# Eclipse

## Eclipse右键New菜单项的自定义设置

大家都知道一般在Eclipse中新建一个项目或者文件都是通过菜单项的File-New来创建，然而有些右键出来的选项可能从来都不会使用到，而有些可能会常用到但是右键中又没有，这个时候就可以自定义New中的右键选项，具体设置方式如下：

**Window --> Customize Perspective... --> Shortcuts --> 选择"New"**  进行相关设置即可(如下图)



## web项目run as 菜单中没有 run on server

eclipse中已安装好tomcat，某maven web项目原本也可以在tomcat中运行，突然发现右击该项目的 run as 菜单中，没有了 run on serven 选项，项目也无法在tomcat中跑了。

原因是该项目的Dynamic Web Module 被我不经意间删除了，如下图。



## 为什么eclipse新建项目的默认编码为何是GBK,而不是UTF-8

我们用的操作系统是中文操作系统，windows操作系统的编码是GBK，所以eclipse默认的新项目的编码是GBK的，你在Window->Preferences->General->Workspace->Text file encoding 将其改为UFT-8即可，同样，如果你想修改html,css等文件的编码格式，你在Window->Preferences->General->Content Types中可以修改

## 如何删除Eclipse弃用的工作空间？

在 eclipse\configuration\.settings\org.eclipse.ui.ide.prefs 文件里找到相应的空间，删除即可。

# maven

## Eclipse+Maven创建webapp项目

Eclipse+Maven创建webapp项目<二>

1、开启eclipse，右键new——》other，如下图找到maven project



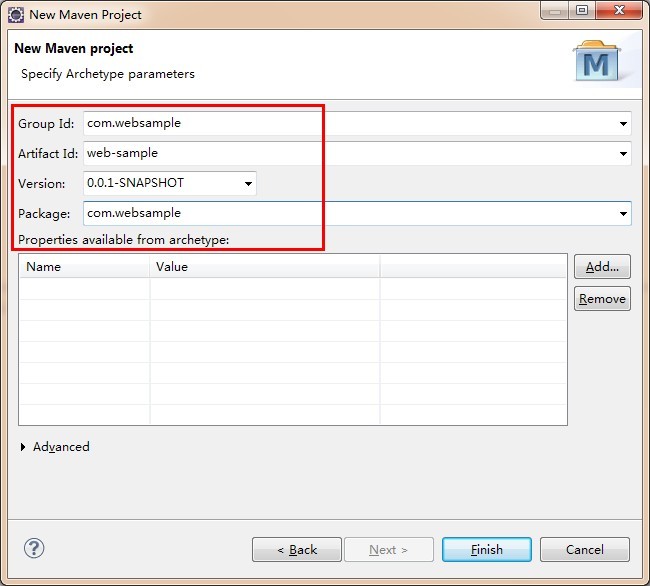
2、选择maven project，显示创建maven项目的窗口，此次不勾选Create a simple project，直接next，如图所示：



