1 回顾

1.1 Maven 的好处

节省空间 对 jar 包做了统一管理 依赖管理
一键构建
可跨平台
应用在大型项目可提高开发效率

1.2 Maven 安装部署配置

1.3 Maven 的仓库

本地仓库
远程仓库（私服）
中央仓库

1.4 添加依赖

从网络上搜索
在本地重建索引，以索引的方式搜索

1.5 项目构建

1.6 依赖范围

Compile struts2 框架 jar
Provided jsp-api.jar 重点
Runtime 数据库驱动包
Test junit.jar

1.7 总结

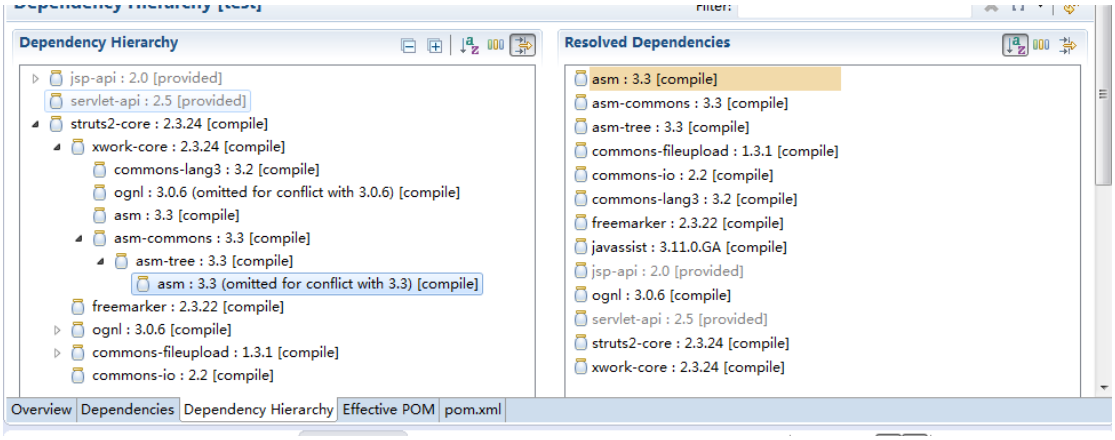
<modelVersion>
坐标　GAV
<groupId>cn.itcast</groupId>
<artifactId>ssh</artifactId>
<version>0.0.1-SNAPSHOT</version>
Packaging　打包方式
Jar　war　pom
<dependencies>
 <dependency>
<build>　里面放的是插件

<plugins>
 <plugin>

2 整合 ssh 框架

2.1 依赖传递

只添加了一个 struts2-core 依赖，发现项目中出现了很多 jar，
这种情况 叫 依赖传递



2.2 依赖版本冲突的解决

1、 第一声明优先原则

```
<dependencies>
  <!--    spring-beans-4.2.4 -->
      <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>4.2.4.RELEASE</version>
      </dependency>


<!--    spring-beans-3.0.5 -->
      <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-spring-plugin</artifactId>
            <version>2.3.24</version>
      </dependency>
```

2、 路径近者优先原则
自己添加 jar 包

```
      <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>4.2.4.RELEASE</version>
      </dependency>
```

3、 排除原则

```
      <dependency>
            <groupId>org.apache.struts</groupId>
            <artifactId>struts2-spring-plugin</artifactId>
            <version>2.3.24</version>
            <exclusions>
              <exclusion>
                <groupId>org.springframework</groupId>
                <artifactId>spring-beans</artifactId>
              </exclusion>
            </exclusions>
      </dependency>
```

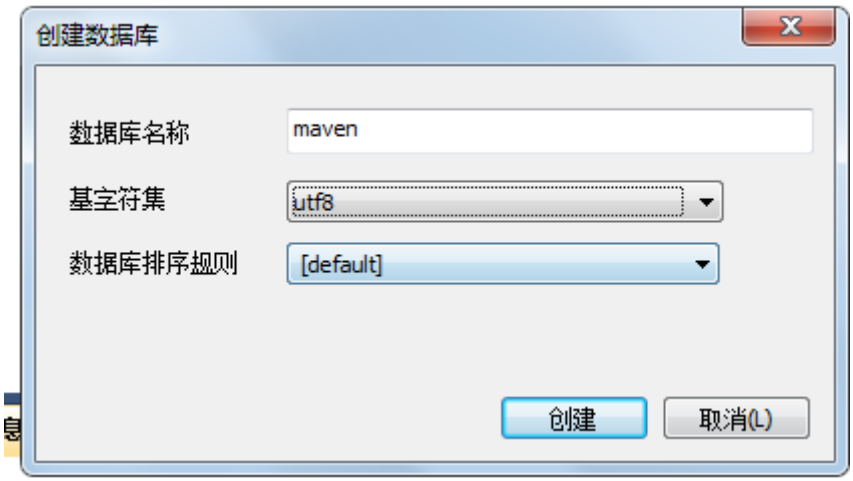
4、 版本锁定原则

```
<properties>
        <spring.version>4.2.4.RELEASE</spring.version>
        <hibernate.version>5.0.7.Final</hibernate.version>
        <struts.version>2.3.24</struts.version>
    </properties>

    <!-- 锁定版本，struts2-2.3.24、spring4.2.4、hibernate5.0.7 -->
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>${spring.version}</version>
            </dependency>
    </dependencies>
    </dependencyManagement>
```

