

# Gradle – Spring 4 MVC Hello World Example – Annotation

By mkyong (<http://www.mkyong.com/author/mkyong/>) | June 4, 2015 | Updated : June 12, 2015

---



In this tutorial, we will take the previous Gradle + Spring MVC XML example (<http://www.mkyong.com/spring-mvc/gradle-spring-mvc-web-project-example/>), rewrite it to support @JavaConfig annotation configuration, no more XML files.

*P.S This example will works in Servlet 3.0+ container only, like Tomcat 7 or Jetty 9.*

Technologies used :

1. Gradle 2.0
2. Spring 4.1.6.RELEASE
3. Tomcat 7 or Jetty 9
4. Eclipse 4.4
5. JDK 1.7
6. Logback 1.1.3
7. Bootstrap 3

## 1. Project Structure

Download the project source code and review the project folder structure :



*P.S No more XML files like web.xml or Spring XML configuration files.*

## 2. Gradle

2.1 Review the build.gradle file, this should be self-explanatory.

build.gradle

```
apply plugin: 'java'
apply plugin: 'war'
apply plugin: 'eclipse-wtp'
//apply plugin: 'jetty' //too old, Jetty 6, use gretty plugin
apply plugin: 'org.akhikhl.gretty'

// JDK 7
sourceCompatibility = 1.7
targetCompatibility = 1.7

repositories {
    mavenLocal()
    mavenCentral()
}

dependencies {
    compile 'ch.qos.logback:logback-classic:1.1.3'
    compile 'org.springframework:spring-webmvc:4.1.6.RELEASE'
    compile 'javax.servlet:jstl:1.2'

    //include in compile only, exclude in the war
    providedCompile 'javax.servlet:servlet-api:2.5'
}

//Gretty Embedded Jetty
buildscript {
    repositories {
        jcenter()
    }

    dependencies {
        classpath 'org.akhikhl.gretty:gretty:+'
    }
}

// Don't use Jetty8, even it's a servlet 3.0+ container,
// but not support non-jar WebApplicationInitializer scanning.
// It will cause "No Spring WebApplicationInitializer types detected on classpath"
gretty {
    port = 8080
    contextPath = 'spring4'
    servletContainer = 'jetty9' //tomcat7 or tomcat8
}

//For Eclipse IDE only
eclipse {

    wtp {
        component {

            //define context path, default to project folder name
            contextPath = 'spring4'

        }
    }
}
```

```
}
```

2.2 Make this project supports Eclipse IDE. Now, you can import the project into Eclipse IDE.

```
your-project$ gradle eclipse
```

## 3. Spring @Configuration

Spring @Configuration and its XML equivalent.

3.1 Spring annotation configuration to scan the service classes.

### SpringRootConfig.java

```
package com.mkyong.helloworld.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan({ "com.mkyong.helloworld.service" })
public class SpringRootConfig {
}
```

XML equivalent.

### spring-core-config.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd ">

  <context:component-scan base-package="com.mkyong.helloworld.service" />

</beans>
```

3.2 Extends abstract class WebMvcConfigurerAdapter .

### SpringWebConfig.java

```
package com.mkyong.helloworld.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;

@EnableWebMvc //<mvc:annotation-driven />
@Configuration
@ComponentScan({ "com.mkyong.helloworld.web" })
public class SpringWebConfig extends WebMvcConfigurerAdapter {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/resources/**")
                .addResourceLocations("/resources/");
    }

    @Bean
    public InternalResourceViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver
                = new InternalResourceViewResolver();
        viewResolver.setViewClass(JstlView.class);
        viewResolver.setPrefix("/WEB-INF/views/jsp/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
}
```

XML equivalent.

```
spring-web-config.xml
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd ">

  <context:component-scan base-package="com.mkyong.helloworld.web" />

  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/WEB-INF/views/jsp/" />
    <property name="suffix" value=".jsp" />
  </bean>

  <mvc:resources mapping="/resources/**" location="/resources/" />

  <mvc:annotation-driven />

</beans>
```

## 4. Servlet 3.0+ Container

Create a `ServletInitializer` class, Servlet 3.0+ container will pick up this class and run it automatically. This is the replacement class for `web.xml`

```
package com.mkyong.helloworld.servlet3;

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

import com.mkyong.helloworld.config.SpringRootConfig;
import com.mkyong.helloworld.config.SpringWebConfig;

public class MyWebInitializer extends
    AbstractAnnotationConfigDispatcherServletInitializer {

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

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

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

}
```

XML equivalent.