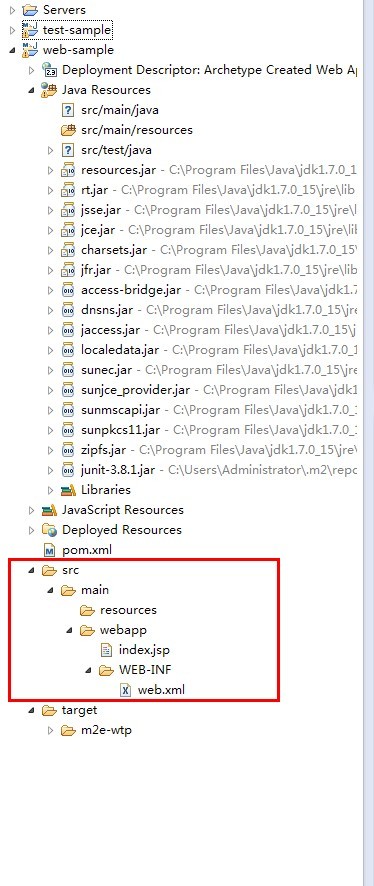
3、在Filter中搜索webapp，选择如图所示，并next：



4、输入maven项目的基本信息，如下图所示：



5、完成maven项目的创建，生成相应的webapp项目结构，如下所示，此处有部分结构是项目不需要的，我们需要去掉：



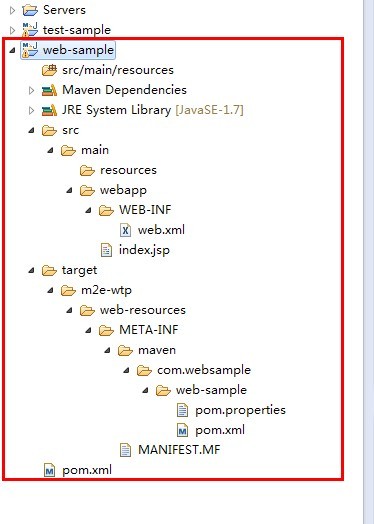
6、选择项目，右键选择Properties，进入属性页面，选择到Maven菜单下，如下图所示：



7、选择java版本为1.7，并去掉其他两项，如下图：



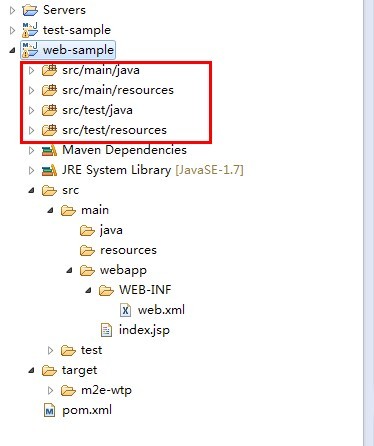
8、点击ok之后，再次回到项目结构，此时项目结构比较清晰，符合我们想要创建的maven项目：



9、创建没有显示的源代码文件夹，如下图所示：



10、最后得到完成的maven webapp项目结构如下：



## Eclipse+Maven构建springmvc项目

一.背景介绍

　　对于初学者，用maven构建项目并不是一件容易的事，springmvc并不是MVC中的主流，但是本人觉得springmvc比struts要好用，真正做到了零配置。一次使用，就放不下了。

二.准备工作

　1.Eclipse 3.7

　　2.maven

　　3.Eclipse 需要安装maven插件。url:maven - <http://download.eclipse.org/technology/m2e/releases> 。安装maven-3.0.4。并且选择本地的maven，如下图：



三.构建工程

　　1.用maven插件构建项目框架

　　maven具有强大构建功能，使用maven可以构建多种不同类型的工程。这里我们构建maven-archetype-webapp类型的项目。在Eclipse->New中选择other,找到maven Project型。如下图：



在选完路径之后，我们选择构建类型，如下图：



接下来，填写工程的Group Id，Artifact Id，如下图：



这里的Group Id就是大项目的id，Arifact Id就是该项目的Id。就像一个大项目中有许多小项目组成一样。此时，我们的项目已经成型了，样子如下图：



接下来，我们要完善项目的目录，配置。

　　2.完善项目

　　首先，完善目录，增加重要的source Folder，这个不是简单的Floder,这些文件夹是会参与编译的。增加src/main/java，src/test/resources，src/test/java目录。让目录变成标准的maven结构。如下图：



接下来，改变一些配置：

让工程的JDK用本地的jdk;

让工程的字符集为UTF-8;

改变工程的目录顺序；

　　这些都完成之后，工程目录应该是如下的样子：



　　3.将工程变成web工程

　　此时，我们的工程还不是标准的web工程，可以在eclipse中增加web工程的特性，选择工程的Properties,选Project Facets,如下图：



这里，我们选择Dynamic Web Module,版本选择2.4，这个版本比较通用。如下图：



此时，我们看到目录中多了一个WebContent目录，由于使用maven构建，web目录是src/main/webapp,所以我们删除WebContent目录。接下来，要配置web项目的发布目录，就是Deployment Assembly，如图：



test目录不要发布，WebContent目录没有了，所以这三项都删掉。并且增加src/main/webapp目录，和Maven Dependenices,完成之后如下图：