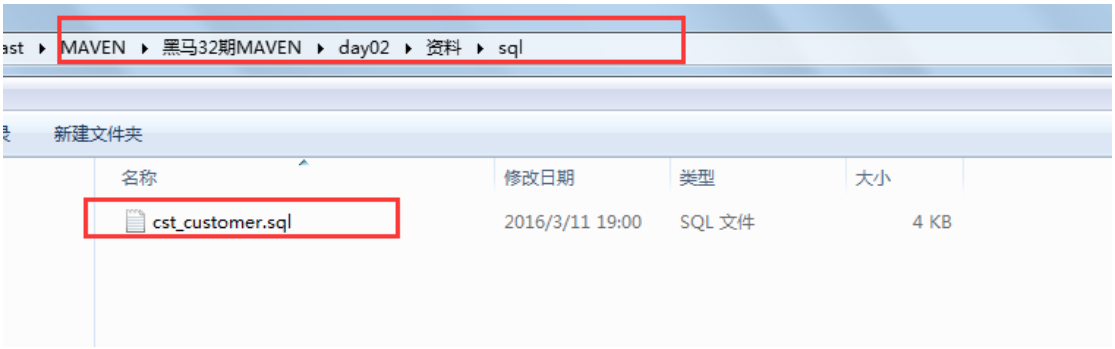
需求：
传客户 ID 页面上显示客户信息
准备数据库

2.3 构建项目

1、 创建数据库，



2、 执行准备好的 sql 脚本
Sql 脚本的位置：



3、 完善 pom.xml 文件，把 ssh 相关的依赖都添加上去

```
<!-- 属性 -->
    <properties>
        <spring.version>4.2.4.RELEASE</spring.version>
        <hibernate.version>5.0.7.Final</hibernate.version>
        <struts.version>2.3.24</struts.version>
    </properties>

    <!-- 锁定版本，struts2-2.3.24、spring4.2.4、hibernate5.0.7 -->
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-aspects</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-orm</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-test</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
```

```
                <artifactId>spring-web</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>${hibernate.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.struts</groupId>
                <artifactId>struts2-core</artifactId>
                <version>${struts.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.struts</groupId>
                <artifactId>struts2-spring-plugin</artifactId>
                <version>${struts.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <!-- 依赖管理 -->
    <dependencies>
        <!-- spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </dependency>
        <!-- hibernate -->
```

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
</dependency>

<!-- 数据库驱动 -->
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.6</version>
        <scope>runtime</scope>
</dependency>
<!-- c3p0 -->

<dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>
        <version>0.9.1.2</version>
</dependency>



<!-- 导入 struts2 -->
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-spring-plugin</artifactId>
</dependency>

<!-- servlet jsp -->
<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.5</version>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jsp-api</artifactId>
        <version>2.0</version>
        <scope>provided</scope>
</dependency>
```

```
        <!-- 日志 -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.2</version>
        </dependency>
        <!-- junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.9</version>
            <scope>test</scope>
        </dependency>
        <!-- jstl -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- 设置编译版本为 1.7 -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>

            <!-- maven 内置 的 tomcat6 插件 -->
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>tomcat-maven-plugin</artifactId>
                <version>1.1</version>
                <configuration>
                    <!-- 可以灵活配置工程路径 -->
                    <path>/ssh</path>
                    <!-- 可以灵活配置端口号 -->
                    <port>8080</port>
```

```
                </configuration>
            </plugin>
        </plugins>
    </build>
```

4、 完成实体类代码

```
package cn.itcast.entity;

public class Customer {

    private Long custId;
    private String custName;
    private Long custUserId;
    private Long custCreateId;
    private String custIndustry;
    private String custLevel;
    private String custLinkman;
    private String custPhone;
    private String custMobile;
    public Long getCustId() {
        return custId;
    }
    public void setCustId(Long custId) {
        this.custId = custId;
```

5、 完成 dao 代码

接口

```
package cn.itcast.dao;

import cn.itcast.entity.Customer;

public interface CustomerDao {

    public Customer getById(Long id);

}
```

实现类

```
package com.itcast.dao.impl;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;
import cn.itcast.dao.CustomerDao;
import cn.itcast.entity.Customer;
```

```
public class CustomerDaoImpl extends HibernateDaoSupport implements CustomerDao {
	@Override
	public Customer getById(Long id) {
		return this.getHibernateTemplate().get(Customer.class, id);
	}
}
```

6、 完成 service 代码

接口

```
package com.itcast.service;
import cn.itcast.entity.Customer;
public interface CustomerService {
	public Customer getById(Long id);
}
```

实现类

```
package com.itcast.service.impl;

import com.itcast.service.CustomerService;

import cn.itcast.dao.CustomerDao;
import cn.itcast.entity.Customer;

