

Maven高级

课程大纲

第1章 回顾[理解]

c 1.1 Maven好处

1. 节省磁盘空间
2. 可以一键构建
3. 可以跨平台
4. 应用在大型项目时可以提高开发效率

1.2 安装配置 maven

注意：3.3+版本需要 jdkj.7+以上的支持

1.3 三种仓库

1. 本地仓库
2. 远程仓库（私服）
3. 中央仓库

1.4 常见的命令

1. Compile
2. Test
3. Package
4. Install
5. Deploy
6. Clean

1.5 坐标的书写规范

1. groupId 公司或组织域名的倒序
2. artifactId 项目名或模块名
3. version 版本号

1.6 如何添加坐标

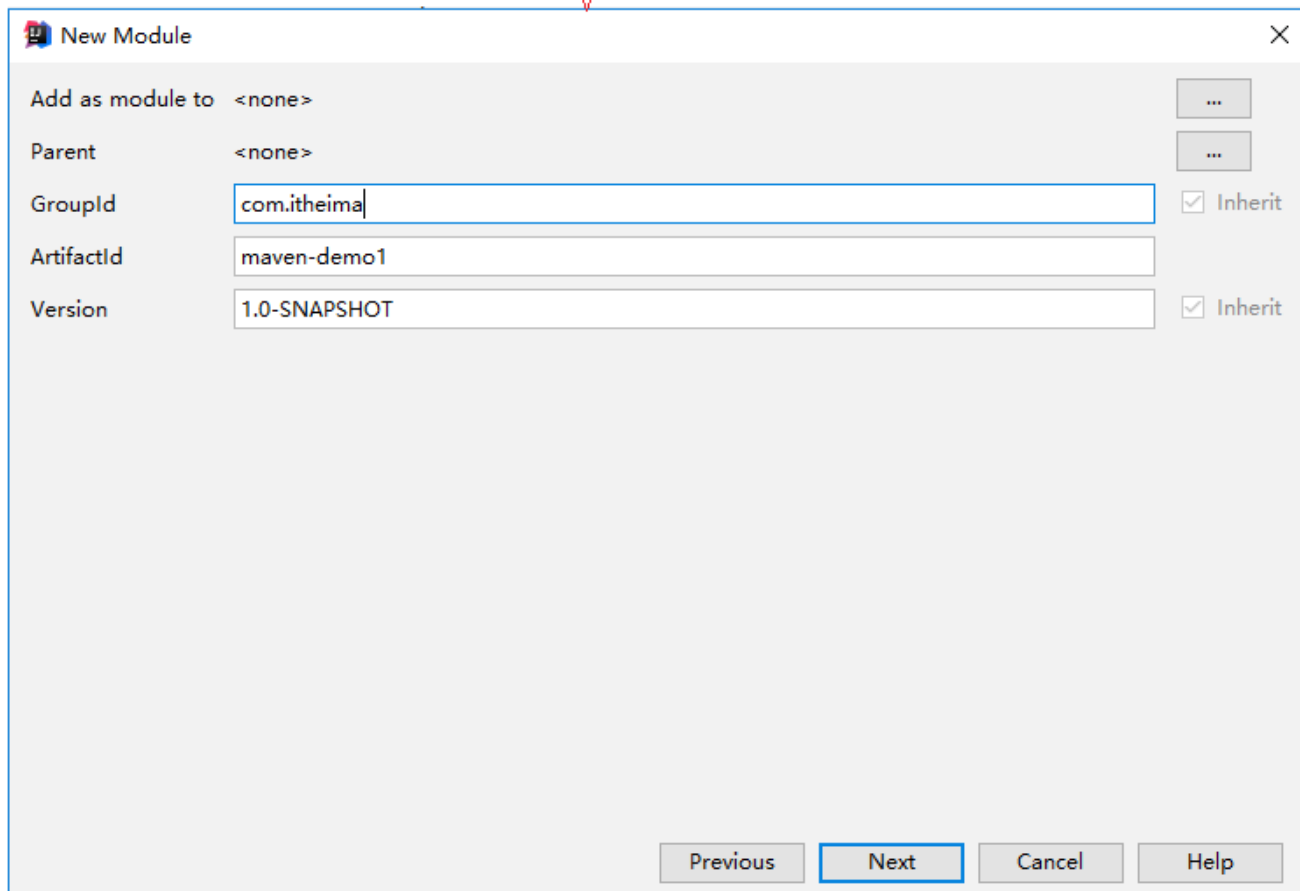
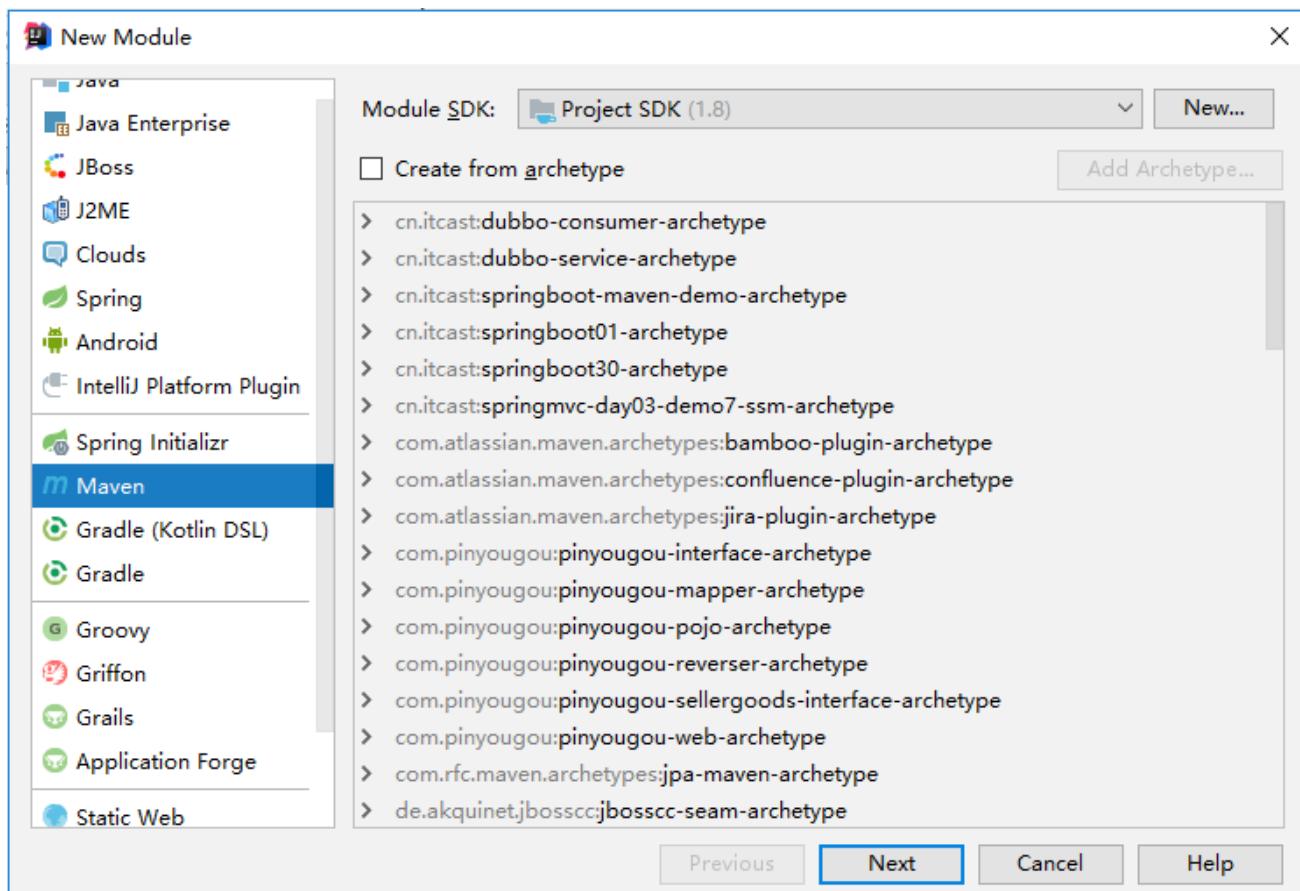
1. 在本地仓库中搜索
2. 互联网上搜，推荐网址 <http://www.mvnrepository.com/>

