Call device driver APIs based on the WebService mode

Preface

WebService mode has the following advantages when call device drivers APIs.

**·**It is highly compatible, friendly to programs developed based on BS architecture, and can be called by any browser.

**·**Low coupling, any language can be called based on its own HTTP, no longer need to focus on the language and platform of the driver itself.

Procedures

Now let's use “A8Capture.dll” as an example to describe how to implement it,“A8Capture.dll”is the device driver。

Environment

·Windows11 64bit

·VS 2017

·eclipse-win32

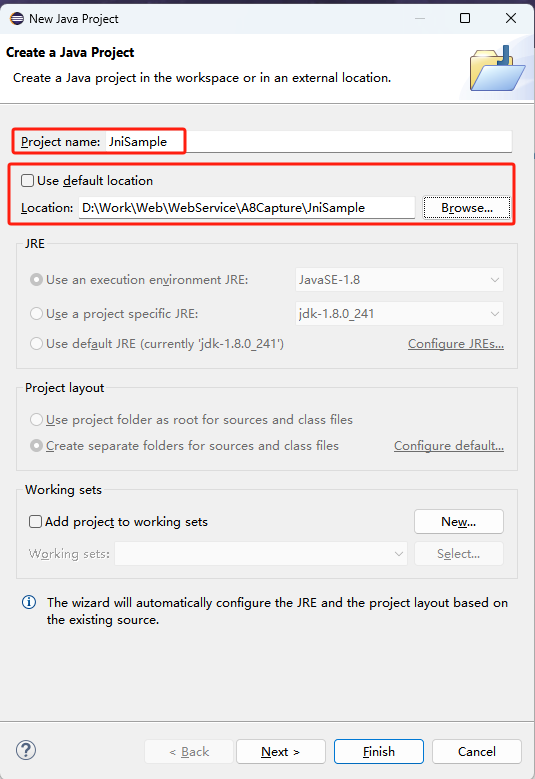
·jdk-8u241-windows-i586

.Tomcat 9.0

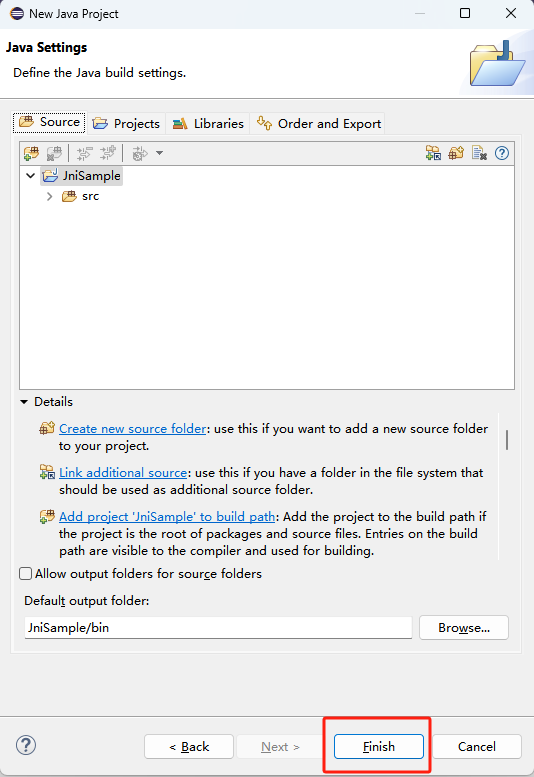
1. How Java calls C++ dlls via JNI

If you're familiar with this, you can ignore this step.However, we recommend writing a simple program to verify the JNI dll, so that it can be compared with the flowlling WebService calling JNI dll.

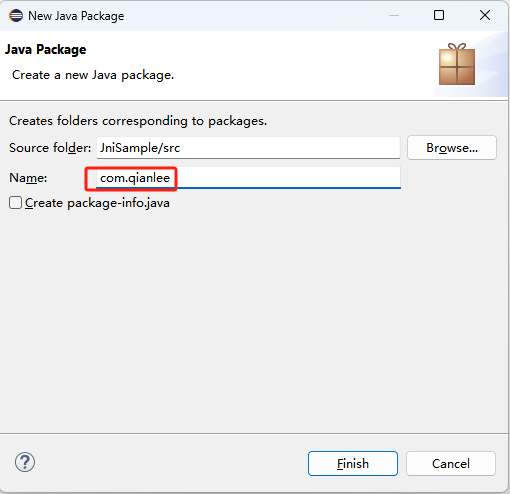
* 1. **Create a new java project**



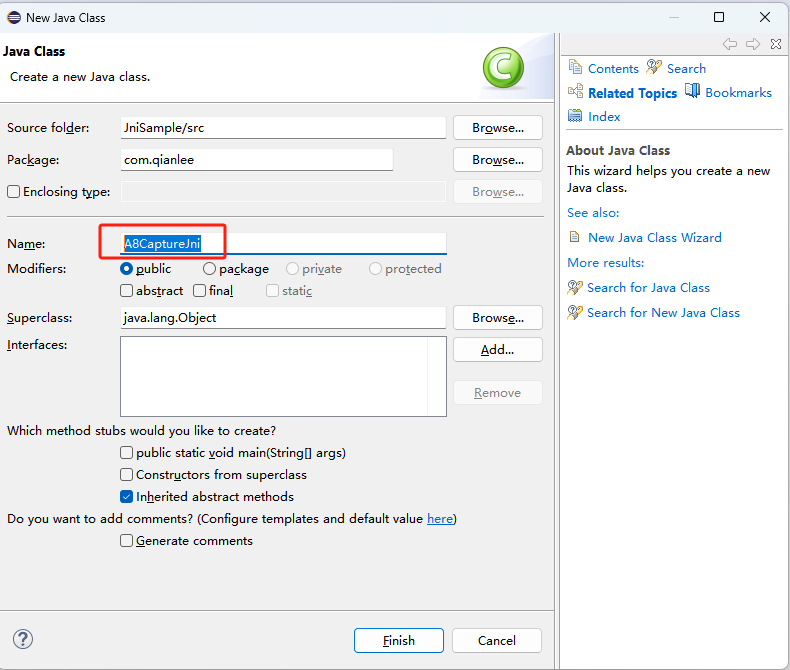
* 1. **Finish create a new java project**



