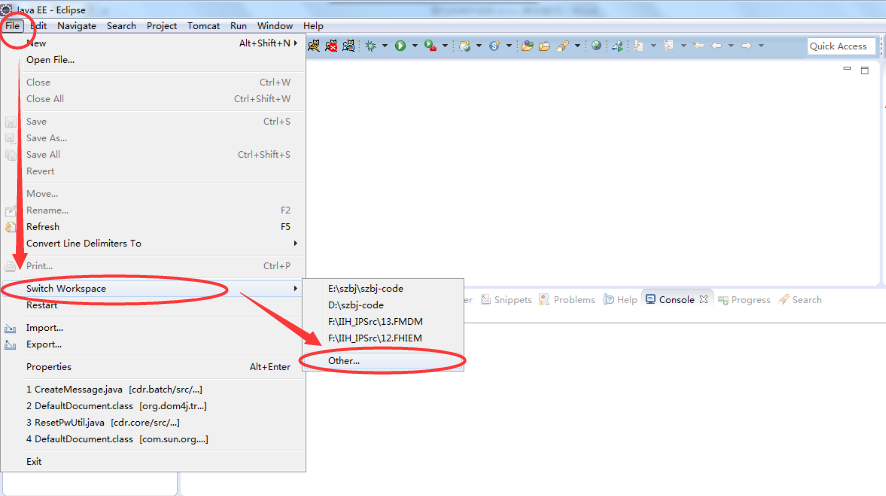
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 深圳邦健生物医疗设备股份有限公司  源代码维护说明   |  |  | | --- | --- | | **项目名称** | 心电适配器 | | **文件名称** | 源代码维护说明 | | **适用范围** | 心电适配器 | | **文件编号** | V1.0 | | **版本** | V1.0 | | **编制/日期** | 2016-12-7 | | **审核/日期** |  | | **审核/日期** |  | | **批准/日期** |  | | **电子档字节数** |  | | **生效日期** |  | |

# ****心电适配器源代码维护****

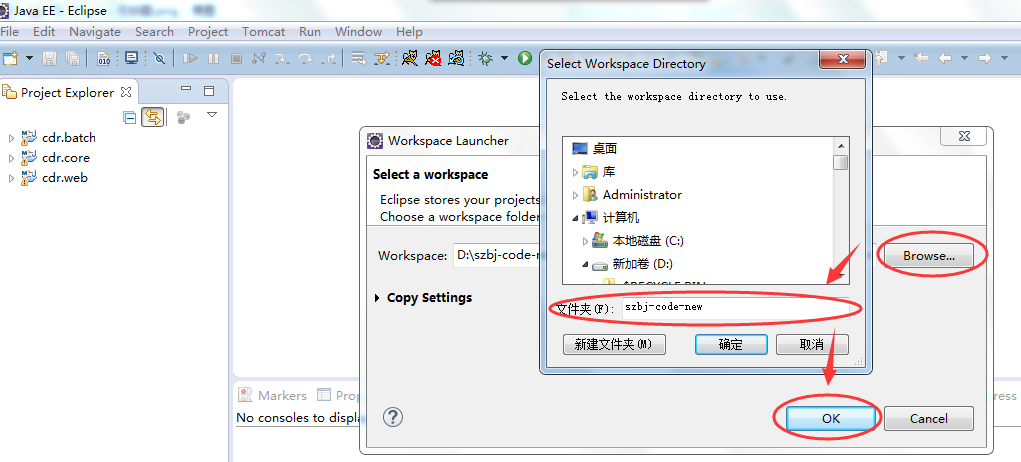
## 注意事项

拿到验收文档和代码后，源代码禁止放置在带有**中文**的目录中。

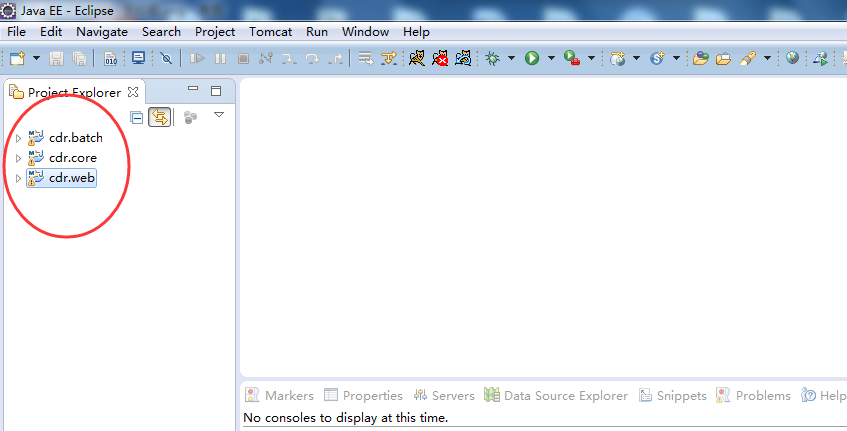
## 选择eclipse工作空间



在弹出框中，点击Browse选择工程所在目录，然后确定



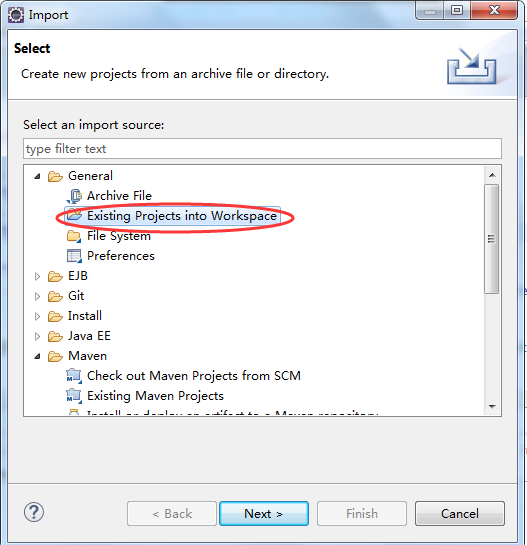
Eclipse重启之后，会重新加载该工作空间的项目，重启后，三个项目直接加载完毕。