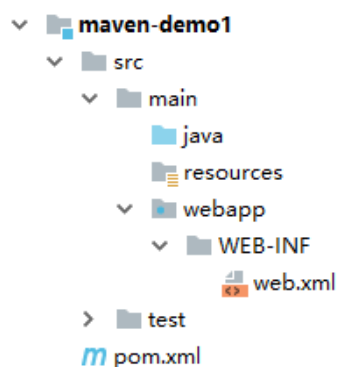
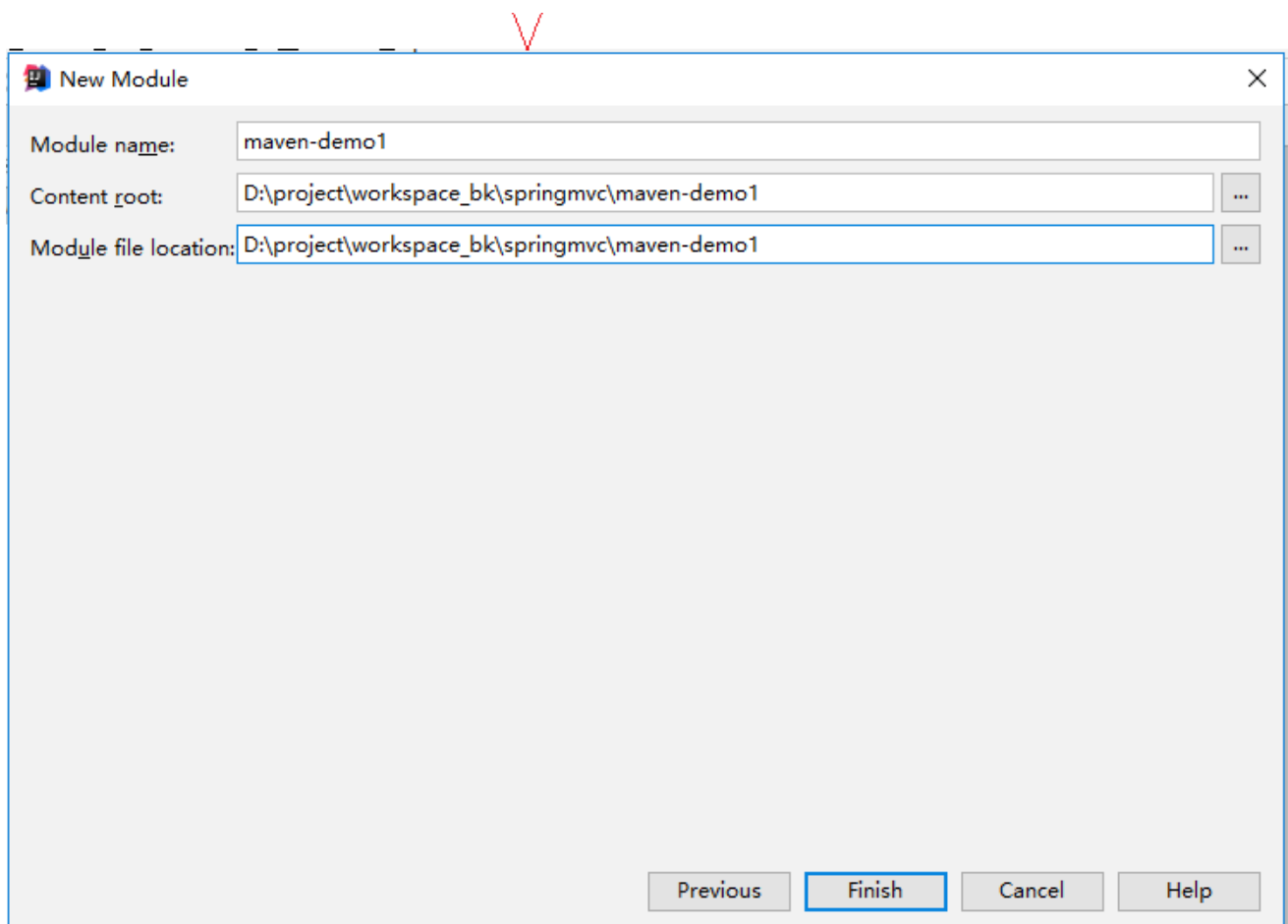
1.7 依赖范围