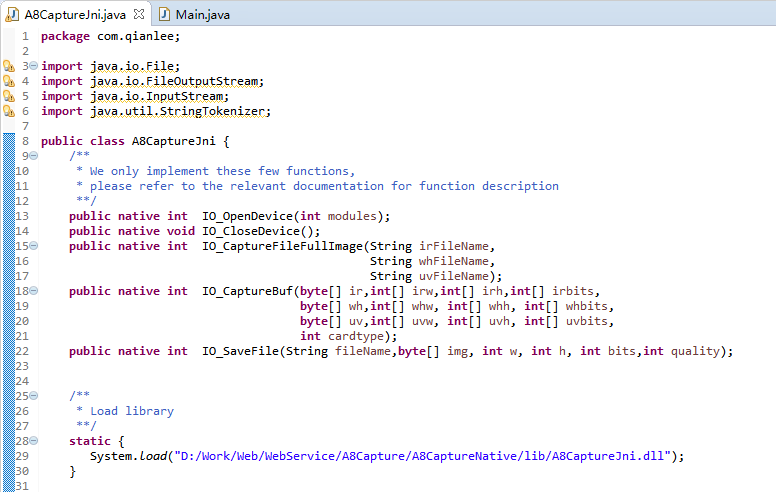
* 1. **Create new java package**



* 1. **Create new java class**

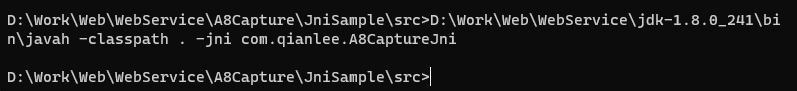


* 1. **Write the code as follows**



* 1. **Generate .h file**

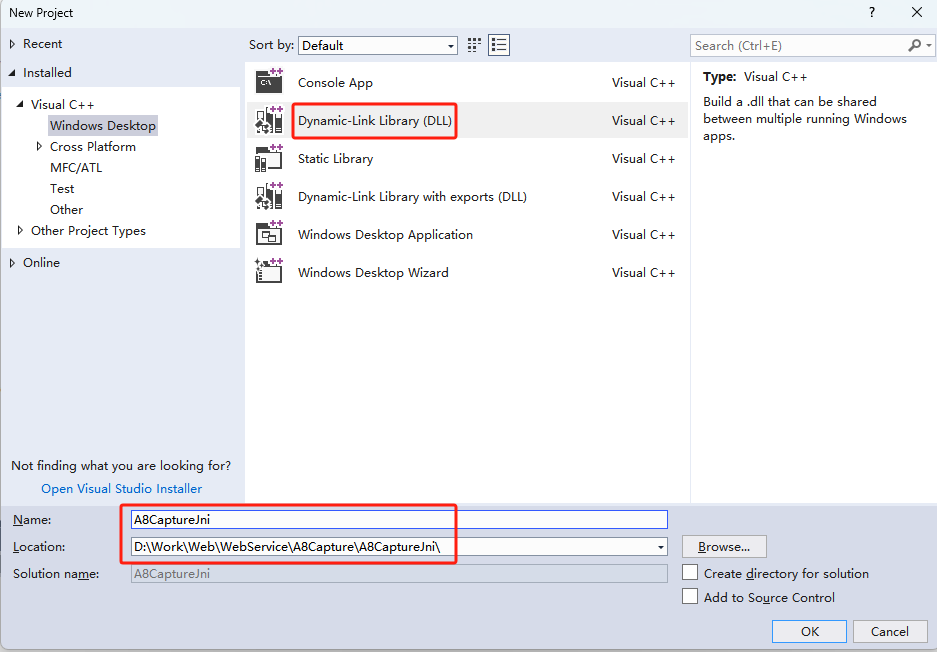
In “D:\Work\Web\WebService\A8Capture\JniSample\src” directory,execture “javah -classpath . -jni com.qianlee.A8CaptureJni”.



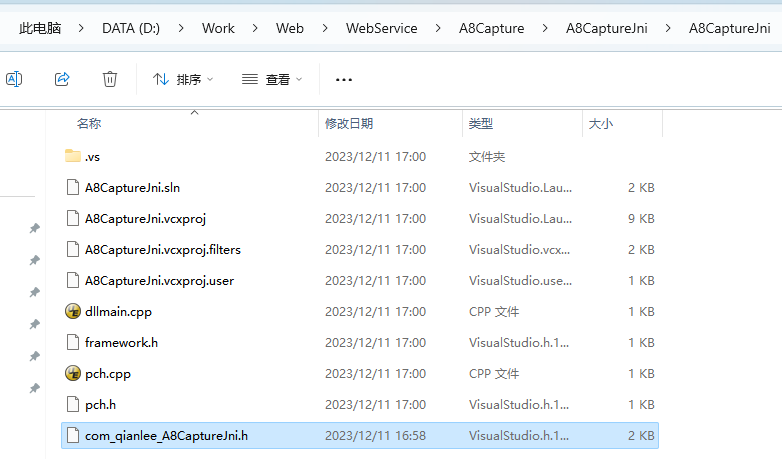
* 1. **Now go to generate c++ jni dll.**