于是，我们的工程就完全是一个web工程了。

　　4.赋予工程的springmvc特性

　　配置web.xml，使其具有springmvc特性，主要配置两处，一个是ContextLoaderListener，一个是DispatcherServlet。代码如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

2 <web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"

3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

4 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee

5 http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">

6

7 <listener>

8 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

9 </listener>

10

11 <servlet>

12 <servlet-name>exam</servlet-name>

13 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

14 </servlet>

15

16 <servlet-mapping>

17 <servlet-name>exam</servlet-name>

18 <url-pattern>/</url-pattern>

19 </servlet-mapping>

20

21 <welcome-file-list>

22 <welcome-file>index.jsp</welcome-file>

23 </welcome-file-list>

24 </web-app>

[复制代码](javascript:void(0);)

　　配置ContextLoaderListener表示，该工程要以spring的方式启动。启动时会默认在/WEB-INF目录下查找applicationContext.xml作为spring容器的配置文件，这里可以初始化一些bean，如DataSource。我们这里什么也不做。代码如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

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

2 <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

3 <beans>

4 </beans>

　　配置DispatcherServlet表示，该工程将采用springmvc的方式。启动时也会默认在/WEB-INF目录下查找XXX-servlet.xml作为配置文件，XXX就是DispatcherServlet的名字，该文件中将配置两项重要的mvc特性：

HandlerMapping,负责为DispatcherServlet这个前端控制器的请求查找Controller；

ViewResolver,负责为DispatcherServlet查找ModelAndView的视图解析器。

代码如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

2 <!-- Bean头部 -->

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

4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"

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

6 xmlns:util="http://www.springframework.org/schema/util"

7 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

8 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd

9 http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd

10 http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.0.xsd">

11

12 <!-- 激活@Controller模式 -->

13 <mvc:annotation-driven />

14 <!-- 对包中的所有类进行扫描，以完成Bean创建和自动依赖注入的功能 需要更改 -->

15 <context:component-scan base-package="cc.monggo.web.controller" />

16

17 <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />

18

19 <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">

20 <property name="prefix">

21 <value>/WEB-INF/jsp/</value>

22 </property>

23 <property name="suffix">

24 <value>.jsp</value>

25 </property>

26 </bean>

27 </beans>

[复制代码](javascript:void(0);)

　　5.让maven自动配置jar包

　　在用maven生成框架时，就生成了pop.xml，这就是maven的配置文件。我们要引入spring-web,servlet等特性的包。代码如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

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

2 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

3 <modelVersion>4.0.0</modelVersion>

4 <groupId>exam</groupId>

5 <artifactId>exam\_3</artifactId>

6 <packaging>war</packaging>

7 <version>0.0.1-SNAPSHOT</version>

8 <name>exam\_3 Maven Webapp</name>

9 <url>http://maven.apache.org</url>

10 <dependencies>

11 <dependency>

12 <groupId>junit</groupId>

13 <artifactId>junit</artifactId>

14 <version>3.8.1</version>

15 <scope>test</scope>

16 </dependency>

17

18 <dependency>

19 <groupId>org.springframework</groupId>

20 <artifactId>spring-web</artifactId>

21 <version>3.0.5.RELEASE</version>

22 </dependency>

23

24 <dependency>

25 <groupId>org.springframework</groupId>

26 <artifactId>spring-webmvc</artifactId>

27 <version>3.0.5.RELEASE</version>

28 </dependency>

29

30 <dependency>

31 <groupId>org.apache.geronimo.specs</groupId>

32 <artifactId>geronimo-servlet\_2.5\_spec</artifactId>

33 <version>1.2</version>

34 </dependency>

35

36 </dependencies>

37 <build>

38 <finalName>exam\_3</finalName>

39 </build>

40 </project>

[复制代码](javascript:void(0);)

