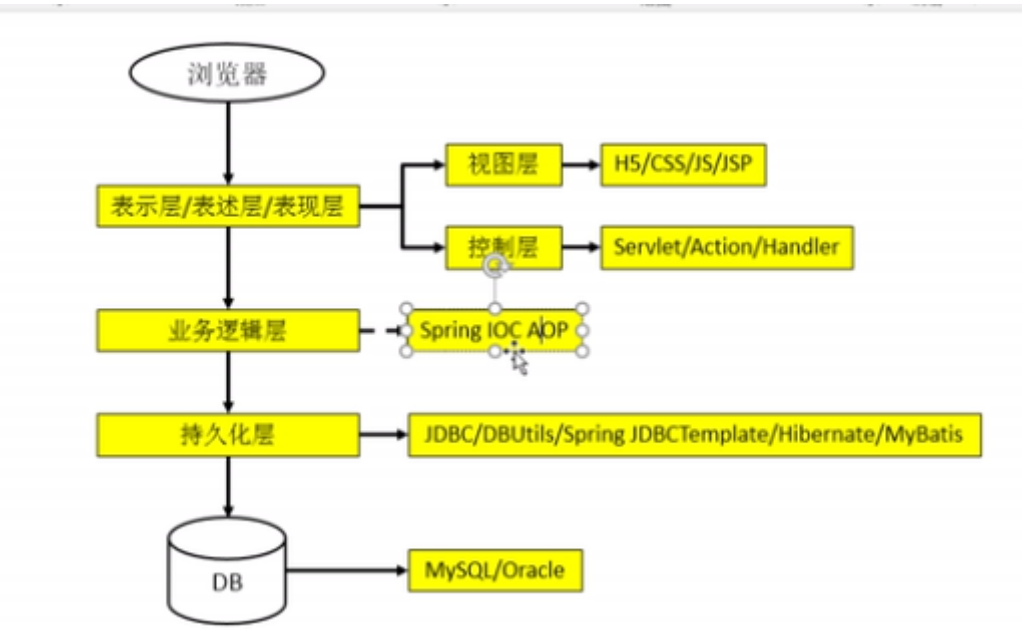


1，目前的技术在开发中存在的问题：（why）

（1）一个项目就是一个工程

如果一个项目非常大，就不适合使用package来划分模块，最好是每一个模块对应一个工程，利于分工协作。一个项目可以借助maven拆分成为多个工程。

（2）项目中需要的jar报必须手动复制，粘贴到web-inf/lib目录下

带来的 问题，同样的jar包文件重复出现在不同的项目工程中，一方面浪费存储空间，另外也让我们的项目看起来很臃肿。借助maven，我们可以将jar包仅仅保存在仓库中，有需要的使用的工程“引用”这个文件接口，并不需要针对把jar包复制过来。

（3）jar包需要别人提供或者去官网下载

①不同技术的官网提供jar包下载的形式是五花八门的。

②有些技术的官网就是通过maven或者SVN等专门的工具来提供下载的。

③如果是以非正规的方式下载的jar包，那么其中的内容很可能也是不规范的。

④借助maven可以以一种规范的方式下载jar包，因为所有知名框架或者第三方工具的jar包已经按照统一的规范存放在maven的中央仓库中，以规范的方式下载的jar包，内容也是可靠的。

⑤统一的规范不仅是对IT开发领域非常重要，对于整个人类社会都是非常重要的。

（4）一个jar包依赖的其它jar包需要自己手动添加到项目中

如果所有jar包之间的依赖关系都需要程序员自己非常清楚的了解，name就会极大的增加学习成本。maven会自动将被依赖的jar包导入进来。

2，maven是什么？（what）

（1）maven是一款服务于java平台的自动化构建工具。

make--->ant--->maven--->gradle

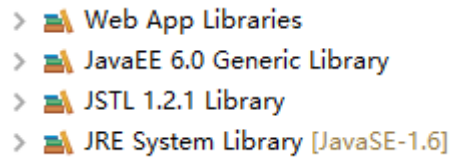
（2）构建

①构建并不是创建，是以“java源文件”、“框架配置文件”、“jsp”、“html”、图片等为原材料去生成一个可以运行的项目的过程。构建中有编译、搭建、部署的含义。