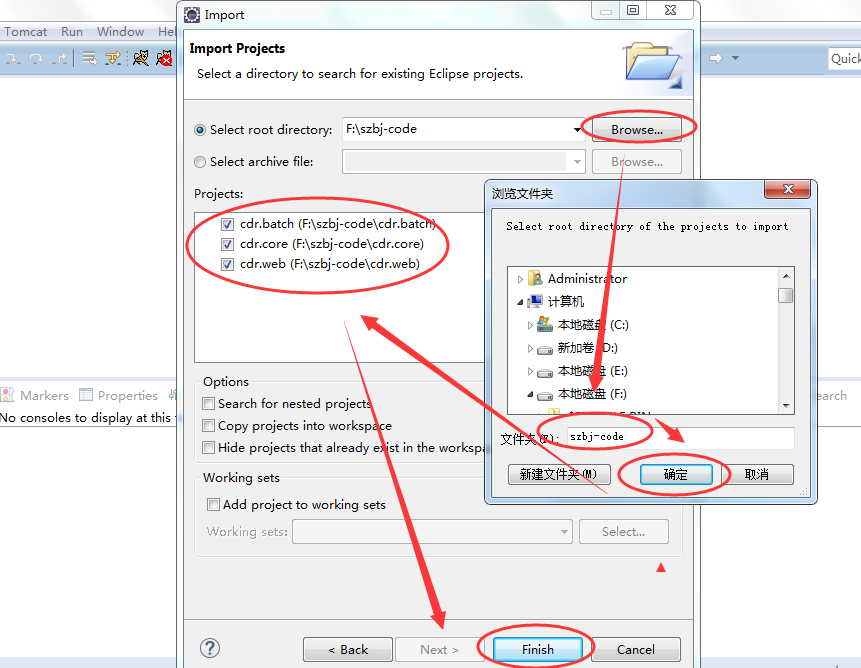
假设未加载完毕，请执行下面第3步导入工程操作，自己手动导入工程。

## 导入工程

通过File-》Import，按照下列所示图片进行操作。



选择项目所在的目录，然后点击确定按钮，三个项目就会加载上，最后点击Finish按钮完成项目的加载

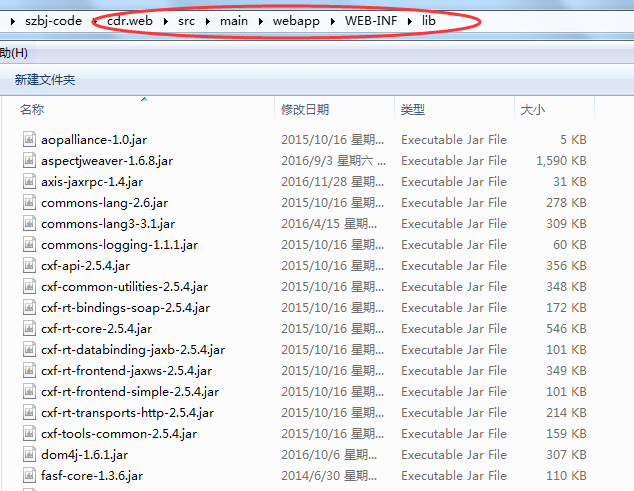


如果关联jar失败，或者因为换路径原因找不到所引用的jar包，需要进行第4步操作。

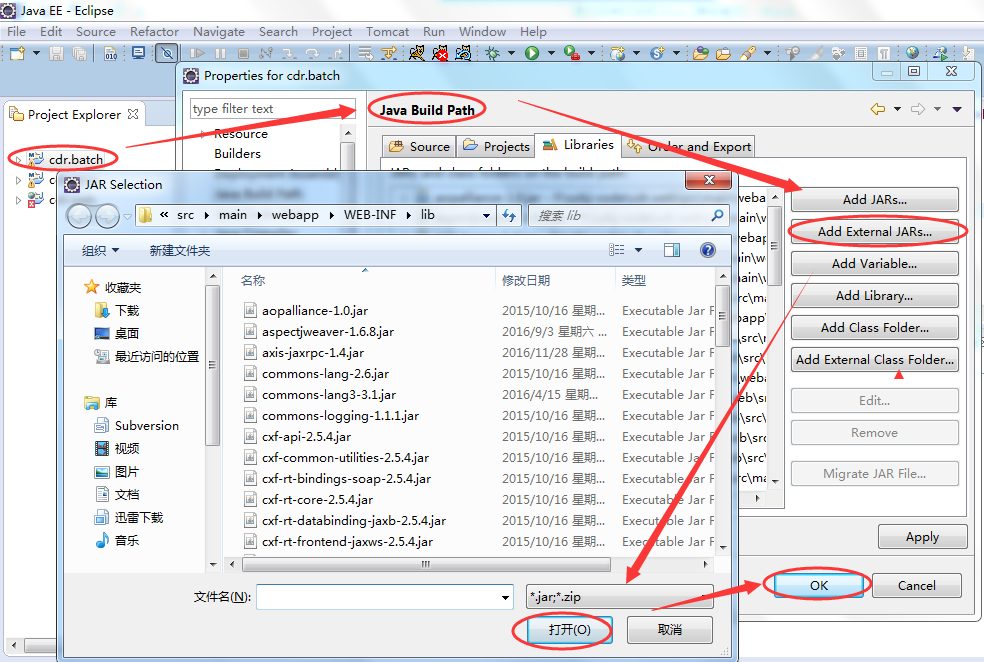
## 引入工程所需的jar包

cdr.batch、cdr.core、cdr.web三大工程所需要的jar包，统一放置在目录：