1. Write C++ JNI DLLs

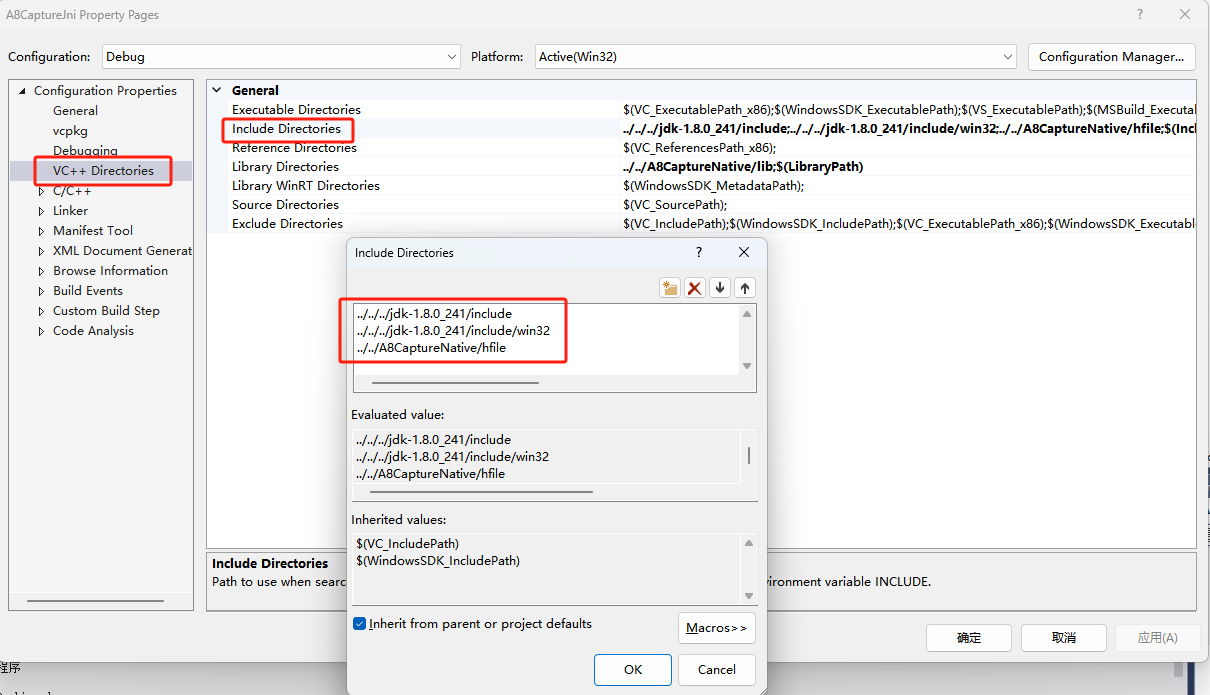
**2.1.Use VS create a new dll projiect.**



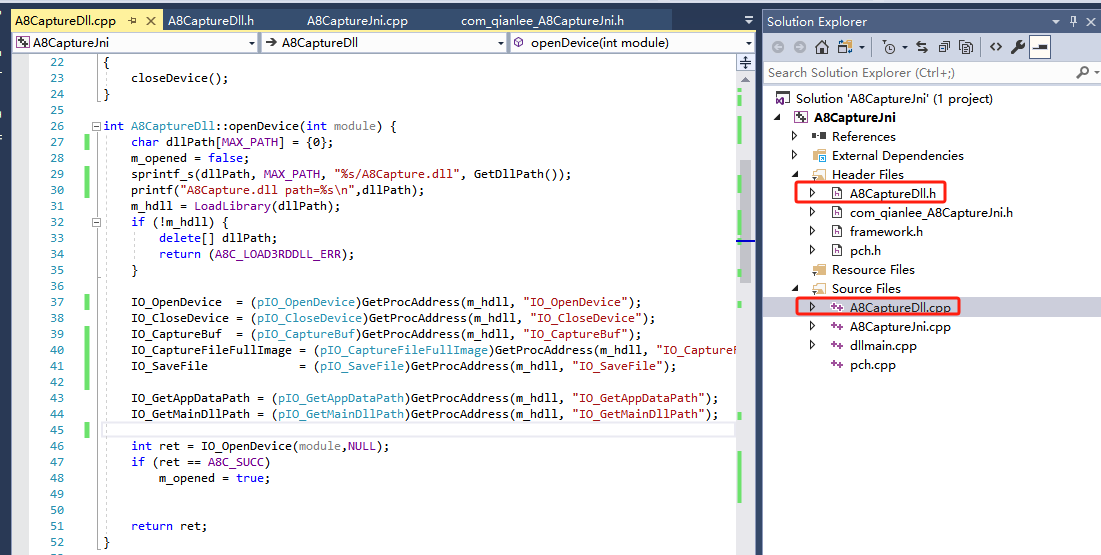
**2.2. Copy “com\_qianlee\_A8CaptureJni.h” to the project directory**