public class CustomerServiceImpl implements CustomerService {
	private CustomerDao	customerDao;
	public void setCustomerDao(CustomerDao customerDao) {
		this.customerDao = customerDao;
	}
	@Override
	public Customer getById(Long id) {
		return customerDao.getById(id);
	}
}
```

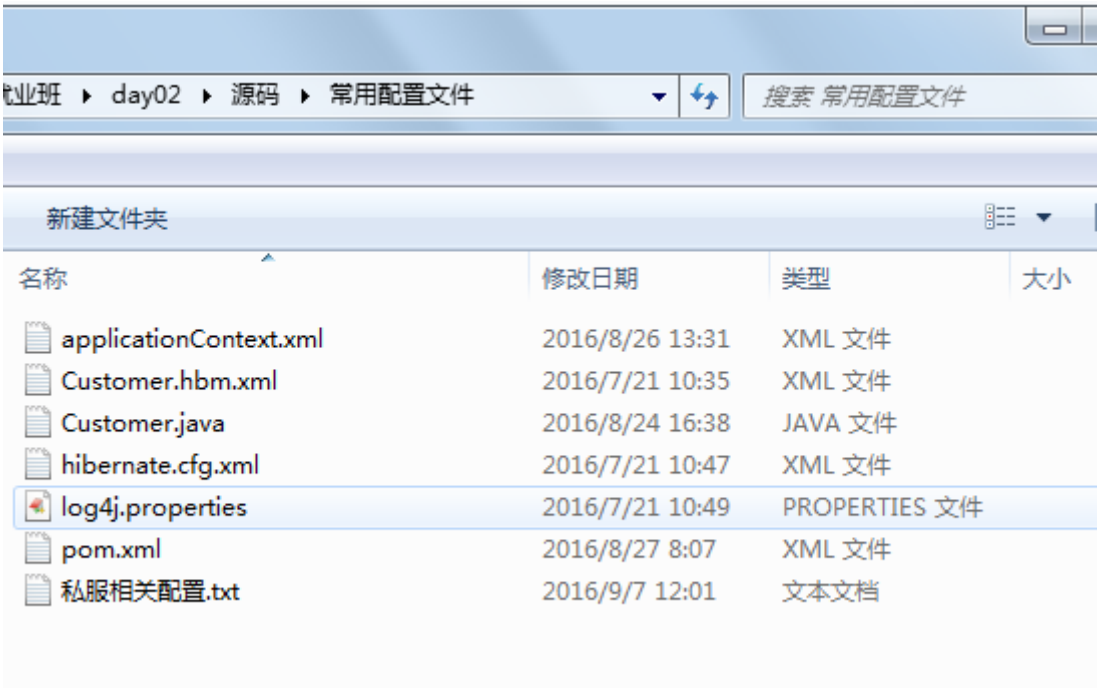
7、 完成 action 代码

```
package cn.itcast.action;
import com.itcast.service.CustomerService;
import com.opensymphony.xwork2.ActionSupport;
import cn.itcast.entity.Customer;
public class CutomerAction extends ActionSupport {
	//两个成员变量
	private Customer	customer;
```

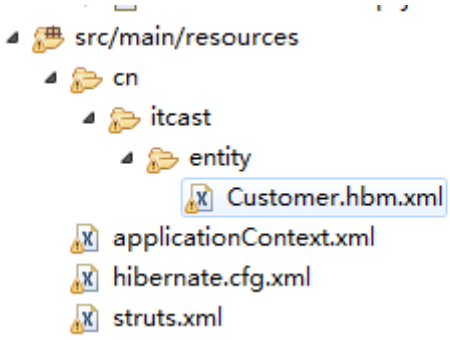
```
    private Long custId;
    public Customer getCustomer() {
        return customer;
    }
    public void setCustomer(Customer customer) {
        this.customer = customer;
    }
    private CustomerService customerService;
    public void setCustomerService(CustomerService customerService) {
        this.customerService = customerService;
    }
    public Long getCustId() {
        return custId;
    }
    public void setCustId(Long custId) {
        this.custId = custId;
    }
    public String findById(){
        customer = customerService.getById(custId);
        return SUCCESS;
    }
}
```

8、 拷贝配置文件并修改

从如下图位置拿到配置文件



放入到 src/main/resources 目录中



修改内容 略

9、 修改 web.xml 添加 spring 的监听

```
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>classpath:applicationContext.xml</param-value>
</context-param>
```

10、 运行项目

3 分模块开发

依赖范围对依赖传递造成的影响（了解）

是因为依赖会有依赖范围，依赖范围对传递依赖也有影响，例如有 A、B、C，A 依赖 B、B 依赖 C，C 可能是 A 的传递依赖，如下图：

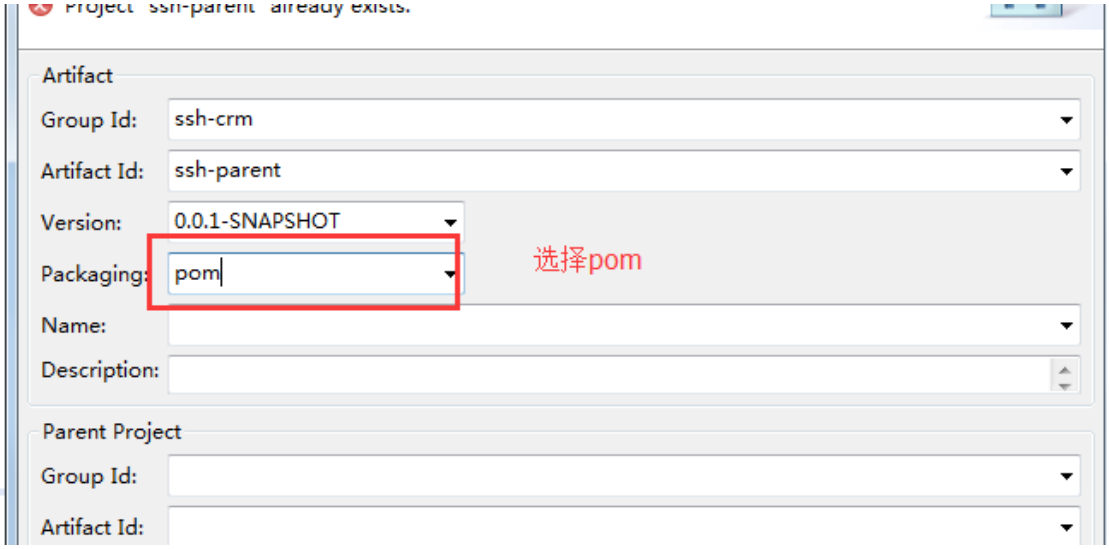
直接依赖 \ 传递依赖	compile	provided	runtime	test
compile	compile	-	runtime	-
provided	provided	provided	provided	-
runtime	runtime	-	runtime	-
test	test	-	test	-

最左边一列为直接依赖，理解为 A 依赖 B 的范围，最顶层一行为传递依赖，理解为 B 依赖 C 的范围，行与列的交叉即为 A 传递依赖 C 的范围。

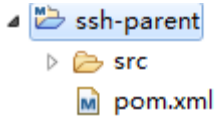
父工程来管理 聚合

3.1 创建父工程：

1、



2、创建出的父工程如下



3、在 pom.Xml 中添加以下信息：