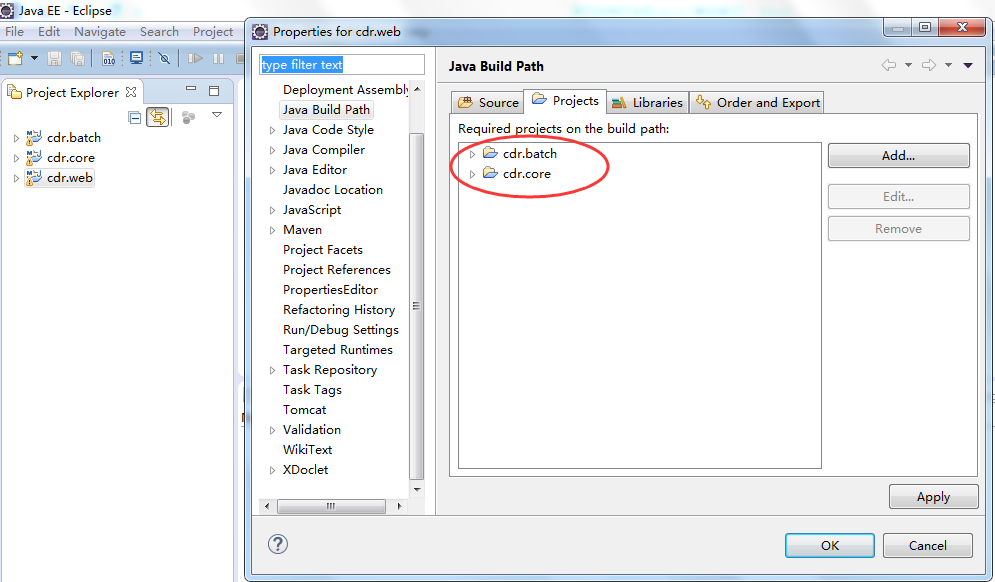
cdr.web\src\main\webapp\WEB-INF\lib，如下图所示。



每个工程都需要加载一下这个目录的jar包，具体操作如下图所示：

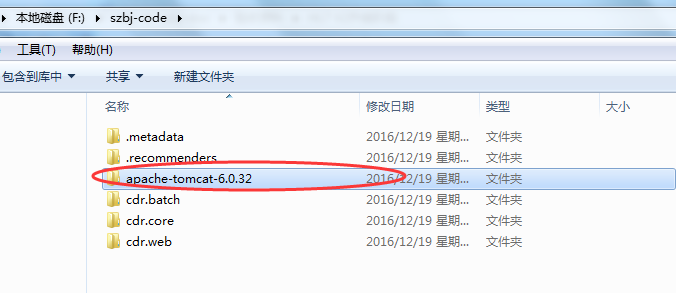


注意：cdr.web工程情况较为特殊，还需要依赖cdr.batch和cdr.core工程，在eclipse中，手动引入即可，如下图所示。

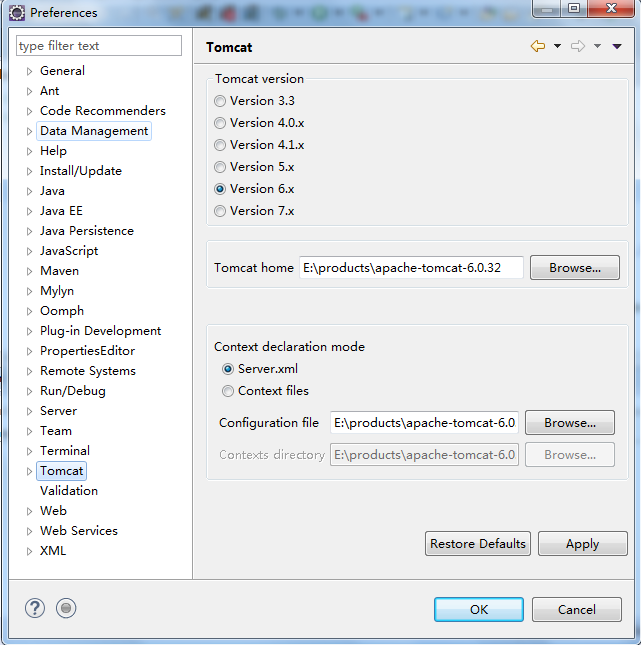


## Tomcat 配置

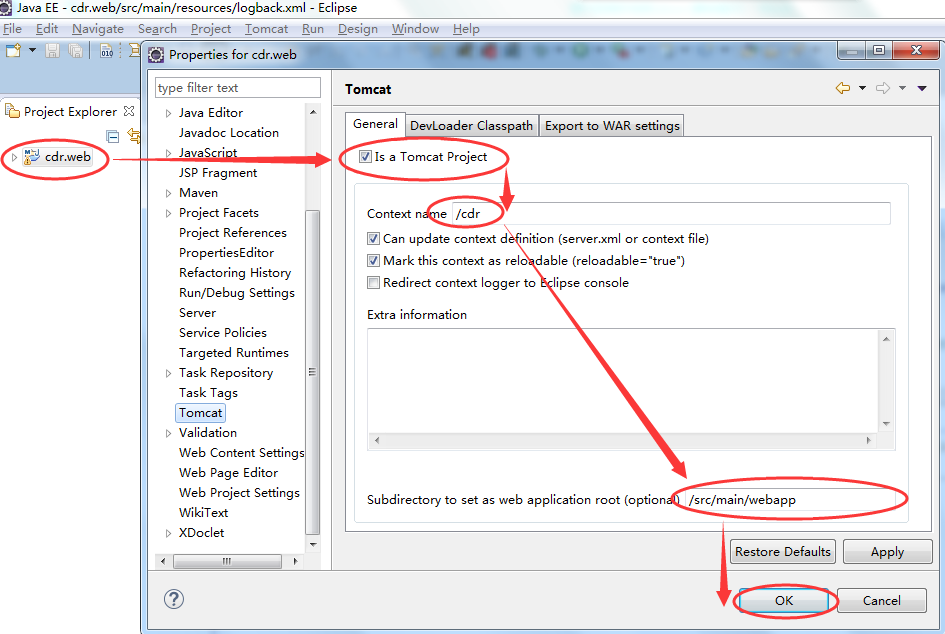
Tomcat我们提供了一个tomcat6，跟源代码放置在同一个目录之下，加载这个即可，具体操作如下：