1. Compile
2. Test
3. Runtime
4. Provided

第2章Maven

创建一个war包工程，流程如下图：





2.1 jar包冲突：第一声明优先原则

哪个jar包在靠上的位置，这个jar包就是先声明的，先声明的jar包下的依赖包，可以优先引入项目中。

我们在pom.xml中引入如下坐标，分别是spring中不同的版本。

```
<!--导入相关依赖包-->
<dependencies>
  <!--引入spring-context，它所以来的包都会导入进来-->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.0.2.RELEASE</version>
```

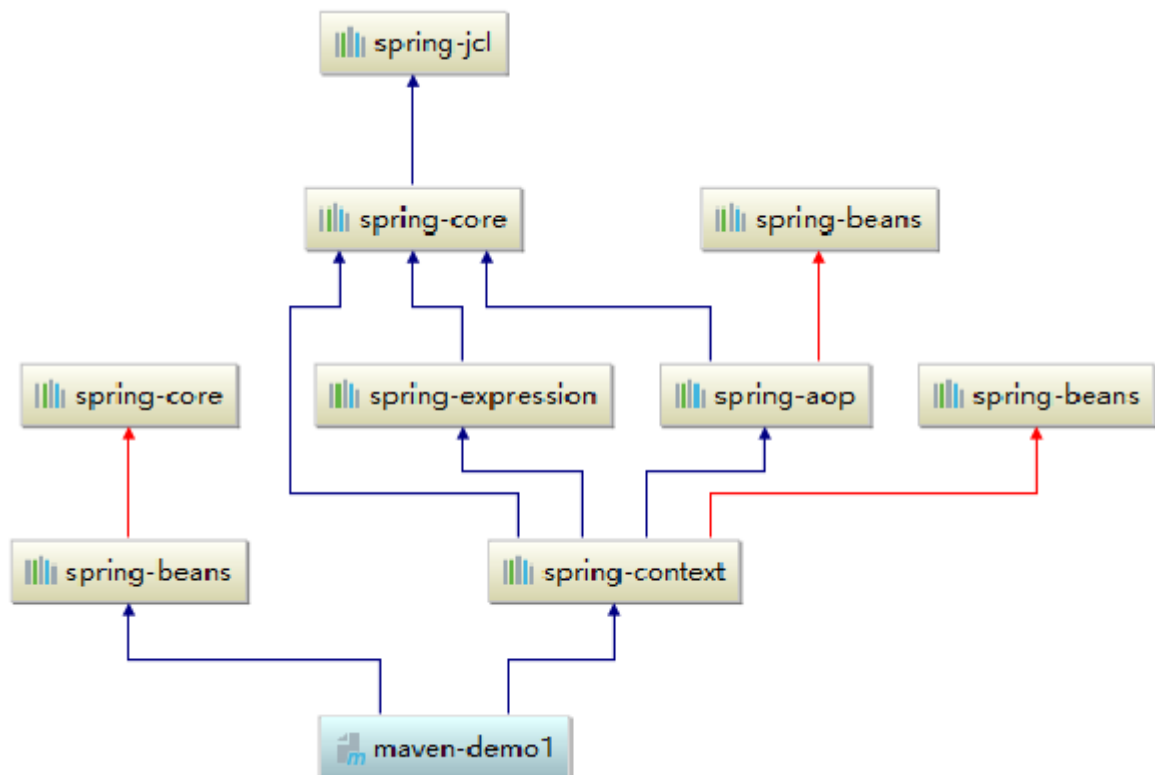
```

</dependency>

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

