Maven 构建和管理 J2ee 项目 1.0

拟制部门：技术部

版本号：V1.0

修改日期： 2011-3-23

1 安装 maven

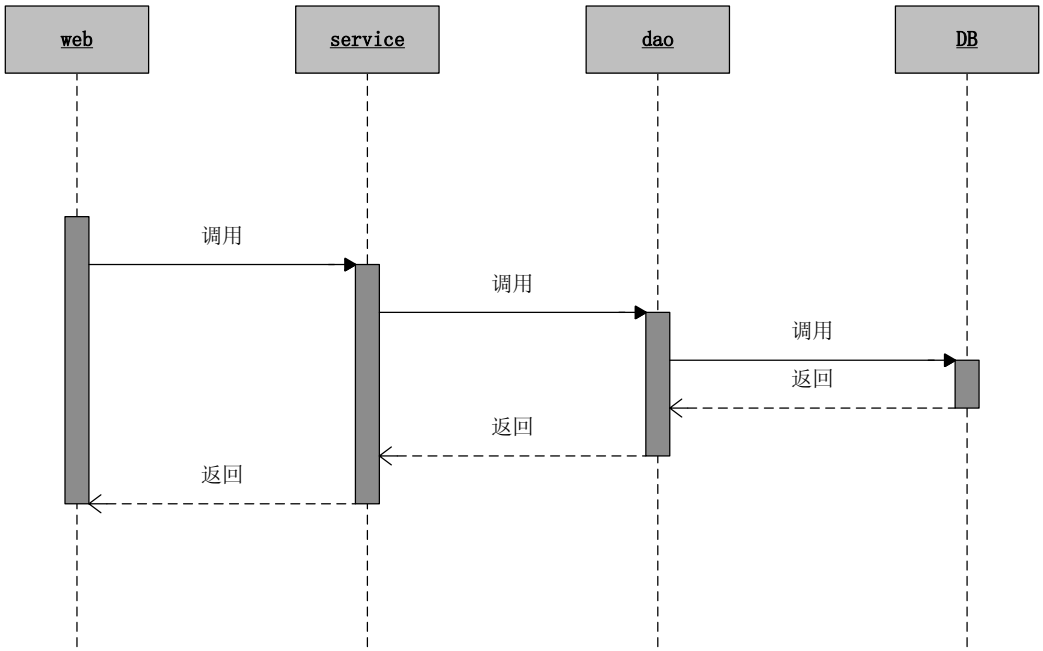
Maven 安装很简单到 http://maven.apache.org/下面下载最新版 maven，解压后在系统 path 目录增加 $:\apache-maven-3.0.3\bin;就可以啦。使用 maven 的前提条件是系统要安装好 java 环境。

2 构建 J2ee 项目

2.1 项目实例

```
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-webapp -DarchetypeArtifactId=maven-archetype-webapp
```

上面命令将创建一个非常常用的 web 项目，项目名称为 my-webapp。这也是非常常用的 maven 命令，很多人都认为这样就可以很好使用了，其实不然。
在创建一个工程后，由于著名的 mvn 模式，很多人都会通过 package 分开层和层之间程序，最常用为：com.mycompany.app.web; com.mycompany.app. service; com.mycompany.app.dao。
结构如下：



在小型系统中改结构比较合理，标准的 mvc 格式。其他 mvc 是说明前端，业务逻辑还有数据层分开。而然业务逻辑是一个系统中最复杂的地方，并不是一个 service 就能搞定。例如：要考虑系统不断壮大，业务越来越复杂，访问量越来越大，这样项目就要拆分，不但要垂直拆分还要水平拆分。下面介绍下，比较好的做法。

2.2 多层次项目构建

例如一个普通电子商务网站，其业务包含商品管理，商品搜索，商品展示；用户注册，商品购买等。
由于开始网站比较小，访问量也低。所以没有必要把系统弄得非常复杂，例如分布式，数据库分片，读写分离，缓存等。这些都没有必要，也没有那么大成本。一切从实际出发，组织团队开始动工。下面讨论下如何合理构建项目。