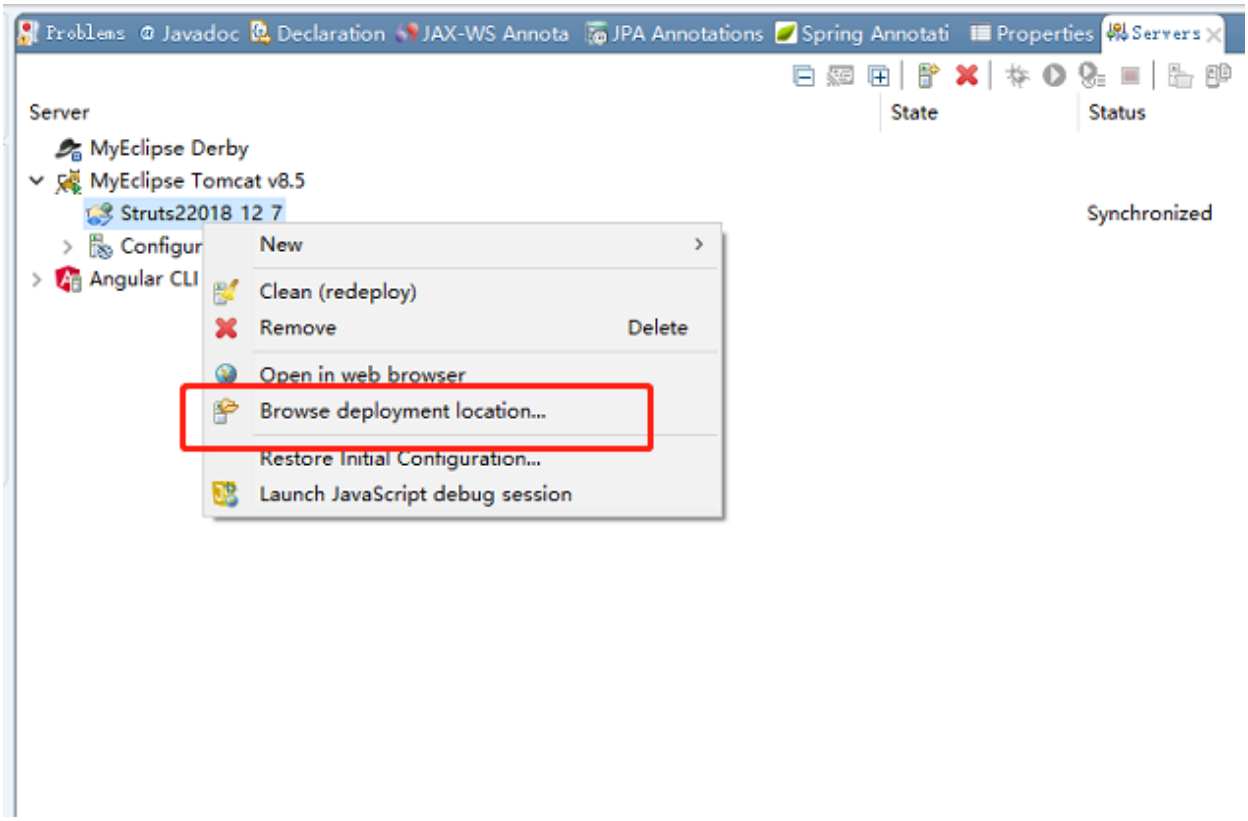
②编译：java源文件user.java--->编译--->class字节码文件user.class--->交给jvm去执行。

③部署：一个BS项目最终运行的并不是动态的web工程本身，而是这个动态web工程“编译的结果”。

运行时环境：其实是一组jar包的引用，并没有把jar包本身复制到工程中，所以并不是目录。



查看web项目的部署位置



web项目的编译结果就是webcontent中的内容。开发过程中，所有的路径或者配置文件中的配置的类路径等的剖视以编译结果的目录结构为标准的。

构建：

以下三个层面来看：

①纯 Java 代码

大家都知道，我们 Java 是一门编译型语言，.java 扩展名的源文件需要编译成.class 扩展名的字节码文件才能够执行。所以编写任何 Java 代码想要执行的话就必须经过编译得到对应的.class 文件。

