增量打包工具算法

增量打包的持久化数据库表设计:

列名	类型	描述
id	int	主键
name	string	模块名称，unique field ： 例如gwcz-biz
package_timestamp	string	最后打包的时间，读取自jar包所在目录的meta文件中的 `lastUpdate`标签
update_string	string	svn上最后的更新时间
sourcecode_path	string	源码所在目录路径
svn_location	string	svn的地址

依赖表

列名	类型	
id	int	主键
module_id	int	模块的主键
dependency_id	string	依赖模块的主键

1. 分析模块的日志,例如要打包车主gwcz，更新gwcz,然后拿到更新日志，例如gwcz有更新，压入要打包的栈中
2. 在sqlite数据库存储各个模块所**直接依赖**的模块，例如：gwcz-biz **直接依赖**的有gwcz-integretion,carrier-facade，uic-facade等等
3. 递归第 1-2步,直到所有需要打包的模块压入了栈内
4. 循环弹出要打包的栈，因为维护一个已经打包模块的set集合，因为有些模块是被重复依赖的，只需要打包一次
5. 打包完成后将svn最后的更新时间，mvn最后的打包时间重新

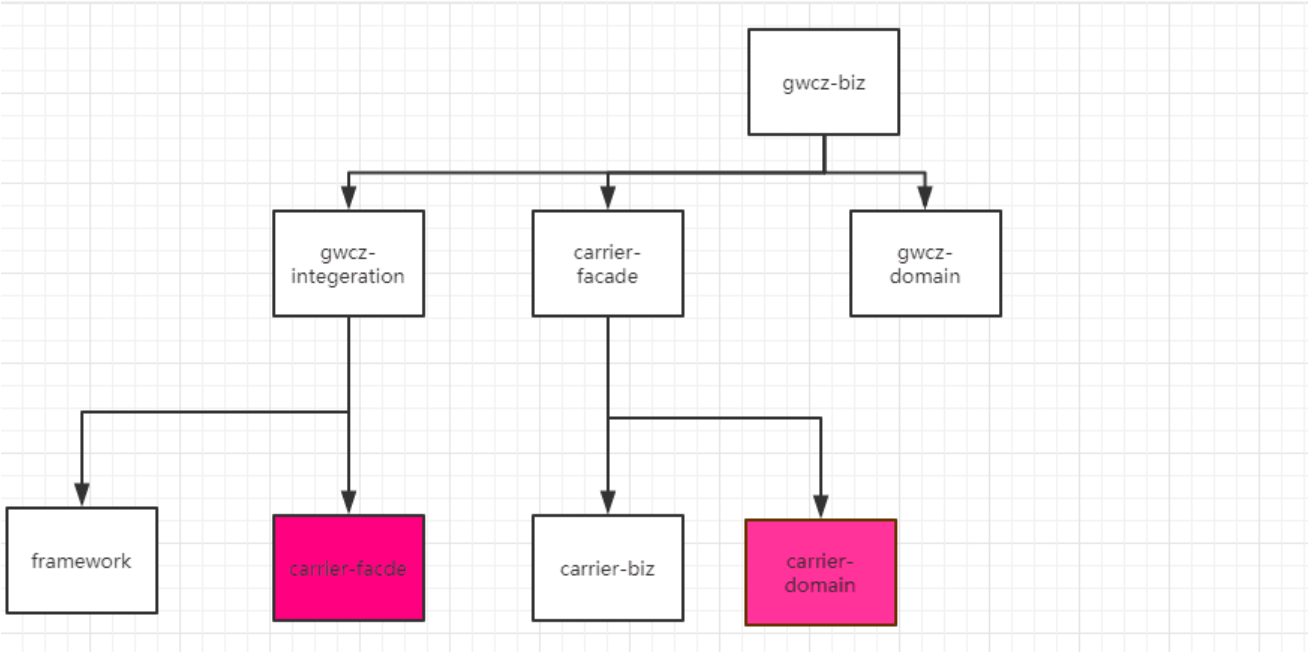
模块是否要重新打包的逻辑

1. 如果svn上面有更新，必须要重新打包
2. 如果svn上面没有更新,但是maven中央仓库的mete问价你的lastUpdate的时间戳的值与数据库中存的值不一致，需要重新打包

在maven工程中，工程间的依赖可以理解为一个树形结构，我们的算法很简单就是从叶子节点开始，层次遍历这个树形结构，对于满足有更新的模块，我们进行打包。将所有的有更新的jar包打好以后，我们重新打包 网关下面的xxx-web 打成war包即可

举例

比如说，现在我们要打包gwcz,车主的gwcz-biz下面有代码更新那么逻辑就是



例如上面的，gwcz-biz的依赖，carrier-facade和carrier-domain有更新，那么我们就打包他们。最后不论怎样