2.2.1 业务划分

从业务来看，这个网站分为：用户，商品，订单。他们之间也会互相之间出现交互，订单中会设计到用户，订单中也会涉及到商品等。
所以在项目中最好把他们分为独立的子项目，例如分别叫：test-biz-item；test-biz-user；test-biz-order。

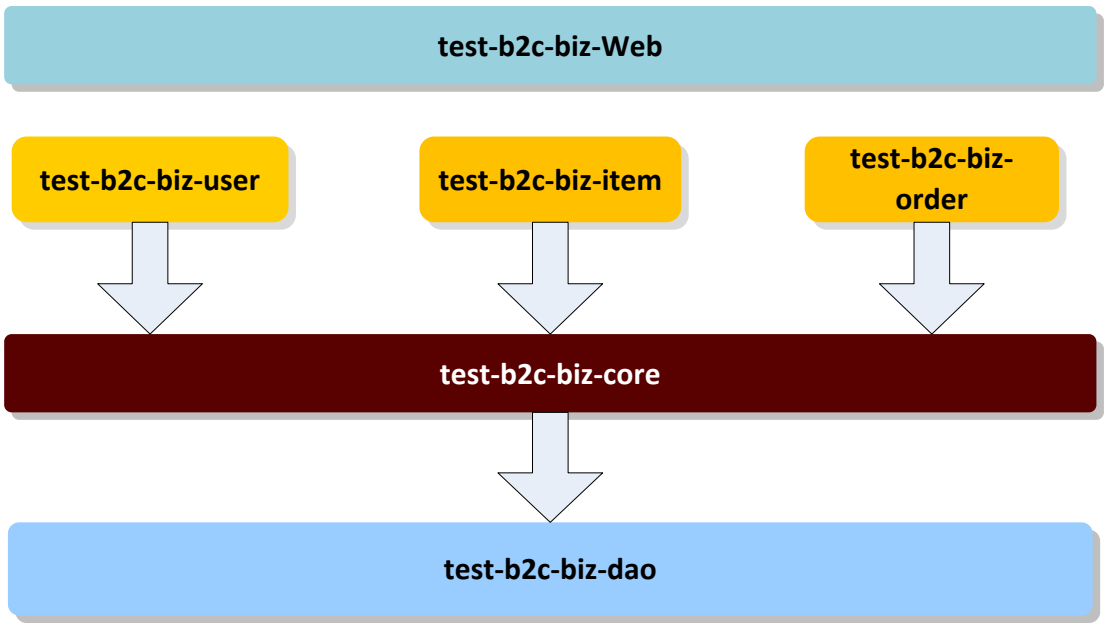
2.2.2 创建项目

Maven 可以针对项目构建结构复杂的多个子项目。由于上面业务已经划分好，创建步骤如下：
创建主工程：
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=test-b2c
执行命令后会生成 test-b2c 文件夹，打开 test-b2c 文件，编辑 pom.xml 文件，把
<packaging>jar</packaging>
改为
<packaging>pom</packaging>
进入 test-b2c 继续创建工程：
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=test-b2c-biz-core
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=test-b2c-biz-user
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=test-b2c-biz-item
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=test-b2c-biz-order
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=test-b2c-biz-dao
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=test-b2c-web -DarchetypeArtifactId=maven-archetype-webapp
执行上面多个操作后，将生成 test-b2c-biz-core，test-b2c-biz-user，test-b2c-biz-item，test-b2c-biz-order，test-b2c-biz-dao, test-b2c-web 文件夹。在 test-b2c 目录下面的 pom 文件中将增加如下内容：

```
  <modules>
    <module>test-b2c-biz-core</module>
    <module>test-b2c-biz-user</module>
    <module>test-b2c-biz-item</module>
    <module>test-b2c-biz-order</module>
    <module>test-b2c-biz-dao</module>
    <module>test-b2c-web</module>
  </modules>
```

下面分别对子模块进行说，
test-b2c-biz-core 针对所有业务的核心业务逻辑；
test-b2c-biz-user 用于网站用户的业务流程
test-b2c-biz-item 用于商品的业务流程
test-b2c-biz-order 订单的业务流程

test-b2c-biz-dao 整个项目的数据层
test-b2c-web 整个项目的 web 前端操作，可以使用 struts 或者 spring-mvc
这样一个简单的项目就初步创建了，其中主要比较难理解的是业务流程和业务逻辑的区别，core 主要是负责具体的业务处理，test-b2c-biz-user 中主要负责组装各种业务操作。调用关系如下：