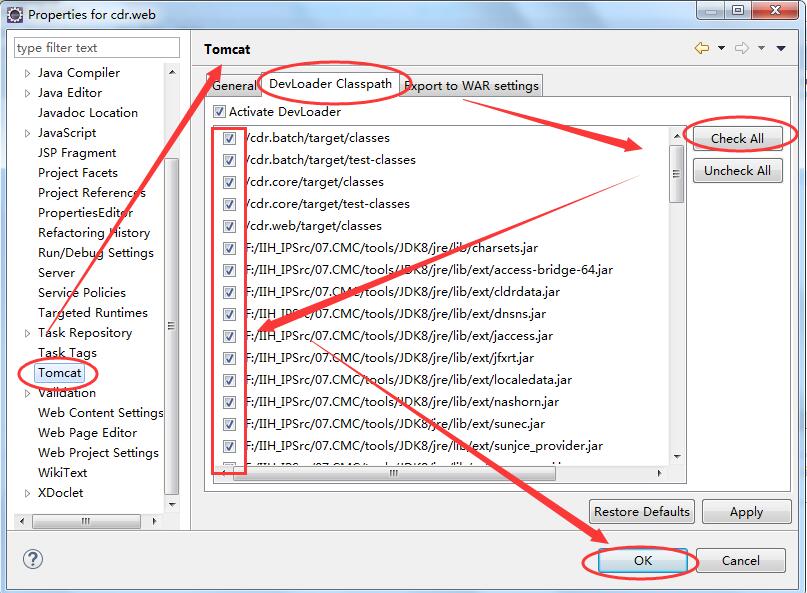
1）在Window-》Preference中，选择tomcat，配置tomcat服务。



2）右键cdr.web工程-》选择Properties，弹出下图所示界面，进行下面配置。（如果将eclipse的工作空间指定到了源代码所在目录，这一步可以忽略，会自动加载上）



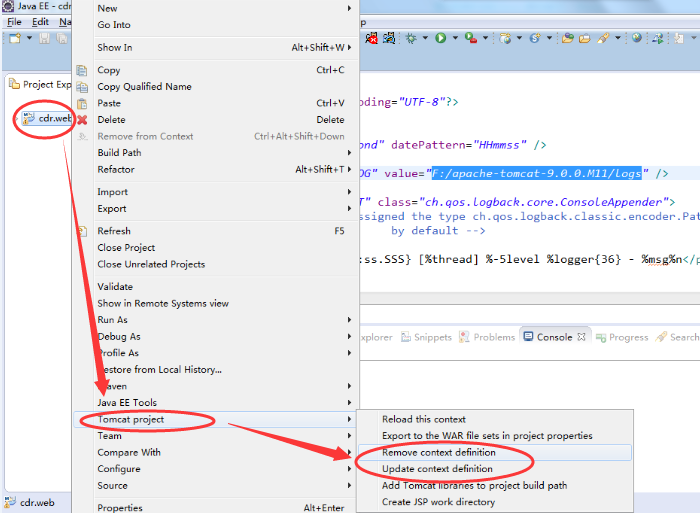
紧接着，切换到DevLoader Classpath页面，将所有的jar包都勾选上，可以通过Check All按钮完成该操作，具体如下图所示：



