.setServletContext(container);

        ServletRegistration.Dynamic servlet = container.addServlet("dispa

        servlet.setLoadOnStartup(1);
        servlet.addMapping("/");
    }

}
```

The content above resembles the content of web.xml from previous tutorial as we are using the front-controller `DispatherServler`, assigning the mapping (url-pattern in xml) and instead of providing the path to spring configuration file(spring-servlet.xml) , here we are registering the Configuration Class. Overall, we are doing the same thing, just the approach is different.

UPDATE: Note that now you can write the above class even more concisely [**and it's the preferred way**], by extending `AbstractAnnotationConfigDispatcherServletInitializer` base class, as shown below:

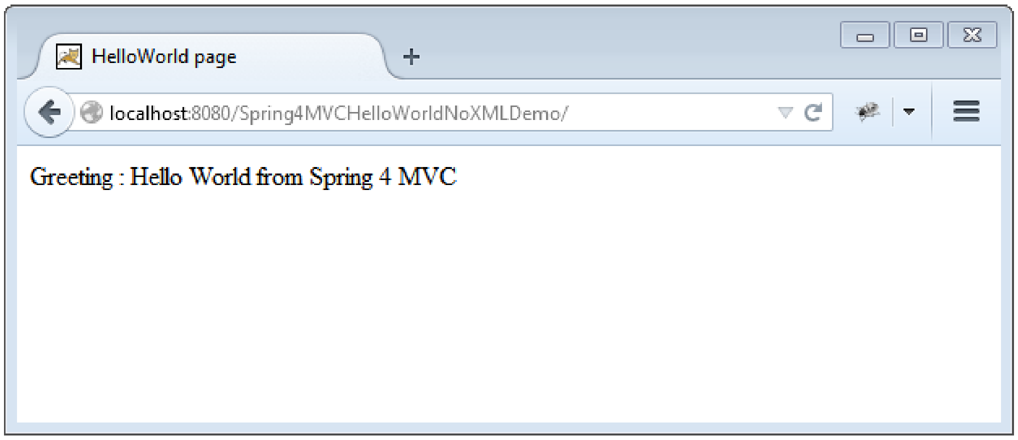
```
package com.websystique.springmvc.configuration;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDi

public class HelloWorldInitializer extends AbstractAnnotationConfigDispat

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] { HelloWorldConfiguration.class };
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return null;
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }

}
```

Step 7: Build and Deploy the application

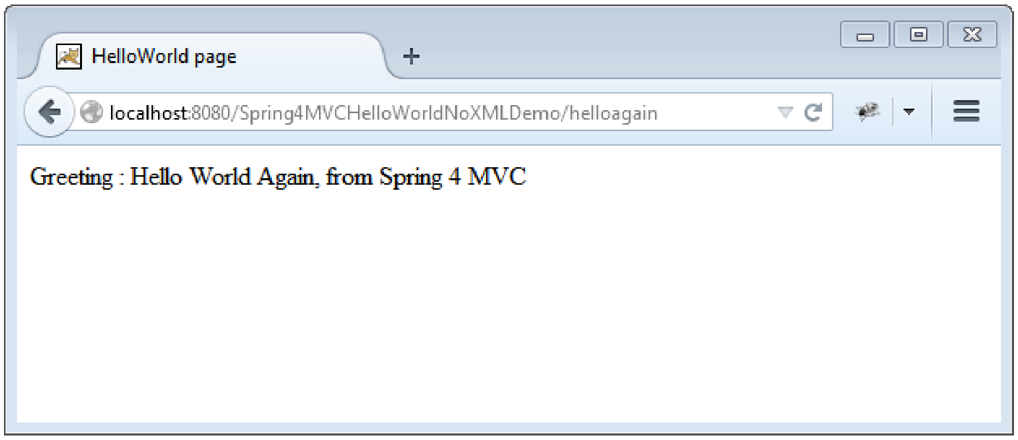
One thing to keep in mind that the Spring java based configuration api's like WebApplicationInitializer depends on `Servlet 3.0 containers`.So make sure you don't have any web.xml with servlet declaration less than 3.0. For our case, we have removed web.xml file from our application.

Now build the war (either by eclipse as was mentioned in last tutorial) or via maven command line(`mvn clean install`). Deploy the war to a Servlet 3.0 container . Since here i am using Tomcat, i will simply put this war file into `tomcat webapps folder` and click on `start.bat` inside tomcat bin directory.

Run the application



Greeting : Hello World from Spring 4 MVC



Greeting : Hello World Again, from Spring 4 MVC

That's it.

Download Source Code

Download Now!

References

- Spring framework

websystiqueadmin

If you like tutorials on this site, why not take a step further and connect me on Facebook , Google Plus & Twitter as well? I would love to hear your thoughts on these articles, it will help me improve further our learning process.

If you appreciate the effort I have put in this learning site, help me improve the visibility of this site towards global audience by sharing and linking this site from within and beyond your network. You & your friends can always link my site from your site on www.websystique.com, and share the learning.

After all, we are here to learn together, aren't we?

Related Posts:

1. **Spring 4 MVC HelloWorld Tutorial – Full XML Example**
2. **Spring 4 MVC + JMS + ActiveMQ annotation based Example**
3. **Spring 4 MVC+Hibernate Many-to-many JSP Example with annotation**
4. **Spring MVC 4 File Download Example**

springmvc. permalink.

← Spring 4 MVC HelloWorld Tutorial – Full XML Example

Spring 4 MVC REST Service Example using @RestController →

26 Comments **websystique** 1 **Login**

Recommend **Share** **Sort by Best**

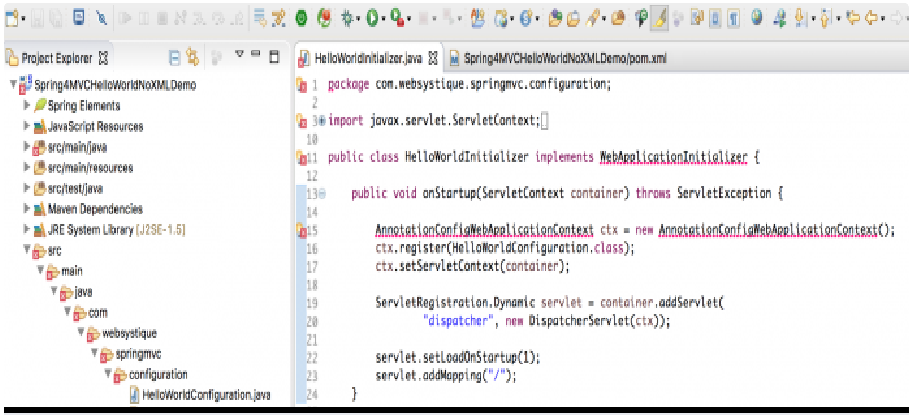
Join the discussion…

David • 6 days ago

Hi.

Unable to create your project from zero. Sometimes not recognize dependencies, sometimes not recognize @RequestMapping, etc...

So I decided to download your code and install using import option in Eclipse Neon.
Everything is ok, but HelloWorldInitializer give full of errors.



```
package com.websystique.springmvc.configuration;

import javax.servlet.ServletContext;[]

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("/");
    }
```

see more

• Reply • Share ›

websystique Mod ➜ David • 6 days ago

Hi David, Firstly update your pom.xml to include maven-compiler-plugin [as shown in post, might be missing in download] with your java version[1.6 or 7/8], then do a maven update, should be fine.

• Reply • Share ›

Amit • 17 days ago

Hi, Thank you for the post. I am struggling with 404 error with Java based configuration. I am using JBoss EAP 6.1 to run this application. URLs are being mapped correctly. However it still gives 404 error while accessing the page. I am using the correct URLs.

http://localhost:8080/Spring4XMLDemo/

http://localhost:8080/Spring4XMLDemo//helloagain

Please don't get confused with the name 'Spring4XMLDemo'. It is actually Java based configuration, not xml based. XML based config is working fine for me.

```
16:18:38,163 INFO [org.jboss.web] (ServerService Thread Pool -- 190)
JBAS018210: Register web context: /Spring4XMLDemo

16:18:38,169 INFO [org.apache.catalina.core.ContainerBase.[jboss.web].
[default-host].[/Spring4XMLDemo]] (ServerService Thread Pool -- 190)
Spring WebApplicationInitializers detected on classpath:
```

see more

• Reply • Share ›

websystique Mod ➜ Amit • 6 days ago

Hi Amit, sorry for being late. I wonder if you are still facing the issue? I don't have a jboss ready setup at moment [used tomcat] but if you are still stuck, let me know, will prepare the env and try myself.

• Reply • Share ›

Amit ➜ websystique • 4 days ago

Hi, Thanks for the response. I have resolved the issue. I used JBoss 7.0 EAP and servlet API version 3.1.0. It resolved the issue.

• Reply • Share ›

pranish • 2 months ago

how to use spring boot and spring 4 together? Or spring boot also has its 4 version

• Reply • Share ›

Gautham Shetty • 2 months ago

getting following error after mvn clean compile