　　maven就是这么简单，一旦保存，maven就会自动下载pop.xml的jar包。此时可以看到目录中Maven Dependencies下生成了jar包。

　　更多的jar包可以在maven中心库下载：[http://mvnrepository.com](http://mvnrepository.com/)。

　　6.做个测试

　　说了一大堆，只有运行起来才有意思，下面写个简单的测试。先写Controller。编写两个类，LoginControler.java，LoginForm.java。代码如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 package cc.monggo.web.controller;

2

3

4 import javax.servlet.http.HttpServletRequest;

5 import javax.servlet.http.HttpServletResponse;

6

7 import org.springframework.stereotype.Controller;

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

9 import org.springframework.web.servlet.ModelAndView;

10

11 import cc.monggo.domain.LoginForm;

12

13 @Controller

14 public class LoginController {

15 @RequestMapping(value="login")

16 public ModelAndView login(HttpServletRequest request,HttpServletResponse response,LoginForm command ){

17 String username = command.getUsername();

18 ModelAndView mv = new ModelAndView("/index/index","command","LOGIN SUCCESS, " + username);

19 return mv;

20 }

21 }

[复制代码](javascript:void(0);)

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 package cc.monggo.domain;

2

3

4 public class LoginForm {

5 private String username;

6 private String password;

7 public String getUsername() {

8 return username;

9 }

10 public void setUsername(String username) {

11 this.username = username;

12 }

13 public String getPassword() {

14 return password;

15 }

16 public void setPassword(String password) {

17 this.password = password;

18 }

19 }

[复制代码](javascript:void(0);)

　　再增加一些jsp,首页的index.jsp,主要是做跳转，代码如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

1 <%

2 request.getRequestDispatcher("/WEB-INF/jsp/login/login.jsp").forward(request,response);

3 %>

　  还有两个jsp，做些简单的功能，一个表单login.jsp,一个表单提交的返回index.jsp，代码如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 <%@ page language="java" contentType="text/html; charset=utf-8"

2 pageEncoding="utf-8"%>

3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

4 <html>

5 <head>

6 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

7 <title>Insert title here</title>

8 </head>

9 <body>

10 <div>

11 <form action="login" methed="get">

12 <input type="text" name="username">

13 <input type="submit" value="SUBMIT">

14 </form>

15 </div>

16 </body>

17 </html>

[复制代码](javascript:void(0);)

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

1 <%@ page language="java" contentType="text/html; charset=utf-8"

2 pageEncoding="utf-8"%>

3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

4 <html>

5 <head>

6 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

7 <title>Insert title here</title>

8 </head>

9 <body>

10 ${command}

11 </body>

12 </html>

[复制代码](javascript:void(0);)

　　整个项目的目录结构如下：



　　在tomcat中运行，即可。不要使用Tomcat插件，可能有问题。就用普通的方式。运行效果如下图：



## Maven项目打包缺少XML文件

解决办法，在pom.xml中把xml文件包含进去。

<build>

<resources>

<resource>

<directory>src/main/java</directory>

<includes>

<include>com/sun/mapping/ChargeMapper.xml</include>

<include>com/sun/mapping/ConsumeMapper.xml</include>

<include>com/sun/mapping/RefundMapper.xml</include>

<include>com/sun/mapping/UserMapper.xml</include>

</includes>

<filtering>true</filtering>

</resource>

</resources>

</build>

## pom.xml更新下载jar包失败





1.问题描述

在通过pom.xml中的依赖坐标,从私服下载依赖包时,往往会遇到如下的类似问题:

The container 'Maven Dependencies' references non existing library '${groupid}/${artifactid}-${version}.jar'

e.g.

|  |  |
| --- | --- |
| 1 | The container 'Maven Dependencies' references non existing library 'E:\Maven\repo\org\apache\hadoop\hadoop-yarn-api\2.0.0-cdh4.5.0\hadoop-yarn-api-2.0.0-cdh4.5.0.jar' |

2.解决方案

上面问题往往是在下载依赖过程中网络出现问题导致的.

