War 包 配置

**一、Application.java 修改：**

1.public class Application extends SpringBootServletInitializer

2. 增加一个方法(可以先不加，有问题再加上也行):

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {

return builder.sources(this.getClass());

}

**二、pom.xml 修改：**

1. <packaging>war</packaging>

2.<!-- 打war包时加入此项， 告诉spring-boot tomcat相关jar包用外部的，不要打进去 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

**三、服务器访问地址：**

[https://localhost/R/#/login](https://localhost/R/#/login/login)

[https://localhost:433/R/#/login](https://localhost:433/R/#/login/login)

<http://localhost:8080/R/#/login>

参考资料：

1. <http://blog.csdn.net/asdfsfsdgdfgh/article/details/52127562>

由jar变成war

<packaging>jar</packaging>

如果是上面的打包方式，启动方式则为

mvn package

java -jar target/mymodule-0.0.1-SNAPSHOT.jar

改变成war

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

<project xmlns="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/xsd/maven-4.0.0.xsd">

<!-- ... -->

<packaging>war</packaging>

<!-- ... -->

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

<!-- ... -->

</dependencies>

</project>

只需要把tomcat的范围改成provided

如果要发布到外部的tomcat同时需要改变启动方式

新增ServletInitializer类

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.context.web.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

return application.sources(Application.class);

}

}

Application.class 为标注有@SpringBootApplication的主启动类

注意的问题

此时打成的包的名称应该和application.properties的

server.context-path=/spring-boot

保持一致

<build>

<finalName>spring-boot</finalName>

</build>

如果不一样发布到tomcat的webapps下上下文会变化

1. <http://blog.csdn.net/u014087707/article/details/51792482>

我修改pom.xml将打包方式改成war

<packaging>war</packaging>

1

完事儿打完包以后， 放到我本地Tomcat上跑了跑，发现没问题。但是部署到服务器上的Tomcat以后， 发现无法启动，错误如下：

org.apache.catalina.LifecycleException: Failed to start component [StandardEngine[Catalina].StandardHost[localhost].StandardContext[

/report]]

at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:153)

at org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:899)

at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:875)

at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:652)

at org.apache.catalina.startup.HostConfig.deployWAR(HostConfig.java:1092)

at org.apache.catalina.startup.HostConfig$DeployWar.run(HostConfig.java:1984)

at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)

at java.util.concurrent.FutureTask.run(FutureTask.java:262)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)

at java.lang.Thread.run(Thread.java:745)

Caused by: java.lang.NoSuchMethodError: javax.servlet.ServletContext.getVirtualServerName()Ljava/lang/String;

at org.apache.tomcat.websocket.server.WsServerContainer.<init>(WsServerContainer.java:150)

at org.apache.tomcat.websocket.server.WsSci.init(WsSci.java:131)

at org.apache.tomcat.websocket.server.WsSci.onStartup(WsSci.java:47)

at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5573)

at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:147)

... 10 more

明显不是应用代码错误，猜想应该是环境问题。经过分析， 我本地的Tomcat版本是8.0.28， 而服务器上的Tomcat是7.0.69。 我在本地下了个Tomcat7.0.70部署也报同样错误。更加确定问题跟Tomcat版本有关。经过多方查找资料，最后在Stackoverflow上看到一个老外说SpringBoot默认Servlet容器是基于Tomcat8的

在打好的war包中确实发现了Tomcat相关jar包，而且是Tomcat8的，拿Tomcat8的embed包在Tomcat7下面那肯定是不能用啊

这里写图片描述

要支持低版本Tomcat需要在maven中指定Tomat版本，配置如下：

<properties>

<tomcat.version>7.0.69</tomcat.version>

</properties>

然后依赖中加上（这个其实不加也行， 官方文档是加上的）

<dependency>

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

<artifactId>tomcat-juli</artifactId>

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

</dependency>

加上以后试了下，果然是没问题了。看了下war包中的lib目录，确实已经变成Tomcat7的包了

这里写图片描述

但是我还是有点疑惑， 这样配置打成包岂不是换个Tomcat版本就要重新打次包？ 既然是由于SpringBoot内部的Servlet容器造成了这个限制， 那我不用行不行？ 又查了很多资料， 还真有办法！

<!-- 打war包时加入此项， 告诉spring-boot tomcat相关jar包用外部的，不要打进去 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

试了下， 加上这个后， 上面说Tomcat的版本无需指定了， 耶! ^\_^

总结

总结下SpringBoot应用部署到Tomcat下的配置方法用于备忘也方便遇到同样问题的朋友

将打包方式改成war

这个没啥好说的， 肯定要改成war

配置嵌入Tomcat中的方式

这里有两种方式可选择：

方式一：用spring-boot内置的tomcat库， 并指定你要部署到Tomcat的版本

<properties>

<tomcat.version>7.0.69</tomcat.version>

</properties>

<dependency>

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

<artifactId>tomcat-juli</artifactId>

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

</dependency>

方式二：不用spring-boot内置的tomcat库（强烈推荐这种方式！！）

<!-- 打war包时加入此项， 告诉spring-boot tomcat相关jar包用外部的，不要打进去 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

maven-war-plugin (可选)

与maven-resources-plugin类似，当你有一些自定义的打包操作， 比如有非标准目录文件要打到war包中或者有配置文件引用了pom中的变量。 具体用法参见官方文档：http://maven.apache.org/components/plugins/maven-war-plugin/