```







我们在控制面板的maven面板，点击查看依赖关系按钮，看到了包和包之间的依赖关系存在冲突，都使用了spring-core包，关系图如下：



我们再来看看他们依赖包的导入，发现导入的包却没有问题，包使用的都是5.0.2的版本。

- > Maven: org.springframework:spring-aop:5.0.2.RELEASE
- > Maven: org.springframework:spring-beans:4.2.4.RELEASE
- > Maven: org.springframework:spring-context:5.0.2.RELEASE
- > Maven: org.springframework:spring-core:5.0.2.RELEASE
- > Maven: org.springframework:spring-expression:5.0.2.RELEASE
- > Maven: org.springframework:spring-jcl:5.0.2.RELEASE

我们把上面2个包的顺序调换后就变成了低版本的依赖导入。

- >  Maven: commons-logging:commons-logging:1.2
- >  Maven: org.springframework:spring-aop:5.0.2.RELEASE
- >  Maven: org.springframework:spring-beans:4.2.4.RELEASE
- >  Maven: org.springframework:spring-context:5.0.2.RELEASE
- >  Maven: org.springframework:spring-core:4.2.4.RELEASE
- >  Maven: org.springframework:spring-expression:5.0.2.RELEASE

2.2 jar包冲突：路径近者优先原则

直接依赖比传递依赖路径近，你那么最终进入项目的jar包会是路径近的直接依赖包。

直接依赖：项目中直接导入的jar包就是项目的直接依赖包。

传递依赖：项目中没有直接导入的jar包，可以通过中直接依赖包传递到项目中去。







修改jar包，直接引入依赖spring-core

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

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.0.2.RELEASE</version>
  </dependency>

  <!--引入直接依赖-->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>4.2.8.RELEASE</version>
  </dependency>
</dependencies>
```

此时优先引入的是直接依赖的引用

- >  Maven: commons-logging:commons-logging:1.2
- >  Maven: org.springframework:spring-aop:5.0.2.RELEASE
- >  Maven: org.springframework:spring-beans:4.2.4.RELEASE
- >  Maven: org.springframework:spring-context:5.0.2.RELEASE
- >  Maven: org.springframework:spring-core:4.2.8.RELEASE
- >  Maven: org.springframework:spring-expression:5.0.2.RELEASE

2.3 jar包冲突：直接排除法

当我们需要排除某个jar包的依赖时，在配置exclusions标签的时候，内部可以不写版本号。pom.xml依赖如下：







```

<!--导入相关依赖包-->
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>4.2.4.RELEASE</version>
        <!--直接排除-->
        <exclusions>
            <exclusion>
                <groupId>org.springframework</groupId>
                <artifactId>spring-core</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.0.2.RELEASE</version>
    </dependency>
</dependencies>

```

依赖导入的jar包如下

- >  Maven: org.springframework:spring-aop:5.0.2.RELEASE
- >  Maven: org.springframework:spring-beans:4.2.4.RELEASE
- >  Maven: org.springframework:spring-context:5.0.2.RELEASE
- >  Maven: org.springframework:spring-core:5.0.2.RELEASE
- >  Maven: org.springframework:spring-expression:5.0.2.RELEASE
- >  Maven: org.springframework:spring-jcl:5.0.2.RELEASE

第3章 SSM回顾

2.1 ssmweb引入pom.xml

```

<groupId>com.itheima</groupId>
<artifactId>ssmweb</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>war</packaging>

<properties>
    <spring.version>5.0.2.RELEASE</spring.version>
    <slf4j.version>1.6.6</slf4j.version>
    <log4j.version>1.2.12</log4j.version>
    <mysql.version>5.1.6</mysql.version>
    <mybatis.version>3.4.5</mybatis.version>
    <aspectjweaver.version>1.6.8</aspectjweaver.version>
    <junit.version>4.12</junit.version>
    <jsp-api.version>2.0</jsp-api.version>
    <servlet-api.version>2.5</servlet-api.version>
    <jstl.version>1.2</jstl.version>

```

