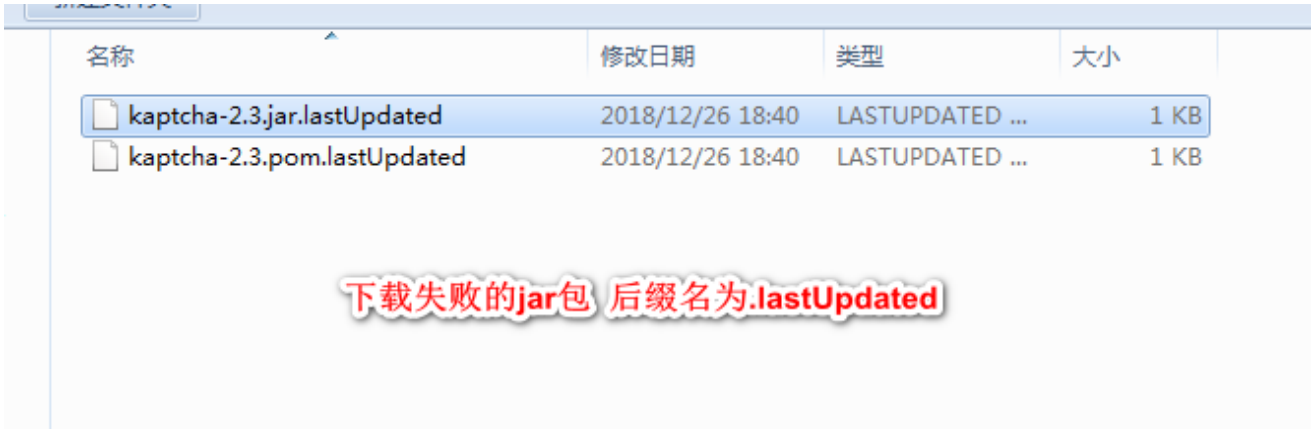
如何重新配置maven环境

1. JDK 1.8
2. maven重装
3. 删掉本地仓库

jar包下载失败怎么办

jar包下载失败的标志 就是文件夹中有 后缀为 .lastUpdated的文件



怎么处理：

第一种方式：删除所有的jar包

第二种方式：使用工具 maven batch.bat

1.配置工具

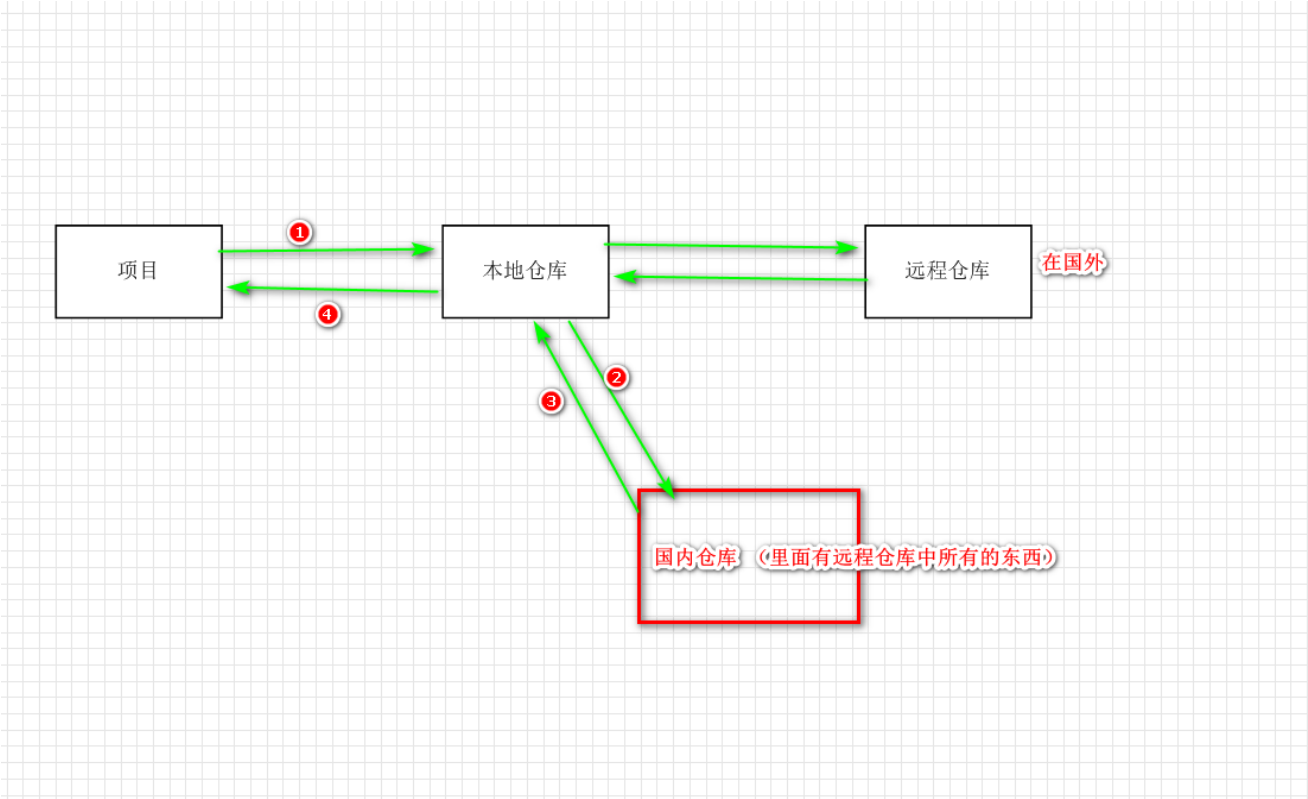
```
@echo  off
rem create by sunhao(sunhao.java@gmail.com)
rem crazy coder
set REPOSITORY_PATH=E:\MavenRepository
rem 正在搜索...
for /f "delims=" %%i in ('dir /b /s "%REPOSITORY_PATH%\*lastUpdated*"') do (
    del /s /q %%i
)
rem 搜索完毕
pause
```

