# GameServer工具开发文档

目录

[GameServer工具开发文档 1](#_Toc434487895)

[前言 1](#_Toc434487896)

[导入项目 1](#_Toc434487897)

[用maven管理项目 8](#_Toc434487898)

[配置项目 13](#_Toc434487899)

[修改jdk版本 13](#_Toc434487900)

[设置source文件夹 13](#_Toc434487901)

[配置项目引用关系 14](#_Toc434487902)

[代码打包 15](#_Toc434487903)

[修改pom 15](#_Toc434487904)

[序列化服务打包 16](#_Toc434487905)

[数据层服务打包 17](#_Toc434487906)

[通讯层服务打包 18](#_Toc434487907)

## 前言

游戏样板间是以开发分布式游戏为demo的一个客户端游戏。

代码分为服务端部署，和客户端部署。

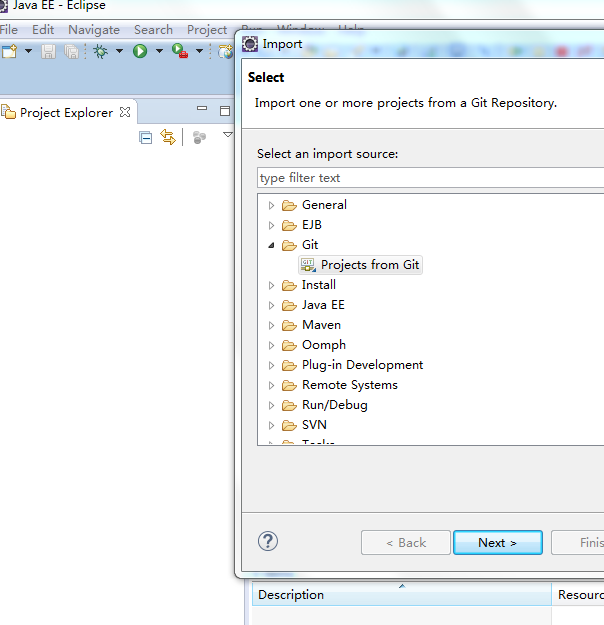
## 服务端部署

服务端代码现在放置在github上。导入的开发工具为eclipse。

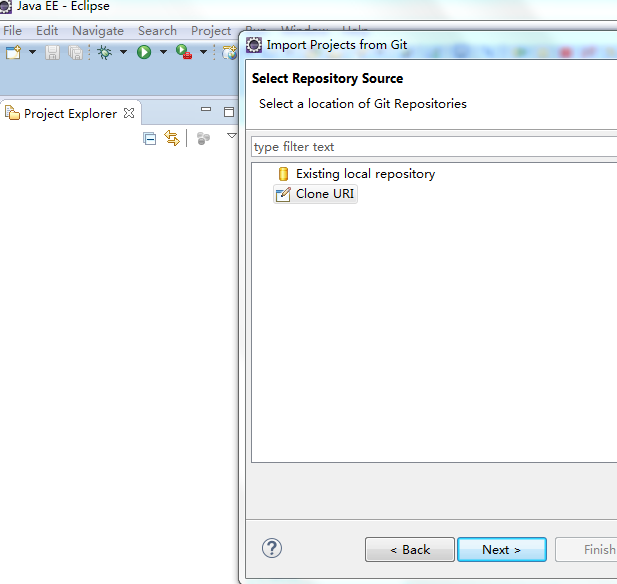
当前java开发中主流工具包括eclipse ,IDE,netbeans 等等，本文以eclipse问例，介绍一下gameServer项目如何导入配置打包，等。默认eclipse已经安装好maven，git相关插件，如果没有安装请自行百度，或者联系作者，tks。