```
<mybatis-spring.version>1.3.0</mybatis-spring.version>
<druid.version>1.0.9</druid.version>
<!--文件的编码格式-->
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
<maven.compiler.source>1.8</maven.compiler.source>
<maven.compiler.target>1.8</maven.compiler.target>
</properties>

<!--jar包管理-->
<!--引入依赖-->
<dependencies>
    <!-- spring -->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>${aspectjweaver.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <!--spring包-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <!--用于SpringMVC-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <!--用于数据库源相关操作-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>

        <artifactId>spring-tx</artifactId>
```



```
        <version>${spring.version}</version>
    </dependency>

    <!--ServletAPI-->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>${servlet-api.version}</version>
        <scope>provided</scope>
    </dependency>

    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>${jsp-api.version}</version>
        <scope>provided</scope>
    </dependency>

    <!--jstl标签-->
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>${jstl.version}</version>
    </dependency>

    <!--MySQL数据库驱动-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>${mysql.version}</version>
    </dependency>

    <!--测试框架-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>${junit.version}</version>
        <scope>compile</scope>
    </dependency>

    <!-- log start -->
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>${log4j.version}</version>

    </dependency>
```

```

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>${slf4j.version}</version>
</dependency>

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>${slf4j.version}</version>
</dependency>
<!-- log end -->

<!--mybatis-->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>${mybatis.version}</version>
</dependency>

<!--MyBatis集成Spring-->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>${mybatis-spring.version}</version>
</dependency>

<!--数据源-->
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid</artifactId>
  <version>${druid.version}</version>
</dependency>
</dependencies>
<build>
  <!--插件-->
  <plugins>
    <!--tomcat插件-->
    <plugin>
      <groupId>org.apache.tomcat.maven</groupId>
      <artifactId>tomcat7-maven-plugin</artifactId>
      <version>2.2</version>
      <!--插件使用的相关配置-->
      <configuration>
        <!--端口号-->
        <port>18081</port>
        <!--写当前项目的名字(虚拟路径),如果写/,那么每次访问项目就不需要加项目名字了-->
        <path>/</path>
        <!--解决get请求乱码-->
        <uriEncoding>UTF-8</uriEncoding>
      </configuration>
    </plugin>
  
```

```
</plugins>
</build>
```

2.2 dao层

2.2.1创建Items实体类

```
package com.itheima.domain;
import java.util.Date;

/**
 * pojo
 */
public class Items {
    private Integer id;
    private String name;
    private Float price;
    private String pic;
    private Date createtime;
    private String detail;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Float getPrice() {
        return price;
    }

    public void setPrice(Float price) {
        this.price = price;
    }

    public String getPic() {
        return pic;
    }

    public void setPic(String pic) {
        this.pic = pic;
    }
}
```

```

    public Date getCreatetime() {
        return createtime;
    }

    public void setCreatetime(Date createtime) {
        this.createtime = createtime;
    }

    public String getDetail() {
        return detail;
    }

    public void setDetail(String detail) {
        this.detail = detail;
    }
}

```

2.2.2 ItemsDao.java

```

package com.itheima.dao;
import com.itheima.domain.Items;
public interface ItemsDao {
    public Items findById(Integer id);
}

```

2.2.3 ItemsDao.xml

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.itheima.dao.ItemsDao">
    <select id="findById" parameterType="Integer" resultType="Items">
        select * from items where id = #{id}
    </select>
</mapper>

```

2.2.4 applicationContext.xml

```

<?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:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:cache="http://www.springframework.org/schema/cache"
    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
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd
    http://www.springframework.org/schema/cache
    http://www.springframework.org/schema/cache/spring-cache.xsd">

<!-- 数据库连接池 -->
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"
    destroy-method="close">
    <property name="url" value="jdbc:mysql://127.0.0.1:3306/maven?characterEncoding=utf8" />
    <property name="username" value="root" />
    <property name="password" value="admin" />
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
</bean>

<!-- SqlSessionFactoryBean -->
<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <!--别名-->
    <property name="typeAliasesPackage" value="com.itheima.domain"/>
</bean>

<!-- 配置接口扫描包 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <!--指定接口位置-->
    <property name="basePackage" value="com.itheima.dao"/>
</bean>

</beans>