3）再次右键cdr.web工程-》Tomcat project，分别执行：

a）Remove context definition

b）Update context definition

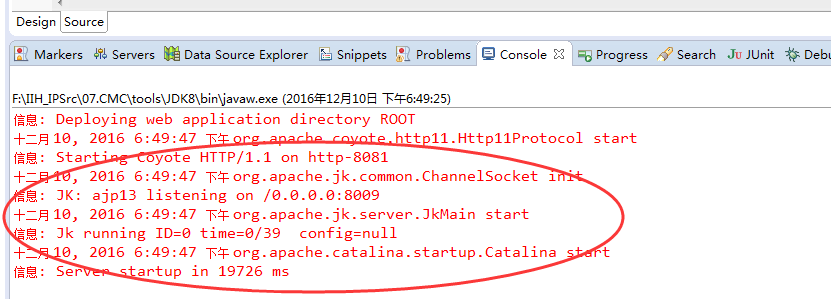


5）如果安装了tom猫插件的话，直接点击运行tomcat。



6）确认服务已经启动OK

当看到控制台输出以下信息，8081端口以tomcat实际端口为准，并检查控制台是否存在报错，没有报错，说明服务启动成功。这时利用soapUI就可以进行测试了。

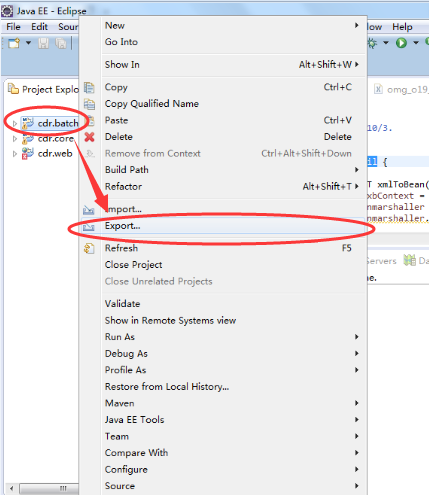


## 打包

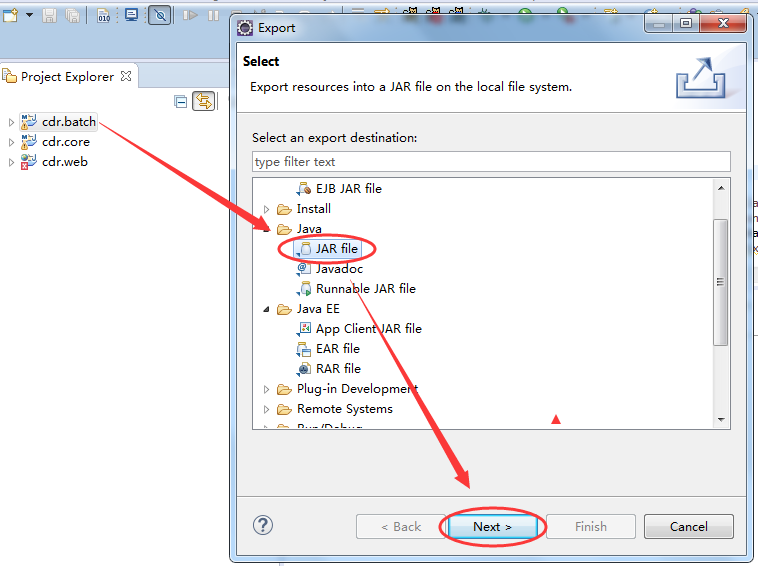
由于项目包含三个工程，所以打包的时候，要将cdr.batch和cdr.core分别打成jar包。cdr.web打成war包，然后将batch和core的jar放入到war包中，具体操作如下：

1. cdr.batch和cdr.core打jar包

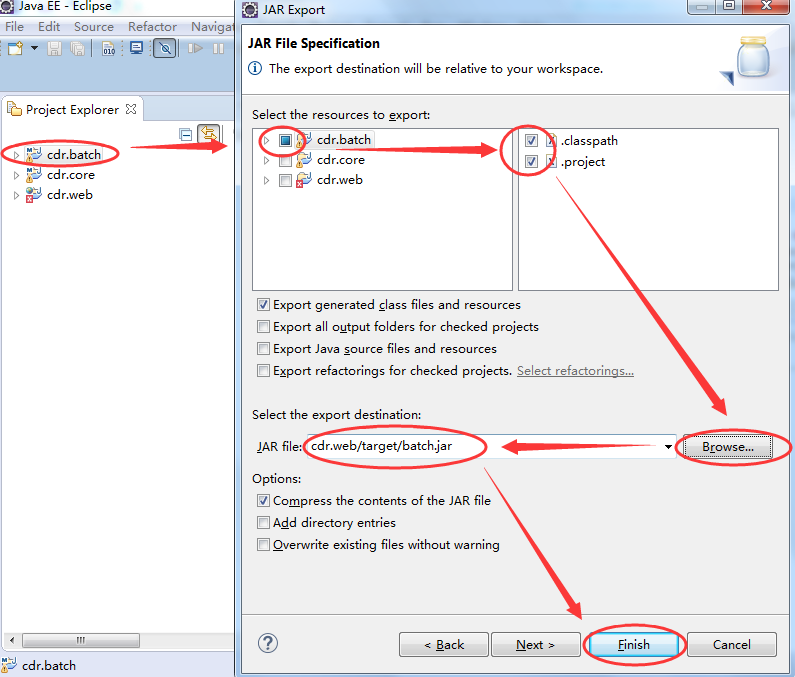
第一步：右键选择该项目，选择Export…



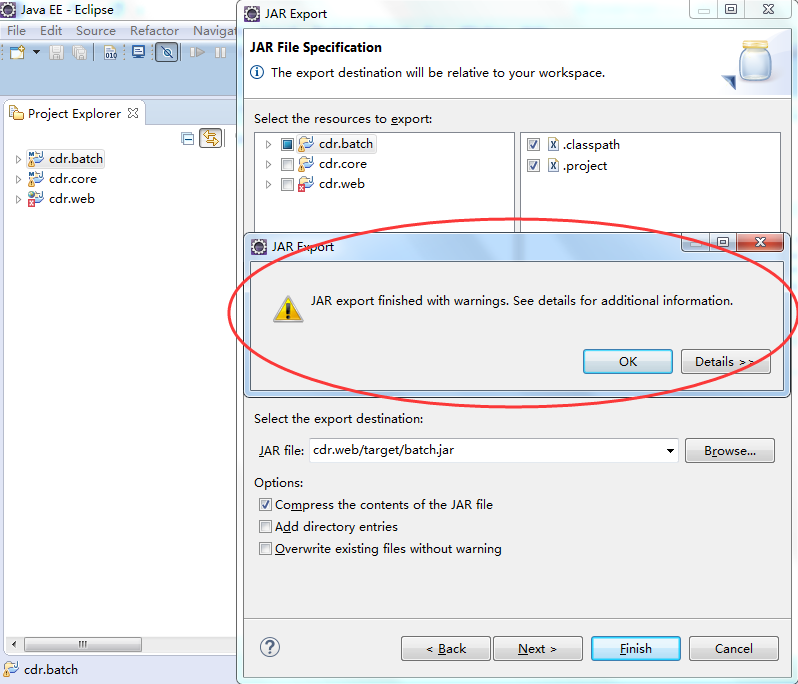
第二步，在弹出框中，选择Java下的JAR file，然后进行next…（下一步），如下图所示。