其中 core 是一个最基本的业务单元，具有复用性。在业务流程中可以复用 core 的操作。业务流程之间不可复用。
在 dao 层中一般我比较喜欢使用 ibatis，ibatis 特别 sql 优化上面可以很好体现。而且 ibatis 可以很好防止 sql 非法注入。Hibernate 也很好，根据团队特点，可以使用不同技术。
其中 user，item，order 之间不能互相调用，core 自身可以随便调用。

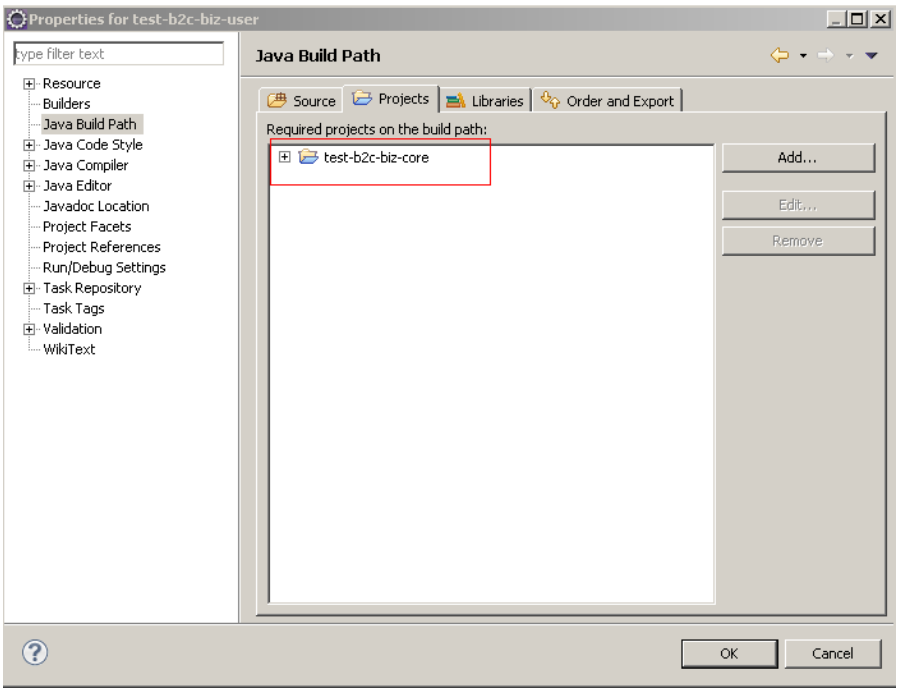
2.2.3 导入 eclipse

工程创建好后，可以导入到 eclipse 之中
Mvn eclipse:eclipse

由于 core 要被 item，order 和 user 调用，可以分别其 pom 中设置依赖：

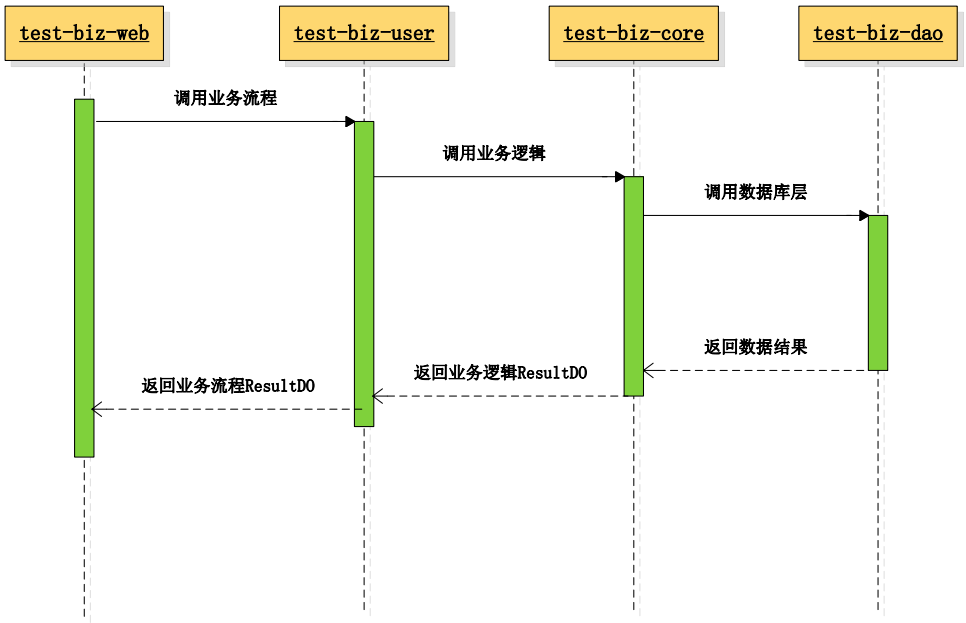
```
<dependency>
        <groupId>com.mycompany.app</groupId>
        <artifactId>test-b2c-biz-core</artifactId>
        <version>${project.version}</version>
</dependency>
```

再 mvn 一把，这样 user 就可以调用 core 模块了



到这里就可以开发啦。

系统调用序列图



2.2.4 接口编程

现在 java 的体系里面最流行莫过 spring，而 spring 使用在多层架构中也发挥得淋漓尽致。在各层次之间可以通过接口调用达到层次分明。

在 user 中创建接口：

```
package com.mycompany.app.user;
public interface UserAO {
    void addUser();
    void deleteUser();