### 导入项目

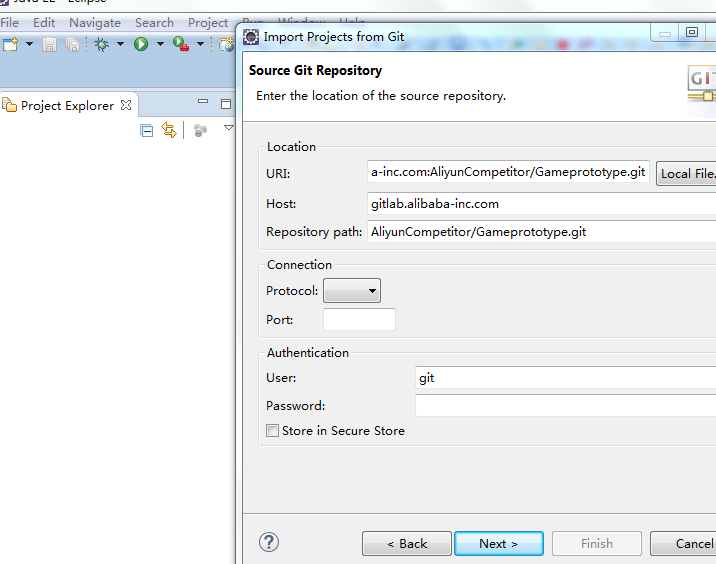
点击左上角File按照选择依次点击File-import-Git-Projects from Git