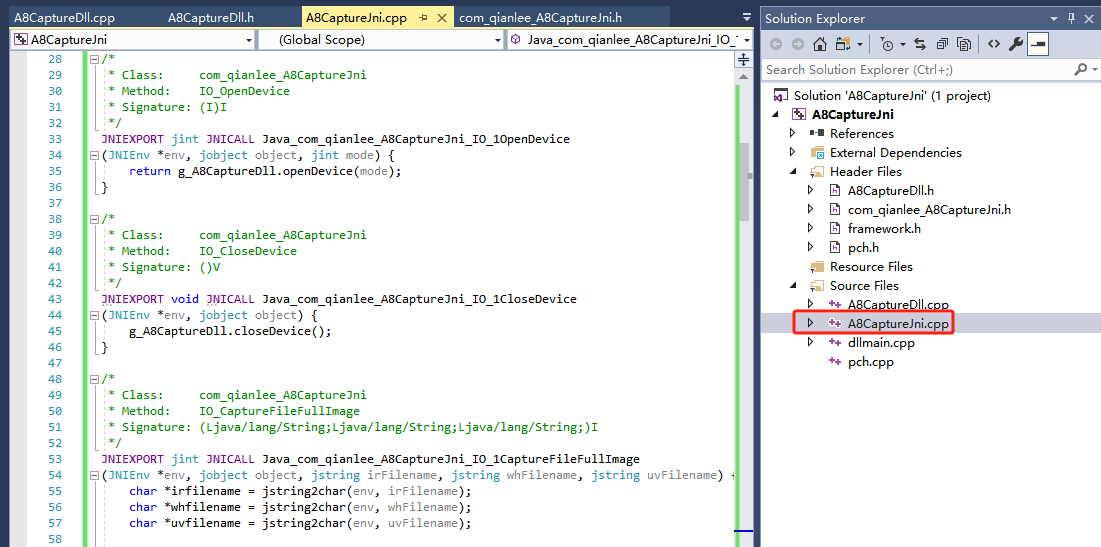
**2.3. Add include directory**



**2.4.We recommend that you call the driver DLL library file dynamically, Please refer to “A8CaptureDll” class.**



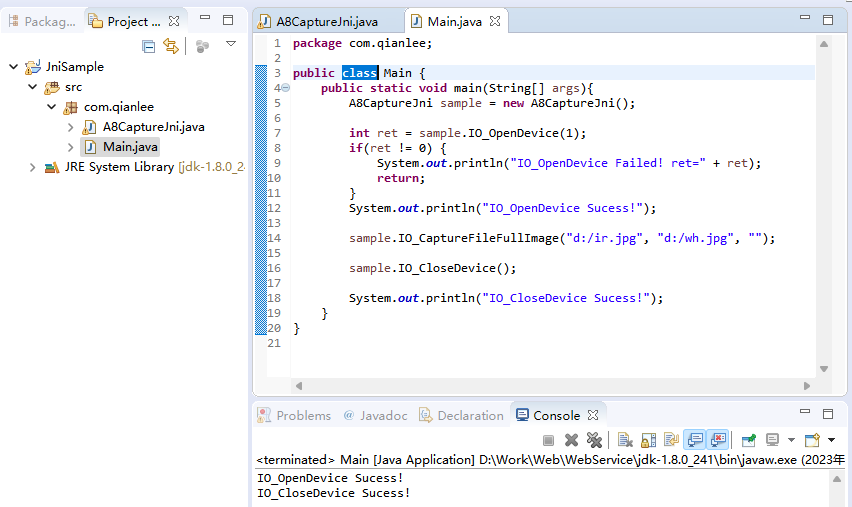
**2.5. Create “A8CaptureJni.cpp” to implement “com\_qianlee\_A8CaptureJni.h” functions**



**2.6. build and then generate “A8CaptureJni.dll”**

**2.7.copy “A8CaptureJni.dl” to “D:\Work\Web\WebService\A8Capture\A8CaptureNative\lib”, we`ll load the dll in this directory.**

**2.8.return eclipse, and open “JniSample” project, create new class “Main”,the test jni dll.**

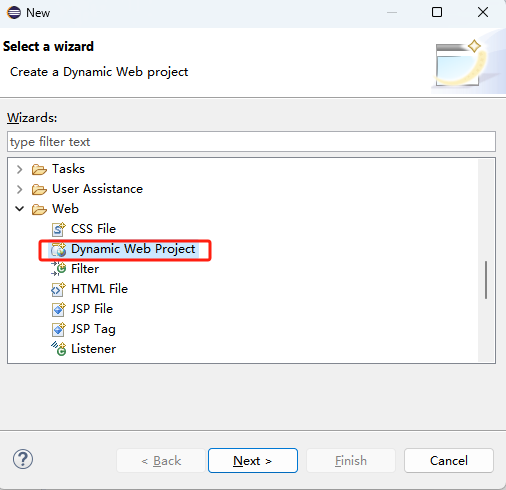


1. Create WebService

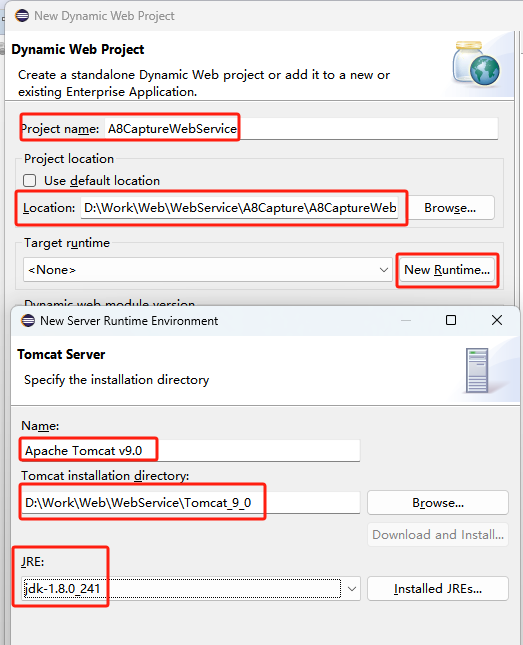
For A8Capture, because the "A8CaptureJni.dll" has been generated, the previous two steps can be ignored.