```

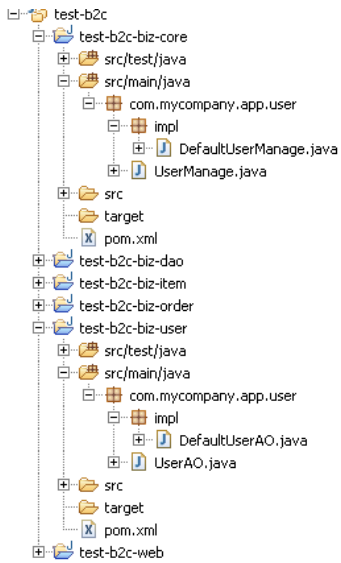
}

再创建实现类：

```
package com.mycompany.app.user.impl;
import com.mycompany.app.user.UserAO;
public class DefaultUserAO implements UserAO {
    public void addUser() {
        // TODO Auto-generated method stub
    }
    public void deleteUser() {
        // TODO Auto-generated method stub
    }
}
```

在core创建分别创建接口和和实现类：

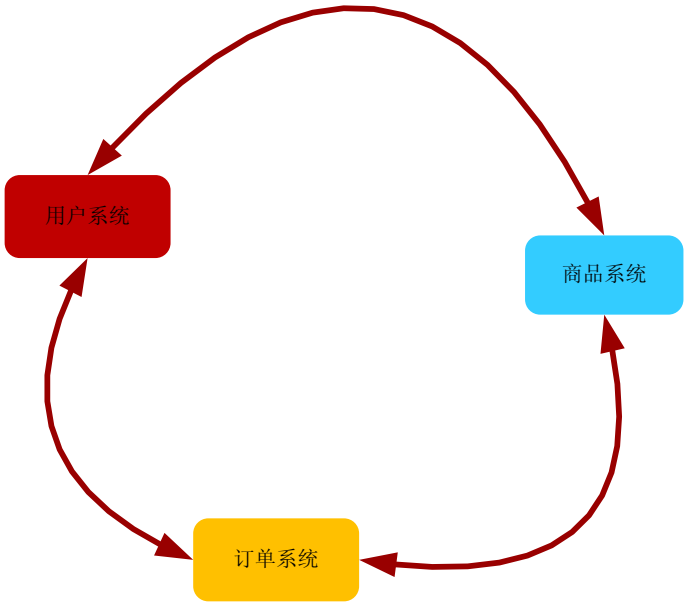
```
package com.mycompany.app.user;
public interface UserManage {
    void addUser();

    void deleteUser();
}

package com.mycompany.app.user.impl;
import com.mycompany.app.user.UserManage;
public class DefaultUserManage implements UserManage {
    @Override
    public void addUser() {
        // TODO Auto-generated method stub
    }
    @Override
    public void deleteUser() {
        // TODO Auto-generated method stub
    }
}
```