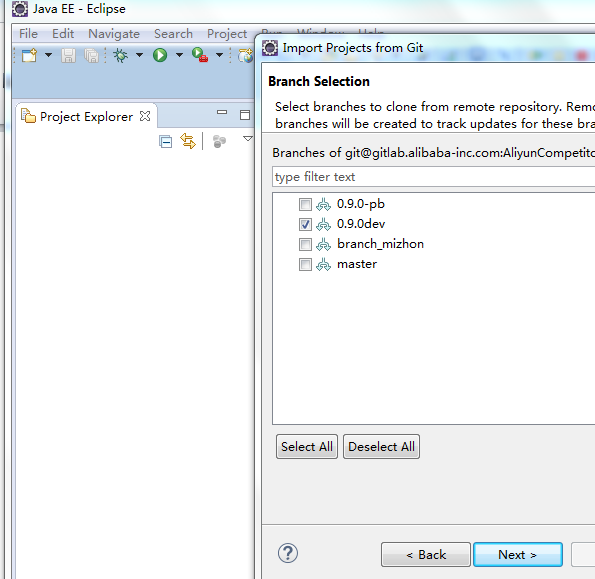
由于当前项目确定是从远程clone本地 因此选择Clone URL



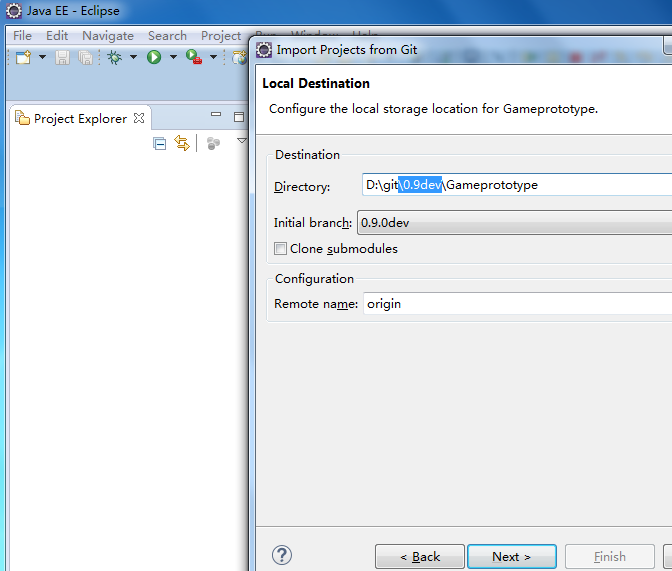
下一步输入git地址



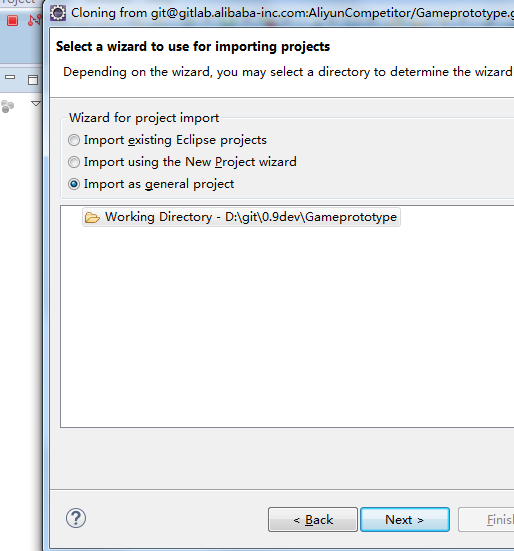
点击next ，选择需要导入的分之，当前部署的是0.9dev 分支



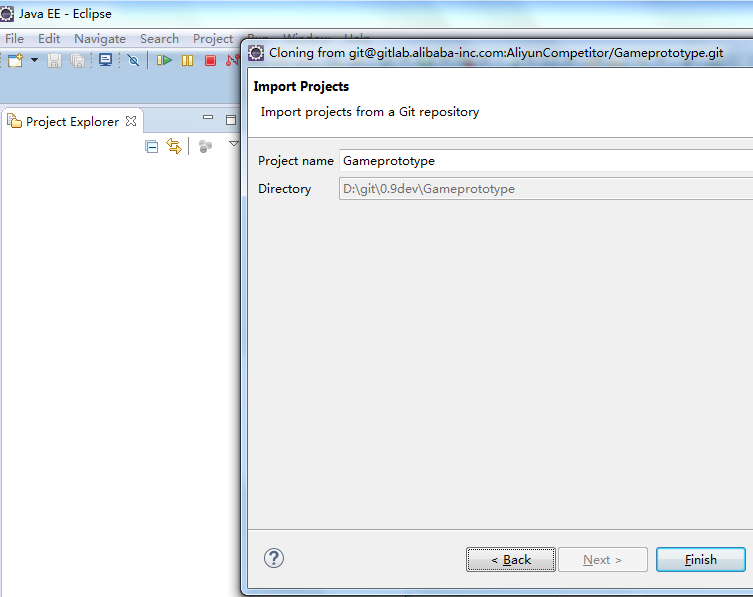
点击next 选择本地目录



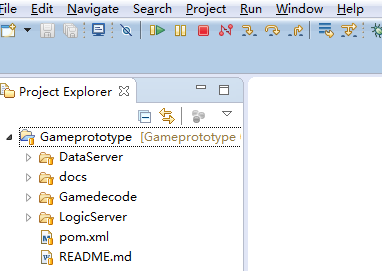
点击next 选择第三个选项



继续点击next



点击finsh即可



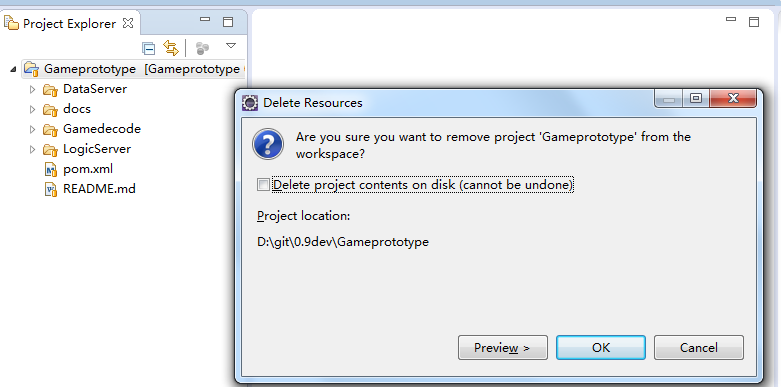
可以看到项目已经导入成功