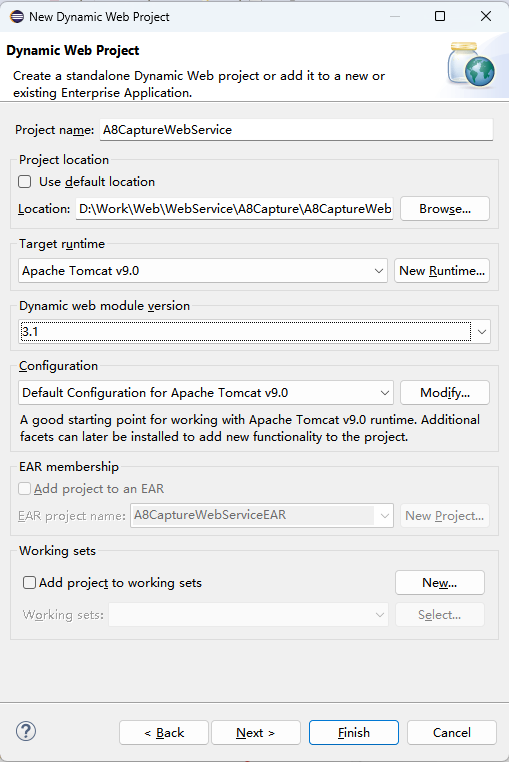
Please refer to ：“[Build RESTful Service in Java using JAX-RS and Jersey (Celsius to Fahrenheit & Fahrenheit to Celsius) • Crunchify](https://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey/)”