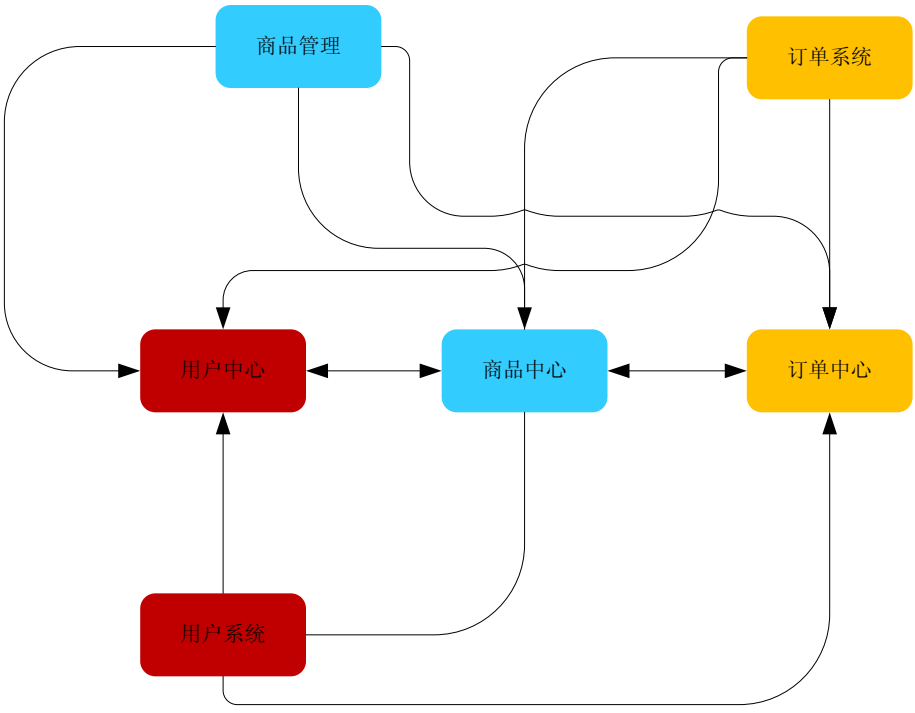
User需要调用core模块的UserManage，可以通过spring的接口注入就可以啦。

2.3 系统水平和垂直拆分

由于业务越来越复杂，访问量越来越大，服务器当然会增加，原来一个系统也会拆分成多个系统；
拆分如下：



原来一个项目可能就要分成三个项目了。这样分别对应三台服务器。由于原来规定在业务流程层不能互相调用，所以拆分起来非常简单。只要在 core 层注意拆分就可以了。
如果访问量再增加，系统业务也随着增加。系统之间交互就非常复杂。这时就考虑水平拆分系统。
结构如下：



上面结构把每个业务系统的核心业务独立出来，提供对外服务。中心之间可以互相调用，言下之意是把原来 core 模块的独立出来。由于开始结构清晰，通过垂直和水平划分，系统很容易就得到扩展。

3 Maven 常用命令

1. 创建 Maven 的普通 java 项目：
   mvn archetype:create
   -DgroupId=packageName
   -DartifactId=projectName
2. 创建 Maven 的 Web 项目：
   mvn archetype:create
   -DgroupId=packageName
   -DartifactId=webappName
   -DarchetypeArtifactId=maven-archetype-webapp
3. 编译源代码： mvn compile
4. 编译测试代码：mvn test-compile
5. 运行测试：mvn test
6. 产生 site：mvn site
7. 打包：mvn package
8. 在本地 Repository 中安装 jar：mvn install
9. 清除产生的项目：mvn clean
10. 生成 eclipse 项目：mvn eclipse:eclipse
11. 生成 idea 项目：mvn idea:idea
12. 组合使用 goal 命令，如只打包不测试：mvn -Dtest package
13. 编译测试的内容：mvn test-compile
14. 只打 jar 包: mvn jar:jar
15. 只测试而不编译，也不测试编译：mvn test -skipping compile -skipping test-compile
    (-skipping 的灵活运用，当然也可以用于其他组合命令)
16. 清除 eclipse 的一些系统设置:mvn eclipse:clean