### 用maven管理项目

上图可以看到虽然项目导入成功并且项目中拥有maven的元素却是以非maven和非java项目的方式存在。

将项目变成maven项目重新import一下

具体步骤为，右击该项目，点击delete，注意不要选择该可选项

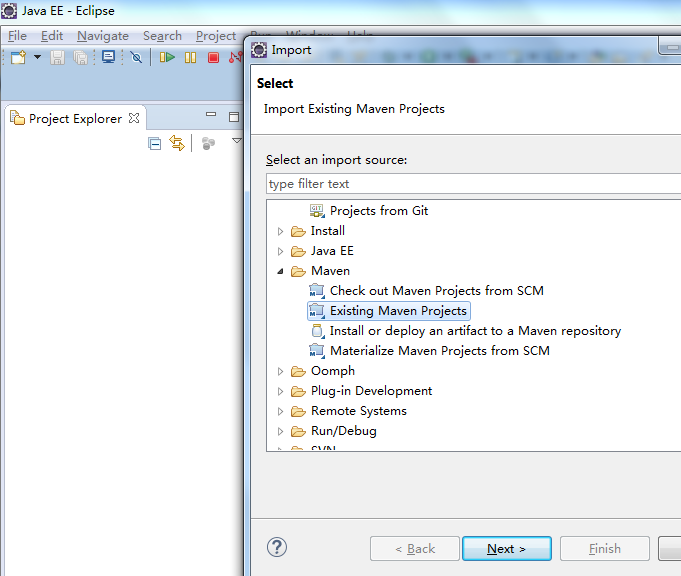


点击ok项目即被删除

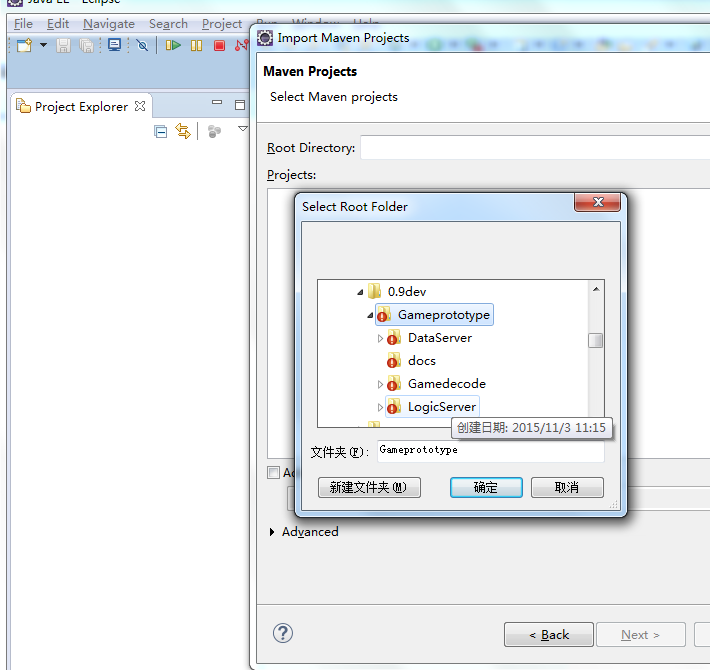
下面用maven方式引入项目

点击左上角File依次选择File-import-Maven-Existing Maven Projects

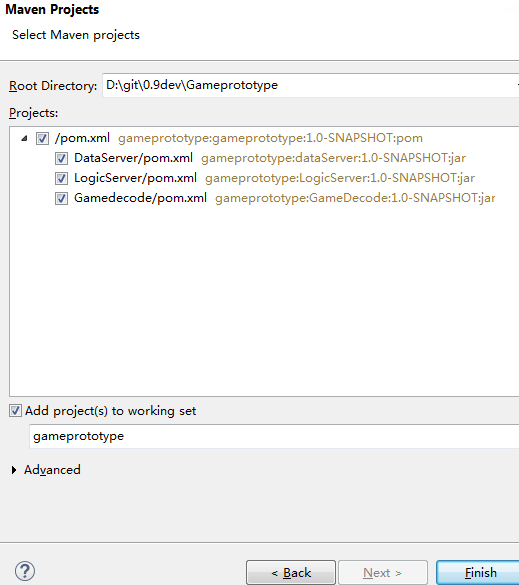
点击next



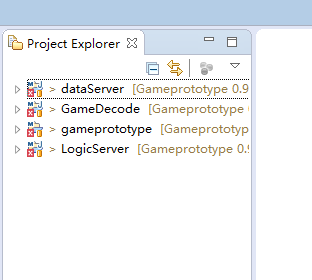
选择项目的地址，点击确定，请记住只要选择对Gameprototype 的File即可，子项目会自行导入