* 1. **Eclipse->File->New->Other->Web->Dynamic Web Project**

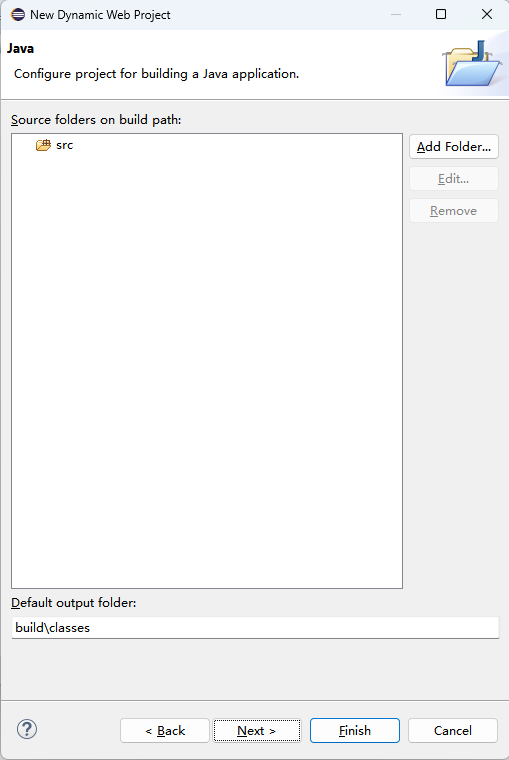


* 1. **Next**

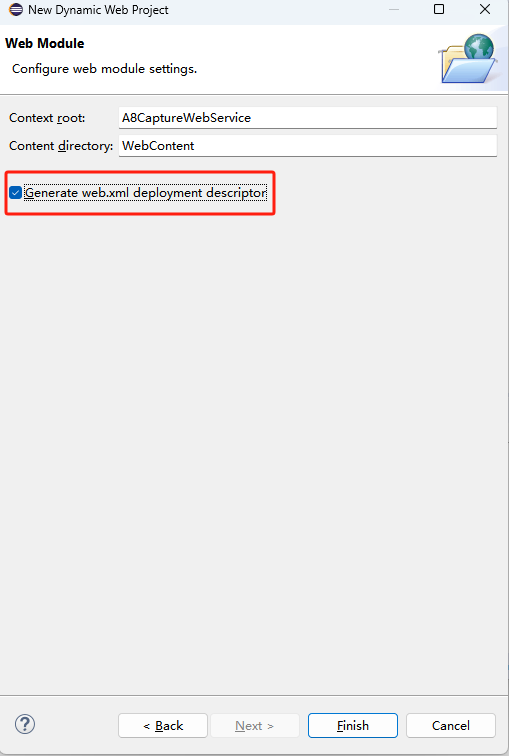




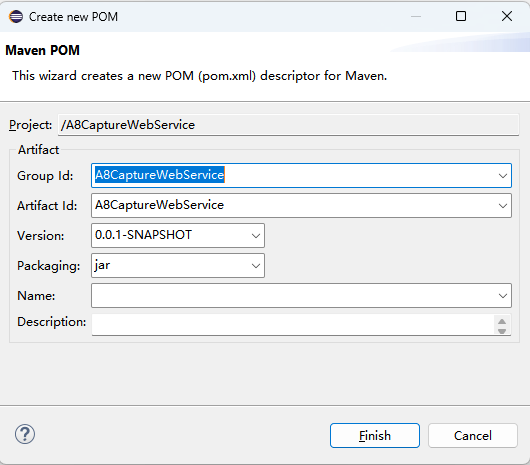
* 1. **Next**



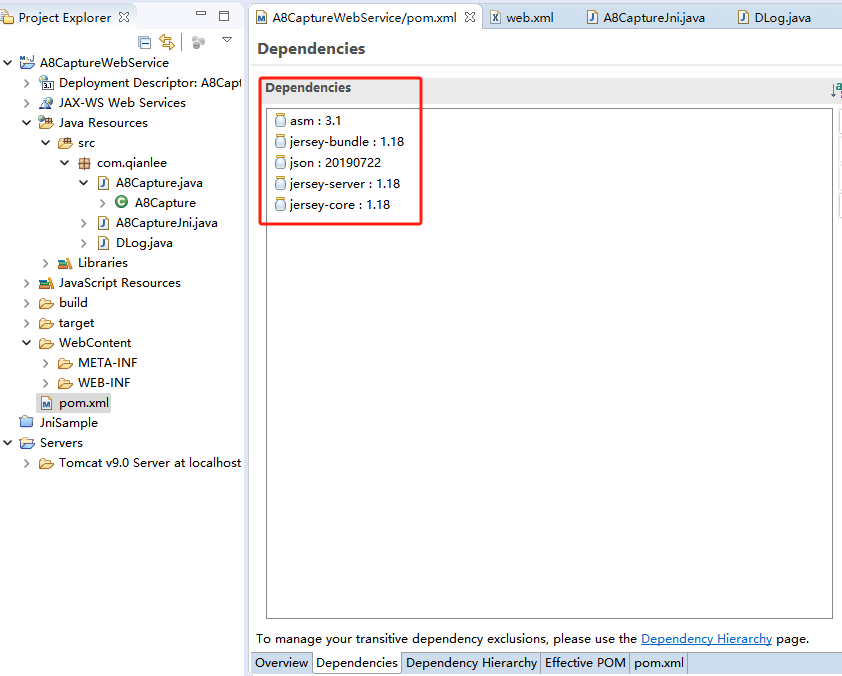
* 1. **Next,then Finish**



* 1. **“A8CaptureWebService” right-click -> Configrue-> Convert to maven project**



* 1. **Open pom.xml file and add below dependencies.**



**This is my “pom.xml”:**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>A8CaptureWebService</groupId>

<artifactId>A8CaptureWebService</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<build>

<defaultGoal>compile</defaultGoal>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

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

<artifactId>maven-war-plugin</artifactId>

<version>3.2.3</version>

<configuration>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>asm</groupId>

<artifactId>asm</artifactId>

<version>3.1</version>

</dependency>

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-bundle</artifactId>

<version>1.18</version>

</dependency>

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20190722</version>

</dependency>

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-server</artifactId>

<version>1.18</version>

</dependency>

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-core</artifactId>

<version>1.18</version>

</dependency>

</dependencies>

</project>

* 1. **Modify “WebContent\WEB-INF\web.xml” like this:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"* id=*"WebApp\_ID"* version=*"3.1"*>

<display-name>A8CaptureWebService</display-name>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>A8Capture Web Service</servlet-name>

<servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

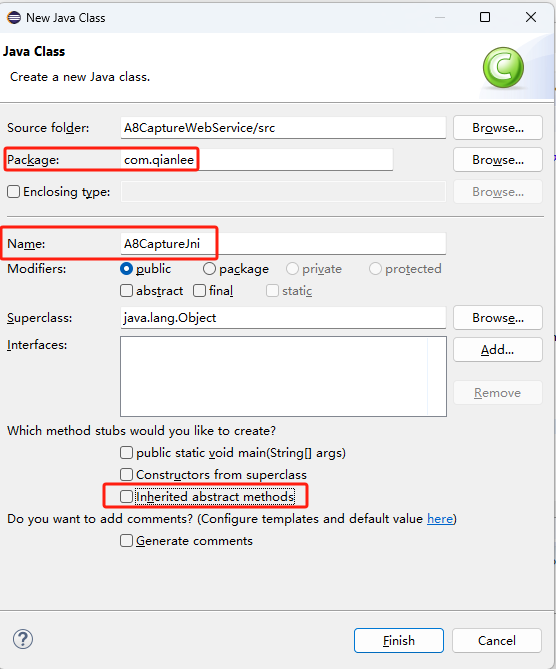
<servlet-name>A8Capture Web Service</servlet-name>