```
INFO] ------------------------------------------------------------------------

[ERROR] No goals have been specified for this build. You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help 1]

org.apache.maven.lifecycle.NoGoalSpecifiedException: No goals have been specified for this build. You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy.

at
org.apache.maven.lifecycle.internal.LifecycleStarter.execute(LifecycleStart

at org.apache.maven.DefaultMaven.doExecute(DefaultMaven.java:307)

at org.apache.maven.DefaultMaven.doExecute(DefaultMaven.java:193)

at org.apache.maven.DefaultMaven.execute(DefaultMaven.java:106)
```

• Reply • Share ›

Dileep Kumar Kottakota • 3 months ago

Getting this error even i have latest jdk.

"dynamic web module 3.1 requires java 1.7 or newer"

• Reply • Share ›

websystique Mod ➜ Dileep Kumar Kottakota • 3 months ago

Hi Dileep, You may need to configure your IDE with required jdk.

• Reply • Share ›

Anil Vighne • 4 months ago

Hello Sir please give example on spring cache. and AOP also.

• Reply • Share ›

ankit chauhan • 6 months ago
Hello websystique,

I just post it to tell you how unbelievably nice this tutorial is .. :)

regards
Ankit
• Reply • Share ›

websystique Mod ankit chauhan • 6 months ago
Hi Ankit,

Glad you liked it.
• Reply • Share ›

Nicolas R.M • 6 months ago
Hi, annotation vs xml configuration?
• Reply • Share ›

websystique Mod Nicolas R.M • 6 months ago
Hi Nicolas,
Both the configurations are shown here, Annotation being the preferred approach. Let me know if i misunderstood your question.
• Reply • Share ›

Nicolas R.M websystique • 6 months ago
Yes, thank you, you are great! :)
• Reply • Share ›

Texan Rock • 6 months ago
type Status report

message /Spring4MVCHelloWorldNoXMLDemo/

description The requested resource is not available.

after deploy in tomcat 8
• Reply • Share ›

websystique Mod Texan Rock • 6 months ago
Your tomcat+Eclipse environment is not ok. Please follow Setup tomcat with Eclipse, to set it up properly.
• Reply • Share ›

Arnish Gupta • 7 months ago
Hello,

i have a problem that it shows ${greeting} as output, but i want to print the value that declared in controller.
• Reply • Share ›

websystique Mod ➦ Arnish Gupta • 7 months ago

Hi Arnish, Could you try adding <%@ page isELIgnored="false" %> in your JSP please? It should fix the issue.

• Reply • Share ›

Arnish Gupta ➦ websystique • 7 months ago

Thank you, it's working.

• Reply • Share ›

mdk • 7 months ago

Both these dependencies should have a scope of provided, otherwise they will be exported to /WEB-INF/lib and contaminate your tomcat

```
<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>
```

should be : -

```
<dependency>
<groupid>javax.servlet</groupid>
```

see more

• Reply • Share ›

websystique Mod ➦ mdk • 7 months ago

Thanks MDK,

I've already said about it in my post above, but it's always better to point-it out loudly.

• Reply • Share ›

Sumit Surana • a year ago

Hi, I tried the above example as it is and even removed the web.xml file with servlet-api version being 3.1.0 and is included in my pom.xml, but the jsp expression syntax ie ${greeting} is not working and is showing "Greeting: ${greeting}" on the web page instead of "Greeting : Hello World from Spring 4 MVC"

• Reply • Share ›

websystique Mod ➦ Sumit Surana • a year ago

Hey Sumit, sorry for late reply.

The attached project itself is fine. The problem you are getting seems related to the container in that your expression are not getting evaluated.
Could you try adding following at top of your JSP and try to build and deploy again?

<%@ page isELIgnored="false" %>

Let me know the outcome.

Reply • Share ›

Aditya Goyal • a year ago

please add any jstl dependency in pom.xml

<dependency>
<groupid>javax.servlet</groupid>
<artifactid>jstl</artifactid>
<version>1.2</version>
</dependency>

Reply • Share ›

websystique Mod Aditya Goyal • a year ago

Hi Aditya, I've added jstl dependency explicitly in pom.xml, and updated the zip. Thanks for reporting this issue. Seems recent version of tomcat [8.x] need explicit JSTL.