此时我们本机已经开始下载依赖代码,但是下载失败,本机仓库中会在${MAVEN\_repo}/${groupid}/${artifactid}/${version} 路径下面生成 \*.lastUpdated 的文件.

\*.lastUpdated实在可恶,下次再下载时由于它的存在Maven不会再去私服中下载

2.1 解决方法:(针对PC与Maven私服之间网络传输问题)

1. 打开本地仓库所在目录, 通过win文件夹的搜索功能,查找 \*.lastUpdated ,然后将找到的文件全部删除
2. 重新 Maven Update Project,
   1. 若在eclipse中操作时勾选上 Force Update of Snapshots / Releases
   2. 若在命令行操作时增加 -U 参数

一般情况下上面方法即可解决问题. 若问题依然存在,请确认下主机与Maven私服之间网络是否畅通.可以通过在浏览器中打开私服地址确认.

若本机与私服之前网络畅通,那这个问题可能是由于私服与中央仓库(或者其他第三方仓库)之间下载依赖时网络故障导致的.

2.2 解决方法:(针对私服与中央仓库或第三方仓库之间网络传输问题)

1. 使用管理员登陆私服页面(Sonatype Nexus)
2. 在仓库的浏览页面找到下载失败的包, 右键菜单 "Update Index", 右键菜单 "Expire Cache"

[](http://static.oschina.net/uploads/space/2013/1206/162311_NoW2_1189023.png)

1. 按照PC与私服之间网络问题的解决步骤处理一遍

# spring

## 通过ClassPathXmlApplicationContext读取配置文件

ApplicationContext context =

**new** ClassPathXmlApplicationContext(**new** String[]

{ "file:D:/eclipse-jee-juno/mavenspace/maven.springmvc/src/main/webapp/WEB-INF/applicationContext.xml" });

SpringPublisher publisher = (SpringPublisher) context

.getBean("springPublisher");

## Windows下ActiveMQ下载、安装部署

1、下载：http://activemq.apache.org/download.html 最新Windows版本

2、安装

(1) 首先配置JAVA环境变量

JAVA\_HOME=D:\Program Files\Java\jdk1.5.0

CLASSPAHT=.;D:\Program Files\Java\jdk1.5.0\lib\tools.jar;D:\Program Files\Java\jdk1.5.0\lib\td.jar;D:\Program Files\Java\jdk1.5.0\lib

PATH=D:\Program Files\Java\jdk1.5.0\bin;

(2)直接解压至任意目录(例如：D:\apache-activemq-5.3.0)

3、启动ActiveMQ服务器：直接运行\bin\win32\activemq.bat

当运行成功后

4、打开ActiveMQ消息管理后台系统 http://localhost:8161/admin/

## 消息队列：Spring JMS

### [SPRING与JMS结合的实例](http://www.cnblogs.com/phoebus0501/archive/2011/02/28/1966896.html)

|  |
| --- |
| package jms.activemq.myexample.spring;  import java.util.Date;  import javax.jms.JMSException;  import javax.jms.Message;  import javax.jms.Session;  import javax.jms.TextMessage;  import org.springframework.jms.core.MessageCreator;    public class MyMessageCreator implements MessageCreator {        /\*\*       \* 消息序号       \*/      private int msgNo;        public MyMessageCreator(int no) {          this.msgNo = no;      }        @Override      public Message createMessage(Session session) throws JMSException {          TextMessage textMsg = session.createTextMessage();          textMsg.setText(new Date() + "第" + this.msgNo + "条消息发出");            return textMsg;      }    } |
| package jms.activemq.myexample.spring;    import javax.jms.\*;    /\*\*   \* Text消息监听   \*   \* @author xiaochuanyu   \*   \*/  public class TextListener implements MessageListener {        /\*\*       \* Casts the message to a TextMessage and displays its text.       \*       \* @param message       \*            the incoming message       \*/      public void onMessage(Message message) {          TextMessage msg = null;            try {              if (message instanceof TextMessage) {                  msg = (TextMessage) message;                  System.out.println("Reading message: " + msg.getText());              } else {                  System.out.println("Message of wrong type: "                          + message.getClass().getName());              }          } catch (JMSException e) {              System.out.println("JMSException in onMessage(): " + e.toString());          } catch (Throwable t) {              System.out.println("Exception in onMessage():" + t.getMessage());          }      }  } |