<url-pattern>/\*</url-pattern>

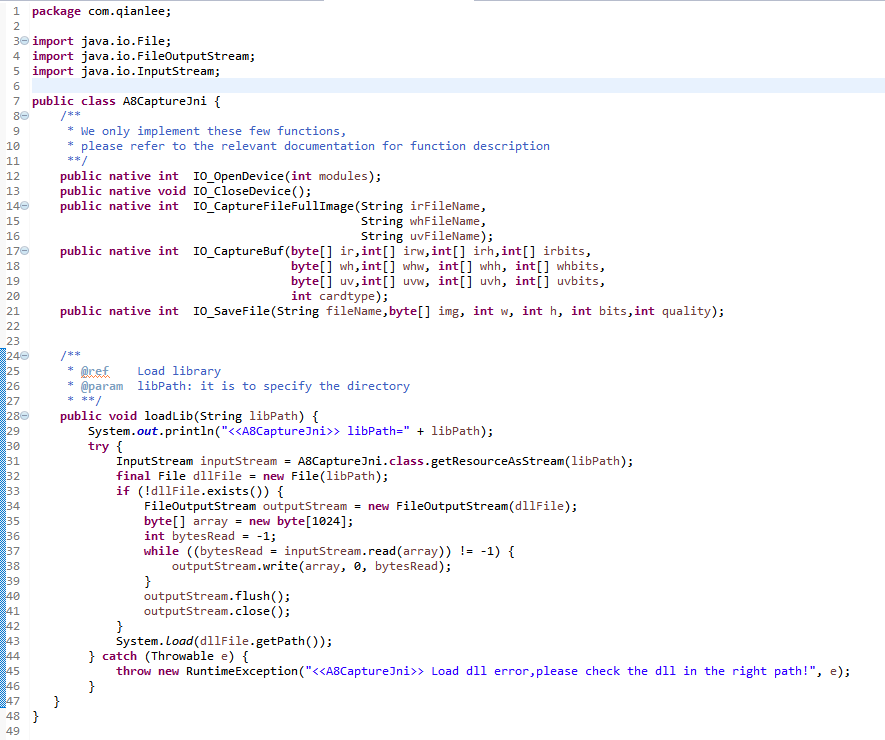
</servlet-mapping>

</web-app>

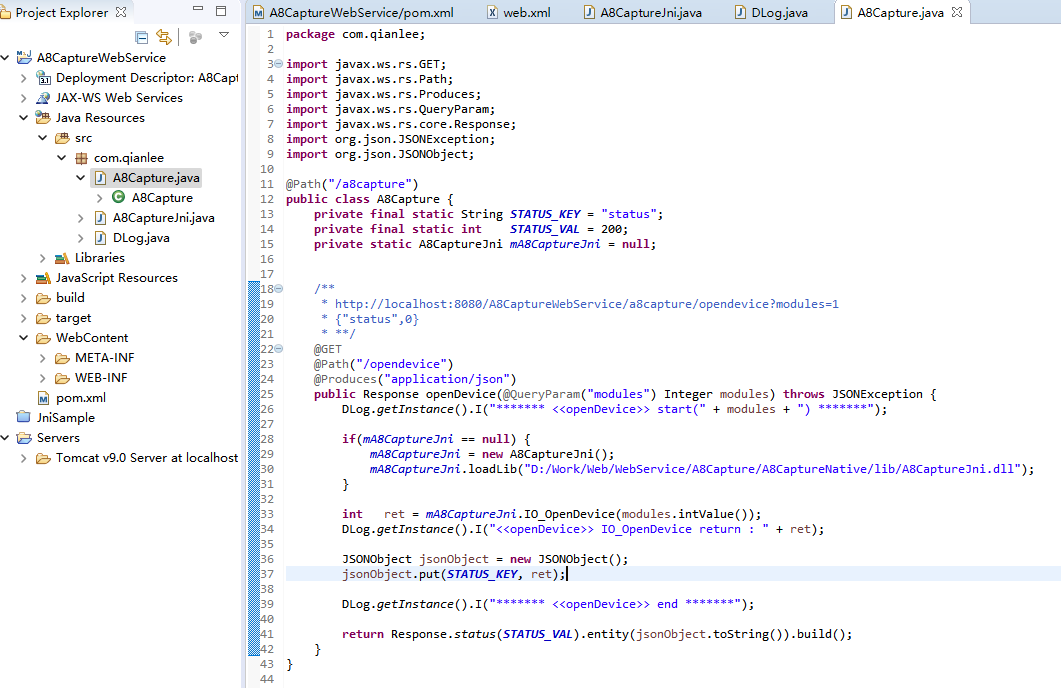
* 1. **Create new java class**



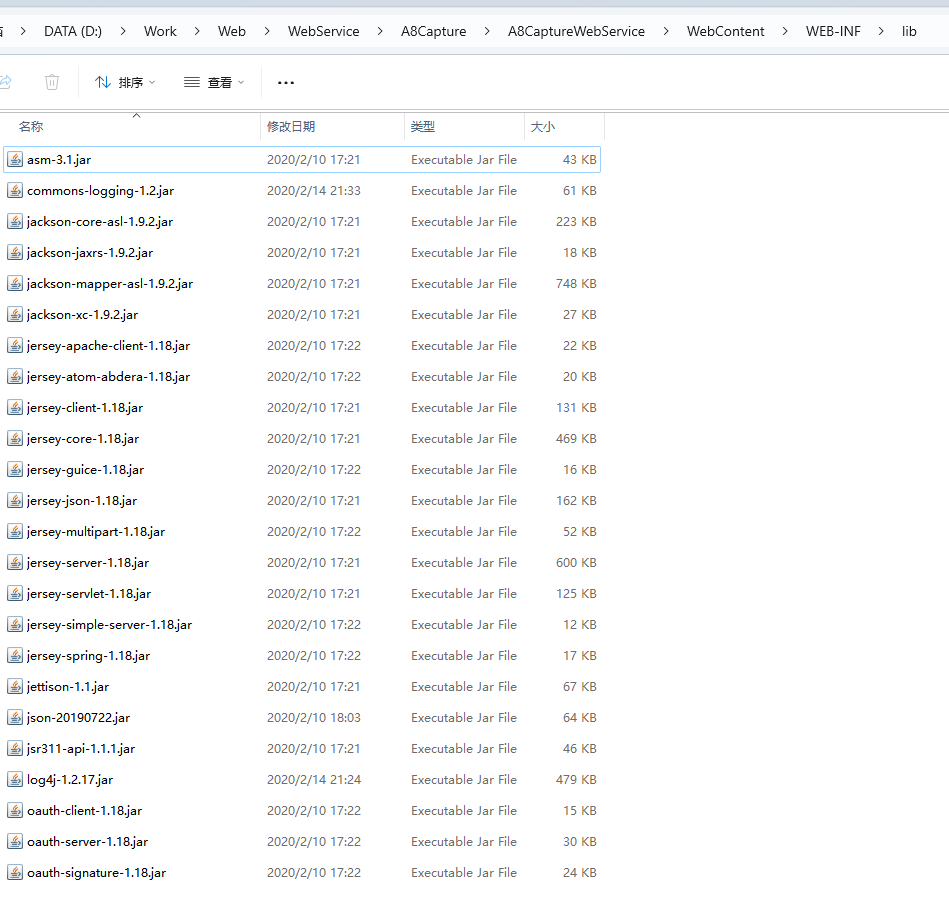
* 1. **Write ”A8CaptureJni” class code like these**



