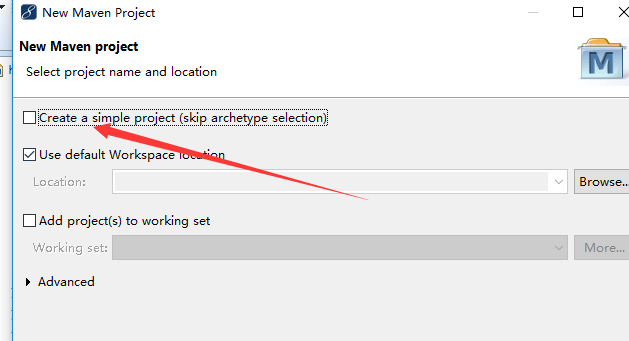
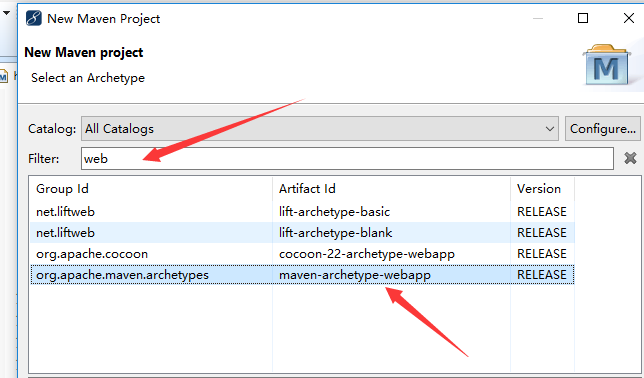
# 1、新建一个war的项目healer-web

## 1、不勾选Create a simple project



## 2、搜索web

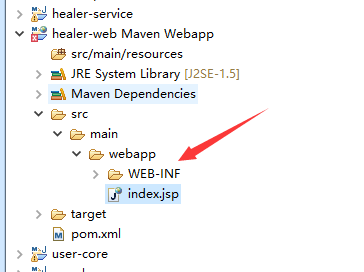


## 3 填写项目相关信息，package 我也不知道干啥的，先试试吧

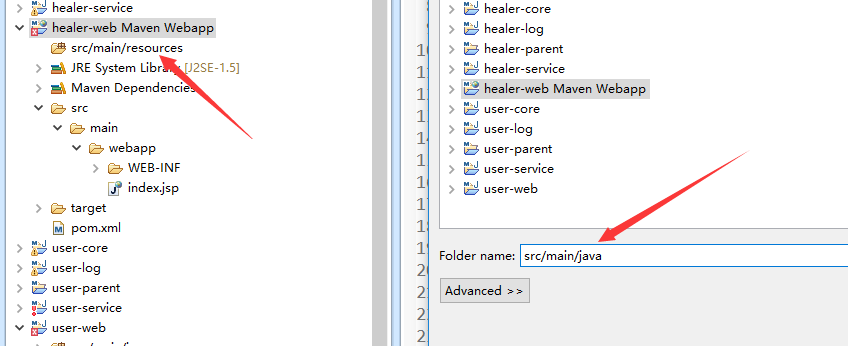
## 4、删去多余的东西，复制下面进去就行

|  |
| --- |
| <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/maven-v4\_0\_0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.hlj.maven</groupId>  <artifactId>healer-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <artifactId>healer-web</artifactId>  <packaging>war</packaging>  <name>healer-web Maven Webapp</name>  <url>http://maven.apache.org</url>      <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  </dependency>  </dependencies>    <build>  <finalName>healer-web</finalName>  </build>      </project> |

## 5.项目结构如图



## 6、没有java的类路径，建立相关目录



# 2、开始建立servlet

## 1、parent中pom.xml添加servlet的依赖

|  |
| --- |
| <!-- 31、servlet 提供给healer-web项目 -->  <dependency>  <groupId>org.mortbay.jetty</groupId>  <artifactId>servlet-api</artifactId>  <version>2.5.20110712</version>  </dependency> |

## 2、子类引入servlet依赖

|  |
| --- |
| <!-- 32、建立healer-web项目中的sevlet -->  <dependency>  <groupId>org.mortbay.jetty</groupId>  <artifactId>servlet-api</artifactId>  </dependency> |

## 3、编写servlet

|  |
| --- |
| **public** **class** HelloServlet **extends** HttpServlet {  **private** **static** **final** **long** *serialVersionUID* = 1L;  @Override  **protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp)  **throws** ServletException, IOException {    }  } |