```

2.2.5 log4j.properties

```

# Set root category priority to INFO and its only appender to CONSOLE.
#log4j.rootCategory=INFO, CONSOLE          debug   info   warn error fatal
log4j.rootCategory=debug, CONSOLE, LOGFILE

# Set the enterprise logger category to FATAL and its only appender to CONSOLE.
log4j.logger.org.apache.axis.enterprise=FATAL, CONSOLE

# CONSOLE is set to be a ConsoleAppender using a PatternLayout.
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%d{ISO8601} %-6r [%15.15t] %-5p %30.30c %x -
%m\n

# LOGFILE is set to be a File appender using a PatternLayout.

```

```
log4j.appender.LOGFILE=org.apache.log4j.FileAppender
log4j.appender.LOGFILE.File=c:\axis.log
log4j.appender.LOGFILE.Append=true
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE.layout.ConversionPattern=%d{ISO8601} %-6r [%15.15t] %-5p %30.30c %x -
%m\n
```

2.2.6 dao测试

```
package com.itheima.test;

import com.itheima.dao.ItemsDao;
import com.itheima.domain.Items;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class AppTest {
    @Test
    public void testDao(){
        ApplicationContext ac = new
ClassPathXmlApplicationContext("classpath:applicationContext.xml");
        ItemsDao bean = ac.getBean(ItemsDao.class);
        Items items = bean.findById(1);
        System.out.println(items.getName());
    }
}
```

2.3 service层

2.3.1 ItemService.java

```
package com.itheima.service;
import com.itheima.domain.Items;
public interface ItemsService {
    public Items findById(Integer id);
}
```

2.2.2 ItemsServiceImpl.java

```
package com.itheima.service.impl;
import com.itheima.dao.ItemsDao;
import com.itheima.domain.Items;
import com.itheima.service.ItemsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class ItemsServiceImpl implements ItemsService {
```

```

@Autowired
private ItemsDao itemsDao;
public Items findById(Integer id) {
    return itemsDao.findById(id);
}
}

```

2.3.3 applicationContext.xml

```

<!--service事务开始-->
<context:component-scan base-package="com.itheima.service"/>
<!-- 配置事务管理器 -->
<bean id="transactionManager"
      class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>
<!-- 配置事务的通知 -->
<tx:advice id="txAdvice">
    <tx:attributes>
        <tx:method name="save*" propagation="REQUIRED"/>
        <tx:method name="modify*" propagation="REQUIRED"/>
        <tx:method name="delete*" propagation="REQUIRED"/>
        <tx:method name="find*" read-only="true"/>
        <tx:method name="*" propagation="REQUIRED"/>
    </tx:attributes>
</tx:advice>
<!-- 配置切面 -->
<aop:config>
    <aop:pointcut expression="execution(* com.itheima.service.impl.*(..))"
id="tranpointcut"/>
    <!-- 事务控制 -->
    <aop:advisor advice-ref="txAdvice" pointcut-ref="tranpointcut"/>
</aop:config>
<!--service事务结束-->

```

2.3.4 service测试

```

package com.itheima.test;

import com.itheima.dao.ItemsDao;
import com.itheima.domain.Items;
import com.itheima.service.ItemsService;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class AppTest {

    @Test
    public void testService(){

        ApplicationContext ac = new

```

```

ClassPathXmlApplicationContext("classpath:applicationContext.xml");
    ItemsService bean = ac.getBean(ItemsService.class);
    Items items = bean.findById(1);
    System.out.println(items.getName());
}
}

```

2.4 web层

2.4.1 ItemsController.java

```

package com.itheima.controller;

import com.itheima.domain.Items;
import com.itheima.service.ItemsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping(value = "/items")
public class ItemsController {

    @Autowired
    private ItemsService itemsService;

    @RequestMapping(value = "/findDetail")
    public String getById(Model model){
        Items items = itemsService.findById(1);
        model.addAttribute("item", items);
        return "itemDetail";
    }
}