```
<!-- 属性 -->
    <properties>
        <spring.version>4.2.4.RELEASE</spring.version>
        <hibernate.version>5.0.7.Final</hibernate.version>
        <struts.version>2.3.24</struts.version>
    </properties>

    <!-- 锁定版本，struts2-2.3.24、spring4.2.4、hibernate5.0.7 -->
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-aspects</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
```

```
                <artifactId>spring-orm</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-test</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-web</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>${hibernate.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.struts</groupId>
                <artifactId>struts2-core</artifactId>
                <version>${struts.version}</version>
            </dependency>
            <dependency>
                <groupId>org.apache.struts</groupId>
                <artifactId>struts2-spring-plugin</artifactId>
                <version>${struts.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <!--  依赖管理  -->
    <dependencies>
        <!-- spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
```

```
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
</dependency>
<!-- hibernate -->
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
</dependency>

<!-- 数据库驱动 -->
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.6</version>
        <scope>runtime</scope>
</dependency>
<!-- c3p0 -->

<dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>
        <version>0.9.1.2</version>
</dependency>


<!-- 导入 struts2 -->
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-spring-plugin</artifactId>
</dependency>

<!-- servlet jsp -->
<dependency>
        <groupId>javax.servlet</groupId>
```

```
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.0</version>
            <scope>provided</scope>
        </dependency>
        <!-- 日志 -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.2</version>
        </dependency>
        <!-- junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.9</version>
            <scope>test</scope>
        </dependency>
        <!-- jstl -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- 设置编译版本为 1.7 -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
```

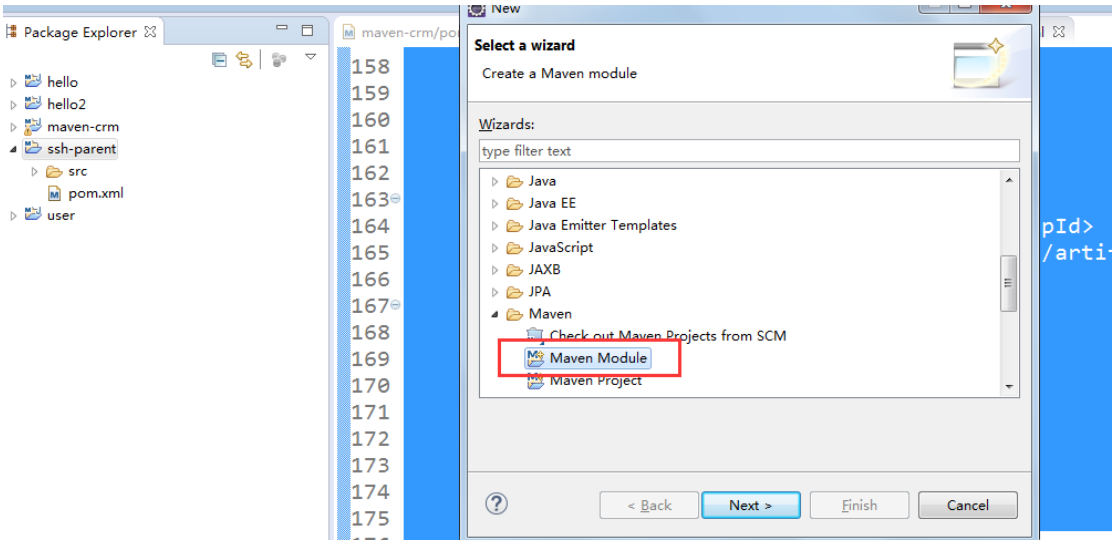
```
            <!-- maven 内置 的 tomcat6 插件 -->
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>tomcat-maven-plugin</artifactId>
                <version>1.1</version>
                <configuration>
                    <!-- 可以灵活配置工程路径 -->
                    <path>/ssh</path>
                    <!-- 可以灵活配置端口号 -->
                    <port>8080</port>
                </configuration>
            </plugin>