## 4、web.xml写入servlet配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  id=*"WebApp\_ID"* version=*"2.5"*>  <display-name>Archetype Created Web Application</display-name>  <servlet>  <servlet-name>HelloServlet</servlet-name>  <servlet-class>zttc.itat.servlet.HelloServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>HelloServlet</servlet-name>  <url-pattern>/hello.do</url-pattern>  </servlet-mapping>  </web-app> |

## 5、hello servler中使用Uer对象，以及service中的方法，需要引入heaelr-service包（它自身依赖于healer-core，healer-log），父类中添加healer-service依赖，子类中添加id即可。这里不写代码

## 6、healer-servlet中写入User对象

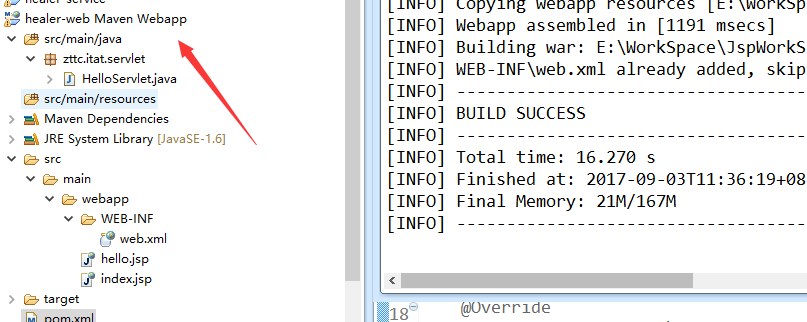
|  |
| --- |
| **public** **class** HelloServlet **extends** HttpServlet {  /\*\*  \*  \*/  **private** **static** **final** **long** *serialVersionUID* = 1L;  @Override  **protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp)  **throws** ServletException, IOException {  req.setAttribute("user", **new** User("admin","123","管理员123"));  req.getRequestDispatcher("/hello.jsp").forward(req, resp);  }  } |

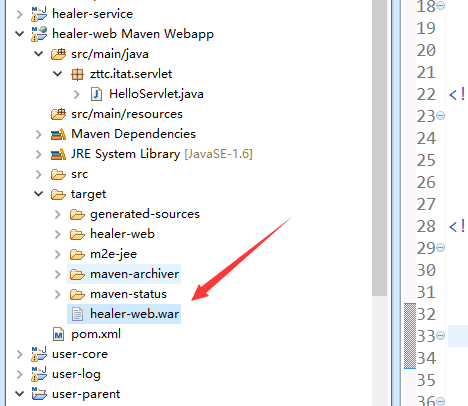
## 7、新建hello.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>Insert title here</title>  </head>  <body>  <h1>Hello---->${user.nickname }</h1>  <hr/>  <h2>ddd</h2>  </body>  </html> |

## 8、对web项目进行打包 成war

### clean package





## 9、将这个war放到tomcat中，启动tomcat，浏览器输入

### <http://localhost:8080/healer-web/hello.do>

# 3、上面的发布来回复制war，比较麻烦，现在解决，打包的时候，自动拷贝war到tomcat下面，但是我开发的时候maven版本比较高，不支持下面的，所以没有成功

## 1、插件copy-maven-plugin添加拷贝的插件，打包时候拷贝

|  |
| --- |
| <!-- copy-maven-plugin 拷贝插件，将war项目自动复制到tomcat目录下-->  <plugin>  <groupId>com.github.goldin</groupId>  <artifactId>copy-maven-plugin</artifactId>  <version>0.2.5</version>  <executions>  <execution>  <id>copy-war</id>  <phase>package</phase>  <goals>  <goal>copy</goal>  </goals>  <configuration>  <resources>  <resource>  <!-- 目标路径 tomcat路径 是个常亮我们在上面propertis中设置-->  <targetPath>${catalina.home}/webapps</targetPath>  <!-- 自己的war文件夹 目录-->  <directory>${project.build.directory}</directory>  <!-- 拷贝的文件 -->  <includes>  <include>healer-web.war</include>  </includes>  </resource>  </resources>  </configuration>  </execution>  </executions>  </plugin> |

## 2、properties中实现上面的tomcat路径

|  |
| --- |
| <!-- 27、配置插件的sql-maven-plugin -->  <jdbc.driver>com.mysql.jdbc.Driver</jdbc.driver>  <jdbc.url>jdbc:mysql://localhost:3306</jdbc.url>  <jdbc.username>root</jdbc.username>  <jdbc.password>147094</jdbc.password>  <catalina.home>D:\ProgramFiles\Study\tomcat6</catalina.home>  </properties> |