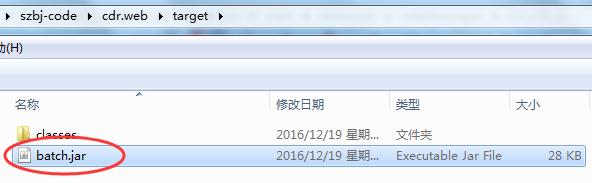
第三步：如下图所示，重点选择一个JAR file的输出目录，并为该jar起名，然后点击Finish按钮。



第四步：紧接着，如果弹出下图所示框，没关系，直接点击OK按钮。

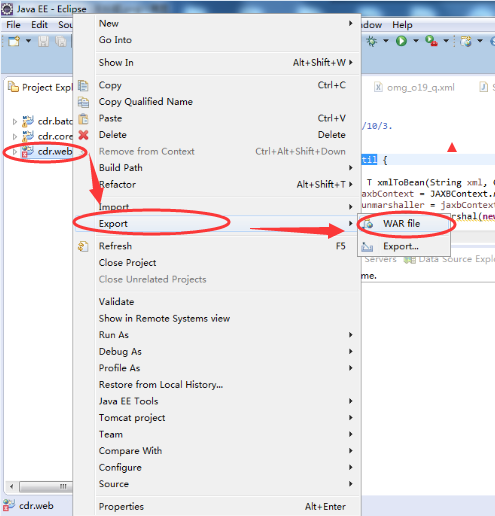


这样，恭喜，jar包就打成功了，打core包的步骤是一样的，这里就不再重复写了。

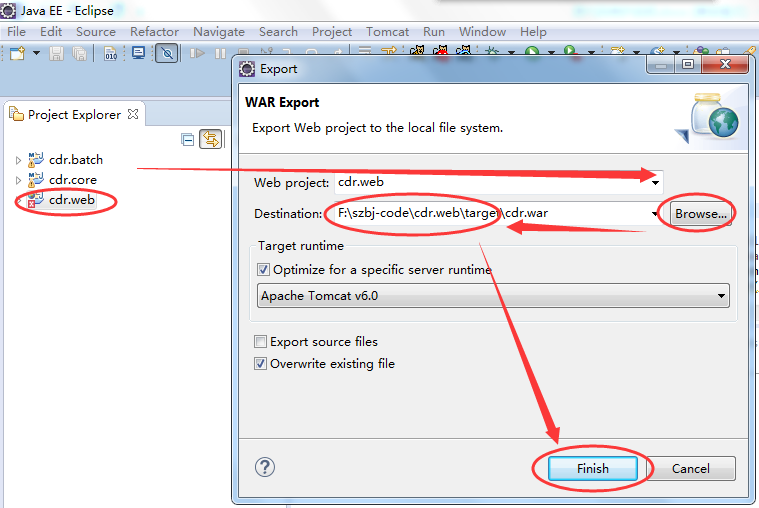


1. cdr.web打war包

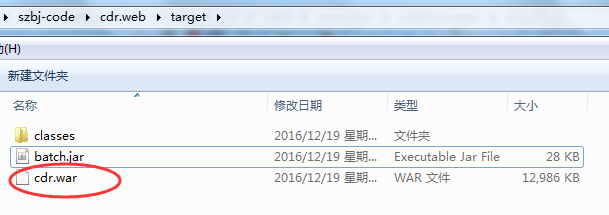
第一步：右键该工程，选择Export..再选择WAR file,如下图所示。



第二步，在弹出框中，选择输出路径，指定文件夹名cdr.war，然后点击Finish完成操作。

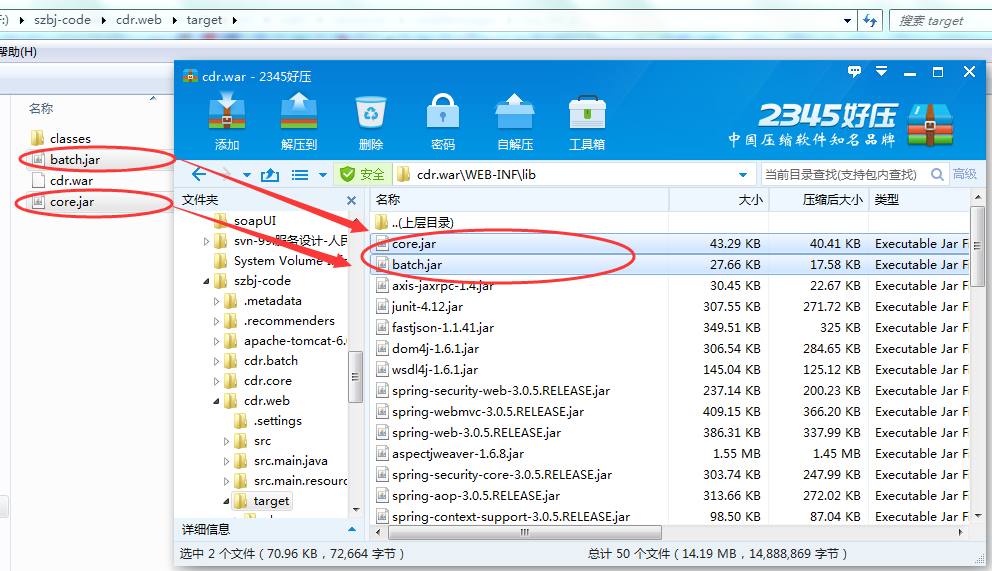


最后，恭喜cdr.war打包成功，见下图所示。



1. jar包和war包整合

用好压打开cdr.war，进入到目录：cdr.war\WEB-INF\lib下，将生成的batch和core对应的jar包拽入。



## 项目日志查看

项目日志分为tomcat日志和系统日志；

1. tomcat日志

tomcat日志是tomcat服务器产生的日志，主要记录在

a）catalina.YYYY-MM-DD.log

b) localhost.YYYY-MM-DD.log

1. 系统日志

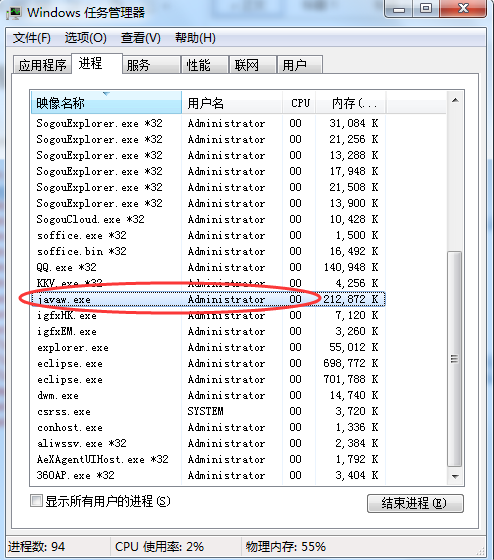
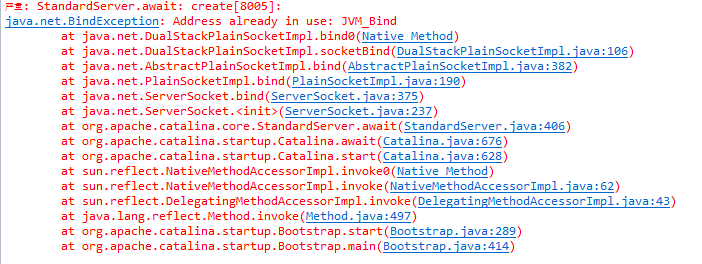
系统日志由下图所示logback.xml文件来控制，记录在

cdr-YYYY-MM-DD.log

## 8.项目日常错误分析

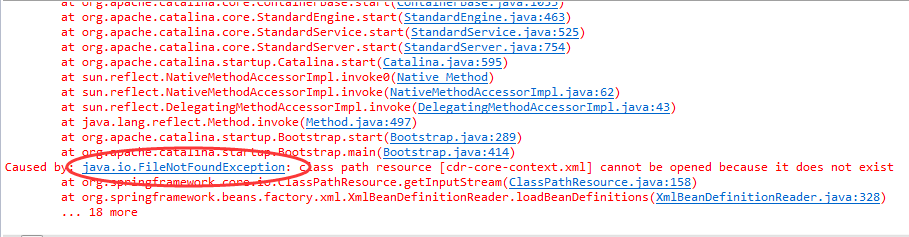
1. 已经开启了tomcat，未关闭，又打开一个，就会报如下错误。

解决办法：关闭之前的，若找不到，那么直接启动任务管理器，删掉javaw.exe进程。