会出现parent project 和 child projdect的pom 点击finsh即可



项目导入完毕

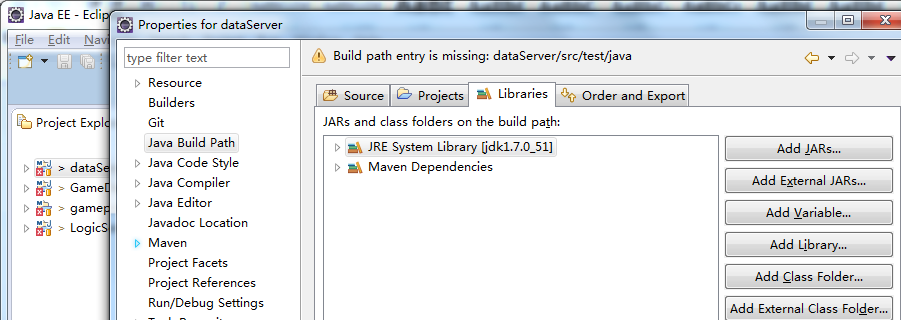


### 配置项目

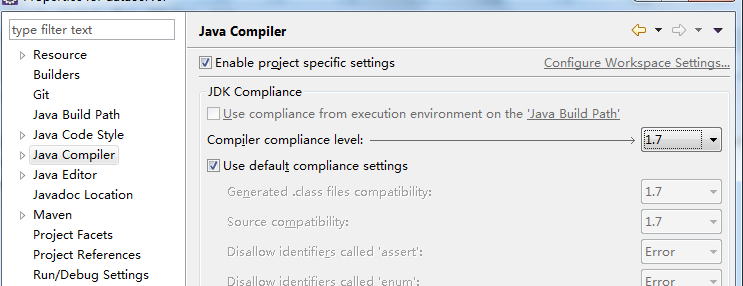
#### 修改jdk版本

由于pom中并未指定jdk和编译版本，因此会出现报错现象，下一步需要将jdk由默认的1.5配置为1.7即可

右击项目properties



更改编译版本



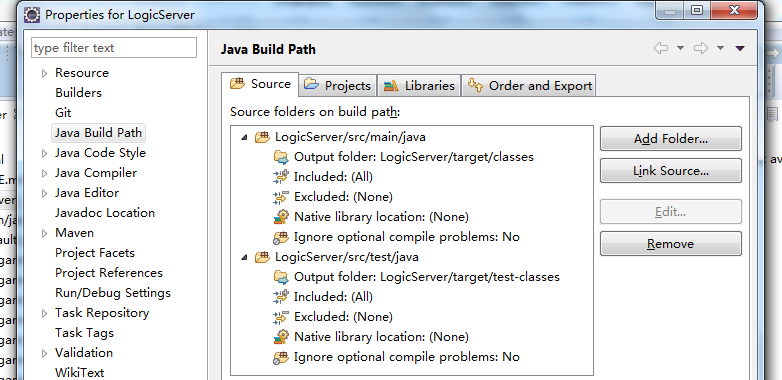
每个项目都按上两图设置下

#### 设置source文件夹

构架时已经考虑到分层打包和分功能打包，因此各个子项目的source Folder 较多，我们需要将代码文件夹设置为source Folder。

具体方式为

右击项目，选择properties – java build path – Source 点击add Folder ，选择需要加入的Folder



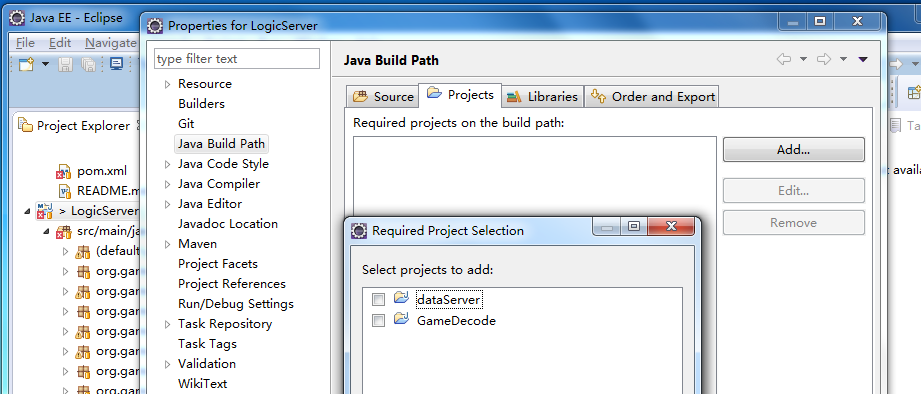
每个项目都需要设置一下。

#### 配置项目引用关系

由于开发时已经考虑到数据层、通讯层、序列化的分离，并考虑过单独sale 通讯层的问题，所以，现在数据和通讯是放置在两个子项目中的，本次开发中，需要在通讯层中引入data子项目和序列化子项目

具体方式为

右击项目，选择properties – java build path – Projects 点击Add，



选择上dataServer 和GameDecode即可

到此项目导入和配置已经全部完成。

下面介绍一下具体的分离分层打包方法。

### 代码打包

#### 修改pom

由于maven的包比较分散，因为为了方便部署拉取三方包，打包时可以讲需要的包打到一个lib文件夹下，因此需要配置一下主项目的pom文件

即在pom文件的倒数第二行插入以下描述

<build>

<plugins>

<plugin>

<!—指定入口类-->

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<classpathPrefix>lib</classpathPrefix>