|  |
| --- |
| package jms.activemq.myexample.spring;    import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;    public class SpringJmsTestMain {        public static void main(String[] args) throws InterruptedException {            ApplicationContext context = new ClassPathXmlApplicationContext(                  new String[] { "SpringJms/SpringJms.xml" });            SpringPublisher publisher = (SpringPublisher) context                  .getBean("springPublisher");          publisher.start();      }  } |
| package jms.activemq.myexample.spring;    import javax.jms.Destination;    import org.springframework.jms.core.JmsTemplate;    public class SpringPublisher {      /\*\*       \* Jms模板       \*/      private JmsTemplate template;        /\*\*       \* Topic       \*/      private Destination topic;        public JmsTemplate getTemplate() {          return template;      }        public void setTemplate(JmsTemplate template) {          this.template = template;      }        public Destination getTopic() {          return topic;      }        public void setTopic(Destination topic) {          this.topic = topic;      }        /\*\*       \* Start       \*       \* @throws InterruptedException       \*/      public void start() throws InterruptedException {            int messageCount = 10;            while ((--messageCount) > 0) {              sendMessage(messageCount);              Thread.sleep(1000);          }      }        /\*\*       \* 消息发送       \*/      protected void sendMessage(int msgNO) {            this.template.send(this.topic, new MyMessageCreator(msgNO));      }  } |

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 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-2.0.xsd">  
  
  
  
 <!-- jms 连接工厂 -->  
 <bean id="jmsFactory" class="org.apache.activemq.ActiveMQConnectionFactory">  
 <property name="brokerURL">  
 <value>tcp://localhost:61616</value>  
 </property>  
 </bean>  
  
  
 <!-- jms 连接池 -->  
   
 <!--   
 <bean id="pooledJmsFactory" class="org.apache.activemq.pool.PooledConnectionFactory">  
 <property name="connectionFactory">  
 <ref local="jmsFactory" />  
 </property>  
 </bean>  
 -->  
  
 <!-- jms Topic -->  
 <bean id="myTopic" class="org.apache.activemq.command.ActiveMQTopic"  
 autowire="constructor">  
 <constructor-arg value="STOCKS.JAVA" />  
 </bean>  
  
  
 <!-- 消息监听器 -->  
 <bean id="myTextListener" class="jms.activemq.myexample.spring.TextListener">  
 </bean>  
  
  
 <!-- jms Consumer -->  
 <bean id="javaConsumer"  
 class="org.springframework.jms.listener.DefaultMessageListenerContainer">  
 <property name="connectionFactory" ref="jmsFactory" />  
 <property name="destination" ref="myTopic" />  
 <property name="messageListener" ref="myTextListener" />  
 </bean>  
  
 <!-- jms 模板 -->  
 <bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">  
 <property name="connectionFactory">  
 <ref local="jmsFactory" />  
 </property>  
 </bean>  
  
  
 <!-- 消息发布器 -->  
 <bean id="springPublisher" class="jms.activemq.myexample.spring.SpringPublisher">  
 <property name="template">  
 <ref local="jmsTemplate" />  
 </property>  
 <property name="topic">  
 <ref local="myTopic" />  
 </property>  
 </bean>  
</beans>