②Web 工程

当我们需要通过浏览器访问 Java 程序时就必须将包含 Java 程序的 Web 工程编译的结果“拿”到服务器上的指定目录下，并启动服务器才行。这个“拿”的过程我们叫部署。

我们可以将未编译的 Web 工程比喻为一只生的鸡，编译好的 Web 工程是一只煮熟的鸡，编译部署的过程就是将鸡炖熟。

Web 工程和其编译结果的目录结构对比见下图：

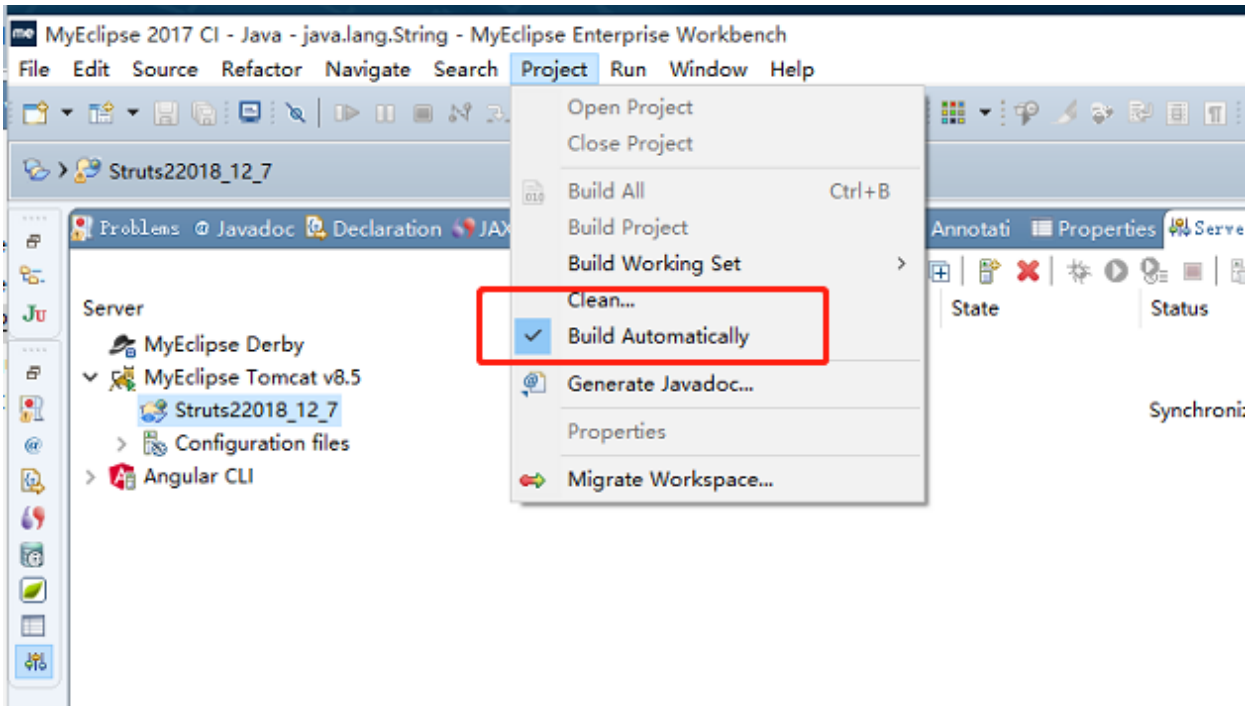
③实际项目

在实际项目中整合第三方框架，Web 工程中除了 Java 程序和 JSP 页面、图片等静态资源之外，还包括第三方框架的 jar 包以及各种各样的配置文件。所有这些资源都必须按照正确的目录结构部署到服务器上，项目才可以运行。