## 3、healer-web中添加好上面的插件之后，进行测试

### clean package

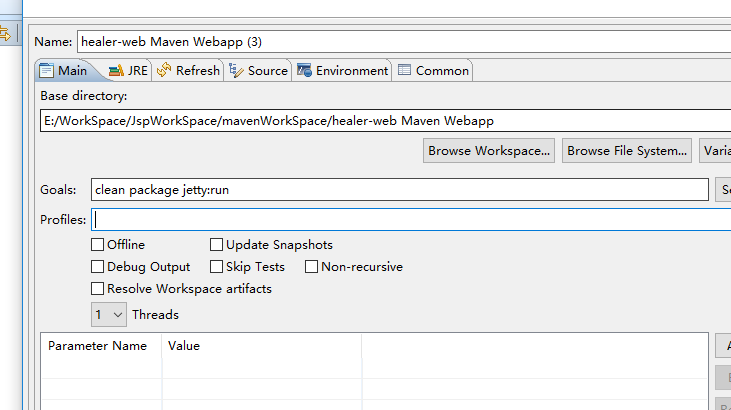
# 4、每次启动tomcat也比较麻烦，现在在解决这个问题，

## 1、自动启动tomcat项目，并且可以设置端口，以及上下文路径

|  |
| --- |
| <!-- 36、自动启动tomcat项目，并且可以设置端口，以及上下文路径-->  <plugin>  <groupId>org.mortbay.jetty</groupId>  <artifactId>jetty-maven-plugin</artifactId>  <configuration>  <!-- 每10秒钟 检测包是否更新，在修改了项目中可以10秒后重新加载 相当于热部署-->  <scanIntervalSeconds>10</scanIntervalSeconds>  <!-- 修改项目上下文路径，也就是项目启动的名字 -->  <webApp>  <contextPath>/healerjean</contextPath>  </webApp>  <!-- 修改端口 -->  <connectors>  <connector implementation=*"org.eclipse.jetty.server.nio.SelectChannelConnector"*>  <port>8787</port>  <maxIdleTime>60000</maxIdleTime>  </connector>  </connectors>  </configuration>  </plugin> |

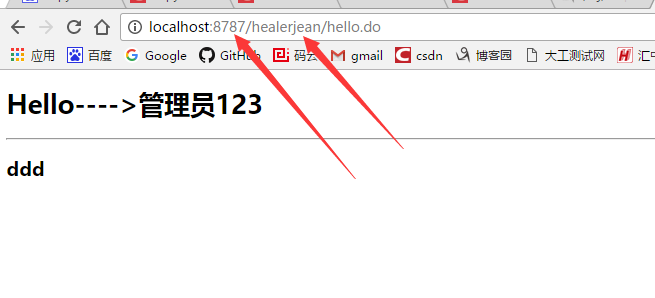
## 2、子类healer-web中引入插件，然后测试运行，需要启动tomcat，与上面的cope 3没有关系，

### clean package jetty:run



## 3、浏览器测试访问

<http://localhost:8787/healerjean/hello.do>



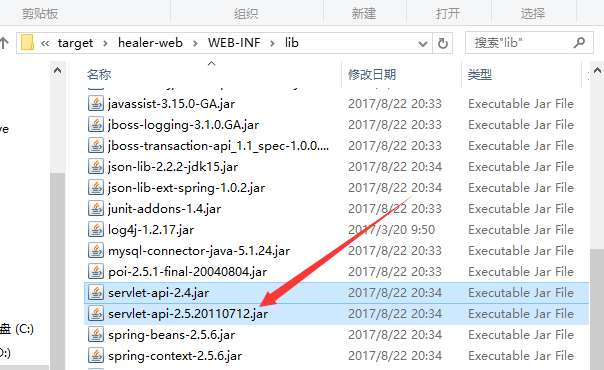
## 4、修改hello sevlet，测试10秒更新一次，

### 不需要重启启动tomcat，



# 5、项目运行，在haeler-web启动的时候不可以debug其他的healer-core，等项目，因为我们的项目中引入的时候，讲他们打成jar包了

## 1、在打包时候讲servlet的包打进了war包，因为sevlet依赖在tomcat容器中已经具备了，所以不需要打入



### 1、servlet-api中配置使用域，在使用java文件的时候提供支持<scope>provided</scope>

|  |
| --- |
| <!-- 31、servlet 提供给healer-web项目 -->  <dependency>  <groupId>org.mortbay.jetty</groupId>  <artifactId>servlet-api</artifactId>  <version>2.5.20110712</version>  <scope>provided</scope>  </dependency> |

## 2、一般debug的时候healer-core用junit单元测试，heaelr-service 用ease-moke单元测试