<mainClass>logicServ</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

<!—将三方包copy到lib内-->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-dependency-plugin</artifactId>

<executions>

<execution>

<id>copy</id>

<phase>package</phase>

<goals>

<goal>copy-dependencies</goal>

</goals>

<configuration>

<outputDirectory>${project.build.directory}/lib</outputDirectory>

</configuration>

</execution>

</executions>

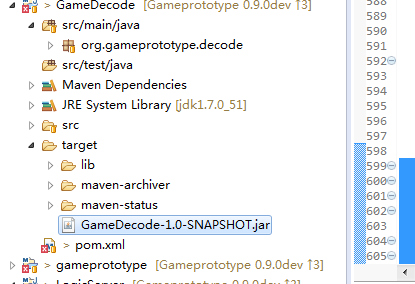
</plugin>

</plugins>

</build>

之所以用maven打包不用eclipse原生方式打包主要原因为，maven打包较为成熟，可以将三方库打入到lib内，方便部署。另外maven可以指定mainClass这样可以大大简化脚本复杂度。

刷新target，可以看到项目已经打包完成



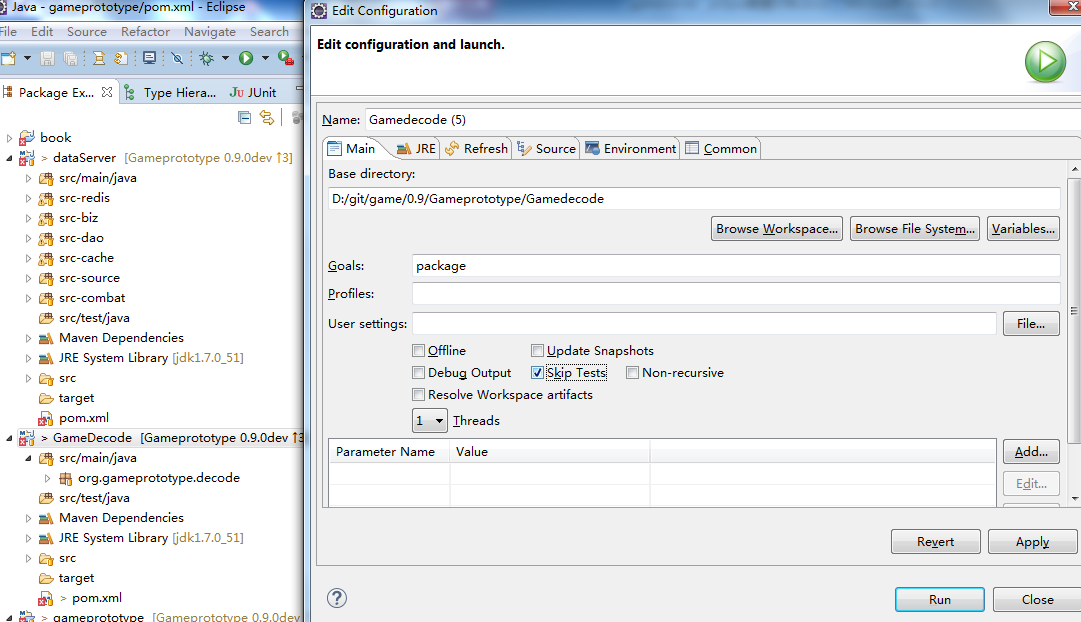
ok打包完成。