        </plugins>
    </build>
```

4、发布到本地仓库

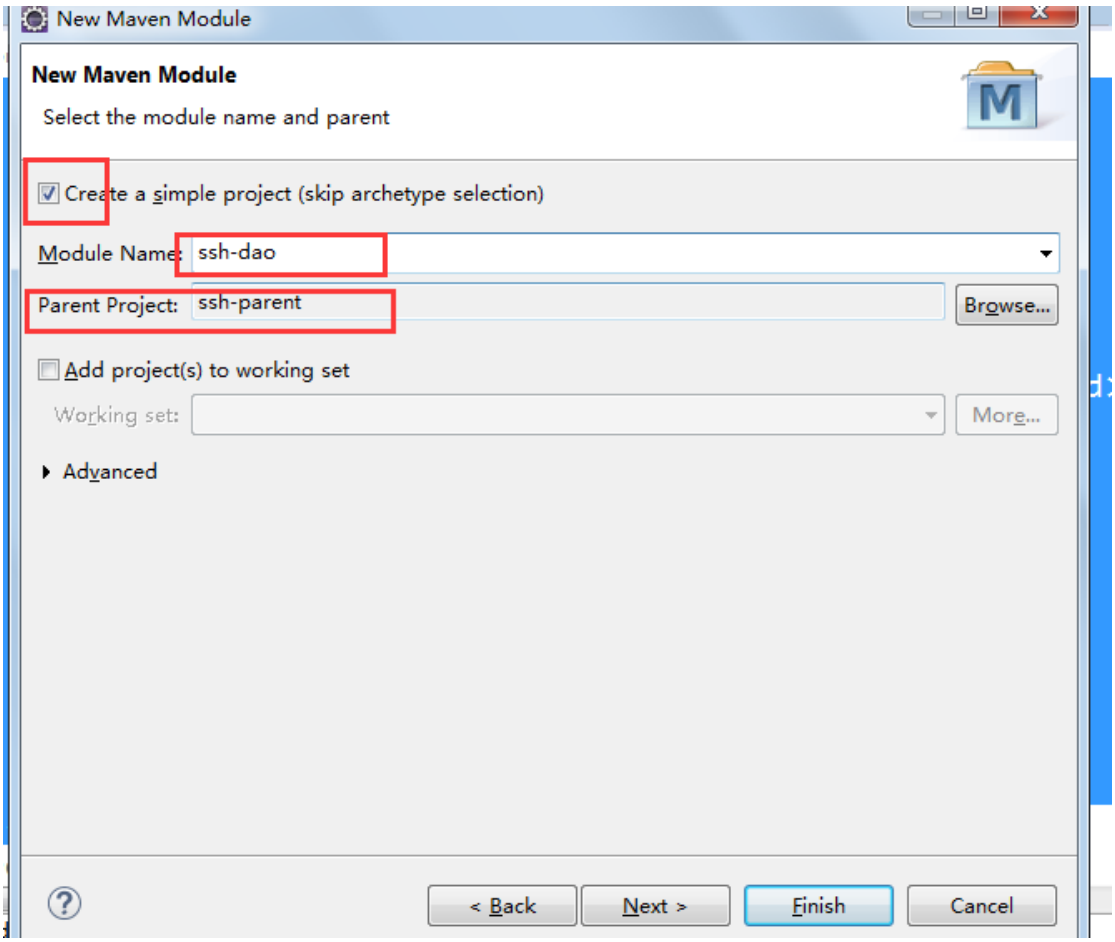
dao service web

3.2 创建 dao 子模块

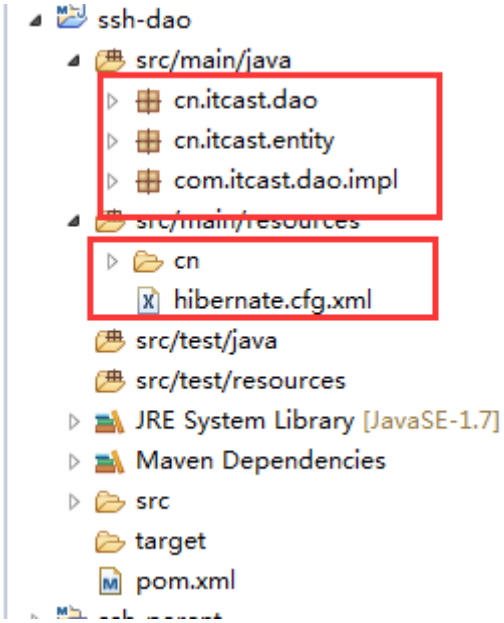
1、在 ssh-parent 项目上右击 ， 创建时选择 Maven Module



2、填写子模块名称 ssh-dao



3、把属于 dao 的代码拷贝到 该模块中：

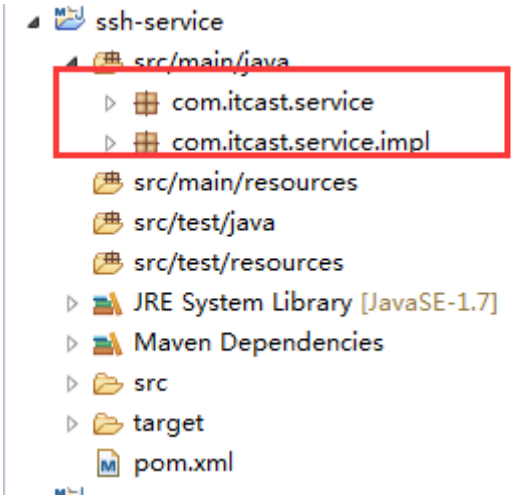


4、完成后发布到本地仓库中

3.3 创建 service 子模块

1、创建方式如上：

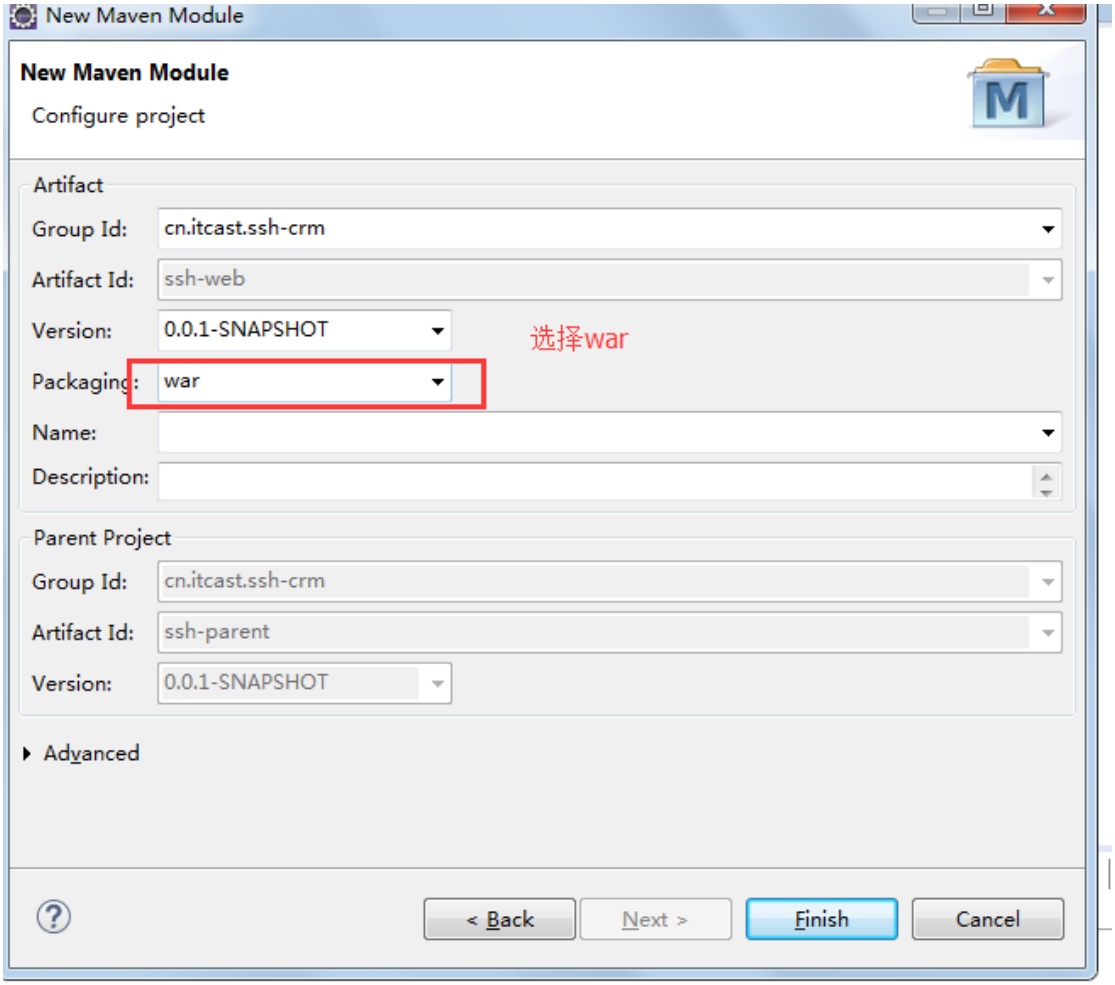
2、把属于 service 的代码拷贝到该工程中



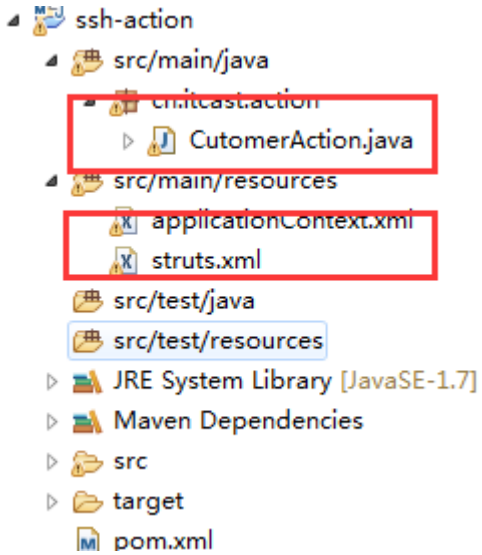
