# Maven项目之Spring.profile.开发.测试.生产环境的配置和切换之一键部署

背景：

软件开发过程一般涉及“开发 -> 测试 -> 部署上线”多个阶段，每个阶段的环境的配置参数会有不同，如数据源，文件路径等。

为避免每次切换环境时都要进行参数配置等繁琐的操作，可以通过spring的profile功能来进行配置参数的切换。

部署阶段最原始的方式是连接服务器，停掉tomcat，备份之前的war，替换war，启动tomcat，这个过程不繁复，但是一天来两次也是够烦的了。

## 环境配置：

jdk8，Maven3.3.9，IntelliJ IDEA，tomcat7。

## 1：springprofile

省略：具体步骤和http://blog.csdn.net/fengyong7723131/article/details/78543822中的第一步一样

1：建立文件夹environment，里面放着三个不同环境的配置文件

2：在配置文件中配置profile-beans。注意：需要放在最下面，并且，所有的配置文件都需要加上profile-beans，不然会失效，切记切记。

<?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:tx="http://www.springframework.org/schema/tx"

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

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

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

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

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

http://www.springframework.org/schema/tx/spring-tx-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/aop

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

<!-- 定义数据源 -->

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"

destroy-method="close">

<property name="driverClassName" value="${jdbc.driverClassName}" />

<property name="url" value="${jdbc.url}" />

<property name="username" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

<property name="minIdle" value="${jdbc.minIdle}" /> <!-- 队列中的最小等待数 -->

<property name="maxIdle" value="${jdbc.maxIdle}" /> <!-- 队列中的最大等待数 -->

<property name="maxWait" value="${jdbc.maxWait}" /> <!-- 最长等待时间，单位毫秒 -->

<property name="maxActive" value="${jdbc.maxActive}" /> <!-- 最大活跃数 -->

<property name="initialSize" value="${jdbc.initialSize}" /><!--

初始大小 -->

<property name="validationQuery" value="select 1" /><!-- 检测语句 -->

</bean>

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="mapperLocations" value="classpath\*:com/jniu/\*/dao/mapper/\*.xml" />

</bean>

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="com.jniu.\*.dao" />

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />

</bean>

<!-- 定义事务管理器 -->

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

<bean id="transactionDefinition"

class="org.springframework.transaction.support.DefaultTransactionDefinition">

<property name="propagationBehaviorName" value="PROPAGATION\_NESTED" />

</bean>

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="com/jniu/\*/dao/mapper/\*"/>

</bean>

<!-- 使用annotation注解方式配置事务 -->

<tx:annotation-driven transaction-manager="transactionManager" />

<beans profile="dev">

<context:property-placeholder ignore-resource-not-found="true" location="classpath:environment/dev\_xyfapp.properties"/>

</beans>

<beans profile="tes">

<context:property-placeholder ignore-resource-not-found="true" location="classpath:environment/test\_xyfapp.properties"/>

</beans>

<beans profile="pro">

<context:property-placeholder ignore-resource-not-found="true" location="classpath:environment/pro\_xyfapp.properties"/>

</beans>

</beans>

3：通过设置spring.profiles.default和spring.profiles.active这两个属性来激活和使用对应的配置文件。

设置方式有多种：

在web.xml中作为web应用的上下文参数context-param；  
在web.xml中作为DispatcherServlet的初始化参数；  
作为JNDI条目；  
作为环境变量；  
作为JVM的系统属性；  
在集成测试类上，使用@ActiveProfiles注解配置。

如果在程序中也使用到了spring.profiles.active，可以按照以下代码获取web.xml中的value值：

import javax.servlet.ServletContext;

import org.springframework.web.context.WebApplicationContext;

public class WebContextListener extends org.springframework.web.context.ContextLoaderListener{

public WebApplicationContext initWebApplicationContext(ServletContext servletContext) {

String env = servletContext.getInitParameter("spring.profiles.active");

//env可以设置在缓存中，供项目使用

return super.initWebApplicationContext(servletContext);

}

}

## 2：Maven配置

### 1：maven本身支持profile，在pom文件里配置profile

<profiles>

<profile>

<id>dev</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<profiles.activation>dev</profiles.activation>

</properties>

</profile>

<profile>

<id>tes</id>

<properties>

<profiles.activation>tes</profiles.activation>

<tomcat.deploy.serverUrl>http://47.92.137.130:8081/manager/text</tomcat.deploy.serverUrl>

<tomcat.deploy.path>/</tomcat.deploy.path>

</properties>

</profile>

<profile>

<id>pro</id>

<properties>

<profiles.activation>pro</profiles.activation>

<tomcat.deploy.serverUrl>http://47.92.158.134:8081/manager/text</tomcat.deploy.serverUrl>

<tomcat.deploy.path>/</tomcat.deploy.path>

</properties>

</profile>

</profiles>

### 2：maven的profile部署，用到了tomcat的manager功能，这里需要修改tomcat的tomcat-users.xml配置,修改如下

<role rolename="manager-script"/>  
<user username="username" password="password" roles="manager-script"/>

### 3：添加maven的tomcat插件和打war包插件

<plugin>

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

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

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

<configuration>

<server>Tomcat</server>

<url>${tomcat.deploy.serverUrl}</url>

<username>username</username>

<password>password</password>

<update>true</update>

<path>${tomcat.deploy.path}</path>

<warFile>target/ce.war</warFile>

<uriEncoding>${project.build.sourceEncoding}</uriEncoding>

</configuration>

</plugin>

<!-- war 打包插件, 设定war包名称不带版本号 -->

<plugin>

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

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

<version>2.4</version>

<configuration>

<packagingExcludes>

</packagingExcludes>

<warSourceExcludes>

</warSourceExcludes>

<webResources>

<resource>

<filtering>true</filtering>

<directory>src/main/webapp</directory>

<includes>

<include>\*\*/web.xml</include>

</includes>

</resource>

</webResources>

<webappDirectory>${project.build.directory}/${project.artifactId}</webappDirectory>

<webXml>src/main/webapp/WEB-INF/web.xml</webXml>

<warName>ce</warName>

</configuration>

</plugin>

### 4：web.xml文件

<context-param>

<param-name>spring.profiles.default</param-name>

<param-value>dev</param-value>

</context-param>

<context-param>

<param-name>spring.profiles.active</param-name>

<param-value>${profiles.activation}</param-value>

</context-param>

### 5：使用命令将项目部署到远程服务器，前提是远程的tomcat正常运行

mvn clean install tomcat7:redeploy -Ptes -Dmaven.test.skip=true  
mvn clean install tomcat7:redeploy -Ppro -Dmaven.test.skip=true

-P后面的参数是pom文件里profile里的id，war插件需要多说的是，可以将-P参数映射到web.xml里的${profiles.activation}

每次部署完记得将本地的编译文件清理一次，不然直接启动的话，连接的是你上一次部署的环境

# [开发环境-测试环境-生产环境-配置自动切换N种办法](http://18810098265.iteye.com/blog/2368914)

http://18810098265.iteye.com/blog/2368914