至此为止，关于gameServer中项目引入，编译，配置，打包，已经全部完成。如果需要修改代码，修改自动编译后，无需修改任何配置，直接打包即可。

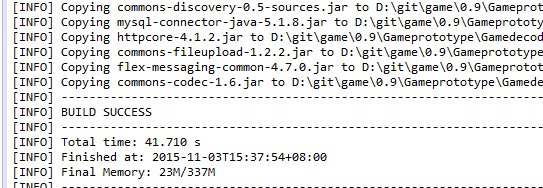
#### 序列化服务打包

先对序列化服务打包。右击序列化服务即GameDecode服务。

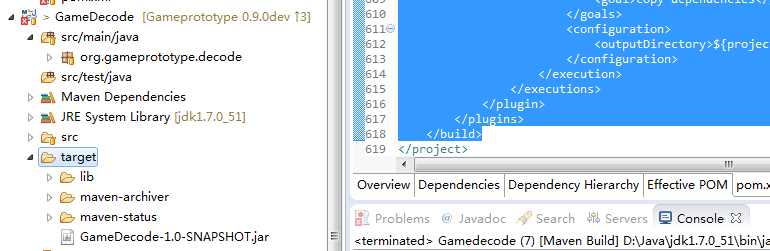
右击GameDecode项目，选择Run As – maven build



观看命令行，出现build success即可



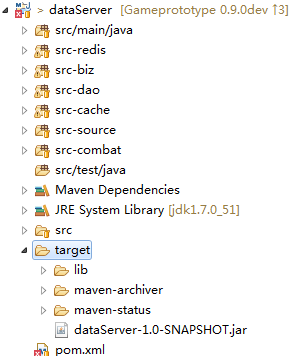
刷新target文件夹，可以看到项目已经打包



Lib下就是我们需要的三方库。

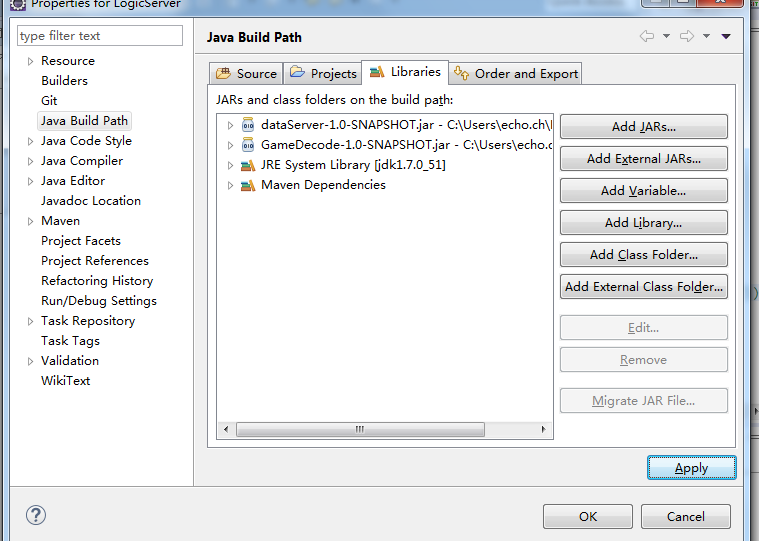
#### 数据层服务打包

按照GameDecode方法一样打包，刷新target，可以看到项目已经打包完成。



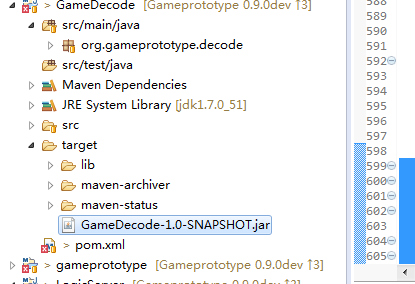
#### 通讯层服务打包

由于通讯层引用了前两个服务的因此需要先将我们打好的2个jar包引入当前通讯层服务，并去除项目依赖，具体方法为

点击add External JARS，添加jar包即可。并删除projects下的项目

配置完成后，按照前两个项目一样打包。

新target，可以看到项目已经打包完成



ok打包完成。

至此为止，关于gameServer中项目引入，编译，配置，打包，已经全部完成。如果需要修改代码，修改自动编译后，无需修改任何配置，直接打包即可。