配置maven本地仓库地址

2.双击运行

下载过慢怎么办

配置镜像仓库



镜像仓库 实际上就是远程仓库的国内备份

1.镜像仓库地址（阿里云私服地址）

```
<mirror>
    <id>alimaven</id>
    <name>aliyun maven</name>
    <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
    <mirrorOf>central</mirrorOf>
</mirror>
```

2.粘贴到settings.xml中

```
  --/

  <mirror>
      <id>alimaven</id>
      <name>aliyun maven</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
  </mirror>

</mirrors>
```

搭建SSM环境

使用maven之前搭建环境

1. 创建一个空项目
2. 添加jar包

3. 完善包结构
4. 配置文件
    1. web.xml
    2. spring.xml
    3. mvc.xml

使用maven之后

1. 创建一个空项目
2. 添加jar包 配置pom.xml中的依赖

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <!-- spring版本号 -->
  <spring.version>4.3.20.RELEASE</spring.version>
  <!-- mybatis版本号 -->
  <mybatis.version>3.2.8</mybatis.version>
  <!-- log4j日志文件管理包版本 -->
  <slf4j.version>1.6.6</slf4j.version>
  <log4j.version>1.2.9</log4j.version>
</properties>


<dependencies>
  <!--spring-->
  <!-- spring 核心包 -->
  <!-- springframe start -->

  <!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
  <dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.7.1</version>
  </dependency>

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

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

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

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</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-jdbc</artifactId>
  <version>${spring.version}</version>
</dependency>
<!-- springframe end -->
<!--mybatis-->
<!-- mybatis核心包 -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>${mybatis.version}</version>
</dependency>

<!-- mybatis/spring包 -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>1.2.2</version>
</dependency>
<!--数据库相关-->
<!-- mysql -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.18</version>
</dependency>
<!-- 数据源包 -->
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid</artifactId>
  <version>1.0.2</version>
</dependency>


<!--测试-->
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
```

```
    <scope>test</scope>
  </dependency>

  <!-- 日志文件管理包 -->
  <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 -->

  <!-- 上传组件包 -->
  <dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
  </dependency>

  <dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.4</version>
  </dependency>

  <dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.9</version>
  </dependency>

  <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.1.41</version>
  </dependency>

  <dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>jsp-api</artifactId>
    <version>6.0.32</version>
    <scope>provided</scope>
```

```
    </dependency>

    <!--jsp相关-->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <!-- scope=compile 的情况（默认compile），也就是说这个项目在编译、测试，运行阶段都需要这个artifact对应的jar包在classpath中 -->
      <!-- scope=provided ，则可以认为这个provided是目标容器已经provided这个artifact,它只影响到编译、测试阶段，运行阶段，假定目标容器已经提供了这个jar包 -->
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/jstl/jstl -->
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>

    <dependency>
      <groupId>org.apache.commons</groupId>
      <artifactId>commons-lang3</artifactId>
      <version>3.4</version>
    </dependency>
  </dependencies>
```

3. 完善包结构
4. 配置文件
    1. web.xml
    2. spring.xml
    3. mvc.xml

注意：

jsp中需要有设置 isELIgnored="false"

```
<%@page contentType="text/html;UTF-8" pageEncoding="UTF-8" isELIgnored="false" %>
```

事务处理

事务处理分类

1. 编程式事务处理 使用代码 【了解即可】
2. 声明事务处理

1. xml配置
2. 注解式

xml配置声明式事务处理

```
<!--声明式事务处理xml方式-->
<!-- 6.1.声明事务管理器    需要数据源-->
<bean class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
id="dataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>

</bean>
<!-- 6.2声明事务增强    tx标签完成   需要事务管理器；在这里面指定事务规则 -->
<tx:advice id="txAdvice" transaction-manager="dataSourceTransactionManager">
    <tx:attributes>
        <tx:method name="get*" read-only="true"  isolation="READ_COMMITTED"
propagation="SUPPORTS"/>
        <tx:method name="set*" read-only="true"  isolation="READ_COMMITTED"
propagation="SUPPORTS"/>
        <tx:method name="*" propagation="REQUIRED"/>
    </tx:attributes>
</tx:advice>

<!-- 6.3通过aop声明切入点 -->
<aop:config>
    <aop:pointcut id="pointCut" expression="execution(* com.baizhi.service.*.*(..))"/>
    <aop:advisor advice-ref="txAdvice" pointcut-ref="pointCut"/>
</aop:config>
```

注解式事务处理

1. spring.xml中配置 事务管理器 开启事务处理

   ```
   <!--1.配置事务管理器-->
   <bean class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
   id="transactionManager">
       <property name="dataSource" ref="dataSource"/>
   </bean>
   <!--2.开启注解式事务处理-->
   <tx:annotation-driven transaction-manager="transactionManager"/>
   ```

2. 使用注解 @Transactional

   ```
   @Transactional 可以添加在方法上和类上
   添加在方法上 代表该方法开启事务控制
   添加类上 代表类中所有方法开启事务控制
   ```

作业：

1. 上课实例代码SSMdemo写一遍
2. 注解式事务处理测试一下