### [Spring+JMS例子](http://www.blogjava.net/wangbing/archive/2010/03/31/317093.html)

第一: 在applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>    
 <bean id="connectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">  
   <property name="brokerURL">  
     <value>tcp://192.168.3.48:61616</value>  
   </property>  
    
 </bean>  
   
 <bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">  
    <property name="connectionFactory">  
      <ref bean="connectionFactory"/>  
    </property>  
 </bean>  
   
 <bean id="destination" class="org.apache.activemq.command.ActiveMQQueue">  
   <constructor-arg index="0">  
     <value>HelloWorldQueue</value>  
   </constructor-arg>  
 </bean>  
</beans>

2.写发送方

package ch13.JMS;

import javax.jms.Destination;  
import javax.jms.JMSException;  
import javax.jms.Message;  
import javax.jms.Session;

import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.jms.core.JmsTemplate;  
import org.springframework.jms.core.MessageCreator;

public class HelloWorldSender  
{  
 public static void main(String args[]) throws Exception  
 {  
  ApplicationContext context = new ClassPathXmlApplicationContext(new String[] { "ch13/JMS/applicationContext.xml" });  
  JmsTemplate jmsTemplate = (JmsTemplate) context.getBean("jmsTemplate");  
  Destination destination = (Destination) context.getBean("destination");  
  jmsTemplate.send  
  (  
   destination, new MessageCreator()  
   {  
    public Message createMessage(Session session) throws JMSException  
    {  
     return session.createTextMessage("大家好这个是测试！");  
    }  
   }  
  );  
 }

}  
3.写接收方

package ch13.JMS;

import javax.jms.Destination;  
import javax.jms.TextMessage;

import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.jms.core.JmsTemplate;

public class HelloWorldReciver  
{

 public static void main(String args[]) throws Exception  
 {  
  ApplicationContext context = new ClassPathXmlApplicationContext(new String[] { "ch13/JMS/applicationContext.xml" });  
  JmsTemplate jmsTemplate = (JmsTemplate) context.getBean("jmsTemplate");  
  Destination destination = (Destination) context.getBean("destination");  
  System.out.println("will wait:" + jmsTemplate.getReceiveTimeout()+ " seconds for message");  
  TextMessage msg = (TextMessage) jmsTemplate.receive(destination);  
  System.out.println("reviced msg is:" + msg.getText());  
 }

}

4.启动activemq中bin 下的activemq.bat

5.先运行 HelloWorldSender

6.再运行 HelloWorld

7.结果:

will wait:-1 seconds for message  
reviced msg is:大家好 这个是测试！

## Spring IOC

1、获取Bean的方法

ApplicationContext ctx = WebApplicationContextUtils.*getWebApplicationContext*(

request.getSession().getServletContext() );

TestService testService = (TestService)ctx.getBean("testService");

testService.sayHello();

# PowerDesigner

## pdb和pdm文件有什么区别

早期的PowerBuilder编译生成的动态链接库的后缀名是PDB，  
但后来PowerDesigner出来后，它的物理模型图（数据表结构）文件后缀名是PDM，但同时它会生成这个文件的备份文件，就是后缀名为PDB的文件。附文件名意思：  
PDB PowerBuilder Dynamic Library   
PDB PowerDesigner Physical Model Backup   
PDM Sybase Power Designer File

# Word 2007

## 标题与编号

1、设置一级标题

输入标题内容，选中内容，点击“标题1”，设置为一级标题。

在选中标题内容，右键-编号，选中要使用的编号即可。

后续同级别的标题，可使用格式刷进行设置。如现在要设置编号为3的标题，则选中编号为2的标题然后点格式刷，注意不能选中编号为1的标题点格式刷，以免编号错乱。

2、设置二级标题

输入标题内容，选中内容，点击“标题2”，设置为一级标题。

在选中标题内容，右键-编号，选中要使用的编号即可。

后续同级别的标题，可使用格式刷进行设置。如现在要设置编号为3的标题，则选中编号为2的标题然后点格式刷，注意不能选中编号为1的标题点格式刷，以免编号错乱。

# 版本管理

## git

## github

### 注册并登陆github

https://github.com/

### 新建仓库







### 配置仓库的用户和key







**在要配置的用户电脑上找到id\_rsa.pub文件，例如C:\Users\sunshine\.ssh\id\_rsa.pub**





### 克隆仓库到本地





### add、commit、push操作

新建文本文档.txt









github的push操作，是针对整个分支的。如果这个分支此前commit过多次，不能选择只push其中某次commit的内容，而是会全部commit的内容一起push.