3、发布到本地仓库中

3.4 创建 Action 子模块

1、选择 war 的打包方式



5、 拷贝属于 action 的代码和配置文件

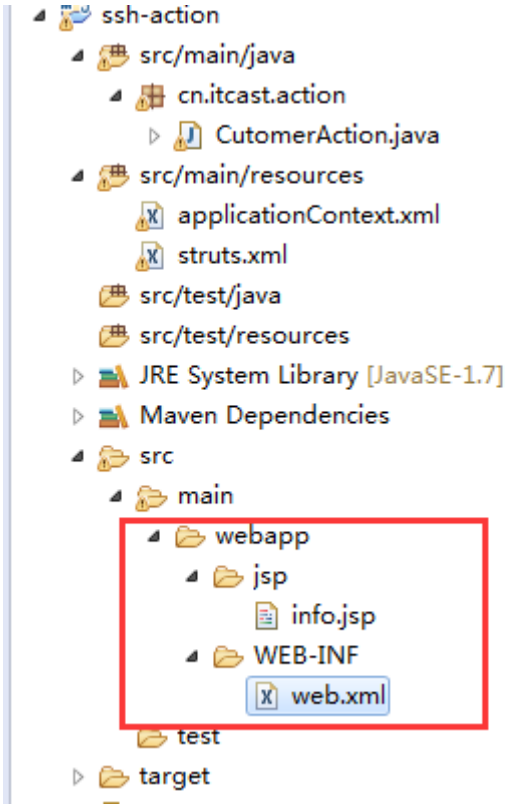


6、 修改 web.xml　添加 spring 监听

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

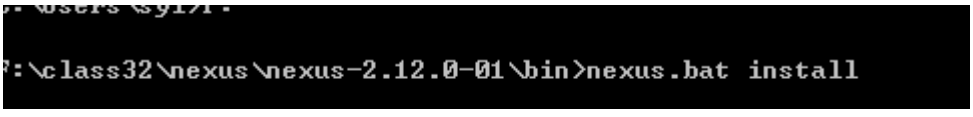
```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath*:applicationContext-*.xml</param-value>
</context-param>
```

4、添加页面：



4 私服 nexus

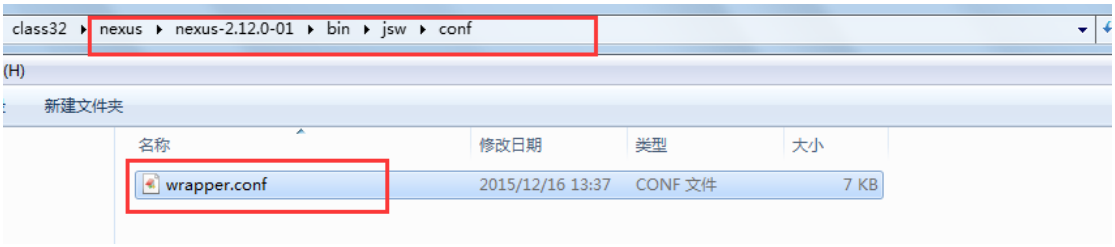
安装 nexus



启动服务