* 1. **Write ”A8Capture” class code like these**



**3.10 Copy these files to “./WebContent/WEB-INT/lib” directory.**



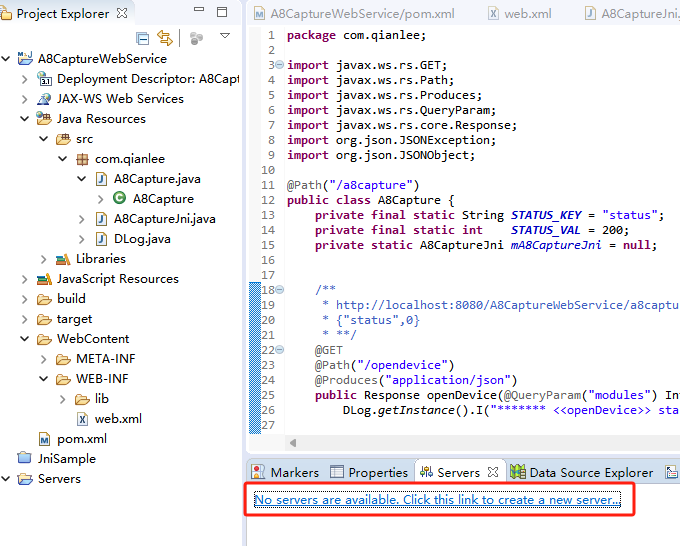
**3.11 Now let’s clean [eclipse workspace](https://crunchify.com/what-is-the-best-code-coverage-plugin-you-should-use-in-eclipse-ide/" \t "https://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey/_blank) and build project.**

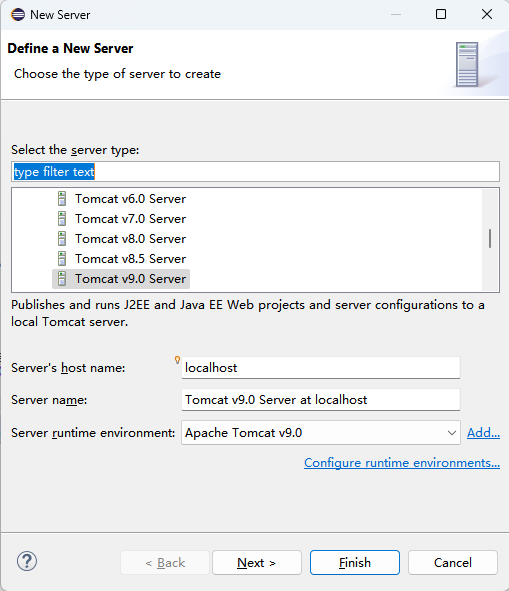
1. Project -> Clean

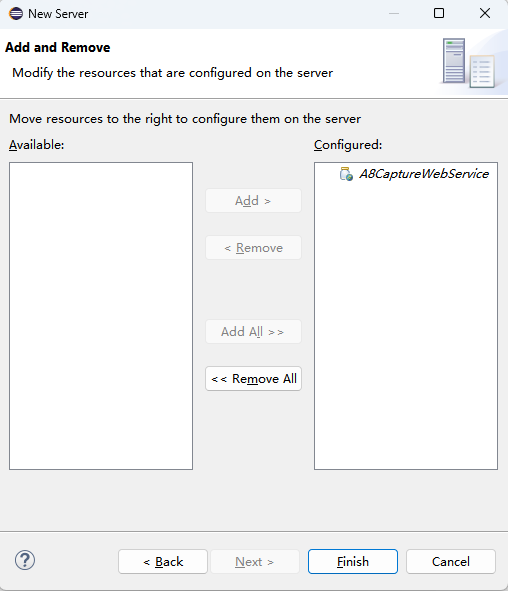
2. Project -> Right click -> Maven -> Update Project

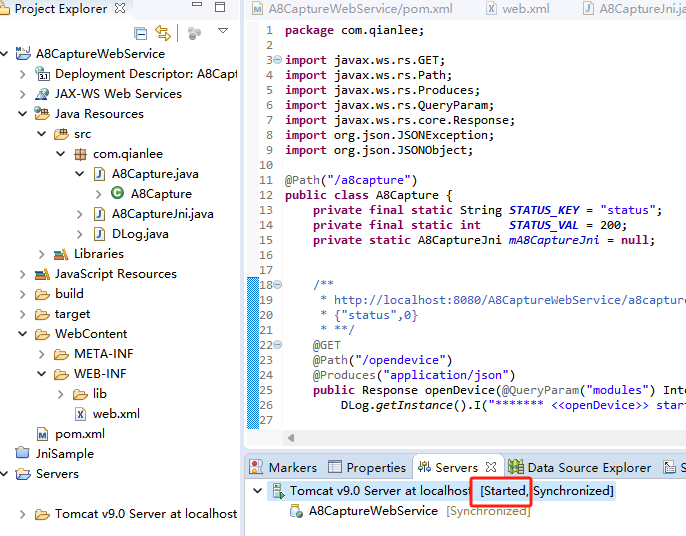
3. Project -> Right click -> Run As.. -> Maven Build (option 5) -> Add "clean install" -> Run

**3.12 Create new server and start it**









**3.12 Test**