所以综上所述：构建就是以我们编写的 Java 代码、框架配置文件、国际化等其他资源文件、JSP 页面和图片等静态资源作为“原材料”，去“生产”出一个可以运行的项目的过程。

那么项目构建的全过程中都包含哪些环节呢？

在IDE中都是自动进行编译的：



（3）构建过程中 各个环节

①清理：将以前编译得到的class字节码删除，为下一次编译做准备。

②编译：将java源程序编译成class字节码文件。

③测试：自动测试。自动调用Junit程序。

④报告：测试程序执行的结果。

⑤打包：动态web打war包，java工程打jar包。

⑥安装：maven的特定概念，将打包得到的文件复制到“仓库”中指定的位置。

⑦部署：将动态web工程生成的war包复制到servlet容器的指定目录下，使其可以运

行。

3，安装maven核心程序

（1）检查java_home环境变量。set JAVA_HOME

（2）解压maven核心程序的压缩包。解压后放在一个非中文无空格路径下。如：F:\learn_maven\apache-maven-3.6.0

（3）配置maven的环境变量。

①配置MAVEN_HOME或者M2_HOME

②path

（4）验证：运行mvn-v命令查看maven版本。（环境变量修改之后命令行窗口需要重新开启）

```
C:\Windows\system32>mvn -v
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-25T02:41:47+08:00)
Maven home: F:\maven\apache-maven-3.6.0\bin\..
Java version: 11.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\java\jdk-11.0.2
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

4，maven的核心概念

（1）约定的目录结构

（2）POM

（3）坐标

（4）依赖

（5）仓库

（6）生命周期/插件/目标

（7）继承

（8）聚合

5，第一个maven工程

（1）创建约定的目录结构

①跟栏目：工程名

②src目录：源程序

③pom.xml文件：maven工程的核心配置文件

④main目录：主程序

main中的java目录：存放java的源程序

main中的resources目录：存放框架配置文件或其他的配置文件

⑤test目录：存放测试程序

test中的java目录：存放测试的java程序

test中的resources目录：存放测试的框架配置或其他的配置文件

（2）为什么要遵守约定的目录结构

①maven要负责我们这个项目的自动化构建，以编译为例，maven要想自动进行编译，

那么他必须知道java源文件保存在哪里。

②如果我们自己自定义的东西想要让框架或者工具知道，有两种解决办法：

a，以配置的方式明确告诉框架；

b，遵守框架内部已经存在的约定。

log4j.properties

log4j.xml

约定>配置>编码

6，常用maven命令

（1）注意：执行与构建过程相关的maven命令，必须进入pom.xml所在的目录。

与构建过程相关的命令：编译、测试、打包……

（2）常用的maven命令：

①mvn clean：清理

②mvn compile：编译主程序

③mvn test-compile：编译测试程序

④mvn test ：执行测试

⑤mvn package：执行打包

⑥mvn install：安装

⑦mvn site：生成站点

7，关于联网的问题

（1）maven的核心程序中仅仅定义了抽象的生命周期，但是具体的工作必须由特定的插件

完成，而插件本身并不包含在maven的核心程序中。

（2）当我们执行的maven命令需要用到某些插件时，maven核心程序会首先到本地仓库中

查找。

（3）本地仓库的默认位置：[系统中当前用户的家目录]\.m2\repository。

（4）maven核心程序如果在本地仓库中找不到需要的插件，那么他会自动连接外网到中央

仓库下载，如果此时无法连接外网，则构建失败。

（5）修改默认本地仓库的位置可以让maven核心程序到我们事先准备好的目录下查找插

件。

①找到maven解压目录\conf\settings.xml。

②在settings.xml中找到localRepository标签。

③将<localRepository></path/to/local/repo</localRepository>从注释中取出。

④将标签体中的内容修改为已经准备好的maven仓库目录，目录要进入以后能够看见具

体的插件。

（6）如果仓库中没有的话，会需要直接连接外网进行下载，如果联网失败的话，就会构建

不成功。

（7）构建产生的产品放在target目录下，指向mvn clean的话，就会直接删除target目录，

就是删除之前所有的构建。