```
F:\class32\nexus\nexus-2.12.0-01\bin>nexus.bat start
wrapper  | Starting the nexus service...
wrapper  | Waiting to start...
wrapper  | Waiting to start...
wrapper  | Waiting to start...
wrapper  | Waiting to start...
wrapper  | Waiting to start...
wrapper  | nexus started.

F:\class32\nexus\nexus-2.12.0-01\bin>
```

启动失败的解决方法：

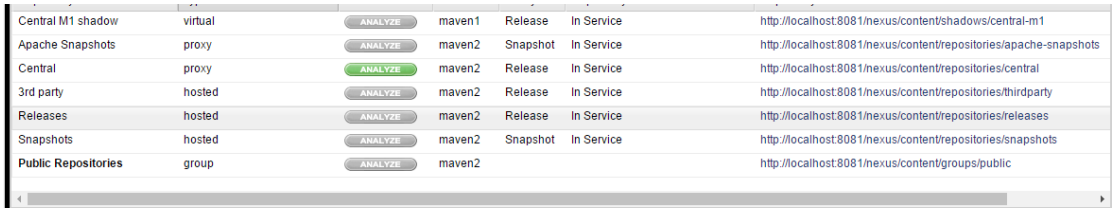



```
12
13  # Set the JVM executable
14  # (modify this to absolute path if you need a Java that is not on the OS path)
15  wrapper.java.command=E:\jdk1.7.0_72\bin\java.exe
16
17  # The main class that JSW will execute within JVM
18  wrapper.java.mainclass=org.sonatype.nexus.bootstrap.jsw.JswLauncher
19
```

登录 nexus

用户名/密码　admin/admin123

仓库类型



Central M1 shadow	virtual	ANALYZE	maven1	Release	In Service	http://localhost:8081/nexus/content/shadows/central-m1
Apache Snapshots	proxy	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central	proxy	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/central
3rd party	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Releases	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/snapshots
Public Repositories	group	ANALYZE	maven2			http://localhost:8081/nexus/content/groups/public

Virtual　虚拟仓库

Proxy　代理仓库

Hosted　宿主仓库　本地仓库

Group　组

需求：

把 dao 放到私服上，然后 service 从私服上下载

需求 ：将 ssh_dao 的这个工程打成 jar 包，并放入到私服上去.

4.1 上传 dao

第一步： 需要在客户端即部署 dao 工程的电脑上配置 maven 环境，并修改 settings.xml 文件，配置连接私服的用户和密码 。

此用户名和密码用于私服校验，因为私服需要知道上传都 的账号和密码 是否和私服中的账号和密码 一致。

```
<server>
  <id>releases</id>
  <username>admin</username>
  <password>admin123</password>
</server>
<server>
  <id>snapshots</id>
  <username>admin</username>
  <password>admin123</password>
</server>
```

第二步： 配置项目 pom.xml

配置私服仓库的地址，本公司的自己的 jar 包会上传到私服的宿主仓库，根据工程的版本号决定上传到哪个宿主仓库，如果版本为 release 则上传到私服的 release 仓库，如果版本为 snapshot 则上传到私服的 snapshot 仓库