# 客户端部署

客户端后台代码使用Unity编译运行。

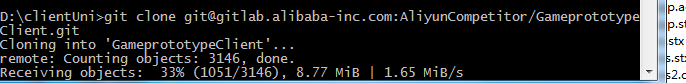
## 下载代码

使用git工具下载代码

进入gitCMD工具

进入需要放置代码的本地目录

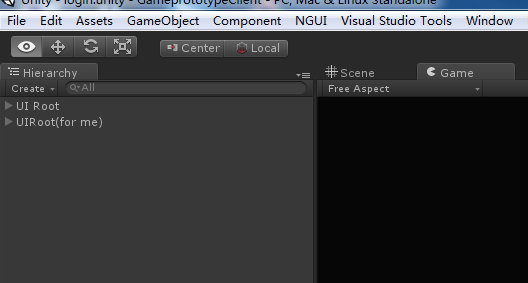
输入git命令

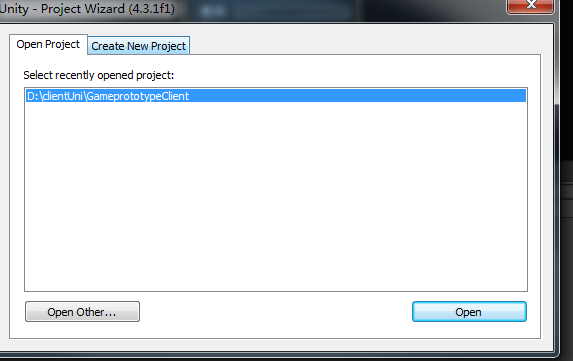
Git clone master git@gitlab.alibaba-inc.com:haihong.xiahh/GameprototypeClient.git

代码下载完毕

## 部署代码

打开unity客户端，点击左上角File-open project 打开项目

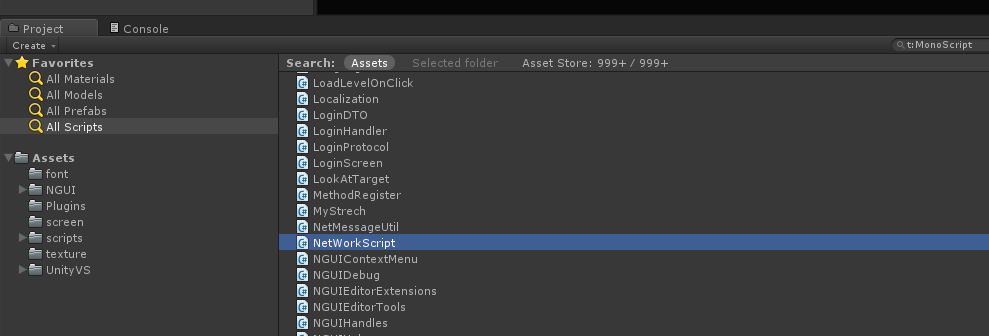




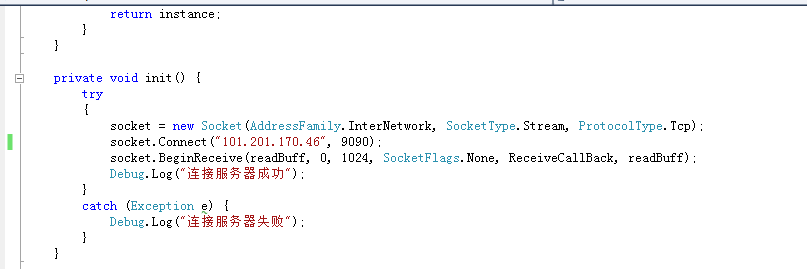
选择代码地址，点击ok，ok代码部署完成。

## 修改服务端ip运行客户端

点击All Scripts 找到NetWordScript



双击打开该文件

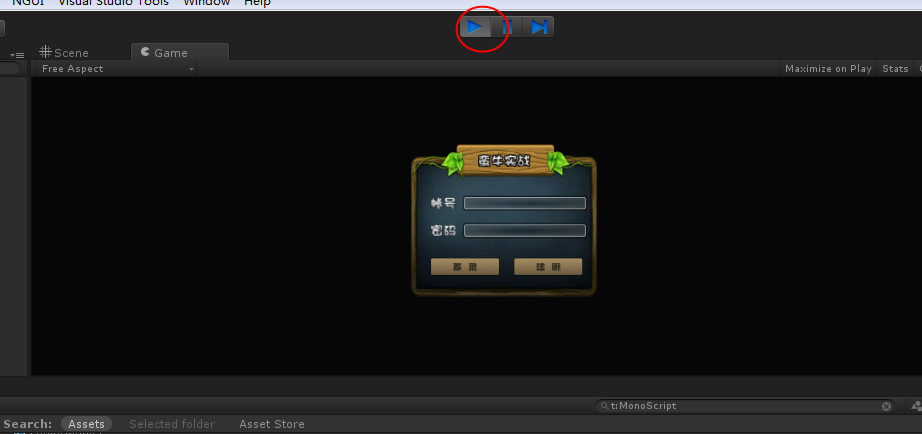


找到 socket.Connect("101.201.170.46", 9090);改行代码 将ip和端口进行修改。

改后保存代码

打开ScreenInt 文件

点击按钮运行代码



运行客户端测试登陆模块