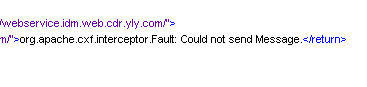
1. 报文件找不到

解决办法：将所需要的jar或者外部工程引入即可。



1. soapUI调用失败

解决办法：确定对方的接口是否能够被调通，调不通就会报下图所示错误。



其它错误，具体问题具体分析。

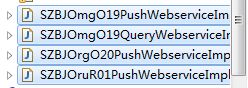
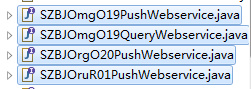
## 源代码

源代码主要分为以下两部分内容：

1. webservice接口文件

分为webservice和webservice的实现类文件，如下图所示，在实现类中，主要包含数据转化的具体逻辑，以及如何调用底层jar包的。

………



其中，消息类型关键字。

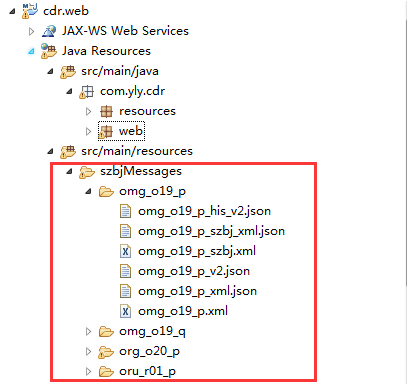
**OmgO19 :** 检查申请单，Push：推送情况 Query：查询情况

**OrgO20：**医嘱执行状态

**OruR01：**检查报告

1. 各webservice对应的json和xml文件

Json和xml文件都放置在下图所示目录，也是用消息类型关键字进行区分。其中，P：Push，推送，Q：Query，查询。



以omg\_o19\_p：检查申请单推送情况为例。

当HIS（目前用soapUI模拟），调用心电适配器的检查申请单推送接口（SZBJOmgO19PushWebserviceImpl?wsdl），V2消息数据到达适配器程序，适配器利用omg\_o19\_p\_his\_v2.json把V2消息被解析成MAP，然后程序利用omg\_o19\_p\_szbj\_xml.json和omg\_o19\_p\_szbj.xml，把MAP数据封装设置到xml模版中，再将该xml推送给心电系统。

心电系统收到xml检查申请单后，会返回一个xml数据的结果给适配器，适配器程序利用omg\_o19\_p\_szbj\_xml.json将该xml解析为MAP，再利用omg\_o19\_p\_his\_v2.json将MAP封装组织，调用hapi的jar包，将map组织为V2消息返回给HIS（呈现在soapUI界面上）。