push成功。

当key还没有配置的时候push操作会出现这个异常：



### 在多个github账号上添加SSH公钥

如果一台计算机的SSH公钥已经在一个github账号（仓库）上配置了，现在又想在另一个账号（仓库）上也配置这个公钥，则github会提示这个信息：Key is already in use.



那么如何在在多个github账号上添加SSH公钥呢？下面是一篇博文（http://justjavac.com/git/2012/04/13/multiple-ssh-keys.html）。

--------------------------------------------------------博文开始--------------------------------------------------------

github使用SSH与客户端连接。如果是单用户（first），生成密钥对后，将公钥保存至github， 每次连接时SSH客户端发送本地私钥（默认~/.ssh/id\_rsa）到服务端验证。 单用户情况下，连接的服务器上保存的公钥和发送的私钥自然是配对的。

但是如果是多用户（first，second），我们在连接到second的帐号时， second保存的是自己的公钥，但是SSH客户端依然发送默认私钥，即first的私钥， 那么这个验证自然无法通过。 不过，要实现多帐号下的SSH key切换在客户端做一些配置即可。

首先cd到~/.ssh 使用 ssh-keygen -t rsa -C 'second@mail.com' 生成新的SSH key：id\_rsa\_second，生成完后将新的SSH public key添加到github。

ssh-keygen -t -rsa -C 'second@mail.com'

默认SSH只会读取id\_rsa，所以为了让SSH识别新的私钥，需要将其添加到SSH agent

ssh-add ～/.ssh/id\_rsa\_second

该命令如果报错：Could not open a connection to your authentication agent. 无法连接到ssh agent，可执行ssh-agent bash命令后再执行ssh-add命令。

完成以上步骤后在~/.ssh目录创建config文件，该文件用于配置私钥对应的服务器。内容如下：

# Default github user(first@mail.com)

Host github.com

HostName github.com

User git

IdentityFile C:/Users/username/.ssh/id\_rsa

# second user(second@mail.com)

Host github-second

HostName github.com

User git

IdentityFile C:/Users/username/.ssh/id\_rsa\_second

配置完成后，在连接非默认帐号的github仓库时，远程库的地址要对应地做一些修改， 比如现在添加second帐号下的一个仓库test，则需要这样添加：

git remote add test git@github-second:second/test.git

#并非原来的git@github.com:second/test.git

这样每次连接都会使用id\_rsa\_second与服务器进行连接。至此，大功告成！

**注意：** github根据配置文件的user.email来获取github帐号显示author信息， 所以对于多帐号用户一定要记得将user.email改为相应的email(second@mail.com)。

--------------------------------------------------------博文结束--------------------------------------------------------

以下是根据博文介绍在本机实际操作的过程记录：

用git命令生成新的公钥并添加公钥：



config文件内容：

# Default github user(first@mail.com)

Host github.com

HostName github.com

User git

IdentityFile C:/Users/sunshine/.ssh/id\_rsa

# second user(second@mail.com)

Host github-second

HostName github.com

User git

IdentityFile C:/Users/sunshine/.ssh/id\_rsa\_second

将新生成的公钥配置到github



使用替换后的URL克隆github仓库：



克隆成功：



### 如何删除github仓库

1、例如要删除项目te，登陆后进入到该项目的页面，点击右边的Settings:



2、进入项目的Setting页面后，页面拉到最下面，就可以看到Delete this repository按钮。



## svn