```
web.xml
```

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">

  <display-name>Gradle + Spring MVC Hello World</display-name>
  <description>Spring MVC web application</description>

  <!-- For web context -->
  <servlet>
    <servlet-name>hello-dispatcher</servlet-name>
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring-mvc-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>hello-dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

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

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring-core-config.xml</param-value>
  </context-param>

</web-app>
```

**Note**

There is no change in the Spring controller, logback and JSP files, so, the source code will not repeat here, please refer to the previous Gradle + Spring MVC XML example (<http://www.mkkyong.com/spring-mvc/gradle-spring-mvc-web-project-example/>) for complete source code.

## 5. Demo

5.1 To run this project. Issues `gradle jettyRun` to start the embedded Jetty container.

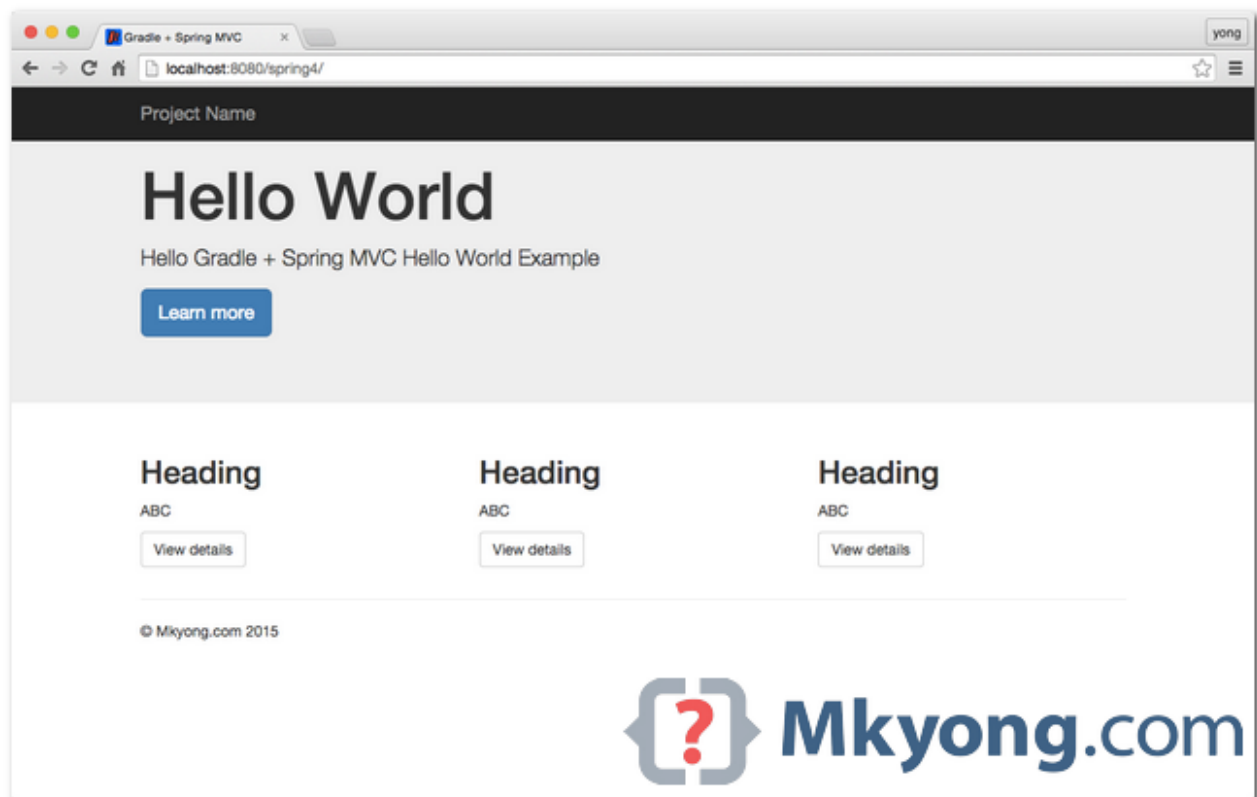


## Terminal

```
your-project$ gradle jettyRun
```

```
21:56:34 INFO Jetty 9.2.10.v20150310 started and listening on port 8080
21:56:34 INFO spring4 runs at:
21:56:34 INFO http://localhost:8080/spring4
Press any key to stop the server.
> Building 87% > :jettyRun
```

5.2 <http://localhost:8080/spring4/>



5.3 <http://localhost:8080/spring4/hello/mkyong.com>