```
<distributionManagement>
  <repository>
      <id>releases</id>
  <url>http://localhost:8081/nexus/content/repositories/releases/</url>
  </repository>
  <snapshotRepository>
      <id>snapshots</id>
  <url>http://localhost:8081/nexus/content/repositories/snapshots/</url>
  </snapshotRepository>
</distributionManagement>
```

注意：pom.xml 这里<id> 和 settings.xml 配置 <id> 对应！

第三步：执行 deploy 命令发布到私服

4.2 下载 dao

第一步 修改 settings.xml

```
<profile>
      <!--profile 的 id-->
     <id>dev</id>
     <repositories>
       <repository>
           <!--仓库 id，repositories 可以配置多个仓库，保证 id 不重复-->
         <id>nexus</id>
           <!--仓库地址，即 nexus 仓库组的地址-->
         <url>http://localhost:8081/nexus/content/groups/public/</url>
           <!--是否下载 releases 构件-->
         <releases>
            <enabled>true</enabled>
         </releases>
           <!--是否下载 snapshots 构件-->
         <snapshots>
            <enabled>true</enabled>
         </snapshots>
       </repository>
     </repositories>
       <pluginRepositories>
    <!-- 插件仓库，maven 的运行依赖插件，也需要从私服下载插件 -->
         <pluginRepository>
           <!-- 插件仓库的 id 不允许重复，如果重复后边配置会覆盖前边 -->
             <id>public</id>
             <name>Public Repositories</name>
             <url>http://localhost:8081/nexus/content/groups/public/</url>
         </pluginRepository>
     </pluginRepositories>
  </profile>
```

```
  <activeProfiles>
     <activeProfile>dev</activeProfile>
  </activeProfiles>
```

第二步 删除本地仓库中的 dao

第三步 update service 工程，出现以下信息说明已经成功

```
16/10/10 GMT+8下午12:07:54: [INFO] Update started
16/10/10 GMT+8下午12:07:54: [INFO] Using org.eclipse.m2e.jdt.JarLifecycleMapping lifecycle mapping for MavenProject: cn.itcast.ssh-maven:ssh-service:0.0.1-SNAPSHOT @ F:\class32\workspace\ssh-parent\ssh-ser
16/10/10 GMT+8下午12:07:54: [INFO] Downloading http://localhost:8081/nexus/content/groups/public/cn/itcast/ssh-maven/ssh-dao/0.0.1-SNAPSHOT/maven-metadata.xml
16/10/10 GMT+8下午12:07:54: [INFO] Downloaded http://localhost:8081/nexus/content/groups/public/cn/itcast/ssh-maven/ssh-dao/0.0.1-SNAPSHOT/maven-metadata.xml
16/10/10 GMT+8下午12:07:54: [INFO] Downloading http://localhost:8081/nexus/content/groups/public/cn/itcast/ssh-maven/ssh-dao/0.0.1-SNAPSHOT/ssh-dao-0.0.1-20161010.040242-1.pom
16/10/10 GMT+8下午12:07:54: [INFO] Downloaded http://localhost:8081/nexus/content/groups/public/cn/itcast/ssh-maven/ssh-dao/0.0.1-SNAPSHOT/ssh-dao-0.0.1-20161010.040242-1.pom
16/10/10 GMT+8下午12:07:55: [INFO] Downloading http://localhost:8081/nexus/content/groups/public/cn/itcast/ssh-maven/ssh-dao/0.0.1-SNAPSHOT/ssh-dao-0.0.1-20161010.040242-1.jar
16/10/10 GMT+8下午12:07:55: [INFO] Downloaded http://localhost:8081/nexus/content/groups/public/cn/itcast/ssh-maven/ssh-dao/0.0.1-SNAPSHOT/ssh-dao-0.0.1-20161010.040242-1.jar
16/10/10 GMT+8下午12:07:55: [INFO] Using org.maven.ide.eclipse.wtp.WarLifecycleMapping lifecycle mapping for MavenProject: cn.itcast.ssh-maven:ssh-web:0.0.1-SNAPSHOT @ F:\class32\workspace\ssh-parent\ss
16/10/10 GMT+8下午12:07:56: [INFO] Adding source folder /ssh-service/src/main/java
16/10/10 GMT+8下午12:07:56: [INFO] Adding resource folder /ssh-service/src/main/resources
16/10/10 GMT+8下午12:07:56: [INFO] Adding source folder /ssh-service/src/test/java
16/10/10 GMT+8下午12:07:56: [INFO] Adding resource folder /ssh-service/src/test/resources
16/10/10 GMT+8下午12:07:56: [INFO] Update completed: 2 sec
16/10/10 GMT+8下午12:07:57: [WARN] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
16/10/10 GMT+8下午12:07:57: [INFO] Copying 1 resource
16/10/10 GMT+8下午12:07:57: [WARN] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
16/10/10 GMT+8下午12:07:57: [INFO] Copying 0 resource
16/10/10 GMT+8下午12:07:57: [WARN] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
16/10/10 GMT+8下午12:07:57: [INFO] Copying 2 resources
16/10/10 GMT+8下午12:07:57: [WARN] Using platform encoding (GBK actually) to copy filtered resources, i.e. build is platform dependent!
16/10/10 GMT+8下午12:07:57: [INFO] Copying 0 resource
```