命令总结：

mvn eclipse:eclipse -downloadSources=ture

加上-downloadSources=ture 把对应的源代码下到本地

mvn package –Dmaven.test.skip=true

过滤测试文件，这样发布项目时不至于有很多无关的测试程序。

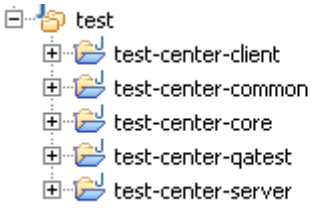
4 二方库管理

构建 maven 二方库的软件有很多，比较有名的算 Nexus 和 Artifactory，工具使用我就不多介绍，主要是讲讲一个项目中那些工程应该放到二方库上面

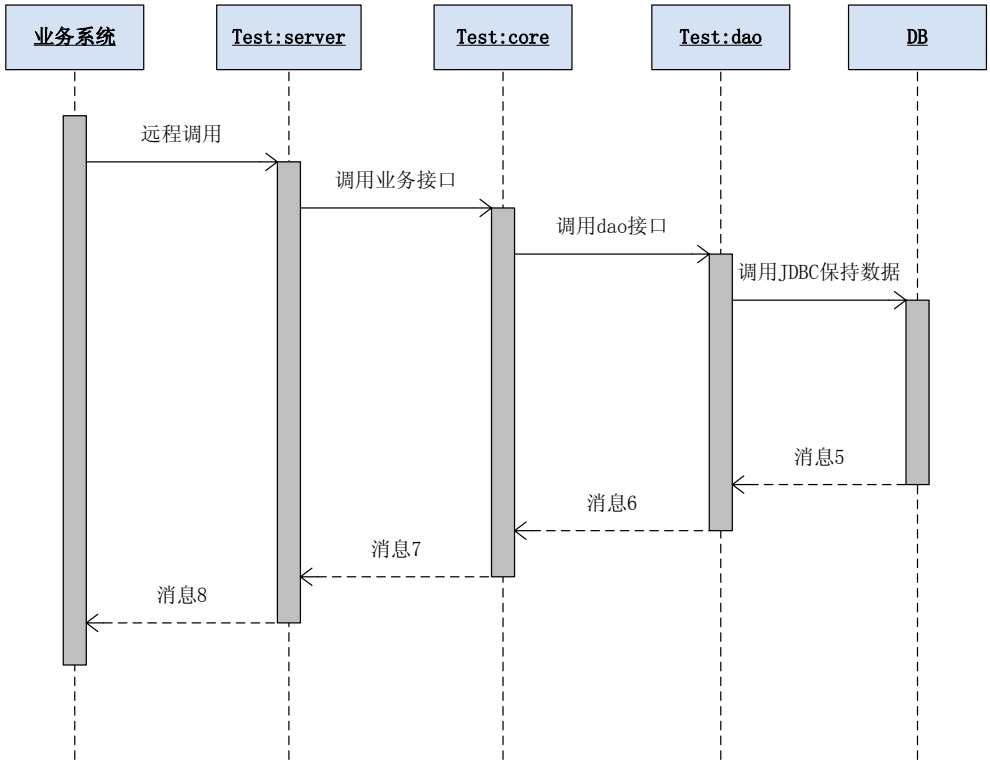
4.1 分布式远程调用

在 java 中分布式远程调用使用，一般都需要提供客户端 jar 给对方调用。下面创建一个提供给别人调用的项目。

项目结构如下



Client 主要用于客户端调用和客户端调用业务控制，common 是项目工具包。很明显 client 和 common 都应该提交到二方库上面，其他系统在调用时通过远程远程接口就可以调用本系统。



在业务系统通过远程调用时，一般情况有些非法业务控制会写在 common 中，这样有利于控制业务数据，还没有通过远程调用时，就已经做非法判断，也可以减少远程调用。