```

2.4.2 itemDetail.jsp

在WEB-INF/pages/目录下

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insert title here</title>

```



```

</head>
<body>
<form>
    <table width="100%" border=1>
        <tr>
            <td>商品名称</td>
            <td> ${item.name } </td>
        </tr>
        <tr>
            <td>商品价格</td>
            <td> ${item.price } </td>
        </tr>
        <tr>
            <td>生成日期</td>
            <td> <fmt:formatDate value="${item.createtime}" pattern="yyyy-MM-dd HH:mm:ss"/>
        </td>
        </tr>
        <tr>
            <td>商品简介</td>
            <td>${item.detail} </td>
        </tr>
    </table>
</form>
</body>
</html>

```

2.4.3 web.xml

```

<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >
<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_3_0.xsd"
    version="3.0">

    <!--POST编码过滤器-->
    <filter>
        <filter-name>characterEncoding</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <!--指定编码-->
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>

```

```

<!--编码映射拦截-->
<filter-mapping>
    <filter-name>characterEncoding</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!--spring核心监听器-->
<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>

<!--前端核心控制器-->
<servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

    <!--指定springmvc核心配置文件-->
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:springmvc.xml</param-value>
    </init-param>
    <!--启动加载优先级-->
    <load-on-startup>1</load-on-startup>
</servlet>
<!--映射拦截-->
<servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>

```

2.4.3 springmvc.xml

```

<?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:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!--包扫描-->
    <context:component-scan base-package="com.itheima.controller"/>

    <!--注解驱动-->

```

```

<mvc:annotation-driven/>
<!--视图解析器-->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <!--前缀-->
    <property name="prefix" value="/WEB-INF/pages/" />
    <!--后缀-->
    <property name="suffix" value=".jsp" />
</bean>
<!--静态资源过滤-->
<mvc:default-servlet-handler/>
</beans>

```

第4章SSM工程注意点

注意点1：新建一个父工程pom 没有选中骨架（一定要有pom文件），本身工程不不要编码

注意点2: 在选中父工程后新建子模块jar

注意点3：web工程选择了骨架（webapp war）

注意点4：在父工程上执行tomcat7:run能够成功，在web工程中执行会失败，在父工程中执行install命令将工程安装到仓库

第5章私服

本地仓库 ==> 远程仓库（私服） ==> 中央仓库

将项目发布到私服

setting配置文件

```

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

```

dao pom.xml

```

<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>
```

从私服下载jar包

父工程pom.xml

```
<repositories>
  <repository>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
    <id>public</id>
    <name>Public Repositories</name>
    <url>http://localhost:8081/nexus/content/groups/public/</url>
  </repository>
  <repository>
    <snapshots>
      <enabled>>false</enabled>
    </snapshots>
    <id>central</id>
    <name>Central Repository</name>
    <url>https://repo.maven.apache.org/maven2</url>
  </repository>
</repositories>

<pluginRepositories>
  <pluginRepository>
    <id>public</id>
    <name>Public Repositories</name>
    <url>http://localhost:8081/nexus/content/groups/public/</url>
  </pluginRepository>
  <pluginRepository>
    <releases>
      <updatePolicy>never</updatePolicy>
    </releases>
    <snapshots>
      <enabled>>false</enabled>
    </snapshots>
    <id>central</id>
    <name>Central Repository</name>
    <url>https://repo.maven.apache.org/maven2</url>
  </pluginRepository>
</pluginRepositories>
```

将jar安装到私服

setting配置

```
<server>  
  <id>thirdparty</id>  
  <username>admin</username>  
  <password>admin123</password>  
</server>
```

命令

```
mvn deploy:deploy-file -DgroupId=com.alibaba -DartifactId=fastjson -Dversion=1.1.37 -  
Dpackaging=jar -Dfile=a.jar -Durl=http://localhost:8081/nexus/content/repositories/thirdparty/ -  
DrepositoryId=thirdparty
```