如果对hapi感兴趣，可以访问以下链接做进一步了解：

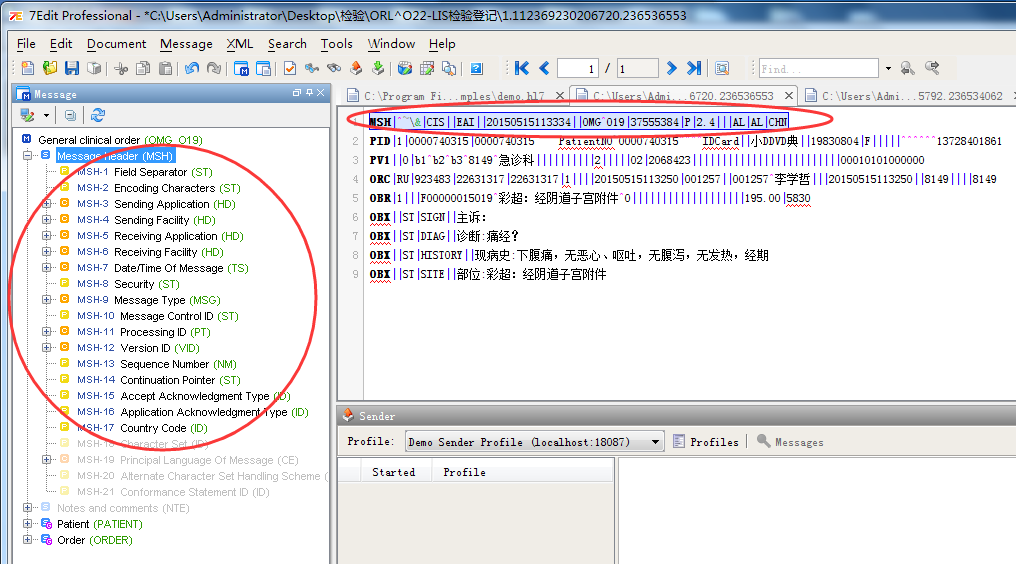
http://hl7api.sourceforge.net/

医嘱执行状态和检查报告跟检查申请单的过程类似。

## 解读V2消息

用7Edit工具打开V2的消息，如下图所示，左侧是对应的树结构，右侧是V2消息，配置的json文件中，path路径就是依据左侧的树结构来的。

具体的配置可参见目前现有的内容。



**10.1 Json文件中的配置说明。**

1）单个节点配置

"FieldSeparator": {

"desc": "字段分隔符", ---字段描述，根据实际情况赋值

"path": "/MSH-1", ---对应的V2消息或者XML的节点路径

"display": "", ---字段样式，赋值：空

"type": "string", ---字段类型，赋值：string

"format": "", ---字段格式，赋值：空

"multiplicity": "0..1" ---字段复杂度，如果是单个字段，赋值：0…1,如果是带有子节点的，赋值：0…n

}

上述各字段变动最大的是path字段的配置。

2）复杂节点配置：

带有子节点的复杂情况如下所示，唯一不同的就是带有一个child，child中也是跟基础节点的配置保持一致即可。

"PIdList": {

"desc": "患者标识",

"path": "/PATIENT",

"display": "",

"type": "complex",

"format": "",

"multiplicity": "0..n",

"child": {

"Id": {

"desc": "标识号",

"path": "/PATIENT/PID-3(#)-1",

"display": "",

"type": "string",

"format": "",

"multiplicity": "0..1"

},

"IdTypeCode": {

"desc": "标识类型",

"path": "/PATIENT/PID-3(#)-5",

"display": "",

"type": "string",

"format": "",

"multiplicity": "0..1"

}

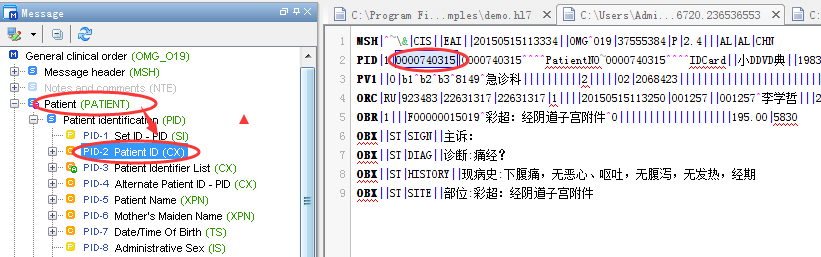
}

},

10.2 **Path路径赋值：**

1. V2消息

V2消息的path路径，一定要借助7Edit这个工具来看。比如给patientId赋值的时候，参看7Edit工具中的路径，取出"/PATIENT/PID-2"进行对path赋值。



"PatientId": {

"desc": "患者id",

"path": "/PATIENT/PID-2",

"display": "",

"type": "string",

"format": "",

"multiplicity": "0..1"

},

其它的节点按此类推。

1. XML消息

记住：从根路径开始，一直到该节点，比如：

/Msg/Head/MSH/FieldSeparator

/Msg/Body/Application/ObsResult

复杂节点也是用"multiplicity": "0..n",附加一个child配置，具体如下所示。

"Obxs": {

"desc": "观察结果",

"path": "/Msg/Body/Application/ObsResult",

"display": "",

"type": "complex",

"format": "",

"multiplicity": "0..n",

"child": {

"ValueType": {

"desc": "值类型",

"path": "ValueType",

"display": "",

"type": "string",

"format": "",

"multiplicity": "0..1"

},

"ObsIdentifierCode": {

"desc": "标识符编码",

"path": "ObsIdentifierCode",

"display": "",

"type": "string",

"format": "",

"multiplicity": "0..1"

},

"ObsIdentifierName": {

"desc": "标识符名称",

"path": "ObsIdentifierName",

"display": "",

"type": "string",

"format": "",

"multiplicity": "0..1"

},

"ObsValue": {

"desc": "观察值",

"path": "ObservationValue",

"display": "",

"type": "string",

"format": "",

"multiplicity": "0..1"

}

}

}